# SECURE SOFTWARE DEVELOPMENT AS A SOCIAL PRACTICE: investigating the dynamics of socio-technical factors in the security practices of developers in SMEs


**Paula Fiddi**

Department of Computer Science

Linacre College

University of Oxford

This dissertation is submitted for the degree of

*Doctor of Philosophy*

Trinity 2025

# Acknowledgements

I am first grateful to the Almighty God for seeing me through this journey. To Him be all the glory and praise.

The completion of my study would not have been possible without the immense guidance and support of my supervisors, Prof. Marina Jirotka and Prof. Ivan Flechais; I am so thankful for their unwavering support and patience. They gave me an opportunity of a lifetime and encouraged me through the journey.

I am also grateful to Dr. Helena Webb, Dr. Ross Gales, and Dr. Carolyn Ten Holter for their valuable advice and for helping me review my thesis chapters. Thank you to Linacre College for hosting me and supporting me throughout my time at Oxford.

I would also like to give special thanks to my loving parents, Mr. Morrison Fiddi and Mrs. Preye Obinyan, my husband, Oluwatoyin Olaoye, my whole family, friends, and the Hillsong Oxford Church for their invaluable support. Finally, thank you so much, Oxford University, for this tremendous opportunity to learn, grow and make an impact over the years.

# Abstract

The software security industry primarily focuses on the technical aspect of the software development process, with limited studies examining the impact of human interactions, organisational contexts, and social factors on developer practices. The software application development industry is worth trillions of pounds, with a user base of billions per year. For decades, software engineering and security research disciplines have focused on end-users' experiences with software applications and increasing security awareness. Unfortunately, there has been little work considering how the developers of these software products navigate the issues of software security within their specific context. Developers in micro, small, and medium-sized enterprises (SMEs), including start-ups and sole proprietorships, are creating a significant proportion of these software products to meet the high demand. These categories of organisations face numerous challenges, including limited resources, a lack of cash flow for exploring costly security technologies, constrained managerial capabilities, working within an informal structure, regulatory burdens, and more. While there is research evidence highlighting existing security best practices designed to cater to software developers within large and structured environments, little attention has been given to the practices adopted by developers in smaller organisations.

The research study presented in this thesis focuses on understanding the security practices of developers in SMEs and how their interactions, dialogues, and negotiations with stakeholders impact these practices during the software development process. The study employed the grounded theory methodology in three phases. The first phase involved an empirical exploration of the current socio-technical factors relevant to the security practices of developers in SMEs, conducted through 25 semi-structured interviews. Next, a conceptual framework was developed to situate this study within the existing literature. And finally, a theory was generated and evaluated to provide a more grounded explanation of the experiences and challenges faced by developers in SMEs, their perception of secure software development practices, and the factors that influence these practices.

This study investigates the experiences and practices of developers in SMEs from different sectors across ten countries and four continents (Africa, Europe, North and South America). Grounded theory methodology was used to generate a theory, and follow-up interviews and scenario-based activities were analysed for theory evaluation. The first outcome of the study

is the C.A.M.S. taxonomy, a classification of socio-technical factors that influence the security practices of developers in SMEs. Then, a conceptual framework is presented that illustrates the relationship between concepts in the C.A.M.S. taxonomy and existing literature. The social theory of Security Practices of Developers in SMEs is then generated to theorise the secure software development practices of developers within their specific organisational context - SMEs.

Key findings from this study indicate that the primary concern for developers in SMEs is navigating human, social, and organisational issues that affect how security practices are operationalised into practical actions and activities. Furthermore, this concern is elevated due to the socio-technical factors such as negotiating and finalising software requirements based on the client's specifications, user needs, access to affordable, usable, and useful security technology, a security knowledge acquisition structure, and support from thriving 'expert' communities. Therefore, implementing security best practices in software development requires developers in SMEs to possess better social skills to navigate communities and improved negotiation skills to communicate effectively with stakeholders, thereby avoiding the loss of financial support for software projects.

# Contents

# List of Figures

# List of Tables

# Chapter 1 Introduction

Software applications have become increasingly ubiquitous in today's society. The increasing need for more software applications has led to a rise in the number of people working as software developers. In addition, advancements in the technology industry and the increased use of smart mobile devices have led to a rise in the use of software applications and strategies that support the efficiency of actors involved in development processes, enabling them to meet high demands.

According to a survey in the longitudinal study by Statista (2021), conducted between 2011 and 2020, the number of software development professionals, including both employed and self-employed developers, registered in the United Kingdom alone has risen to over 400,000, with a growth rate of approximately 22% (Statista 2021b). The number of software developers worldwide is estimated to exceed 20 million, with this figure expected to rise to approximately 45 million by 2030 (EDC 2021; Statista 2021a). Concurrently, more software companies and technology service firms are emerging, and developers build most software applications in micro-businesses, small to medium-sized companies with fewer than 250 personnel (Statista 2021b). These numbers are astounding, but not surprising, given the increase in demand for software products around us. It is therefore crucial that software application products, their creators, and the environments in which they are studied, understood, and supported for improvement, as they form core elements for advancements in technology (Fuggetta and Di Nitto, 2014).

The expectations for how developers should work with code, use tools effectively, or adhere to the latest development practices are very high (Wurster and van Oorschot, 2009). In addition to meeting these high demands and expectations, developers must ensure that the software products are delivered to a certain standard, which includes excellent functionality, efficiency, a visually appealing design, and resistance to malicious attacks and vulnerabilities

(Green and Smith, 2016). Researchers such as Wurster & van Oorschot (2009) have even accused developers of being villains in the secure software development agenda, claiming that they are not willing to receive sufficient security training to improve app security (Wurster and van Oorschot, 2009). However, other researchers such as Green & Smith (2016) have argued that developers should not be considered the enemies because just like end-users, developers are individuals too, who need more developer-centred approaches that support their software development and security practices (Green and Smith 2016; Acar, Stransky, et al. 2017; Assal and Chiasson 2019). This study subscribes to the school of thought that developers also need support in their craft, practices, and activities during the different stages of the software development process.

For decades, studies in software engineering and security research have focused on changing the security behaviours and experiences of software application end-users. However, not enough attention has been given to the developer. There has been little work done to explain developers' experiences, how they face the challenges of building software within their organisational context or how they navigate the security activities in the development process effectively, to meet the high demands and expectations for software products. Developers are key individuals within various organisations or owners of small businesses, who play different roles during the development process of an application software product; hence, this study focuses on the activities and practices of developers as both users and practitioners. Supporting the security practices of application developers is particularly important due to the growing development of cybercriminals who use malicious programs to infect smart devices, with the sole purpose of collecting private data (Arabo, 2016). Fast-paced environments and the poor implementation of secure software development practices during development processes may lead to vulnerable applications that hackers can exploit (Arabo, 2016). Furthermore, the absence of security expertise within the development team makes the products susceptible to security attacks.

Security provides a way to address threats and breaches with measures, technologies, and policies to support and protect users and organisations. In recent years, it has been widely acknowledged in various studies that security cannot be fully achieved through technology

growth alone; it must take into full consideration the practices of all users involved, both internal (professional users) and external (end-users) users (Dhillon and Backhouse 2001; Stanton et al. 2005; Lebek et al. 2013; Stedmon et al. 2016). Furthermore, studies that have investigated security practices within organisations focus mainly on the technological infrastructures within larger companies and how the decisions that arise from the development of secured systems impact their external (end) users' needs only (Osborn and Simpson 2015) if humans are considered the weakest link in the security (Schneier 2000), it unrealistic to exempt the developers from human errors or insufficient knowledge of security topics when they work on completing their tasks.

Furthermore, developers in small to medium-sized organisations, in particular, may be further constrained by human, social, or organisational factors that can influence their security activities and decisions. Finally, security issues and challenges are experienced differently in various organisations, which may be due to differences in organisational structures, policies, practices, or decision-making processes (Backhouse and Dhillon, 1996). It is therefore essential to have a holistic view of developers' security practices during the software development process, not just from a technological angle. For simplicity, software applications may be used interchangeably with the popular words "app or apps" in this thesis.

## 1.1 Motivation of this Study

This study is motivated by the limited evidence explaining the secure software development practices of developers within SMEs and the significant gap in security research in identifying issues that are particular to these developers. Thus, this study takes a closer look at the activities and interactions of developers in SMEs to understand how their practices are formed during the software development process. Factors such as limited resources (workforce and finances) and infrastructure, which particularly influence the operations of small-scale organisations (Amrin, 2014), have been considered in this study to determine the role various factors play in the development process for SMEs. The evolution of security has relied heavily on larger organisations that have access to more resources and infrastructure to

produce requirements for security standards, tools, and the development of policies (Osborn and Simpson, 2015).

Although numerous valuable and successful tools and models have been developed through research funded by companies with higher asset value (Lee and Jang 2009), small-scale businesses continue to struggle to incorporate these security initiatives and tools into their day-to-day activities (Osborn and Simpson 2015). There is little to no theoretical basis for understanding the secure software development practices of developers within these small organisations. Also, developers working outside large mainstream organisations may not have access to professional security teams or frameworks that support secure programming (Acar, Stransky, et al. 2017). This gives rise to a significant research gap in software security research, specifically in identifying issues particular to developers in SMEs, to provide support that is well-suited for them. Developers require support in various ways that enable them to adopt effective security practices throughout the development process.

In summary, many studies done in secure software development research do not generate theories or frameworks to conceptualise practices and explain the perspective of developers in a way that informs secure software development, particularly within smaller social and organisational environments. Therefore, generating theories and frameworks that explain security practices through secure software development can serve as theoretical bases for more empirical research in software security and software engineering domains. Other mature disciplines have established theories that support their studies, and researchers in software engineering and security are now pushing for the creation and application of more theories, which is the motivation behind this study. Therefore, this study aims to develop a theory that explains how developers navigate security practices for secure software development while working through the dynamics that exist within their business ecosystem.

## 1.2 Research Objectives

This study is motivated by the question: '**How do software application developers in SMEs practice security effectively?'** The objective is to understand and explain the practices of developers in SMEs (and micro-businesses) by building a theory that explains the core

concepts that shape these practices. The main research question is further broken into following sub-questions:

i. **RQ1:** What current key socio-technical factors are relevant to the security practices of software application developers in SMEs? – To scope and frame the study.
ii. **RQ2:** How can we theorise the relationship between these socio-technical factors and the security practices of app developers in SMEs? – To build and evaluate a theory of the security practices for developers in SMEs.
iii. **RQ3**: What recommendations can insights from the theory and study provide developers in SMEs? - To support the security practices of SME developers during the software development process.

## 1.3 Thesis Structure

This thesis is presented in eight chapters, with the first chapter introducing the research area and my motivation for conducting this research. The directions for other chapters are summarised below:

**Chapter 2** presents a preliminary review of current research and related works in secure software application development, framing and establishing the context of this study. The chapter starts by outlining the need for a preliminary literature review before implementing grounded theory methodology. The interplay between the software application development landscape, the role of small and medium-sized enterprises and the social nature of software security is described. Finally, the developer's role in the development process, as well as human behaviours, social and organisational issues related to security activities, taking into consideration the peculiarities of small, medium-sized organisations and micro-businesses.

**Chapter 3** describes my methodological approach to addressing the research objectives and how the study was executed. This study reviews well-established knowledge and research paradigms and methodologies commonly used in the software engineering and cybersecurity research disciplines to support its philosophical stance and define the context. Informed by the various paradigms and methodologies in the literature, the grounded theory methodology

was chosen for this study, along with its execution. The chapter concludes with an approach to addressing researchers' biases. Semi-structured interviews were the primary source for data collection, and conceptual framework analysis was conducted to synthesise findings presented in Chapters 4 and 5**,** probe the data further in comparison to existing literature, and situate the study within the security and software engineering bodies of knowledge.

**Chapter 4** presents the initial result of the grounded theory process and an overview of the first set of conceptual categories developed, which are represented in the taxonomy. These initial conceptual categories describe the current socio-technical factors that may influence the security practices of developers in SMEs. The results of the analysis are classified in a taxonomy called the CAMS taxonomy, which stands for Context, Attitude, Motivation, and Support. The CAMS taxonomy is a classification of socio-technical factors identified as relevant to current security practices during the software development process in SMEs. In addition, the taxonomy is an organisational system and a representation of the relationship between initial categories and their sub-categories. This process contributes towards the construction of the theory presented in **Chapter 5**.

**Chapter 5** presents further data analysis, leading to the development of a theory on *Socio-technical Security Practices of Developers in SMEs*. This social construct explains how developers in SMEs and micro-businesses manage and implement continuous security activities as part of their software development process. The theory comprises theoretical categories that illustrate how activities influenced by socio-technical factors shape the security practices of developers. The theory also draws insight from the initial categories captured in the CAMS taxonomy presented in **Chapter 4**, which have been refined to provide context for understanding the various elements of the theory.

**Chapter 6** situates and further elaborates on the theory of *Socio-technical Security Practices of Developers in SMEs* within the context of software development and security research. A conceptual framework is employed to compare the outcomes of this study with those of other empirical studies and existing theories in this research area.

**Chapter 7** discusses broader implications of the theory and framework. This study demonstrates the impact of its outcomes for secure software development initiatives, responsible innovation, and security in agile methodologies.

**Chapter 8** summarises the results and contributions of this study. Some recommendations and a road map for further studies in secure software development research are also presented in this chapter.

# Chapter 2 Literature Review

## 2.1 Introduction

This chapter presents a review of the current state of the software application development landscape, framing and establishing the context for this study. It starts with an outline of the need for a preliminary literature review when implementing grounded theory methodology. Then, it describes the interplay between software application development practices, security practices, and the role of software developers in small and medium-sized organisations in meeting the demands for reliable and secure software artefacts. The concept of responsible innovation, as well as human and social dynamics in software development practices, is introduced to highlight research gaps. The chapter concludes with a review of the developer as a practitioner and user within the development lifecycle, presenting the current state of secure software development in practice.

## 2.2 Traditional Literature Review and Using Grounded Theory

Contrary to traditional research practices, Glaser and Strauss (Glaser and Strauss 2017) in *The Discovery of Grounded Theory* urged researchers using grounded theory to conduct a literature review after a substantive theory emerges from independent data analysis. They advocated for qualitative research conducted using grounded theory methodology to delay extensive review of existing research and allow their work to be viewed through fresh lenses, thereby minimising existing biases and forcing the data to fit existing theories. Other grounded theorists, such as Charmaz (Charmaz 2006), build on Glaser and Strauss' classic grounded theory strategies and agree on avoiding an in-depth initial literature review process. However, there are no restrictions on conducting a prior literature review of areas that guide

the principles and practices of the methodological process and the terrains of the research area.

In response to the recommendations of expert grounded theorists, the initial literature review process was conducted before data gathering to provide the researcher with sufficient background on relevant problem areas, frame the study area, and highlight the contribution of this body of work to both industry and academia. Grounded theorist Charmaz recommends that researchers conduct further rounds of rigorous literature review to situate and strengthen the theoretical framework that emerged (Charmaz 2006). The next round of the systematic literature review process is presented in **Chapter 5** as a conceptual framework that provides an in-depth examination of existing relevant empirical studies and extant theories in state-of-the-art secure software development research. The subsequent sections below aim to cover discussions within the secure software application development landscape, identify the distinct role and responsibilities of SMEs and developers in this landscape. This review highlights how further research, utilising appropriate social methods, can enhance the growing field of secure software development.

## 2.3 Software Application Development Landscape

Before addressing issues relating to software application development, it is essential to first review the development process to gain insights into the landscape that developers need to navigate. The software development process or lifecycle has been studied in different capacities in computing research. Traditional development process models such as the waterfall, spiral, iterative, or prototyping models have been used for developing software products since the term' software development process' gained significant visibility in the eighties (Woodhouse 2005). Existing approaches to software development have been revolutionised by the widespread adoption of concepts such as open-source software development, cloud computing, agile methods, and various development platforms for mobile software and game applications (Fuggetta and Di Nitto, 2014).

Both large and small organisations are continuously adopting these relatively new development methodologies, processes, and frameworks. Agile methodology has been particularly adopted by small-scale organisations in software development projects as a means of providing more flexible, customer-focused, and rapidly delivered software products or services (Beck et al., 2001). Generally, the software development process involves a complex management of organisational policies, tools, support environments and people. According to Alexandre et al. (Alexandre, Renault, and Habra, 2006), SMEs are more likely to follow simplified development processes, which may not prioritise further training or improvements in project and risk management. Consequently, software developers involved may not be fully aware of the security implications of their work or may not have received adequate training on security topics. With the rapid growth rate of software development organisations worldwide, improving software development processes and practices has become increasingly prevalent and gained momentum in both industry and academia.

The typical software development lifecycle of modern software applications involves the following stages: feasibility study, requirement elicitation, design, programming or implementation, testing, deployment, documentation, and maintenance of application software products (Pranam, 2018; Leau et al., 2012). These stages collectively make up the Software Development Lifecycle (SDLC). The discipline concerned with all elements of the software development lifecycle is called Software Engineering (Wasserman, 2010). Software Engineering comprises a mixture of tools, methodologies, practices, architectures, and technologies (such as programming languages, operating systems, database management systems) that are pulled together to ensure a successful outcome for a software application product (Cockburn 2002). The software engineering field involves finding solutions to various customers' problems through a systematic development of software applications within cost, time, and other constraints (Lethbridge and Laganiere 2005).

Software applications commonly referred to as "apps" are a type of computer software designed and developed to perform specific tasks and facilitate a variety of functions such as managing and manipulating data, coordinating resources, gaming, or communicating, which are beneficial to their users (Philipson 2004). For most parts of this thesis, the term 'app' or 'apps' will be used to denote software application products and systems.

Over the past decade, the development of apps has shifted from being primarily driven by large companies to mass production models adopted by small to medium-sized companies and independent developers. (Philipson 2004; Fuggetta and Di Nitto 2014). This is due to the rapidly evolving technology of smart and mobile devices to meet the needs of users. Apps can be built on various mobile platforms, desktop computers, or run on the Internet using web technologies. All these development environments pose multiple challenges to the software engineering community. One of the significant challenges facing software app development is the impact of the Internet on the development process, execution, and use of apps (Fuggetta and Di Nitto 2014).

Many developers in software production lines are unfamiliar with best practices for developing secure software applications (Kazymyr and Mokrohuz 2016), thereby exposing software applications to potential attacks. In addition, the rapid growth of the mobile, gaming and service software market comes with various challenges in the development process of these software from technical constraints (such as cross development platforms) to management issues, limited resources (time, personnel, budget), testing and support facilities and the frequent change in customer requirements (Joorabchi, Mesbah, and Kruchten 2013). Other challenges that need to be considered during software development include security, ease of use, memory capacity, energy consumption, multiple stakeholders, and other characteristics specific to the type of software, which are addressed at different phases of the development process (Khalid, Zahra, and Khan 2014).

### 2.3.1 Software Development Practices

Generally, software development practice refers to a collection of concepts, principles, processes, methods, tools, and techniques that a software engineer calls upon daily to complete a task (Humphrey, Kitson, and Kasse 1989). Professionals implement these practices to ensure the successful development, delivery, and maintenance of a software product. The core of software development and engineering practices is said to be built on the four phases of problem-solving introduced by mathematician George Polya: understand the problem, plan a solution, execute the plan, and look back and reflect (Polya 2004). In the software engineering context, these can be compared to identifying and analysing the

problem, software modelling and design, code generation, testing and validating that the features and functionalities match the specified requirements (Van Vliet and Van Vliet 2008; Humphrey, Kitson, and Kasse 1989). Practices are essential because they provide the details needed to navigate the road from idea to reality, showing how to implement principles, methods, processes, and tools, while also highlighting possible roadblocks (Van Vliet and Van Vliet, 2008).

Practices have been primary subjects of discussions and debates in various fields and domains, including anthropology and sociology, with definitions extending beyond daily short-term activities to more structured patterned activities adopted by multiple fields and institutions (Rouse 2007). Practice theorists include philosophers (such as Wittgenstein 1953), social theorists and sociologists (Bourdieu 1977, Giddens 1979, Schatzki 1996), anthropologists (Ortner 1984) as well as science and technology theorists (Callon 1991, Latour 1992 and Rouse 2006). Despite the differences in views on practice, practice theories aim to explain the concept of practice. The concept of 'practice' has been debated by different theorists and philosophers. For example, Bourdieu's works are widely known for presenting 'practice' as a means of explaining and generalising the notion of *habitus* – a concept that describes the habits, skills, and dispositions of an individual as socially ingrained (Bourdieu 1977). Bourdieu is credited with influencing the social theoretical arguments of the concept of practice (Shove, Pantzar, and Watson 2014). Another philosopher, Theodore Schatzki presented diverse theoretical positions of 'practice' in a collective study called the '*The Practice Turn in Contemporary Theory',* that describe practice theories as "a collection of accounts that promote 'practices' as the fundamental social phenomenon"(Karin Knorr Cetina et al. 2001; Theodore R Schatzki and Schatzki 1996). Schatzki's work builds its basic premise on the works of philosophers Ludwig Wittgenstein and Martin Heidegger (Theodore R. Schatzki, 1993), who argue that practice is an "open-ended spatial-temporal manifold of actions" and a set of organised activities, tasks, and projects (T.R. Schatzki, 2005).

Shove et al (Shove, Pantzar, and Watson 2014) gave a more recent definition of practice, which is the "interdependent relations between materials, competences and meanings" and practices evolve as these elements change. This definition builds on the work of Reckwitz, a cultural theorist, who defines practice as a "way of understanding or behaving around a

specific act that can be normalised by routine at different times and places" (Reckwitz 2002; Ashenden and Ollis 2020). Materials represent tools, technologies, infrastructures, or physical objects used in practice, while competencies are the skills, techniques and knowledge individuals need to execute the practice; and meanings are cultural understandings, ideas expectations and shared meanings disseminated through practices (Ashenden and Ollis 2020; Shove, Pantzar, and Watson 2012).

In software engineering, various practices have been developed to help teams communicate, plan, design, develop, manage, and maintain software, ensuring that the resulting product is accurate, efficient, and adaptable (Zelkowitz et al., 1984; Van Vliet and Van Vliet, 2008). Software engineering practices are primarily analysed and discussed across the various phases of software development, or within the methodologies and development strategies implemented throughout the lifecycle. These practices can be represented in the following categories:

i. **Communication and collaboration practices**: Software requirements and specifications must be gathered thoroughly and communicated within software teams, as well as between stakeholders and potential users (Pikkarainen et al., 2008). Effective communication practices, whether in-person or remote, are essential to ensure that customers' requests are accurately understood, and the proposed software solutions and expectations are clearly communicated. Adequate communication mechanisms must be in place to support both formal and informal conversations and negotiations throughout the software development lifecycle (Jarzębowicz and Sitko 2019). Furthermore, studies have shown that development teams rely heavily on collaboration between colleagues to support knowledge sharing and manage dependencies across various technical activities (Whitehead, 2007). Different organisations have various communication and collaboration approaches; however, these practices can be challenging in larger software development projects and teams.
ii. **Management practices**: Software projects involve people, processes, and tools over extended periods and require proper planning and control to ensure the successful completion and delivery of the product (Van Vliet, 2008). Studies have shown that software projects are considered successful due to effective planning, cost estimation,

measurement, milestone tracking, quality control, risk, and change management (Jones 2004). The ever-evolving nature of the software application development landscape is primarily characterised by the effective management of the project itself, the development methodology, the team's profile, organisational structure, and the technology that supports everything (Van Vliet 2008; Wysocki 2010). These management practices vary across different organisations and project sizes to ensure that the resulting software product is delivered as effectively and efficiently as possible.

iii. **Modelling and design practices**: These practices in software engineering help engineers construct software effectively by representing its features in a textual or graphical manner to express the customer's requirements (Bjørner, 2006; Van Vliet, 2008). Software design and modelling provide different ways to view the data, architecture, user interface and components of the software before actual construction and implementation (Otero 2012). Software engineering teams apply agile methodologies, such as the Rapid Application Development (RAD) model, to swiftly deliver software prototypes and gather feedback from customers for an iterative software development process (Van Vliet, 2008; Naz and Khan, 2015).

iv. **Development and deployment practices:** involve coding, testing, deployment, and maintenance of software (Humphrey, Kitson, and Kasse 1989; Van Vliet 2008). Agile development, continuous integration, delivery, and deployment practices are examples of practices adopted by many organisations to frequently release new software products (Shahin, Babar, and Zhu, 2017). Furthermore, variations in software project sizes and types have also been described as key factors in determining the approaches, tools, and practices that organisations implement in the software development process (Jones, 2003).

The practice of software engineering and development encompasses various activities that describe the problem domain and purpose of the software, its structure, computation, and behaviour in response to user interactions and the environment (Päivärinta and Smolander 2015; Coleman and O'Connor 2008). We cannot focus primarily on computational activities;

attention must also be given to other activities, such as business decisions, human interactions, and negotiations, that are pertinent to the success of the software product.

### 2.3.2 Communities of Practice and Knowledge Building in Software Development Teams

Research studies in software engineering recognise that various human activities are involved in the software development process, as it typically requires coordination and communication among individual developers, teams, or organisations (Easterbrook et al. 2008). Communication and coordination require individuals or teams to share a common interest in the development of a reliable software product hence laying the foundation for a community of practice. Wenger defines a community of practice (CoP) as a "group of people who share a concern or passion for something they do and learn how to do it better as they interact regularly."(Wenger 2011)

Members of software development teams join various communities and engage in shared activities to feel supported, discuss challenges and share information. These communities are comprised of experts who share a common interest in the domain, seek to expand their skills and knowledge in specific areas, and thereby advance the body of knowledge within the domain (Wenger, 1999, 2011). In studies on the communities of practice within large software development organisations, various examples of cross-team communities are created as initiatives within these organisations to facilitate agile transformation and continuous improvements (Paasivaara and Lassenius 2014). Communities such as Developer CoP, Feature Coordination CoP or User Experience CoPs are examples of CoPs are built by people from different parts of the organisation to effectively solve problems that cut across various teams or share development best practices (Kahkonen 2004; Paasivaara and Lassenius 2014). These CoPs are either intra- or inter-organisational based on individual needs and the organisation's purpose and policies (Pattinson and Preece 2014). In small and medium-sized organisations, studies have shown that communities of practice are primarily used for knowledge acquisition, knowledge sharing, and innovation support (Pattinson 2011; Rossignoli et al. 2023). Compared to larger organisations, communities of practice in SMEs appear to be unplanned due to the fast pace, resource constraints, or lack of a formal strategy,

necessitating the need for members to seek opportunities to learn outside the organisation (Pattinson 2011). This facilitates the adoption of new tools, techniques and methods that support the personal and professional development of team members (Kahkonen 2004). Building software applications requires developers to be part of various communities or practices that help them evaluate their practices, experiences, and approaches in specific development areas, and educate them about emerging technologies. This may also provide them with a sense of belonging as they feel supported when faced with complex challenges during the development process.

### 2.3.3 Security Practices in Software Development

Building software applications that are secure and protect the interests of users requires developers to follow security practices at different phases of the development process (Howard 2004). Like other practices in software engineering, security practices can also be studied beyond computational activities. According to the literature, one approach is to closely examine the role of security in each phase of the Software Development Lifecycle (SDLC). The Software Development Lifecycle is a process that encompasses planning, requirements gathering, design, implementation, testing, and validation activities for a software application (Pranam, 2018). In the past, security has always been considered outside the focus of SDLC. Still, with the recent rise in security breaches, cyber threats and attacks, the software engineering community recognises the value of integrating security in the different phases of the development process (Geer 2010; Assal and Chiasson 2018). Although the importance of security in the software development process is being acknowledged, software functionality and features are still prioritised over security, with security measures implemented as an afterthought when designing software applications (Geer 2010).

Integrating security at the application level means that developers are faced with a long list of recommendations from security frameworks, expensive guidance tools and an overload of theoretical information (Apvrille and Pourzandi 2005). This results in only a few developers adhering to best practices for producing secure software. Although an increasing number of developers are becoming more aware of the importance of integrating security in coding and designing of software, some developers still struggle with making decisions about supporting

technologies, secure programming techniques, advice resources and the best approach to ensuring that they build secure software products (Balebako et al. 2014; Acar, Stransky, et al. 2017). The following section describes some security initiatives that have proposed integrating security practices throughout the software development process. The strengths and limitations of these initiatives for developers in small and medium-sized companies are also highlighted.

### 2.3.4 Secure Software Development Initiatives

Several studies have been done to produce a variety of processes, recommendations, and guidelines with the intent to support software companies and independent developers in implementing best security practices in the software application development lifecycle. Some of the common initiatives recognised in the software development landscape are discussed in the sections below, highlighting the challenges developers in SMEs face when attempting to implement frameworks, guidelines, or recommendations designed by these initiatives.

**Microsoft's Security Development Lifecycle**

In 2004, Microsoft introduced one of the first initiatives to encourage early integration of security practices in the software development lifecycle, called the Security Development Lifecycle (SDL) (Microsoft Corp. 2004). The Microsoft SDL comprises 12 recommended practices (**Figure 2.1**) for various stages of the development process, supporting security compliance requirements to enable developers to build more secure software, regardless of the methodology or platform used. However, this initiative was originally designed for internal use by Microsoft developers and engineers, who are part of a large, structured, and well-resourced organisation. These practices were further grouped into three categories to accommodate the heavy SDL requirements within the lightweight Agile Framework, known as SDL-Agile. In the first category, the Microsoft SDL-Agile recommends that developers consider certain practices as critical to every sprint requirement, such as completing at least one security training and modelling threats for all new features. Verification tasks, including analysis tools, security design reviews, and planning activities, should be considered 'bucket-requirements'-performed regularly but not compulsory for every sprint. The third category

consists of one-time requirements, such as identifying the team's security expert and establishing a standard incident response plan, that must be performed at the beginning of the development process, within a set period.

It may be challenging for developers in smaller organisations to fully implement Microsoft's SDL practices due to various constraints and limited resources. Even with the introduction of the SDL-Agile, Microsoft only accounts for fitting the SDL requirements into shorter release cycles, not for leaner teams or the impact of human interactions and constraints associated with smaller organisations.

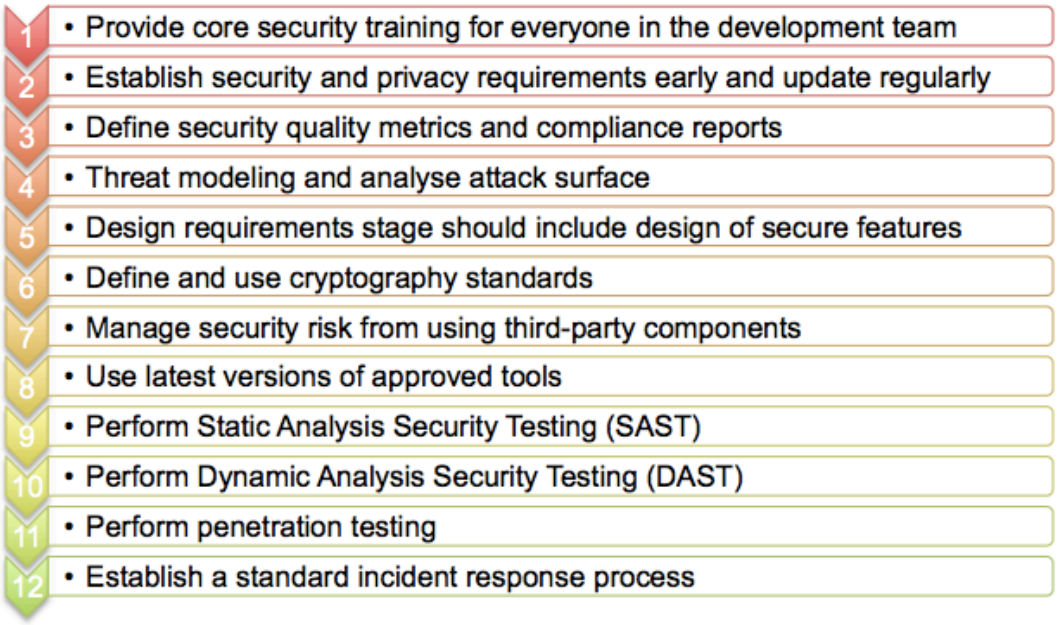



Figure 2.1– Microsoft Secure Software Development Practices (Microsoft Corp. 2004)

**The BSIMM Framework**

The Building Security in Maturity Model (BSIMM) (BSIMM by Synopsys 2022) was first introduced in 2008 and created to consolidate and analyse the activities and terminologies employed by various software security initiatives or programs used by different organisations. Currently in its 13th iteration, the model quantifies the practices employed by 130 organisations in building secure software. The BSIMM presents high-level insights from the analysis of security program activities in these organisations, highlighting current top

software security activities, growth areas in code review, cloud and attack intelligence, supply chain trends, and recommended actions for organisations to improve their existing security programs.

The BSIMM framework (**Figure 2.2**) comprises 12 main security practices organised into four categories: Governance, Intelligence, Software Security Development Lifecycle (SSDL) Touchpoints and Deployment. Like the Microsoft SDL initiative, the BSIMM is intended for use by software application development organisations of any size, providing a way to present high-level insights into how a company's security practices compare to others and how they can plan for their software security initiative (McGraw, Migues, and West 2018). However, the BSIMM framework does not provide detailed instructions for implementing these security practices or how smaller companies can adopt them. The expected outcome of using the framework is based on the premise that the participating organisation already has set security practices or dedicated security teams in place.

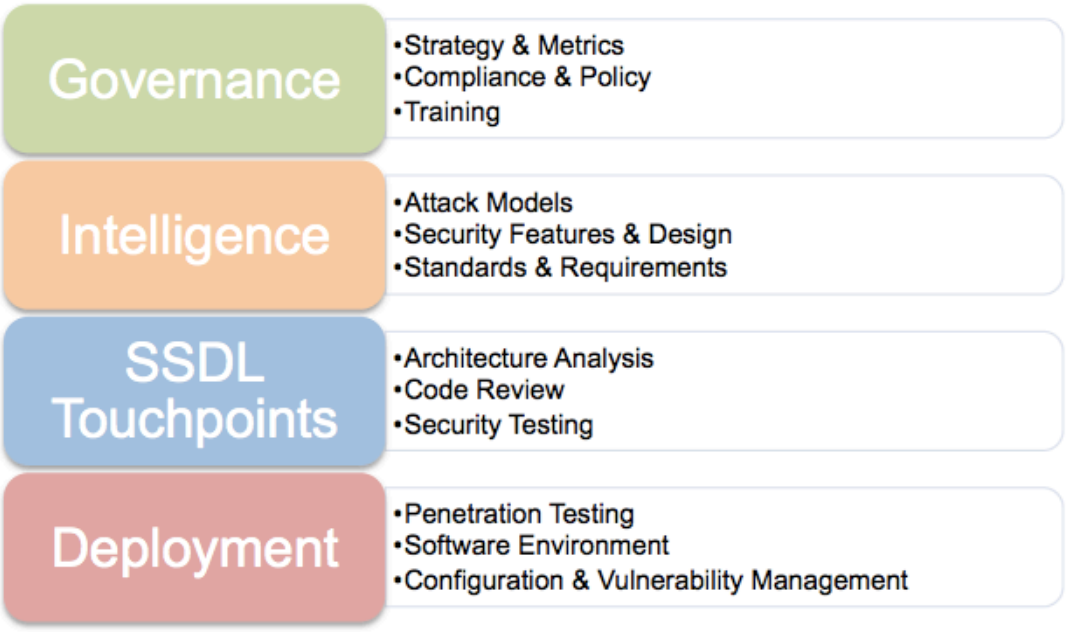



Figure 2.2– BSIMM Software Security Framework (BSIMM by Synopsys 2022)

### OWASP Top-10 and Application Security Verification Standard

The Open Web Application Security Project (OWASP) Top 10 is an initiative established in 2001 by the OWASP Foundation, aimed at raising awareness about the ten most critical security risks in web applications (OWASP 2017). It provides free and open-source tools, standards, books, videos, and checklists on common application security issues for developers. They promote approaching application security as a people, process, and technology process. The Top-10 project materials aim to provide insights into security problems considered 'simple' to give developers a starting point on the journey to building more secure apps. Using a rating scheme, the OWASP Top 10 provides developers with the most significant web application security risks with information about the technical impact and some preventive measures for these risks. The top security risks, including injections, broken user authentication, sensitive data exposure, and improper asset management, are grouped into three main categories: access control, network security, and app security (OWASP 2023).

Other initiatives run by the OWASP Foundation include the Application Security Verification Standard (ASVS) project, which provides a standard for testing web application security controls and outlines a set of requirements for secure development (OWASP 2018). The Software Assurance Maturity Model (SAMM) (Chandra and OWASP 2013) is an initiative that provides a template for organisations with recommended best practices across the entire software lifecycle. The OWASP SAMM categorises fifteen security best practices into five main business functions: governance, design, implementation, verification, and operations. Like the BSIMM framework, the OWASP SAMM claims to be technology, process, and organisation agnostic; however, it does not address how other factors influence the adoption and implementation of these security practices. The various initiatives created by the OWASP Foundation are community-written materials that incorporate contributions from developers, security specialists, and practitioners from multiple organisations.

| Governance | Design | Implementation | Verification | Operations |
|---|---|---|---|---|
| Strategy and Metrics | Threat Assessment | Secure Build | Architecture Assessment | Incident Management |
| Policy and Compliance | Security Requirements | Secure Deployment | Requirements-driven Testing | Environment Management |
| Education and Guidance | Security Architecture | Defect Management | Security Testing | Operational Management |

Figure 2.3– OWASP SAMM Model Overview (OWASP SAMM 2019)

**SAFECode – Fundamental Practices for Secure Software Development**

Another initiative referenced in academic literature and industry is the Software Assurance Forum for Excellence in Code (SAFECode), which published its Fundamental Practices for Secure Software Development in 2008. The SAFECode initiative aims to provide software practitioners with a guide for creating or enhancing their software security programs and improving the adoption of security development methods (Belk et al. 2014). The publication includes guidelines for four core areas: requirements identification, management of third-party components, security issue management, and vulnerability response and disclosure, as part of foundational considerations for a successful secure software development lifecycle (Simpson 2014; SAFECode 2018). Much of the SAFECode guidelines focuses on the software development practices of large companies, with potential benefits for small software companies.

Other initiatives and frameworks such as the National Institute of Standards and Technology - Secure Software Development Framework (Souppaya, Scarfone, and Dodson 2021), the SANS Institute Framework for Secure App Design and Development (McCown 2003), and the National Cyber Security Centre – Secure Development and Deployment Guidance (NCSC 2018) are built on the consolidation of established recommendations and guidelines presented in the BSIMMS, OWASP and SAFECode publications. These government initiatives claim to provide practitioners with a common language for describing secure software development practices for various organisational sizes; however, if they are built mainly on the experiences of large organisations, they might pose a challenge for smaller organisations to adopt.

### 2.3.5 The Concept of Responsible Innovation in Software Development Practices

The concept of Responsible Innovation (RI) is relatively new to the software development industry. However, the idea of responsibility in science and research practices has been around for some time (Stilgoe, Owen, and Macnaghten 2013). The definitions of responsible innovation are still evolving, however, von Schomberg (2012) provides the following definition: "a transparent, interactive process by which societal actors and innovators become mutually responsive to each other with a view to the (ethical) acceptability, sustainability and societal desirability of the innovation process and its marketable products (to allow a proper embedding of scientific and technological advances in our society)" (Von Schomberg 2012). Stilgoe et al. (2013) defines it as a "means of taking care of the future through collective stewardship of science and innovation in the present" and developed a framework made up of four elements – anticipation, reflexivity, inclusion, and responsiveness (Stilgoe, Owen, and Macnaghten 2013).

i. **Anticipation** involves thinking systematically to identify issues related to the innovation or research outcomes.
ii. **Reflexivity** means that individuals and organisations need to reflect on their motivations,
iii. **Inclusion** is about involving all stakeholders throughout the innovation process, and
iv. **Responsiveness** refers to the ability to adjust the product based on feedback from stakeholders.

Another responsible innovation framework adopted by research and industry is the AREA framework, comprising four main components that can be implemented in practice (EPSRC, 2013; Jirotka et al. 2017).

i. **Anticipate** the possible impact and implications of research and innovation,
ii. **Reflect** on the motivations, purpose, processes, and products.
iii. **Engage** with stakeholders and the wider audience in broad deliberations and

iv. **Act** according to the responses received from stakeholders to change the direction of the research or innovation.

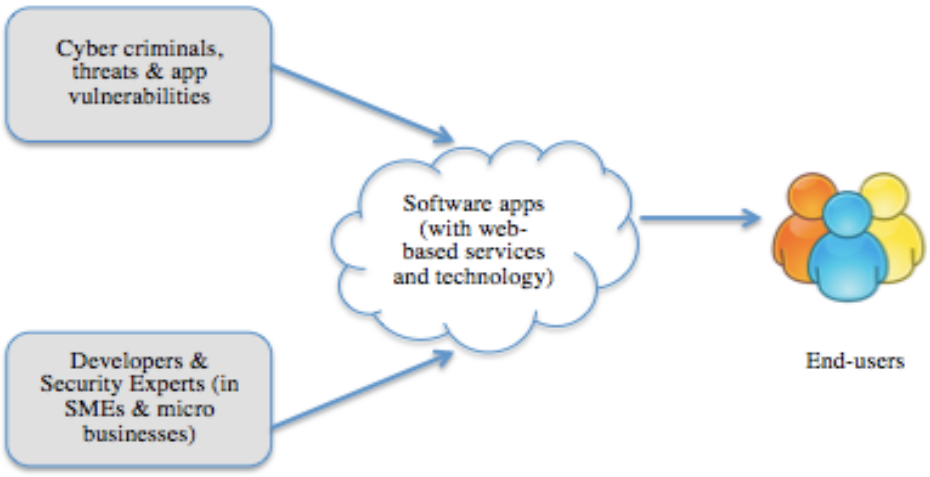
In the software industry, responsible innovation is described as a "set of practices in software development that anticipate and address the potential negative impacts of technology on people" (Microsoft Corp. 2023). It remains unclear what the concept of responsible innovation entails across the software application development landscape and how it can be effectively applied in practice. However, some studies have indicated that stakeholder engagement in the innovation process plays the leading role in how responsible innovation should be approached (Pavie, Carthy, and Scholten 2014; Koops 2015).

As software applications become increasingly pervasive in our society, they are being developed and released to the public in short timeframes, with the ability to influence human activities, strengthen or weaken democracy, or even negatively impact people (Jirotka et al. 2017). It has become prevalent that software development innovators and researchers need to be exposed to more ways of ensuring that products are created in society's interest. Responsible innovation frameworks and toolkits claim to offer a way of enabling software development teams by incorporating social and ethical human-centred practices in the development process for the future of software engineering (Pavie, Carthy, and Scholten 2014; Von Schomberg 2019).

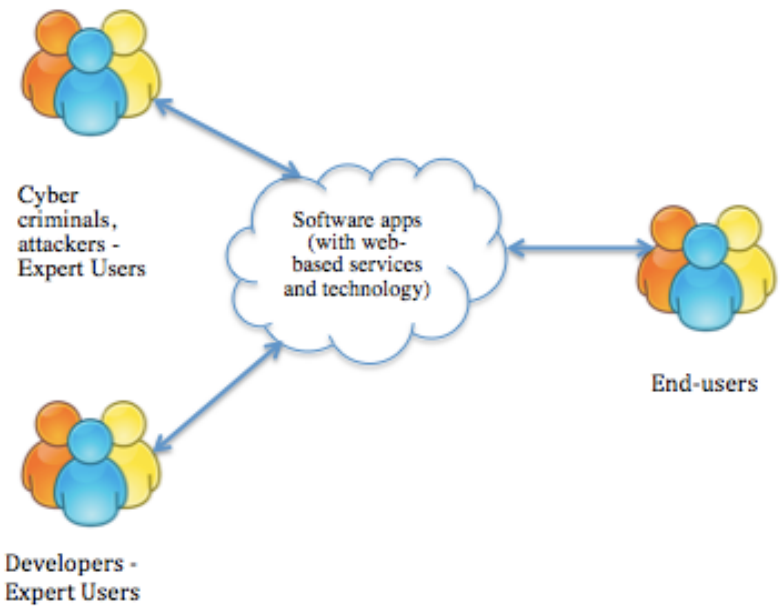
### 2.3.6 Human and Socio-Technical Aspects in Secure Software Development

For software applications to be ultimately helpful to any individual, they need to be built with people in mind. A focus on human and social dimensions in secure software development offers a potential means of attaining the best practices and advancement in the domain (Colwill 2009). This is because the multifaceted role of human factors in cybersecurity shows that vulnerabilities and attacks are not solely a result of technological problems or programming errors (Kraemer, Carayon, and Clem 2009). In the context of this study, security activities encompass practices and measures designed to protect infrastructure, information, and professional users (developers) from attacks, disruptions, or other events

that could intentionally or unintentionally exploit a specific vulnerability in the information system (Fischer 2005). The influence of human factors (such as individual characteristics, attitudes, diversity of perspectives, experiences, and behaviours) on cybersecurity has been studied in various capacities. Issues arising within the software development ecosystem of small and medium-sized enterprises will be discussed in more detail in the later parts of this dissertation.




(a)




(b)

Figure 2.4– Relationship diagrams between end-users, app developers and attackers

Security breaches are a continuous competition between attackers who probe for weaknesses and defenders who constantly try to protect organisations or individuals (Fischer 2005). The effective remediation of vulnerabilities and damages from cyberattacks is a significant issue in cybersecurity, but organisations focus mainly on technological methods to address these issues (Kraemer, Carayon, and Clem 2009). Even when the remedies for vulnerabilities are known, Fischer (2016) argues that in many cases, they are not implemented due to budgetary, human, or operational constraints. This presents a significant gap in cybersecurity for organisations, such as SMEs, who face these constraints (Fischer 2016).

Assets are valuable not only to members of an organisation, but also to the attackers behind organisational threats (Faily 2011). In a recent study on security perceptions by inter-organisational groups, it was identified that a culture of security involves more than just policy communication; it involves factors such as understanding the values and norms of an organisation's culture and sub-cultures, and an appreciation of the different work contexts (Schlienger and Teufel 2002; Faily 2011; Wamala 2013). The study conducted by Kraemer et al. on understanding the pathways to vulnerabilities highlights the need to adopt a socio-technical approach to addressing security issues within organisations (Kraemer, Carayon, and Clem 2009). The high level of stakeholder participation associated with this approach was found to raise awareness of security and increase subsequent ownership of selected security measures (Faily 2011).

In recent years, the importance of social aspects of software development and cybersecurity has been increasingly recognised in research and practice (Stedmon et al. 2016). Studies such as that of Tamburri have analysed organisational social structures to identify areas beneficial to software engineering (Tamburri, Lago, and Vliet 2013). Kraemer and others have argued that human and organisational factors play a significant role in the development of security vulnerabilities, as evidenced by results from qualitative studies (Kraemer, Carayon, and Clem,2009). Studying cybersecurity and software development practices through a socio-technical lens reveals the interplay between technology and human and social dynamics (Fuggetta and Di Nitto 2014). A technological approach may focus solely on tools, system performance, and software security, whereas the social approach primarily focuses on human factors. These elements should not be studied in isolation, but rather through a socio-

technical approach that integrates human aspects and technology to define the dependencies between developer activities and cybersecurity (Safa, Von Solms, and Futcher 2016).

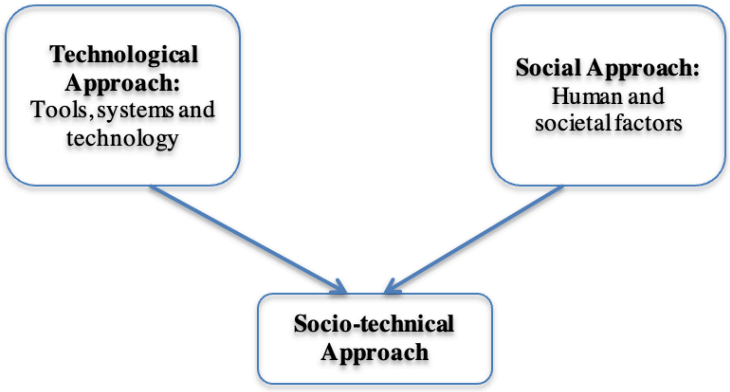



Figure 2.5– An Integration of Social and Technological Approaches

It is essential to understand that the software application ecosystem encompasses the relationships, practices, motives, intentions, and actions of various categories of humans, as illustrated in **Figure 2.4**. Unfortunately, the prevailing approach identified in the literature to tackling human aspects of software security mainly focuses on large, well-structured organisations and end-users. Less attention is given to expert users, in this case, developers and less structured organisations. This is problematic because developers in smaller organisations may not be implementing sufficient security measures, and they underestimate the risk of cyber-attacks, making them easier targets for cybercriminals (Renaud 2016). Furthermore, developers in small organisations may not have access to the expensive security training and are not given enough support to address their security decisions or issues. If humans are the weakest link in the security process, as Schneier (Schneier 2006) claims, then it is crucial to consider this specific group of users - the software developers – as individuals who need support, guidelines and tools in the secure software development and app security ecosystems. This study focuses on the software application developers as individuals who play a significant role in the development process; their organisational contexts, current practices, the challenges they face and the social aspects of implementing these practices.

Existing literature and research investigating the dynamics within software engineering and

the security domain often focus on the design of the products, stakeholders' satisfaction, and end-users' experiences. Although these are fundamental aspects, understanding the different categories of individuals – the developers - who are also stakeholders and users, plays an ultimate role in the success of a software development process. Furthermore, the structures within larger organisations may provide developers with clear task assignments and guidance for navigating secure software development. This may not be the case for developers in smaller organisations with varying structures and dynamics.

## 2.4 Why Small and Medium-Sized Enterprises (SMEs)?

To better explore software application development and security practices within SMEs, it is essential to describe what SMEs are and why this study focuses on the activities of this category of software organisations. SMEs have become a particular focus of governments and academia because they account for 99.9% of all U.K. private sector businesses as of 2020[1]. The European Commission[2] and the United States Small Businesses Administration[3] also claim that SMEs make up 99% of the businesses in Europe and America.

The definition of SMEs across various continents has been based on specific criteria such as the number of employees, annual turnover, and balance sheet statistics (Amrin 2014). These criteria may apply to new or existing businesses, with start-ups being a subset of this group (Osborn and Simpson 2015). The European Commission categorises SMEs as businesses that have fewer than 250 personnel with an annual turnover of less than 50 million euros, with the micro-businesses (such as start-ups) having fewer than 10 employees and less than 2 million euros (Amrin 2014). These size-based definitions (turnovers, number of employees, total assets) vary from country to country, which makes it challenging to have a single definition for SMEs; however, this variation helps differentiate them from larger organisations (Munro

[1] https://www.gov.uk/government/statistics/business-population-estimates-2021

[2] http://ec.europa.eu/growth/smes/business-friendly-environment/sme-definition_en

[3]Many countries including the US, organisations with less than 500 employees are also considered as SMEs https://www.sba.gov/sites/default/files/advocacy/United_States.pdf

2013). In addition, size-based definitions also provide a basis for differentiating business operations and the influence of size on the organisational structure, business processes, strategies, and overall company culture (Osborn and Simpson 2015). Consequently, the size and structure of the organisation may influence various aspects of security operations such as decision-making processes, information sharing and responsibility structures.

Some researchers have also classified SMEs into different types based on their role within the economy. For example, Kyophilavong classifies SMEs into three groups: production, trade, and services (Kyophilavong 2007). Mills (2015) goes further and classifies SMEs into four types: non-employee or sole proprietorship businesses, main street or local businesses, suppliers, and high-growth businesses (Mills 2015). Small-scale companies exist to create jobs, produce products, and provide services within various societies. Many SMEs are in the service sector, which now accounts for two-thirds of economic activity and employment in countries such as the United Kingdom (Observer 2000). These companies are primarily found in wholesale and retail trade, the hotel and restaurant industry, manufacturing, communications and business services, and construction. SMEs also account for an increasingly high percentage of technology-intensive sectors such as information and communications technology (ICT) and biotechnology, according to the U.K. Department for Business, Energy, and Industrial Strategy - Business Population Estimates (2016). They dominate crucial strategic business service subsectors, including services relating to computer software and information processing, research and development, marketing, business organisation and human resource development (Storey 2016).

The computer software SMEs (such as those involved in mobile software, web development, game development, IT consultancy, and support services) focus on providing technical products and services for individuals or other businesses. These tech SMEs may be either sole proprietor micro-businesses, start-ups with less than 10 employees or software and hardware suppliers. They have specific strengths and weaknesses that may require tailored policy responses, as indicated in recent literature reports (Munro 2013). Many of the conventional problems facing SMEs (such as limited resources, lack of financing, difficulties in exploiting expensive technology, constrained managerial capabilities, low productivity,

and regulatory burdens) may contribute to the reasons why tech SMEs are unable to implement cyber security measures designed for larger, well-resourced organisations (Osborn and Simpson 2015; Storey 2016). Even SMEs with comparable sizes and annual turnover to some large organisations can have significantly different decision-making processes or organisational structures that influence the practices they implement (Amrin 2014). The characteristics of SMEs increase the potential for incompatibilities between implementing security best practices and responsibilities in software development.

The challenges particular to smaller organisations may influence software development practices. For example, even though roles may exist, in SMEs, they are not distinct or clear, and employees are assigned multiple roles and responsibilities (Koutsoumpos and Marinelarena 2013). Thus, work is done in an informal setting, relying heavily on collaborations. In addition, pressing deadlines and limited resources make the adoption of standard techniques or practices during software application development difficult for many computer software and service SMEs (Mishra and Mishra 2009). Although most of these studies report some of the organisational issues that software SMEs may face, there are still debates about what impact these have on the developers' experience, practices, and the software artefact. We can't discuss the concept of practices in software application development without recognising that the individuals involved in executing these practices, the knowledge required and the decisions they need to make (Shove, Pantzar, and Watson 2014).

## 2.5 The Software Application Developer

Software application developers are major stakeholders in computer and software systems, and therefore have responsibilities and interests in the success of these systems (Flechais 2005). Other stakeholders include end-users, administrators, owners, and other employees who are involved in an organisation or society. It is common knowledge in various fields of computing, particularly in human-centred computing, that all stakeholders play different yet crucial roles in the design and development of software and hardware systems (Ghaoui 2005). A *software developer* is an individual responsible for designing, coding, testing, and

maintaining software products or services (Birrell and Ould 1988). The US Department of Labour defines a software developer as one who develops applications that allow others to perform a specific task on a computer or other devices, or those who create the systems that run these devices (US Department of Labour 2017). These developers are human actors who coordinate and manage software processes; therefore, understanding what they do, how they communicate and collaborate, as well as their roles and responsibilities within the software development lifecycle, is necessary to comprehend their practices. It is therefore essential to describe developers' activities, including the tools and technologies they use, communication and information-sharing methods, team collaboration techniques, and their individual or collective responsibilities, to explain how their security practices in software development are formed. Furthermore, human actions, behaviours, decision-making strategies, and various forms of team archetypes should be considered (Yilmaz 2007) to better understand the influence on the practices developers form.

Developers are often referred to by various titles, including computer programmers, coders, and software engineers, both in literature and the industry. As the software application development landscape constantly evolves with new specialisations and technological innovations, developers are now categorised based on different specialisations (Shaw and Garlan 1996; Philipson 2004). Developers are classified into various categories of specialisation, including mobile, web, front-end, back-end, full-stack, desktop, application programming interface (API), game, graphic, software test, embedded, or security software developers. This research focuses on app developers who specialise in designing, developing, and deploying software applications that require web technologies and services. In recent times, app developers do not require specific formal training or a certification process to develop apps. With the vast market for innovation and no barriers, they represent a vulnerable group for security research (Balebako et al. 2014).

### 2.5.1 The Developer as a Practitioner

In recent times, developers no longer have to build software applications from scratch. They can utilise a combination of functionalities from APIs (*Application Programming Interfaces*) of different programming libraries to develop their software product or service (Wurster and van Oorschot 2009). During the software development process, other developers may be responsible for different parts of the project, or in the case of SMEs, a single developer may be responsible for every part (Gerogiannis et al. 2013). In fostering the development of tools, techniques or frameworks that support developers' security practices within their organisations, it is vital to understand their context of use as the roles and responsibilities of these developers are continuously evolving. For example, a developer can be assigned to focus on the security aspect of the software development process in addition to design (McCown 2003).

Since developers influence the security of their software products or the organisation's systems, it is also essential to understand their interactions with various factors such as resources or organisational culture. Developers can benefit substantially from the outcomes of research studies that focus on how all the factors influence their practices and the resulting impact on the security of their software products (Wurster and van Oorschot 2009). Although developers are considered experts in computer use, it is unrealistic to assume that they are aware of all the best security practices and are willing or able to follow them consistently. In a qualitative study on developers by Joorabchi et al. (2013), it was reported that as practitioners, developers are faced with challenges of fragmentation across platforms and devices, keeping up with programming languages on different platform types as well as the lack of support and time pressure for monitoring, analysing, and testing applications (Joorabchi, Mesbah, and Kruchten 2013). Despite these challenges, studies often overlook the fact that developers, like any human, are prone to mistakes, influenced by their social and organisational context, and require as much support as end-users, if not more. Several significant security incidents have been reported due to human errors by developers. Green and Smith (2016) highlighted several examples, including the vulnerabilities faced by Apple users due to developers downloading XcodeGhost malware through malicious developer

tools, as well as the Internet-wide attack cycles caused by the Heartbleed and Shellshock vulnerabilities (Green and Smith 2016). The examples show that various factors may contribute to security risks and vulnerabilities in software applications.

Software application development encompasses a combination of technical and human components, with all stakeholders playing crucial roles in collaboration to achieve the best possible user experience by meeting various standards. The combination of human and technical dimensions in the development process shows the need for a socio-technical approach to addressing issues such as security in this domain. This study explains the practices of developers within the development lifecycle by focusing on the interplay between human, social and organisational factors.

### 2.5.2 The Developer as a User

Although there are many interesting and important research studies on end-users, software developers represent a different group of individuals who are still being overlooked in current security research (Green and Smith 2016). Developers encompass a wide range of different attitudes, characteristics, behaviours, and motivations in the software process who make mistakes or need help as much as end-users (Flechais 2005). The developer's actions, like those of any other user, can influence system security, and poor practices can actively compromise security in any organisation (Vidyaraman, Chandrasekaran, and Upadhyaya 2008).

Generally, a user can be characterised as an individual with legitimate access to a personal and/or organisation's information (Albrechtsen 2007). The software developer, as a user in this context, has access to code, shares information about projects with team members, and manages the software development process. The attitude, behaviour, and practices of the developer must be adequately understood to address their contributions to the secure software development domain. Several behavioural models and theories for cyber and information security have been developed, drawing contributions from interdisciplinary domains such as sociology, psychology, and criminology, to study the security practices, approaches, and behaviours of end-users in both domestic and organisational settings. For example, in a study

that reviewed the security behaviour of end-users in an organisation, a conceptual model of factors affecting their behaviours in an organisational environment was developed and used for implementing security initiatives. These factors include body of knowledge (security policies, communication practices, awareness), management influences, peer influences, deterrence efforts, rewards, employee participation, users' knowledge, self-efficacy, attitudes, beliefs, psychological ownership, organisational commitment, trust, procedural justice, security technology ease of use and effectiveness (Abraham 2011).

Another example is a study of professional users' behaviours that influence their acceptance of cybersecurity strategy and policies within their organisations. Results from a survey of 2,000 professional users supported the hypothesis that users' behaviours and characteristics influence the effectiveness of cybersecurity strategies within various organisations (Hasna and Mustapha 2016). Popular theories referenced in software security research include the Theory of Planned Behaviour, used to predict, and understand human behaviour, and the Theory of Reasoned Action, which attempts to explain the relationship between attitudes and behaviours within human actions (Madden, Ellen, and Ajzen 1992). These models and theories have been predominantly used to study the behaviour and attitude of end-users for decades (Lebek et al. 2013). However, they have mainly drawn upon quantitative research methods and theories within the psychology discipline to study human or organisational factors that influence security practices (Casaca and Florentino 2014). Furthermore, these studies have been aimed at end-users; however, the factors identified are also applicable to developers who are users.

Quantitative methods have been primarily used in cybersecurity research, based on the notion that security research is solely technical and can only be described through the measurable properties that quantitative techniques offer (Casaca and Florentino 2014). However, security practices in software development involve complex human interactions and activities, affected by both social and technical factors (de Bruijn and Janssen 2017), hence the need for more studies that investigate and explain how to navigate them. Furthermore, just like end-users, developers are part of social environments that influence how they practice secure software development. Therefore, this study aims to develop a theory that explains how developers navigate the practices for secure software application development while working

through the social dynamics that exist within their business and organisational ecosystem.

## 2.6 Summary

This chapter presents a review of the software development landscape and the various practices that contribute to the success of a software application. The issues surrounding the implementation of security practices, including human and social factors, were also discussed to identify what makes up the melting pot of secure software development practices. An understanding of the software application development process, the roles of different organisational structures and the activities of various stakeholders in the process is needed to gain insights into how we can build more secure software applications. To do this, we also need to understand the interactions between stakeholders and actors in the process, as the interactions may or may not impact practices in secure software development.

The preliminary literature review in this chapter reveals a gap between human interactions, social factors, and organisational structure, and the practice of secure software development. In recent years, developers in micro-businesses, small and medium-sized companies, have created most of the software applications used today (Gerogiannis et al. 2013). These developers face several challenges that are particular to smaller companies, which may influence how effectively they implement security practices during software application development (Ponsard and Deprez 2018). The setback is that smaller organisational structures find it challenging to invest in the cost of implementing new methodologies for improving the software development process, as well as standard security measures, due to limited resources and time constraints (Mishra and Mishra 2009).

There is a need for more diverse and exploratory approaches to studying the practices and experiences of developers, particularly in SMEs, in the software engineering and security research fields. Grounded Theory is one approach that enables us to explore the broader issues surrounding human, social, and organisational factors that shape the security practices of developers in a specific context, specifically SMEs. In the next chapter, the grounded theory process, justification for its use, and the tools used in this study are described.

# Chapter 3 Methodology

## 3.1 Introduction

This chapter describes my methodological approach to addressing the research objectives and how the study was executed. The chapter starts with a review of research approaches in software engineering and security research, definition of the grounded theory methodology and the justification of use in the software engineering and security research disciplines. The following sections describes the specific type of grounded theory methodology implemented in this study and how it was implemented.

## 3.2 Software Engineering and Security Research Approaches

### 3.2.1 Software Engineering Research Approaches

Several studies within software engineering research have traditionally used empirical methods such as experimentation, surveys, or case studies. For example, studies using these methods have focused mainly on issues such as end-user programming ((Burnett 2009; Leblanc 2012)), log monitoring of applications, numerical descriptions of how end-users interact with software products (Burnett 2009; Henninger 2003), and other activities within the software environment with the aim of improving the software development process, and user experience. Little attention is being given to other actors (the developers) within the development lifecycle. Sjøberg et al. (Sjoberg, Dyba, and Jorgensen 2007) suggests that the future of software engineering will depend on empirical research methods that enable the development of scientific knowledge about the usefulness of different technologies and

systems as well as the importance of different actors or stakeholder and their activities in software engineering.

Software engineering research needs to account for the various human activities involved in the process of developing software as the process usually involves coordination between individual developers, teams, or organisations (Easterbrook et al. 2008). With inputs from disciplines such as sociology and psychology, we can use multiple research strategies and analytical methods such as Thematic Analysis, Grounded Theory or Participatory Action Research, to better understand how developers continuously build and maintain complex software systems and the challenges they face in the environments where these systems are built. The use of empirical methods has become quite popular in software engineering research and as such has been incorporated in other sub-fields such as the security.

### 3.2.2 Security Research Approaches

Security research has been discussed as one of the issues under the software engineering research domain with a broad range of research approaches. In recent years the importance of applying empirical research methods to security have been increasingly recognised in research and practice (Stedmon et al. 2016). Particularly the human and social aspects in security and software engineering have been focused on more because human play a major role in the design, development, implementation, acceptance, and use of technology. Consequently, the impact of humans on security activities can be severe if best practices are not put in place to address the issues and challenges faced by individuals or organisations in attaining security. Organisations striving to achieve security who focus majorly on systems with little or no regards for the humans involved, open the door for various forms of attack (Smith 2003; Kraemer, Carayon, and Clem 2009). Hence, the need for a shift from the traditional technical focus to a socio-technical approach that explores the practices of different categories of stakeholders in security is essential.

Studying security and software application development practices through a socio-technical lens shows the interplay between technology, humans, and social dynamics (Fuggetta and Di Nitto 2014). A research study that uses a technological approach tends to focus mainly on

tools, systems performance and software security which is an approach used by most researchers in security-related studies such as cryptography, threat models, vulnerabilities, firewalls, multi-factor authentication, cloud, and network security. Some software security related studies have used psychological and social approaches to explore end-user security behaviours and factors that influence these behaviours, within their personal environment (e.g. (Stanton et al. 2005; Albrechtsen 2007)) or in an organisation (e.g. (Abraham 2011; Balebako et al. 2014; Hasna and Mustapha 2016)). Regardless of the strength of access control policies or technical controls, if humans do not implement and use them appropriately, because of the possible influence of various human or organisational factors, the effect on security can be severe (Kraemer, Carayon, and Clem 2009). Security cannot be fully achieved through technology growth only; it must take into full consideration the users involved – both professional and end users.

Studies that have integrated social and technological approaches have used various qualitative methods to examine factors that influence usable security, security requirements in the software development lifecycle, end-user security practices and data privacy issues. Flechais (Sasse and Flechais 2005) and Joorabchi (Joorabchi, Mesbah, and Kruchten 2013) demonstrated how to approach technical security issues in the areas of usable security using a mixture of case studies, surveys, and Grounded Theory. They used Grounded Theory method to give a better understanding of how people manage, understand, and make security decisions within various situations. Assal and Chiasson (Assal and Chiasson 2018) also applied a mixture of Qualitative Content Analysis and Grounded Theory methods to explore software security practices during each stage of the development lifecycle in both large and small organisations in North America. Using semi-structured interviews with 13 developers, the outcome of their study showed a disparity between real-world security practices and best practices identified in literature (Assal and Chiasson 2018). Grounded theory method offers a set of unique procedures, including theoretical sampling and focused coding aimed towards theory development (Sikolia et al. 2013). This enables researchers study practices extensively in order to develop fresh theories that closely fit the data; are useful, conceptually dense, durable, modifiable, and explanatory (Strauss and Corbin 1994; Charmaz 2006).

## 3.3 Research Approach

To navigate the research problem area, the broader opening question '**How do software developers in SMEs practice secure software development effectively?'** was further divided into three main research questions. It is important to note that the research questions evolved as more data was collected and analysed. This study sought to address the questions:

iv. **RQ1:** What current key socio-technical factors are relevant to the security practices of software application developers in SMEs? – To scope and frame the study.

v. **RQ2:** How can we theorise the relationship between these socio-technical factors and the security practices of app developers in SMEs? – To build and evaluate a theory of the security practices for developers in SMEs.

vi. **RQ3**: What recommendations can insights from the theory and study provide developers in SMEs? - To support the socio-technical security practices of developers in SMEs during the software development process.

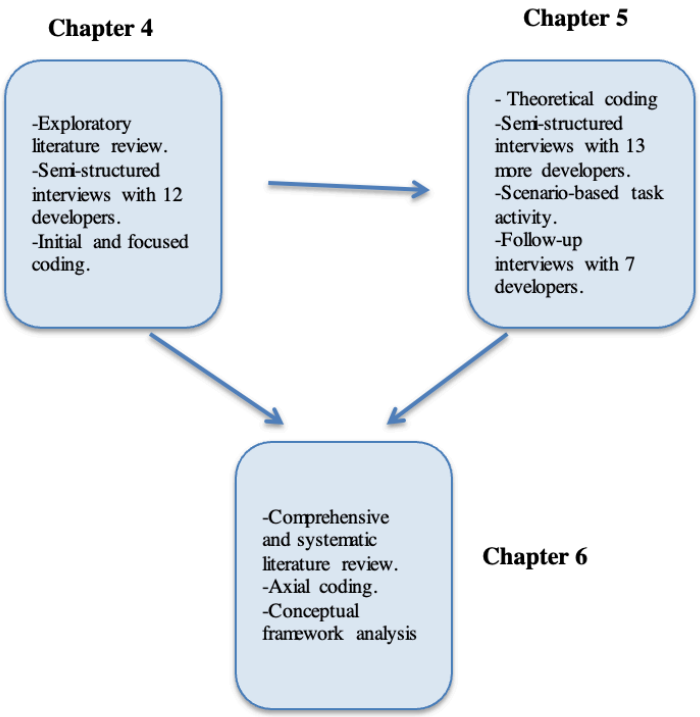



Figure 3.1– Research Approach

Informed by research approaches in literature discussed in **Section 3.2**, the research approach presented in **Figure 3.1** made up of a combination of techniques guided by the grounded theory methodology was organised systematically to address the questions and evaluate the outcomes of this research. In this study, a combination of ground theory methodology and conceptual framework analysis were used. Software engineering, security and application development cut across social and technological boundaries hence the need for a multi-dimensional approach to exploring the social dynamics in this study. To better understand developers' experiences during the development lifecycle and the impact of these experiences on security practices, it is important to investigate not just the tools they use but also the factors surrounding these processes within their social environments. Various software engineering researchers also agree that studies related to software application development should consider the importance of human activities and draw from various empirical research methods in fields such as the social sciences, to study humans at team and organisational levels (Easterbrook et al. 2008).

### 3.3.1 Scoping, Framing and Refining

This study is motivated by the question: **How do software developers in SMEs practice secure software development effectively?** Addressing this overarching research question require the study to be properly framed, with a clear scope and strategy. An ideal research strategy for this study is one that allows us to explain the actions, interactions and processes of some individuals and generates an analytical schema of the phenomenon (McCaslin and Scott 2003). The grounded theory methodology enables us to generate a context-specific theory that explains the actions and interactions of developers within the context of their organisational size and security actions in software development process.

To better frame and scope this study from the broad research area, the initial stage of the grounded theory methodology process was set in motion. It started with preliminary literature review of the secure software development landscape. This was followed by semi-structured interviews, using an interview guide with open-ended questions that allowed for an exploration of the research topic. Participants were asked questions about their roles, company size, structure, security awareness, activities and tools used during software

development. The interview data was then analysed with the initial and focused coding techniques described in **Section 3.4.1.2**, to generate the initial conceptual codes and categories that shape the remainder of the research process. This initial data analysis was done through an inductive approach that present concepts formed from the views and perceptions of the participants.

The outcome of this scoping study is presented in **Chapter 4**. The grounded theory process encourages constant comparison of data to better understand how concepts and categories are interrelated (McCaslin and Scott 2003; Charmaz 2006). At this phase of the study, it was important to compare the initial findings to previous related studies to further refine the research questions and solidify the study objectives and contributions.

### 3.3.2 Conceptual Framework and Theory Generation

The conceptual framework analysis work focused on the relationship between the initial findings of this study and existing literature in secure software development research. The resulting conceptual framework presented in **Chapter 5** provides an interpretation of the realities of this study participants compared to the outcomes of previous related studies. This provides an understanding of the phenomenon, describes existing data, identifies further research gaps, and presents a rationale for theory building.

This phase of the study involved systematically sourcing for data consisting of research studies relating to practices in secure software development, cyber security, and software engineering disciplines. A total of 60 published articles in addition to initial findings of this study – CAMS taxonomy - were analysed. The Context, Attitude, Motivation and Support taxonomy is a classification of participants' perspective of the factors that influence their current security decisions and practices. The proposed framework then describes the relationship between human, social, organisational, and technical factors relevant to the practices of developers and other software professionals during the development process within various organisational context.

A conceptual framework is a network of related concepts, key factors, or constructs (Jabareen 2009). The conceptual analysis technique traditionally focuses on examining a chosen

concept, then "quantifying and tallying its implicit and explicit occurrence in texts but neglects the meaning" (Palmquist and Carley 1997). Researchers argue that this technique is inadequate for theorising concepts because it is difficult to relate specific textual data to other types of data and it does not provide the necessary rigor and depth but is good for providing descriptions of concepts (Carley 1993; Beckwith, Dickinson, and Kendall 2008; Jabareen 2009). Jabareen proposes the conceptual framework analysis as a grounded theory technique that not only describes but generates, identifies, and traces major concepts of a phenomenon (Jabareen 2009). A theory generation methodology then provides the adequate depth, rigor and replicability needed for the interpretation of multidisciplinary data within a specific context (J. Corbin and Strauss 2014). Theories require concepts that are related by "means of statements of relationships"(Jabareen 2009; Glaser and Strauss 2017). In this study, the concepts were not only generated and described but statements that explain the pattern of relationship between these concepts were used to generate a theory of the security practices of developers in SMEs.

Using conceptual framework analysis in a grounded theory process requires the selected texts represent practices and social processes that relate to the phenomenon being study. It should also come from a variety of relevant existing articles, interviews, books, standards, and practices from multidisciplinary literature (Jabareen 2009). The analysis is an iterative process that involves mapping the selected data sources; reviewing and categorising the selected data; identifying and naming concepts; deconstructing, categorising, and integrating concepts; synthesising, resynthesising, and validating concepts. A steady comparison between data from this study (presented in **Section 3.5**), previous related studies, available software securities initiatives and any guidelines on secure software practices targeted at developers. This body of work was used to guide subsequent data collection for theoretical sampling and, to identify and generate a theory of core concepts that explain the security practices of developers in SMEs during the software development process. Further details of the implementation and outcomes of the conceptual framework data, process and procedure are presented in **Chapter 5**.

### 3.3.3 Theory Evaluation

The generated theory from this study generally attempts to explain some aspect of developers' practices during software development within the SME context. One approach to evaluating the theory is to look at the extent to which the theory is applicable and explains one or more examples of the practices or phenomenon in question. [ref] It also involves identifying instances where the theory cannot be applied. Grounded theorists have also suggested various criteria for evaluating grounded theory studies: Strauss and Corbin recommend checking the validity, reliability, and credibility of the data (J. M. Corbin and Strauss 1990; Strauss and Corbin 1997); Glaser suggests checking the fit, workability, relevance, and modifiability of the theory (Glaser 1998; 2001); Charmaz adds that the originality, usefulness and resonance of the theory should be evaluated (Charmaz 2006; Charmaz and Thornberg 2021).

Theory evaluation criteria are described in **Section 3.4.1.5** and outcomes are presented in **Chapter 5**. It describes the evidence that the emergent theory is credible and original because it fully represents an interpretation of what was gathered from the interview data and offers new insights to secure software development research. It also resonates with the intended audience, is modifiable, extends the software engineering and security bodies of knowledge and presents opportunities for further research.

## 3.4 Why Grounded Theory Methodology?

Grounded theory is defined as a *"general methodology for developing theory that is grounded in data, systematically gathered and analysed through the research process"* (Strauss and Corbin 1994). The grounded theory methodology originated in the sixties from the social sciences as a means of focusing on the generation of a theory as opposed to experimental theory testing (Charmaz 2008).

The process typically begins with an inductive approach for collecting and analysing qualitative data, which involves gathering data, breaking it down, conceptualising it and reassembling it into new categories. And the goal of grounded theory is to draw patterns of

behaviours and experiences by generating concepts and categories from the data gathered from those involved in that problem space (Strauss and Corbin 1994). In the process of building the theory, the researcher does not begin the study with any preconceived theory in mind but begins with an area of study and allows the theory to emerge from the data.

In software engineering and security research, the idea of developing theories is still unclear, at least for most of us. This is probably a result of our backgrounds in the physical and natural sciences, and objective views. A "theory" is usually seen as a universal truth (Adolph 2013), for example Einstein's Theory of General Relativity. Scientists have continuously challenged this notion, stating that theories especially within a social context do not have to be universal truths, they can vary in their pervasiveness (Adolph 2013). Engineers are also faced with the challenge of separating the influence of the researcher's personal values and experiences on the outcome of the concepts and categories created during the grounded theory process. This challenge is not particular to grounded theory but with interpretative qualitative research. Hence, why quantitative researchers are constantly contesting the results of qualitative research in the software engineering and security fields. In grounded theory, the researcher's pre-existing values, interpretations and views are taken into consideration and seen as contributions to the richness of the data.

According to Schreiber and Stern (Schreiber and Stern 2001), grounded theory is best suited for research areas that are exploratory, addressing the What and How questions such as "What is going on here? Or how do people make sense of their experiences in certain situations?" In answering these exploratory questions, researchers begin to make sense of the meanings and actions of participants (Charmaz 2006). Grounded Theory is useful in developing a theory (theoretical framework) about a specific research area that has not been previously studied or an already researched area in a specific context where new perspectives are beneficial (Schreiber and Stern 2001; Adolph 2013).

One of the core features of grounded theory is the process of theoretical sampling. Sampling is aimed at the theory construction and not for population representation, so researchers must constantly refine their sampling strategy to be more focused during the iterative process of data collection and analysis. Therefore, grounded theory will be inappropriate for research

studies that are not discovery-oriented, inquiring for answers from a question such as "Are developers in large organisations more productive than developers in SMEs?" However, if we are interested in understanding the experiences of developers, then grounded theory may be more appropriate by asking, "How do developers in large organisations manage their development process compared to those in SMEs?" Answering such question may help inform policy changes or the creation of techniques that support the adoption of more efficient development processes in these large organisations. Grounded Theory enables us to explore the broader issues around how developer's practices are impacted by social dynamics, interactions, and negotiations within the SME context. This contributes to the software engineering and security bodies of knowledge.

## 3.5 Implementing the Grounded Theory Process

The following sections describe the implementation and execution of the ground theory process in this study.

### 3.5.1 Clarifying the Research Problem

Grounded theorists agree that the methodology allows researchers work with a general area of interest and not necessarily with a specific problem (McCallin 2003) making the methodology even more appealing for studies in secure software development, a relatively new domain. Not because a specific problem cannot be clearly defined from the inception of the project, rather the researcher gets the opportunity to define the specific problems highlighted by participants from the initial stages of the study. However, it is recommended that researchers define the research problem area with some opening questions from a general area of interest (Charmaz 2006).

Rather than having research questions set based on extant theories or frameworks, grounded theory method allows the researcher to identify and clarify research questions from the data gathered. Here, research questions do not depend solely on in-depth review of literature, the researcher also has the advantage of structuring questions from preconceived knowledge of

the problem area and/or personal experiences. The study started with a broader problem statement: “How can we support the security activities of developers in small businesses?”

The problem area was refined based on an understanding of the secure software development landscape and the question was changed to: “**How do developers in SMEs practice secure software development effectively?”** To address this question, there is a need to understand the current security activities and experiences of software application developers in SMEs or micro-businesses during the software development process. Furthermore, it is also important to identify the factors and potential challenges that are particular to them.

### 3.5.2 Ethical Consideration

Due to the sensitive nature of participants’ data and information, ethical considerations are taken to ensure that participants are fully aware of the research process and there is transparency about the use of their data. All research studies involving human participants require approval from the University’s ethics board. Participant recruitment and interview data collection was approved by the University of Oxford Central University Research Ethics Committee (CUREC), with reference numbers R54135/RE001 and CS_CIA_20_009. Research data was handled carefully according to the University’s data protection guidelines and policy and continuous guidance and feedback was sort as the research progressed.

### 3.5.3 Data Collection and Tools

The nature of the research problem area generally dictates the data collection methods the researcher gets to choose (Charmaz 2006). For example, some research problems might require the researcher to fully immerse themselves and observe the daily operations and activities of participants, and so ethnography might be the method of choice. Researchers can also use a combination of data collection methods such as interviews, elicited texts, depending on the research problem and group of participants(Gibbs 2007).

The nature of this study required more qualitative methods for data collection, particularly interviews because it gives developers the opportunity to describe their experiences and allows the researcher to collect data in-person and virtually with the advancements in various

communication technology. Although interviews only give partial descriptions of the phenomena and reports of what participants says they do - which may not fully reflect what they do – they are still one of the most dominant techniques for data collection (Opdenakker 2006). Collecting data through interviews does not go without its weaknesses such as the cost, time-consumption, biases or less anonymity for participants but they are one of the most effective and widely accepted techniques for qualitative research (Alshenqeeti 2014).

All interviews, face-to-face and virtual interviews done via teleconferencing tools, were recorded digitally using a voice recorder, then later transcribed using Microsoft Word. Notes taken during the interviews also contributed to memos used during analysis. Transcribing the interview data was tedious but improved the interaction with the data. The ATLAS.ti qualitative research support tool was used to manage the transcribed data during the coding process. The ATLAS.ti tool doesn't generate any codes; it only helps organise the documents, codes and categories created by the researcher with the options to colour codes and label your documents. It also has a feature that allows you visually represent the connection between the categories created.

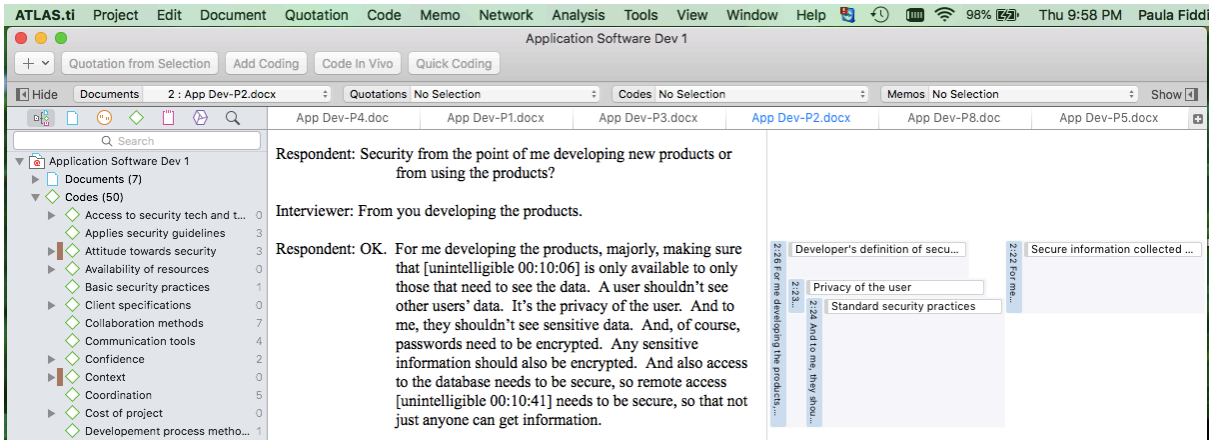



Figure 3.2– Coding Process using ATLAS.ti Tool

Conducting interviews require skills to ensure that the sessions are productive, and conversations flow seamlessly (Adams 2015). Semi-structured interviews were implemented because it was more effective to have some open-ended questions ready to start the conversation. It also encouraged the participant to open slowly without feeling like the interviewer was being too intrusive and it gave the interviewer the opportunity to probe

responses even further. After a few interviews, it got easier to talk to participants since the data analysis and initial coding process had already begun, the conversations got better as more insights were gathered from the data. An interview guide was used because the research required in-depth exploration of secure software development experiences based on the developer's account and interpretation of their activities. The interview guide contains a set of guiding questions presented in **Appendix C.1** that were developed based on preconceived ideas and preliminary review of literature on the general activities of developers in the software development ecosystem. The questions were open-ended to allow the conversations to flow better and discussions were led by the participants' responses. The interview guide in **Appendix C.1** is split into four sections:

1. Introductory questions, which includes some probing questions about the company size and services offered, the participants' development platforms and the type of software products created.
2. Questions about the developers' activities and tools used during the development process, including software development methodologies, daily tools, frameworks, and technologies used for development, communication, collaboration, and project management.
3. Cyber security knowledge, awareness, and concerns with questions about their perception of security in software application development, the frameworks, or guidelines they know and implement, and policies that are being followed within their organisations.
4. Software development processes and security considerations, including questions about what developers consider good or bad practices, and security decisions, negotiations, and responsibilities.

A total of 25 individuals participated in the interview process and **Appendix B** shows an example of the consent form used in the study. The interviews lasted for 45 minutes to 1 hour per interview subject depending on their schedules and the level of details they were willing to give. Follow-up interviews for theory evaluation were done in a similar format, however, worksheets were used during the conversation to engage the participants further (see **Appendix D** for follow-up interview worksheets). Interviews were conducted face-to-face or

virtually depending on distance or participant's choice. No identifying information such as personal or company details were intentionally collected during the interview, only information provided by the participants, and sensitive information was removed from the transcribed materials. Each interview was audio-recorded, and participants were informed about the recording since they had the option to refuse to be recorded. All participants consented to being audio-recorded.

### 3.5.4 Participant Recruitment and Selection

The data collection process started with participant recruitment after ethics approval. The recruitment process included in-person recruiting at local tech innovation events through social networks. This was accompanied by the snowballing technique (Colin Robson 2002), where interviewees were asked to recommend colleagues in similar roles and job occupations, who are willing to be part of the study. Attending and interacting with developers at social events organised by software development communities and local tech conferences was also an effective way of recruiting developers for the study, it however did not guarantee that the developer will eventually commit. The goal at this stage of the study was to recruit developers that built software applications within small, medium-sized organisations. According to the definition of SMEs and micro-businesses described in **Chapter 2,** participants had to be either sole proprietors of their micro-businesses or employees of software development companies with fewer than 250 personnel (Munro 2013). No further selection criterion was used to select participants at this stage of data collection.

Recruiting developers in SMEs was no easy task because was a challenge to get developers to commit to the interview time needed for the study. One primary reason for this is that most of the developers contacted were concerned about spending any time being part of a research study, because they had ongoing projects with strict timelines. Initially, 40 developers were contacted (see **Appendix A** for an example of the contact letter), 15 of the developers responded, and 12 agreed to participate in the interview process. Two of the participants in the pool of 15 respondents dropped out at the last minute and explained that they could not spare time for the study. They were also uncomfortable about discussing their software development processes and security-related questions. The interviewing process started with

the first 12 respondents presented in **Table 3.3**, while more recruitment activities through snowballing and attendance at developers' events were simultaneously ongoing. Data transcription and the initial and focused coding processes were also being performed. **Table 3.4** represents the group of participants that make-up the theoretical sample and saturation phase. Participant recruitment was a bit more strategic at this phase because categories had been identified and theoretical sampling requires data that focuses specifically on the categories of the developing theory (Thornberg and Charmaz 2014). More details about theoretical sampling in **Section 3.6.8**.

### 3.5.5 Participants Demographics

**Tables 3.3** and **3.4** represent are lists of individuals who participated in the interviews. Participants were from different sectors across 10 countries in 4 continents (Africa, Europe, North and South America). The focus was on members in the software engineering department of the various organisations. The participants also work across different business sectors such as telecommunications, financial technology (FinTech), gaming and entertainment. In addition, they considered themselves to be either mobile, web or desktop app developers or belonging to more than one category. Finally, participants described themselves as experts in developing apps on specific platforms such as Windows, Android, or iOS platforms.

| Participant's ID | Company size | Type of Developer | Sector | Developers' Base Country |
|---|---|---|---|---|
| P1 | 5 | Mobile & Web | Telecommunications | England |
| P2 | 10 | Mobile & Web | FinTech | Nigeria |
| P3 | 15-20 | Mobile (Android) | Art & Lifestyle | Ghana |
| P4 | 5 | Desktop, Mobile, Web | Education/Learning platforms | Nigeria |
| P5 | 15 | Mobile | Marketing | Ireland |
| P6 | 80 -100 | Web (Full stack) | Legacy Systems | England |
| P7 | 50 | Desktop, Mobile, Web | Trading & Money Market systems | Nigeria |
| P8 | 30-40 | Mobile (iOS) | General mobile apps | US/Canada |
| P9 | 15 | Mobile (Android) | Banking | Ghana |
| P10 | 6 | Mobile (Android) | Gaming | Nigeria |
| P11 | 6 | Mobile & Web | Entertainment | England |
| P12 | 15 | Web & APIs | Payment Technology | Sweden |

Table 3.1 – Participant Demographics – Dataset 1

| Participant's ID | Company size | Years of Experience | Type of Developer | Domain/Sector | Developers' Base Country |
|---|---|---|---|---|---|
| P13 | 40 | 15 | Mobile & Web | E-Commerce, Gaming | Brazil |
| P14 | 25 | 13 | Front-end | Finance/Banking | U. K. |
| P15 | 20-25 | 10 | Mobile (Android) | Art & lifestyle | US/Canada |
| P16 | 5 | 9 | Back-end, Databases | Learning Platforms | Nigeria |
| P17 | 6 | 6 | APIs | Software Services | Ireland |
| P18 | 20 | 4 | Mobile (iOS, Android) | Legacy Systems | England |
| P19 | 7 | 4 | APIs | Trading Systems | Nigeria |
| P20 | 5 | 7 | Web | General mobile apps | Brazil |
| P21 | 5 | 3 | Web | General mobile apps | Brazil |
| P22 | 40-50 | 10+ | Infrastructure, Web | Education/Learning | Nigeria |
| P23 | 10 | 8 | Web (Full stack) | Marketing | Spain |
| P24 | 25 | 9 | Web | Finance/Banking | U. K. |
| P25 | 12 | 12 | Web | FinTech & EdTech | England |

Table 3.2 – Participant Demographics – Dataset 2

### 3.5.6 Observations

Observing participants was not initially considered as a data collection method in this study because of concerns that may arise with observing security activities. However, it is worth noting incidents that were observed during the process of recruiting study participants at

events and conferences organised for developers. A researchers' account of these observations supplements the interviews discussions and contribute to the construction of the theory. These observations were the researchers' personal account of the events as an attendee and were more passive. No specific protocols were followed since attendees at these events were not asked any questions relating to the study until they formally gave consent to be part of the study. The organisers of the conferences and events were informed about my research and verbally consented to observing conference discussions and no individual data about attendees were collected.

A total of five developers' events and conferences was attended mainly to recruit interview subjects, however, activities, interactions and topics developers discussed at these community gatherings were noted. There were opportunities of informal conversations during networking sessions and breaks with a few attendees over a cup of tea, but no direct questions related to the study were asked. The conversations were mainly about the topics discussed during the event and to exchange contact details, inviting them to participate in the study. The events included attendees from different organisations as well as independent developers. Some of the attendees that were engaged worked for large organisations, and so were out-of-scope for this study.

One of the events was specifically called an application security event targeted at developers. This was a relatively small event with less than 50 attendees, so it was not tedious interacting with more than half of the attendees. Of the 35 individuals engaged in informal conversations, only 2 of them were software application developers while others were security professionals. Observations such as this are invaluable because they bolster statements made by participants during interviews about the lack of security prioritisation and conversations in developers' communities. Only a handful of developers attended an application security event to learn more about new security strategies.

### 3.5.7 Sampling for Theoretical Categories and Conceptual Framework

In grounded theory, sampling for theoretical categories supports the development of an emergent theory. It is used to gather more data to advance the analytical thinking process

after the initial and focused coding stages have been completed (Seale 2004). According to Charmaz, this process is done when the researcher has developed some categories, but they are not robust enough and more specific questions have been raised from the data already gathered (Charmaz 2006). This means that the researcher needs to gather more data that focuses on developing and refining categories for the emerging theory.

The first phase of interviews was open-ended, with participants discussing their various software development experiences and activities they considered to be security related. This made developing the initial codes less problematic because the generated codes represented a broad view of the different areas of development practices in SMEs. It was also easier to filter through the initial codes to identify the most significant recurring codes and concepts that make more analytic sense. The initial conceptual categories were used to develop the CAMS taxonomy presented in **Chapter 4**, which shows the challenges faced by developers in SMEs and factors that may influence software security practices. At this phase of the analysis process, it became more prevalent to identify how these developers navigated the factors represented in the taxonomy by engaging in a more comprehensive and focused data collection and review of literature. This initiated the need for a conceptual framework analysis and a theoretical sampling strategy, which involved the development of a conceptual framework model presented in **Chapter 5** and collecting data that elaborated on the concepts of context, support, and people dynamics in secure software application development.

### 3.5.8 Generating Initial Codes and Concepts

There are mainly two phases of coding in the constructionist grounded theory process – initial and focused coding. Charmaz, however, recommends that a third theoretical coding phase should be done to help "*clarify and sharpen*" the analysis and ensure that categories interpret the data completely (Charmaz 2006). Initial grounded theory coding generates the provisional codes which are then refined and categorised, while focused coding generates core categories that explain the different areas of the data to construct the theory. And finally, theoretical coding further refines and conceptualises the relationship between categories which is then incorporated into the theory. Codes generated in the initial grounded theory analytical process will form the building blocks of the theory. During initial coding, the researcher asks broad

questions (Glaser and Strauss 1967; Charmaz 2006) such as "What is this data a study of?", "What does the data suggest?", "From whose point of view?".

After transcribing the first round of interviews, the initial coding process involved an inductive approach, reading through all the transcripts to get a general impression of the data using a combination of line-by-line and incident-by-incident coding. An inductive approach enables the researcher to openly code the data for new theoretical categories with little or no existing knowledge of the topic from literature while the deductive approach involves developing codes based on the analyses of previous research studies (Gabriel 2013). Grounded theory encourages using a more inductive approach during the initial coding phase enabling the researcher to identify what the data suggests and what theoretical categories the data specifies (Hodkinson 2008). This led to an exploratory view of developers' security activities within their specific development environments.

A combination of line-by-line and incident-by-incident coding was used because not every line of the data described the experiences of participants and incidents that occurred. However, some lines of data contained specific words that describe the key point that section of the data. For example, one of my initial codes was "*Project Uniqueness*":

"*Each project comes with its own uniqueness. So, if you have general standard security guidelines, it might not be specific enough for that project*". [P1]

This statement by participant P1 highlights one of the characteristics of software application projects. At the focused coding stage, codes about project size, cost or uniqueness were refined and conceptualised as "*Project Characteristics*" to describe larger segments of the data.

The initial coding process resulted in a total of 116 codes representing a collection of themes identified in the data. These 116 initial codes describe the general experiences and actions of participating during their development processes. The data analysis and data collection processes ran simultaneously. The number of codes and concepts generated at this phase were overwhelming, and as such needed to be refined and pruned to identify the core issues in the data.

### 3.5.9 Identifying Core Issues using Focused Coding

The focused coding process move initial codes to conceptual categories to identify how participants express and resolve core issues(Belgrave and Seide 2019). It also determines the appropriateness of the initial codes. The move to focused coding is not an entirely direct or rigid process because as more and more data are collected and analysed, you may identify areas that require you to return to data from earlier respondents to explore areas that you may have overlooked or struggled to interpret initially (Charmaz 2006).

New data was collected while coding and continuously compared to existing data as the interviews progressed to refine the initial codes. Memos were also written about the generated codes from the start to support the thought process about the terms used to label the codes and to help understand what is happening in the data. Memos also help to identify and interpret participants' implicit meanings about their experiences (see **Appendix D** for initial codes and **Figure 3.3** for memo snippet). Generated code and concepts were also discussed, and peer reviewed during Human Centred Computing research group data sessions to debate and scrutinise the interpretation of the data.

To narrow down the core issues in this study, the initial codes were assessed again and then regrouped from 116 codes into 30 tentative categories using the tree-view of the ATLAS.ti tool for organisation. After further evaluation, 4 core conceptual categories with 19 sub-categories that show the relationship and connection between them were created. The resulting core categories, as well as supporting sub-categories, are presented as the Context, Attitude, Motivation and Support (CAMS) taxonomy in **Chapter 4** (see **Figure 4.1).**

For context, during the focused coding analytical process, the concepts of "*Collaboration* and *Coordination*" were initial codes that seemed like they could be core categories when participants described how they executed projects. Although, these are key aspects of the development process, they do not properly explain or paint the big picture of incidents in the data that are relevant to the security practices of these developers in SMEs. In the memo written, "*Collaboration* and *Coordination*" approaches by participants was dependent on the type and size characteristics of the project. Project characteristics are also determined by

context in which the software application product will be used. In this case, more effort was put into identifying categories that explain contextual factors surrounding software projects. These codes were then conceptualised as the "*Context*" category influenced by "*Project Characteristics*". This abstraction process was also based on further interviews done and showed that collaboration and coordination were not core concerns for participants. Through this reasoning and consistent analytical process, the category "*Context*" was constructed to represent the information participants gather regarding the circumstances surrounding how their software projects will become functioning products.

### 3.5.10 Memo-writing

Memo-writings are intermediate steps between data collection, analysis and writing up the results (Charmaz 2006). Memos are personal to the primary researcher because they offer an engaging phase with the data without writing a full-blown draft of the analysis. They allow you to go back and forth with the data to fine-tune further data collection strategy and theory development. Memos are messy because they are usually written in the spur-of-the-moment during the interviews or analysis, but they support the writing phase of the study. They are filled with random thoughts, grammatical errors, sketches as part of implementing the grounded theory methodology diligently. **Figure 3.3** shows a snippet of one of my memo notes written during coding.

The memo writing process started during the initial coding phase of the research (see **Appendix D** for more memos) and ran through out theory development. The memos were speculative at the start and showed scribbles of ideas. As the data collection and analysis processes progressed, more concept-based memos formed, written in a separate notepad and on plain A4 papers. Some memos were discarded, particularly those formed based on interactions with external information such as discussions with other researchers, they were not grounded in the data. This process was necessary because they exposed biases, but it ensured that ideas and concepts that made it to the theory were fully based on the data and not speculations.

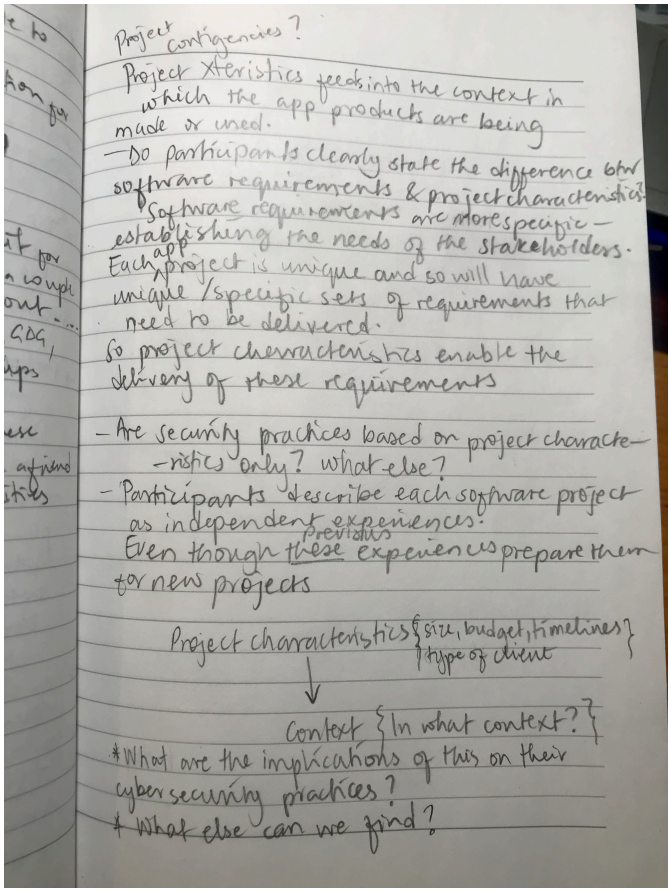



Figure 3.3– Memo Snippet

### 3.5.11 Sorting, Writing and Diagramming Concepts

The sorting, writing, and diagramming processes involve examining all memos to identify connections and relationships between concepts that build the theory. After sorting and analysing the data, the initial categories are presented diagrammatical in the CAMS taxonomy (see **Figure 4.1**). Diagrams serve as useful tools to show the scope and direction of concepts created during analysis.

## 3.6 Addressing Researchers' Biases

In grounded theory, it is expected that the researcher does not force their preconceived ideas or biases about the problem area on the emerging theory in any capacity. Researchers' biases

can be is a threat to the credibility of any study, especially for qualitative studies that depend on the researchers' interpretation of the data. However, it is difficult to separate yourself from the analysis process because the researcher is part of what they study and are not isolated from it. For social constructivist grounded theory the process should be flexible, interactive, and open-minded. Data analysis and interpretation is a social process can be influenced by personal views and peer reviews. It is important readers are aware of my personal biases.

Steps taken to address biases including discussing codes and concept from the analysis with other researchers through data review sessions and paper submissions. Memo-writing was also a major way to clarify what was happening in the data and to curb personal biases. The memos also helped to further scrutinise codes as they developed into categories and concepts that earned their way into the theory. The conceptual framework presented in Chapter 5 also provided a way to compare the initial outcomes of this study with state-of-the-art literature in the secure software development domain, thereby reducing the influence of personal biases.

## 3.7 Trusting the Theory

One of the main concerns in this study was how to get readers at the Department of Computer Science to trust the emergent theory as an interpretation grounded in the data gathered. Hence, why many doctoral students shy away from fully implementing grounded theory methodology to produce a theory. What you might largely see is the use of some of the coding techniques in grounded theory followed by quantitative methods to validate results. Based on interaction with other researchers, studies done in science departments lean heavily on quantitative, positivist methodologies and validation testing. Therefore, quantitative methods are preferred to detach the researcher from respondents and require larger datasets for the generalisation.

To trust the results presented, readers will need an interpretivist perspective to understand the influence of human and social dynamics on technical practices which is at the core of this study. Interpretivist claims that our realities are socially constructed through languages, consciousness, and shared meanings (Shoib, Nandhakumar, and Jones 2006). The claim of positivism is that there is only one reality, and it can be observed empirically and explained

with logical analysis (Kaboub 2008; Pham 2018). The interpretivist paradigm also known as constructivism argues that an objective reality is created because of the subjective mental construction of an individual based on their social experiences (Creswell and Tashakkori 2007). Guba and Lincoln (Guba and Lincoln 2005) further claim that these mind constructions are also connected to 'physical and tangible entities', and constructed realities may be conflicting and diverse (Bailey and Bailey 2017). Physical and tangible entities, such as computing artefacts and their relationship to humans in various social environments cannot be measured in the same way positivist approach physical and natural sciences. This is because humans interpret the different experiences in their world and act according to their interpretations even when the world doesn't (Pham 2018). Instead of generalising the base of understanding for a whole population, interpretivist researchers try to gain a deeper understanding of the complexity of multiple interpretations of a phenomenon in its unique context (Creswell and Tashakkori 2007).

So, positivist criteria are not appropriate for evaluating theories generated from interpretivist studies. The results presented in this study describe the experiences of the developers and the construction of security practices within their reality. Recommended criteria for evaluating the resulting theory of this study (credibility, originality, resonance, and usefulness) have been described in previous earlier in this chapter. To implement these criteria practically, another round of follow-up interviews was conducted with study participants to evaluate the emergent theory as part of their reality and not as a confirmation of a priori hypotheses. They were sent a worksheet with summaries of theoretical categories and scenarios that represented multiple instances of the theory. Then the interviews were guided by questions about resonance and usefulness of the theory. The goal here was to check that the theory resonates with the participants and it useful to their daily secure software development activities as developers. Details of the theory evaluation process and feedback from participants are presented in **Chapter 7**.

## 3.8 Conclusion

The grounded theory methodology to research has been contested various researchers and some consider it to be a non-traditional way of explaining the technical essence of software engineering and security practices (Stol, Ralph, and Fitzgerald 2016). The argument is that the researcher may not have sufficient background knowledge of the relevant problems in the study area and consequentially, not produce novel contributions to the body of knowledge. However, there's a lack of theoretical knowledge around the work and practices of software professionals and theories can help us explain software engineering and its practices from different perspectives (Päivärinta and Smolander 2015). One of the contributions of this study is in the full implementation of the grounded theory methodology resulting in a theory that explains secure software development from the perspective of developers in SMEs. This also serves as a reference point to compare the different approaches to studying the software application development and security practices, people, and artefacts in different organisational environments.

This chapter presents how a social constructivist variant of grounded theory methodology was implemented, providing detailed account of the core grounded theory procedures and techniques used and how the theory emerged. Majority of the data gathered were from semi-structured interviews with support of observational data. The process involved concurrent data collection and constant comparative analysis between codes, concepts, and categories. Feedback on the interpreted data was also solicited from other software engineering, cyber security, and social science researchers.

# Chapter 4 Socio-Technical Factors and Secure Software Development Practices in SMEs

## 4.1 Introduction

This chapter frames the research problem area and scopes the study based on the outcomes of the initial data analysis. The outcome of the initial round of data analysis resulted in a classification of conceptual categories, known as the CAMS taxonomy, which stands for Context, Attitude, Motivation, and Support. The CAMS taxonomy is a classification of socio-technical factors found to be relevant to current security practices of developers in SMEs during the application development process. The taxonomy is an organisational system that shows the relationship between initial conceptual categories and their sub-categories resulting from coding phases in the grounded theory process. Four main conceptual categories were identified: Context, Attitude, Motivation, and Support, with nineteen sub-categories (shown in **Figure 4.1**) that are intertwined to reflect factors influencing how secure software development is practised by developers in SMEs.

## 4.2 Approach

The work presented in this chapter shows the outcome of implementing the initial phase of the grounded methodology described in **Section 3.5**. This study aimed to scope this research and gain an understanding of the socio-technical factors that influence the practices of developers in SMEs. The process involved semi-structured interviews with 12 developers in SMEs across Europe, Africa, and North America. The data was then analysed using initial and focused coding techniques described in **Chapter 3** to identify core concepts about security practices in the development process of developers in SMEs. The findings are the

initial conceptual categories developed from analysis and visually represented in the taxonomy. The socio-technical factors described in the taxonomy address the first research scoping question: "**What current key socio-technical factors are relevant to the security practices of developers in SMEs?"**

This process contributes to the construction of the theory presented in **Chapter 5,** which aims to explain developers' experiences in SMEs and micro-businesses, providing a better understanding of the activities, security knowledge, motivation, and perceptions that shape their security practices.

## 4.3 The C.A.M.S. Taxonomy

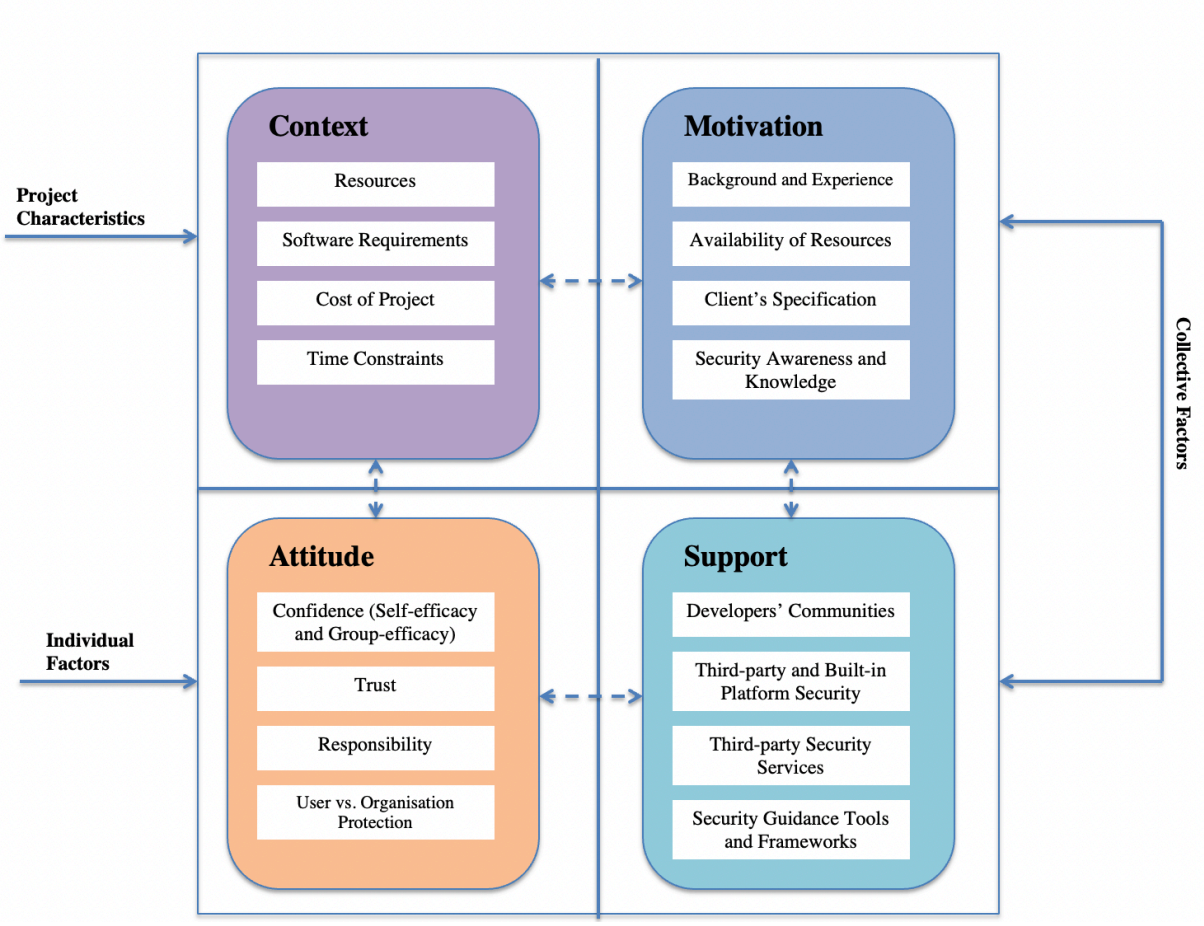



Figure 4.1– C.A.M.S. Taxonomy

The CAMS taxonomy represents core categories and subcategories identified at the end of the initial coding phase of the analysis, leading to the construction of a substantive theory. **Figure 4.1** is a visual representation of the taxonomy of initial concepts. The word CAMS was coined from the four categories considered to be key factors at this phase of the analysis. *Context*, *Attitude*, *Motivation* and *Support* represent participants' perspectives of the factors that influence their current security decisions and practices. Each main category and the supporting subcategories are described in the following sections.

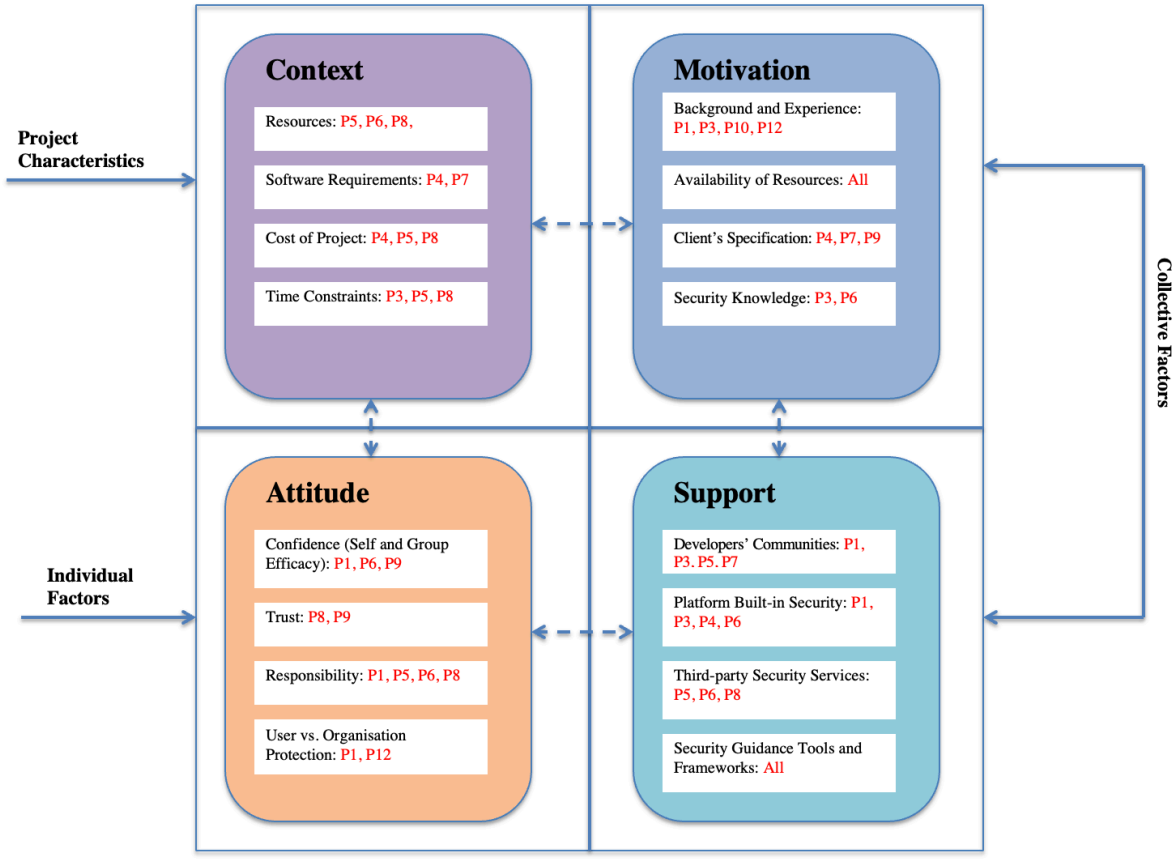



Figure 4.2– Participants Distribution in C.A.M.S. Taxonomy

**Figure 4.2** illustrates the distribution of participants in this study, which strongly aligns with the categories created, highlighting the key factors presented by the participants. Although the taxonomy may not encompass all existing factors that influence developers' security

practices, the CAMS taxonomy enables further exploration of secure software development activities within the SME organisational context.

### 4.3.1 Project Characteristics

The codes and concepts relating to software project size, budget, and timelines were categorised as project characteristics. This is because these elements shape the approach, tools, and technologies that developers choose to use in completing a software project. Developers who participated in this study primarily work in small and medium-sized software companies, and their software projects consist mainly of the services they offer, including mobile and web-based applications. Every software project comes with different specifications, requirements, and characteristics from various clients. Therefore, the developers must define the project's scope, determine the necessary methodology to meet the client's request or the end-user's problem, select the most suitable tools for the project, and identify the required security measures for the product's success.

> "*Each project comes with its own uniqueness. So, if you have general standard security guidelines, it might not be specific enough for that project*". [P1]

Due to the uniqueness of each project, developers must undergo various decision-making processes, which participants describe as complex and challenging. The software project size and type were key characteristics used in estimating teams, tools, technologies, and security measures required. According to participants, different project types and sizes had two to eight developers working on them concurrently, depending on their level of expertise in the relevant problem area. Where more developers were needed, the work was outsourced to freelance or external developers. In mobile and web app developers' PI and P2 companies, the distribution of software project tasks was particularly dependent on the size of the project:

> "*My company is a really small company. I'd say…Well, two directors, but mainly me running the day-to-day operations. But when projects do come in, I have people that work on various parts of the project. On average, I'd say between three and five, depending on the size of the project. When I start out a project, the first thing I go through is to make sure the requirements are properly understood across both myself*

> *… I understand the requirements, and what needs to be done, as well as the rest of the team. Depending on the specific type of project, I then decide how best to split the work within the team.*" [P1].
>
> "*The company has six developers…. sorry, seven developers. And depending on the projects, oftentimes, it's two per project*" [P2]

Participants often made references to the size of their companies during the interviews. For example, saying, "*My company is tiny*" to emphasise the challenges they face as developers in SMEs, juggling every aspect of the development process, including practices, tools, and techniques. In addition to coding, participant P1 was also responsible for project estimations, including team sizing and resource allocation. Developers in these SMEs often have to play multiple roles due to limited human resources, ensuring that projects are completed successfully. Whether it was a mobile app, web app, system software or game development project, estimation and allocations were crucial tasks for developers in this study. This highlights the need for software project management techniques and personnel to help manage the entire project, which also includes ensuring that developers implement best practices throughout the development process.

Another key aspect of project characteristics as a factor in determining the proper infrastructure for development was that Participant P5, a mobile app developer, described the complexities involved in deciding whether to use a public cloud service or their servers. This ensured that they benefited from the server security services provided by the cloud service provider during the app development process. Risk, budget, timelines, and limited resources are also highlighted as factors that primarily affect their decisions to move their live mobile app development projects to the cloud.

> "*So, instead of me deploying my server, I would actually put it on like a cloud service because I know I don't have the budget to think about A, B, C, D, E, F or whatever. So instead of managing my server security, I would use something like that. Yes, aside from security reasons, I use cloud services for everything else because you are literally putting your whole business in their hands, in a way. For us, at that size, it was an easy decision. One thing we know is that we don't have the budget to do any*

> *of those things (any of the security ourselves), or even the knowledge to do it effectively, at the very least. We knew that these guys do have the budget and the motivation, so we are trusting that they will. Those are some of the ideas we used.*
>
> *Depending on time, we would run on Microsoft Azure as an instance for our stuff. Instead of us having to deal with all those things on our own, we tried to push it over to whoever is running the cloud service*." [P5]

For other participants, estimating and having the proper budget to cover project contingencies were also relevant factors that contributed to the challenges they faced during the development process. Project contingencies are risk estimates and allowances set aside for changes or unexpected events (McManus 2012; DeMarco and Lister 2013). According to participants, this can be due to unforeseen costs arising from additional requirements, changes in clients' specifications, and the need for new or additional personnel when implementing new security guidelines, such as those of the General Data Protection Regulation (GDPR). These project contingencies are seen to contribute to the overall constraints on the software project characteristics and have the potential to influence security decisions.

> "*It all comes back to the cost of implementing changes. So, say for example, in some of our projects, where we charge per hour or something, "Oh, definitely, why not? It's more money for the business." But if it is my own cost going in, I might allow changes once or twice. But I would frown at it further down the line*." [P1]

For developers in this study, whether it involves clearly defining project characteristics, making estimations, or having a contingency plan, software application projects in larger organisations may be better prepared compared to those in SMEs and micro-businesses. Participants also expressed concerns about the inability to have contingency plans, combined with the cost of implementing security measures. The analysis shows that project characteristics are factors that directly relate to the context and motivation surrounding the development process.

Further discussions about *Context* and *Motivation* are presented in later sections. Although project characteristics, constraints and contingencies may not seem like new insights in security or software engineering research, we must present the exact findings gathered from the data. This is to ensure that participants' experiences in software application development are adequately described and their relevance to supporting security practices is noted.

### 4.3.2 Individual Factors

The data showed a relationship between participants' practices and their individual experiences. We cannot ignore the role that a person's values, knowledge, and experiences play in their professional practices. Although the analysis did not involve a deep dive into the psychological implications of personal characteristics on security practices, recognising statements made by participant P3 concerning his approach to addressing security issues was a key point in this study.

> "*The truth is I rely on my own imagination and on other people's experiences. I talk to people, yes. I ask questions. "OK, when you were hacked, how were you attacked? How did you fix it*?" [P3].

Participant P3 is a mobile app developer who sometimes relies on his imagination as a way of considering security measures. P3 also depends on the experiences of other developers who are probably part of his team or close community. According to participants, developers generally work on several software projects concurrently, and so considering security activities for multiple live projects running on different platforms can be taxing. Therefore, it is not out of place for developers to rely on their intuition, personal experiences or quick responses from peers to make sense of potential security threats and countermeasures.

Further analysis of the interview data reveals that security sensitivity, stemming from past negative experiences, was another crucial factor contributing to the adoption of security practices in the development process. For example, P3, who was previously a hacker, describes the effect of their experience on their current views of security practices in their role as a mobile app developer:

> "*It means a lot of things to me personally, because before I started developing, I used to be a hacker. So, what I did was just to break into vulnerable servers on the web and go in through the back of it. When I was in school as well, I found several vulnerable points on some of our web services. So, when I did that, I reported to the IT team to fix them. I got rewards for a couple of them, so it was kind of interesting. From experience, I try every way I used to hack. So, I try it on my apps to see if they are vulnerable. Even while I'm still coding, I consider those things and I block them. "OK, can the user go in through this? Can a hacker go in through photo upload? OK, why would he be able to? Oh, yes, I left out this extension. Let me add that*." [P3]

Individuals, such as participant P3, reflect on their personal experiences, alongside other factors, to shape their approach to addressing security measures, which becomes part of how they practice secure software development. Personal experiences with either having your app attacked or being the attacker contribute to how they view security in software development. Most participants describe the impact of individual factors and experiences on security practices as both positive and negative. It might enable developers to be more security sensitive; however, it limits the developers' capacity to see beyond their own experiences. Consequently, this may influence their attitudes towards implementing software security measures. Some participants also described how shared experiences of other developers influence their security practices.

### 4.3.3 Collective Factors

Collective factors represent the collective expert knowledge and personal experiences of cyber threats or attacks that developers have shared within their teams and communities. Similar to the *Individual Factors* discussed in the previous section, participants rely on the collective knowledge, expertise, values, and experiences of their peers as they navigate issues encountered during the development process. These shared experiences and knowledge were categorised as *Collective Factors* and represent one of the key factors relevant to the security practices of developers in this study. The nature of the interaction between the social aspects

of sharing collective knowledge and its implications for the security of the software products that developers create makes this a key socio-technical factor.

During the analysis, participants were found to rely on collective knowledge, experiences, and resources within their developer communities. Almost all the study participants belonged to at least one developers' group, both virtually and physically. These communities are described as being social and/or professional, providing developers with a space to share their development experiences and discuss issues related to tools, techniques, and practices. These discussions shape their approach to implementing security measures and determine which practices they consider relevant to their development process.

> "*This is a very important layer regarding security, so this is a network layer that, yes, [answers] all the communication. So, I would feel nervous if this library is only used by like 3% of the development population. So, I will use this popularity as a measure. If this is a popular project, then ... I mean, if I ever encounter the issue, then 50% of the world will also face the issue, so yes, so I will take that bet*." [P8]

Participant P8 described using libraries and technologies suggested by other developers. They described a feeling of 'nervousness' when using a library that people in their community do not frequently use. Shared collective expert knowledge and experiences with libraries help to ease the tension a developer might feel when introducing a new tool, technology, or technique into their development process.

When asked about the popularity measure, P8 further describes the confidence in collective knowledge:

> "*Popularity, and ... Not only popularity but also how it's regarded in the developers' community. There are a lot of smarter people in our community, and if they're saying this is safe, then okay, I will believe it…not random other developers, but people that are considered top in our community*." [P8]

Within these communities, some developers are considered to be highly skilled and more experienced than others. In vivo, participant P9 referred to them as "celebrity developers". The security or technical guidance and recommendations from celebrity developers can be

said to have a social impact on the practices of the developers within their community. P9 explains further:

> "*In my experience with using Google Firebase [a mobile platform for developing apps], I've just not heard of attacks on them. Based on reviews from other developers and things that I read online." We have celebrity developers around…I think if they get more involved in advocating for more secure apps, we will see a gradual change in the outlook of app developers towards security. Developers who give talks, publish regular articles and are popular within the developers' circles.* [P9]

It was interesting to note that developers in this study did not consider themselves to be high-skilled 'celebrity developers' in their communities. They relied on the support of other developers within their communities to support their security practices. The implementation of specific security measures and technology is influenced by recommendations based on the collective knowledge of their communities.

### 4.3.4 Context

Humans generally require some context to make decisions about any situation, and this is not different for developers. Developers in this study explain that when approached by a client or have an idea for a software product, they consider several factors, including who the software is for, where or when it will be used, and the tools and technologies required to create it. Deciding the 'who', 'what', 'where', 'when', and 'how' of the software product may require extensive information gathering to ensure the task's completion. For participants, answering these questions is particularly important in a small business context because there is a possibility of not having a contingency plan.

Context here refers to the information gathered by developers regarding the circumstances surrounding how their apps will become valuable and functional products. It can be in two folds: the context with which their clients will interact with or use the software products developed, and the context surrounding the development process itself. At this stage of the interview process, participants focused on the latter. Contextual factors, therefore, are core influencers in the adoption of software development and security practices.

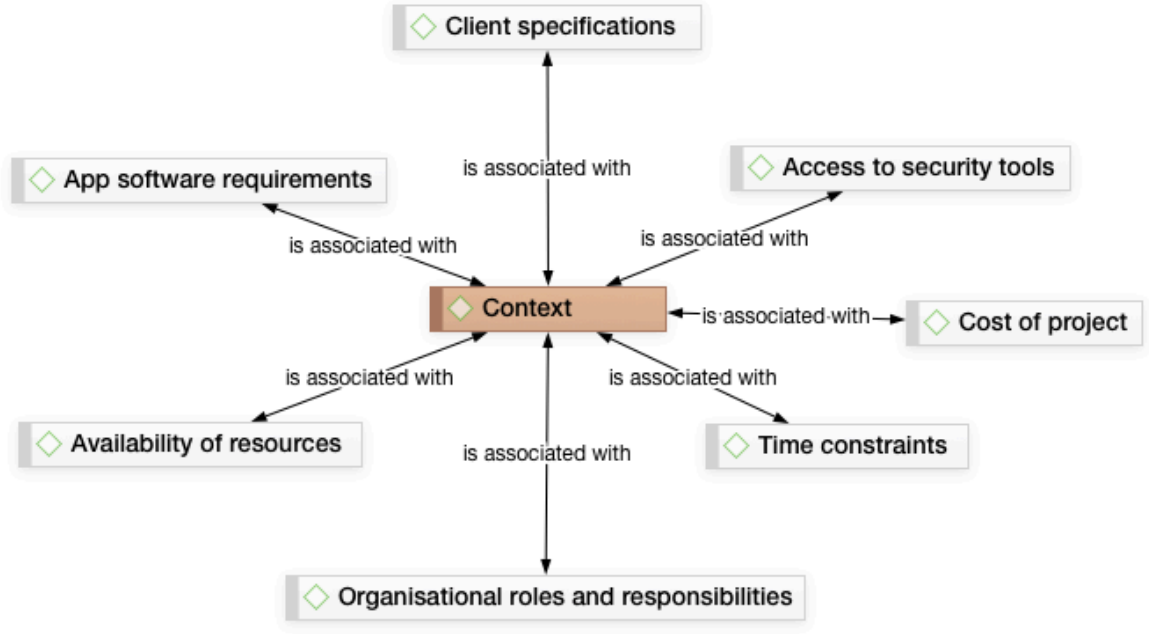



Figure 4.3– Context Surrounding Developers' Security Decisions

**Figure 4.3** is a network of factors that contribute to the context surrounding the development of a software product. This visual representation illustrates the relationship between the initial codes analysed as factors and explains the thought process involved in developing this category. For every software product that the participants reported having worked on, the project terms were either formally or informally set. The formal cases involved a project belonging to a paying client, while the informal cases arose from a personal interest in an app idea. Each software project is then characterised by the context surrounding its development process, and this, in turn, contributes to the security measures considered for implementation. The following paragraphs explain how participants recounted their experiences and provided examples of these security considerations, based on their specific contexts. Additionally, as discussed earlier in **Section 4.2.1**, *Project Characteristics* addresses these contextual factors, providing a starting point for explaining the different areas that participants consider necessary when the terms of the software project and product goal have been established. Further descriptions of each relevant factor are presented based on discussions during the interviews.

### i. Resources

Resources can be the amount of money, tools, technology, or the number of people needed to complete a variety of tasks during the development of a software product. The availability or limitation in the flow of these resources represents contextual factors that affect the effective and efficient functioning of the development teams. Participant P8, an iOS mobile app developer, expressed his views about the impact of limited resources in their small software company. Participant P8 gives insight into how limited resources deter them from investing in security during the app development process.

> *"So, as a small company with smaller resources, and when you have a lot of incoming jobs, you have to just keep constantly developing, and you don't have time for security training, etc. As I said in a previous example, the security [incident] happens ... could happen at one point, like, three years later, and so, it's not something that happens in a normal situation. So, yes, it is not a very good thing, but I think in a lot of cases, we put a higher priority on delivering the job on time to the client and don't really dig deep into security. In a lot of situations, that is okay, probably 90%, 95%, that is okay, and my company, a small and medium company, we probably take a bet on that 95% because we just need to keep moving, keep going, and keep the cash flowing. For the big companies, they have more resources, so they can probably spare some resources for [security] training while projects are going on*." [P8].

During the interviews, probing participants further about software security considerations and training, P8 explains that the reason their development team did not consider security training was that they couldn't afford it. They further made a bold statement that there is a 95% chance of no security incident due to the implementation of security knowledge gained from their past experiences. Developers, such as P8, rely on their personal experiences as a substitute for using other security measures that may require additional funds. This illustrates the resources needed to integrate security measures into the development process.

Other participants, such as P5 and P6, also made references to resources, mainly financial resources. When asked about the considerations for security implementation during the

development process, P5 provided an example of using third-party services for data storage, along with the number of resources required when choosing a service.

> "*Another one you think about is your data centre and how you're storing the data. There are so many places where there are different good and secured recommendations and, usually, it depends on the number of resources you have, and I guess your motivation*." [P5]

In addition to resources, P5 mentions motivation as a contributing factor that influences security decisions. Data from P6 also adds security knowledge and experiences of developers to the list of factors that contribute to long-term security decisions and practices. We explore the issues around motivation and security knowledge later in the sections.

> "*I think it's mostly knowledge and experience that makes developers better with handling security issues and may be available resources. Not all companies, but some small ones that have some money like to get penetration testing services to go through the application, that is what I think*." [P6]

For these developers, the availability of resources was a contextual factor that informed the decisions they made around software security measures when they received a project. In summary, participants report that the planning and execution of a software project primarily depend on the resources available for the project, as well as the security experience of the developers working on that specific type of software product. More details are needed to specify the software's purpose and justify the required resources. The software requirements specification process enables this stage of the development process.

### ii. Software Requirements

During the interview with participant P4, they explained the thought process of their team during the process of eliciting software requirements for their apps, and implementing any security measure was primarily context-based or not a priority.

> "*It depends on the choices we make when we are deciding on our software requirements. For example, when we were choosing which server to go with, security*

> *was number one because we had to make sure the server was secure. Now, when we were choosing the frameworks to go with, security was not number one, because there are way more important things at that point than security. So, it depends on what it is, in the context of what we are looking at, and then we try to make the best decision for us at that point*." [P4]

The participants' response was probed further:

> "*So, what I'm saying is, security is not 100% of what's happening in our heads; we are not security specialists or experts, always looking for security everywhere, but it's just a part of what we do*." [P4]

Specifying security requirements during the planning phase of the development process may not have been instinctive to the participants, as they are not security experts, or it may have been beyond the scope of the project deliverables. This is not because they do not care about the security of their product, but, like developer P4 explained, it doesn't take priority until some context is provided to support the security decision. Participant P7 also shares some thoughts about security considerations from the start of the development process:

> "*From the initial phase of the development process, gathering the tools you need, identifying what the application will do and designing, we have to be mindful of particular security checks. These checks can actually tighten the security as it concerns cyber-attacks. Initial security checks can be done at the beginning of the development process. I think it's all about building solution applications and knowing the particular places...or particular tools you should put in place to avoid vulnerabilities and restrict all these things (like access)*." [P7]

P7 is an app developer who works in a team of about 3 to 5 developers, depending on the size of the project. He also works for a medium-sized company with approximately 50 employees in the trading and financial markets sector. His company has provided security training for its employees due to the peculiarities of the industry. The initial analysis of P7's report suggested that they may be presenting their perception of security based on experience in a

more formally structured organisation, where security discussions are conducted from the start of the project or when developing any financial software application.

As described in **Chapter 2**, Recommendations, guidelines, processes, and frameworks for incorporating security tools, technology, and practices as part of the software requirements phase of the software development process have primarily been represented by large organisations and software companies. However, how applicable are they to developers in SMEs?

### iii. Cost of Project

Participants explain that software projects are typically assigned a relatively low budget due to limited resources. Budgets typically depend on the client's willingness to pay for the project or the amount of funds they are willing to invest in it. According to the data, low budgets coupled with potentially high costs of delivering a software product with high-quality security technology were one of the main reasons why investments in security may have been given less priority when compared to delivering other features and functionalities of the software product. Many participants expressed that, among the costs associated with each project, the cost of implementing security was a key challenge. One participant P5 said:

> "*Basically, your thing [app] is only as secure as how much money you can put behind it*." [P5]

For participant P5, the quality of security in a software product reflects the funds allocated for its implementation. Whether an app is secure or not is mainly dependent on the amount of money the company or client is willing to invest in it. This alludes to the fact that security comes at a high cost for developers in SMEs. For participant P8, the additional cost of security to a project's budget may dissuade clients from offering them the job.

> "*However, a lot of our projects are fixed [bid]. It's because of the business requirement. A lot of client companies want to just pay a fixed fee upfront, and you don't want to pay more, even if the scope has changed a little bit over the course, even if we have found new things on the course, and it has increased the cost and scope, but yes, the company doesn't really want to*." [P8].

Participants who have clients from other larger companies agree that the project cost is paid based on a fixed agreed price when bidding for that project. Therefore, if any security considerations or measures need to be introduced during the development process, it is challenging to make changes or add the security cost due to the agreed-upon fixed project price. This proves to be a difficult decision for most developers in this study, as they do not want to lose their clients due to competing bids from other companies. Developers such as participant P4 also agree that money and cost drive the outcomes of the software.

> *"Money is a major thing in every decision. The software we build and how we sell…I am – I'm very cost-driven. Everything is about cost*." [P4].

As the central decision-maker in a small company of six developers, Participant P4 describes their software business as cost-driven, as this reflects the company's goals and overall vision. The availability of funds and proper allocation of these funds during the software development process are key factors that drive software development decisions. Furthermore, P4 admits that they had never really considered the cost of implementing security during the software development process.

In other organisational contexts, such as that of Participant P1, the costs of implementing specific security standards, such as the PCI DSS, contributed to security decisions made.

> "*We launched a product called MW. It allows us to process payments on our own … I think, at the time, it was kind of cost-effective in that it was 0.2% lower than what we were being charged by PS [an integrated payment solution company] and the interest rates. We made the call not to use MW because it was taking the credit card details of the customer and passing that onto our API. The fact that you are taking the payment details on your own phones, we really need to be PCI DSS compliant. So, we had to go through that PCI DSS process. And although we would probably have been PCI DSS Level 2 or Level 3, for us, as an organisation, the funding wasn't really there at the time to go through that. We just thought, "Well, why go through all that process for 0.2%?" So, it just didn't make any sense for us*." [P1]

The Payment Card Industry Data Security Standard (PCI DSS) is a standardisation scheme created by the PCI Security Standards Council for organisations that handle debit and credit cards ("Payment Card Industry Data Security Standard" 2009). It is also considered to be a cybersecurity compliance certification and requirements checklist for integrated payment technologies. For participant P1, paying to be PCI DSS compliant and certified was necessary to some extent due to the type of software they built, but it incurred additional costs for the project. P1 and their team decided to pay a third-party payment platform provider (PS) to host their app. The third-party company, PS - already possesses the PCI DSS certificate. Whether or not they were comfortable with the level of security provided by the PS company, it seemed like a better option compared to the expensive and tedious process of becoming PCI DSS certified themselves. To P1, the PS company is now responsible for the security of their software.

For developers in SMEs, such as P1, they can cover the cost of assigning security responsibilities to third-party security service providers. The liability of assigned security responsibilities can put developers and SMEs in a tight decision-making position during the development process due to potential costs associated with security breaches. During the interviews, our participants were clear about the importance of having available resources that cover all costs of the app project, including security measures.

**iv. Time Constraints**

All participants mentioned that there was not enough time for security considerations during the software development process, as it was more important to deliver a functioning app to their client within the set timeframe than to implement any security measures that might extend the time. Participants claim that it is difficult to anticipate the time required for security checks, and they may not be able to incorporate it into the agreed-upon timeline. For a developer focused solely on features and functionality, there is a willingness to disregard recommended security practices or easily make them an afterthought due to the timelines set by the client. When asked about their security decision process, P3 was more concerned about delivering the software product on time:

> "*Sometimes, I'm like, "I could have done this in a better way or more securely, but who cares? This client wants it on time. When I'm upgrading, I would focus more on security*." [P3]

Participant P3's statement initially draws your attention to the phrase and question, "*but who cares*?" It can be assumed that the developer thinks that if no one cared about security, including the client, then there may not be a need to prioritise it. However, upon further analysis of the entire statement, P3 initially questioned their actions and considered secure software development to be a better approach. Delivering the app on time was more important. Another participant, P5, also expressed concern about time constraints and their impact on the thought process of developers regarding security in app development.

> "*I think that it is probably the biggest constraint, like how much time can you really afford to give towards thinking about the security of something*?" [P5]

Fast project delivery times and stiff launch dates are significant factors that contribute to what participants describe as efficiency in their line of business. According to participants, project timeline expectations and client approvals are usually shorter than what is needed to deliver the software product but are less strict for personal interest projects. The context of the time frame for delivery depends on whether the product is client-driven or individual, and this may influence how these developers consider implementing security measures in the software development process. The developers could include an estimated timeframe required for additional security processes in the overall project timeline as a default; however, other factors, such as the project's overall cost, available resources, and clients' specifications, must still be considered before a security decision is made.

> "*Speed over quality, including the security and stuff. Yes. So, yes, it's the same situation, so you have to deliver the project within this certain time limit, in order to keep the company going, and in a lot of cases, you plan for the quality, for sure, but not a very detailed level of security is planned, I'd say*." [P8]

Participant P8 agrees that implementing proper security measures for their software applications is considered an added quality to the product. However, if security measures are

considered to add quality to an app, it is crucial to understand why developers decide to deprioritise the quality of their software product. The opening phrase in P8's statement, "speed over quality", explains a reason for the trade-off that some developers have to make during the decision-making process of an ongoing software project. Even with good intentions to deliver a secure app product, developers in SMEs, such as P8, must decide whether to implement security measures or no security at all to keep developing their apps at a fast pace and ensure a steady flow of jobs from clients.

### 4.3.5 Attitude

It is complex to generalise the attitude that developers, like every other individual, express towards anything. For participants in this study, it was essential to understand their thoughts and feelings regarding security in software development, considering their SME context and how this becomes a practice. Participants expressed their attitudes towards software security practices in various forms.

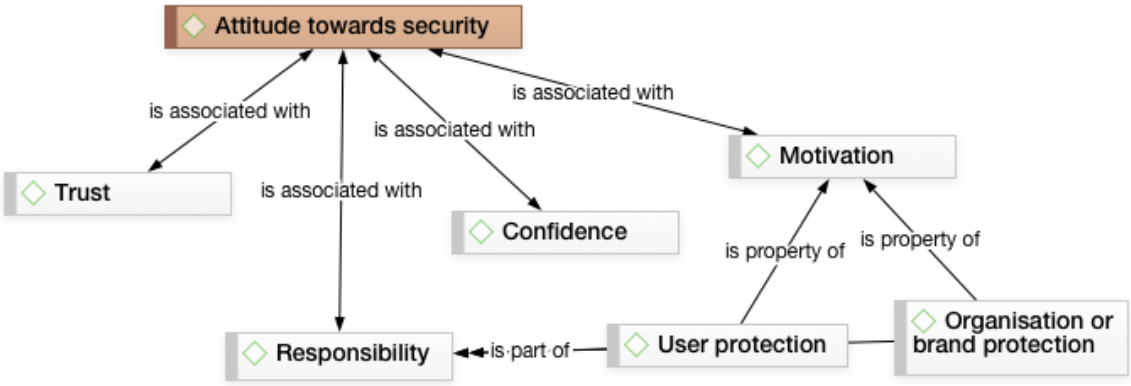



Figure 4.4– Developer's Attitude Towards Security

#### i. Confidence

One form of expression of attitude identified during analysis is confidence in self, group, or community expertise. Participants had a certain level of confidence in their past security experiences, software engineering skills, the security standards of the platforms and development kits they use, and in members of their developer communities. Self-confidence

and efficacy were expressed mainly by participants with a background and training in security. Some participants were confident in their ability to identify and address vulnerabilities and threats to their apps. In comparison, others were confident in the security standards of the tools and technologies they used.

> "*I've got a master's in computer forensics, and I've got security experience and background. I kind of take security seriously and consider it. So when I'm doing architecture, I'm thinking, "OK, how is that information going to be stored? Are we making sure that we're using encryption? What kind of encryption is being used? I consider all that. I'm confident in deciding what needs to be done*." [P1]

Participant P1, a mobile and web developer, describes their confidence in making security decisions concerning software projects as a result of the knowledge gained from security education and experience. They claim to have the correct set of skills and abilities needed to handle any security challenge faced during the development process, regardless of any organisational context. This informs considerations for the proper encryption type, data storage, and access policies that comprise their security practices. P1 also claims to take a proactive approach to addressing security in software projects because they are confident in their abilities. In contrast, participant P9 takes a more reactive approach to implementing software security, as their apps have never been attacked, which builds confidence for P9.

> "*I have not had any intrusions in my apps. I have not had any major cyber security concerns, even from my users. So, I think there is just that level of slight confidence…maybe not confidence but this relaxed atmosphere about what I'm doing and if it isn't broken, what is the point of fixing it? Yes, that kind of mentality. I think that adds to it*." [P9]

The interview with participant P9 was fascinating because they didn't want to be perceived as being too confident in their security knowledge or in the fact that none of their apps had experienced any security breach. The statement, "if it isn't broken, what is the point of fixing it?" may be interpreted as meaning that if there are no threats or security breaches, then why bother with security now? This reactive approach and laid-back attitude were common across most of the developers in this study.

Participants also described confidence in the frameworks, platforms, tools, and technologies they use. They had confidence in the security standards of the larger organisations that build these tools and create platforms for curated app security guidelines and information. Platforms such as the Open Web Application Security Project (OWASP, 2021) are trusted to provide recommendations for security best practices, guidance tools, and frameworks needed to develop secure applications. However, some participants do not consider it necessary to implement the OWASP recommendations.

> "*For me, what I get from OWASP is more than enough really. And OWASP is quite a big organisation that keeps me informed about what exactly is happening in the app security world and things like that. So, I don't generally tend to look anywhere else*." [P1]

> "*One of the things we do every year is, we revise the OWASP top 10 standard guidelines, but then with the OWASP standards, you still have to be very aware of the data you apply to your daily coding and who has access. In terms of security, for example, we do things like communicate via HTTPS, which is an encrypted protocol*." [P6]

Apart from security information sources such as OWASP, participants expressed confidence in the development kits and platforms they use, as well as the impact these have on their security practices. They explained that these software development kits have been created by large organisations with the necessary structures and resources in place and are automated to include the best security measures in their tools, so that developers can rely on them to build secure apps.

> "*I monitor our app security when using Google Firebase. I delegate most of my security concerns to reliable third parties like Google. For instance, Google's Firebase is a structure that has been standing for several years. Although it wasn't initially for Google, Google acquired it and in my experience with using Google products, I believe they work hard to ensure that your details (projects) are safe with them. Also, reviews from other developers, I've not heard of any cyber-attacks while using this platform*." [P9]

P9 is a native Android app developer who works for a medium-sized company and also works on independent software projects. In the statement above, P9 expresses confidence in knowing which security technique to implement based on the recommendations of the software development tool used, Google Firebase. P9 describes Google Firebase as a secure platform used in developing their mobile and web applications. For developers in SMEs such as participant P9, there is a dependency on platform-specific security measures for their software.

### ii. Trust

It was challenging to interpret what trust meant to developers in this study, because trust can be defined in multiple ways. A connection between confidence and trust in one's own and/or group's software security abilities was identified during the analysis. To gain an understanding of trust in the context of this study, the focus was on how participants described trust based on their experiences or used the word in their statements. In one way, participants described trust as a form of group-efficacy or in-group expertise. They were willing to implement security recommendations from members within their developers' communities because they trusted these recommendations to come from the genuine experiences of community members. To some participants, these communities represent a form of social structure characterised by hierarchies based on expertise within the software development ecosystem. Members within this social structure relied on each other to share experiences, resources such as information about platforms or new technology, security recommendations, codes, and code reviews. Some developers in these communities were considered to be more trusted experts than others in the field. The opinions of trusted experts are given higher weight, and their suggestions inform the practices and decisions of other developers within their communities. P9 referred to these trusted experts as "*celebrity developers"* because of their reputation within the community.

> "*If the celebrity developers get involved and advocate for more secure apps, we would see this gradual change in the outlook of developers towards security."* [P9]

Not only do developers' communities play a major role in shaping the practices and beliefs of participants but specific individuals in these communities are considered to be major

influencers. P8 explains these further, describing trust in the words of developers they consider to be "*smarter*" in their community, to recommend secure software development tips.

> "*There are a lot of smarter people in your community, and if they're saying this is safe, then okay, I will believe it*." [P8]

Although participants, such as P9, describe trust and confidence in the information gathered in developers' communities, they also expressed concerns about the lack of discussions around security within these communities and a divide between developers and security experts.

> "*I also think that security is just something that is not discussed enough in developers' circles. I meet with some other developers almost every week, and we rarely have discussions about it…or rather I do not hear people talk about security. I think it is not properly ingrained in the app developers' culture*." [P9]

Whether they are 'smarter people' or celebrity developers, analysis showed a sense of hierarchy in the security expertise levels among developers within these social structures and communities. This may place a responsibility on the developers at the top of this hierarchy to ensure that they raise software security and cybersecurity awareness through ongoing conversations.

### iii. Individual and Collective Security Responsibility

While some participants assumed the role of security expert and became responsible for the security measures practised by their team, others delegated security responsibility to third-party services, other team members, or "celebrity developers" whom they assumed to be the security experts. Participants expressed their sense of responsibility in various forms, both individually and collectively. For some participants, individual security responsibility comes with a sense of authority and control over access to data during the development process due to their roles within the organisation.

> "*Server details need to only be shared with people [team members] that really need access to it. So, if you are not working on a feature where you really need it, then I am the custodian of that information, or the project manager is the custodian of that information, and you only can request it when you need it*." [P1]

In P1's statement above, the participant refers to themself as the "custodian of that information" to emphasise ownership and control over the distribution of and access to information within their team. P1's statement can be interpreted as a form of individual security responsibility, ensuring that other team members are compliant. However, participants such as P6 and P8 believe that specific development roles, such as DevOps (Development Operations) or Quality Assurance (QA) personnel, are responsible for and have the authority over security decisions.

> "*So, in my current team, there are the DevOps guys in my team who make the security decisions*" [P6]

P8 describes a form of collective security responsibility shared between the development team and the client. P8 argues that it isn't the responsibility of developers alone to know about security frameworks and guidelines, instead, the onus is on the entire development team including the client, to ensure that the final software product is secure.

> "*I think it is a QA rather than the developers. So, there will be the QA on the client's team and then on our team, definitely following security guidelines more diligently than we developers are doing*." [P8]

P6 then further describes security responsibility as an activity that involves the entire development team, other members of the organisation, as well as the end-users of the app product.

> "*I think the whole development team should be thinking about security. Just the same way the whole company should be about GDPR and data protection. Everybody has the same sort of responsibility... I think the users should be responsible too. When you are using a password, I would say users should be responsible. If you put in a*

> *password, the person using the application should make sure that the password is strong. I think the platform providers would also be responsible*." [P6]

One additional form of collective security responsibility, as described by participant P5, involves a shared responsibility between developers and platform providers or third-party services. According to participants, it is the duty of organisations that create software development kits and third-party service providers who develop APIs and libraries to ensure that their tools meet specific security standards.

> "*So, if you push your code to a repository, what you've done is you've put the responsibility of securing your code to them [third-party services]. Now, if you were really paranoid, what you'd probably do is, before sending the code, you would probably encrypt it locally...I think the whole team is responsible for security. The users should be responsible, and the platform providers should also be responsible*" [P5].

In summary, the analysis shows that developers in this study agree to both individual and collective security responsibility. With collective security responsibility, security decisions, measures, and practices are shared across development teams, clients, end-users, and involve the role of third parties and communities in these security decisions. Individual security responsibility represents the role and duty of the developer in ensuring that the final product meets acceptable security standards. Furthermore, it is essential that software development kits, platform providers, and other third-party suppliers understand that developers in SMEs rely on them to assume the majority of security responsibility for their development tools. Whether a developer feels individually responsible for security or depends on the shared responsibility of team members or platform providers, these varying levels of responsibility may influence developers' attitudes towards security.

#### iv. User versus Organisation Protection

The analysis showed that it was important to developers to protect their end-users, clients and organisations from potential attacks that could occur due to vulnerabilities in the apps they had built. Apart from deciding between protecting clients or end-users and reducing project

costs, participants also described their reactions to the impact of these decisions on their business integrity and brand reputation.

For participant P1, there was a concern about clients not heeding their security advice and recommendations. Clients may decide to cut down on or avoid additional security costs, which contradicts the developer's security advice. As a developer with security knowledge, P1 explains the burden of responsibility for protecting both their small software company and its clients. Clients cannot be required to pay an additional charge for implementing specific security measures. So, whatever the consequences may be, the reputation of this small business is also at stake.

> "*My job is to provide technical guidance and security advice, develop the software and help the client understand the length and breadth of what we're working on. So, if I've provided the advice and the advice is not taken, as long as it's not something that would generally tarnish my company's image, then I am in business to make money.*" [P1]

P1's statement above was particularly interesting because of the roles of the client and the business context for making security decisions in the software development process. Although security practices are still expected to be done at a certain level regardless of the cost implications, developers are left with deciding to trade-off security for other features when the cost isn't being covered by the client. These business dynamics may contribute to how developers in SMEs are able to practice security effectively.

> "*We deploy to different platforms depending on the client's capacity and scope of the project. We worked with a company in the past that outsourced their app development services to a separate IT service company that outsourced a part of the project to us. So, we had to depend on the IT company that provides the client with the tech for information with limited access to the client themselves. Sometimes, as developers, we might need to use certain security APIs and they will say it's not within the scope of what the client had paid for and then everyone starts arguing back-and-forth about how to move forward. It was a terrible experience. Everyone ends up doing what they think is right for their business.*" [P12]

P12 also describes an experience that highlights how business dynamics contribute to security considerations when working on software projects. Developers need to work within the client or business budget, and so if there are any security measures that require extra funding, they are likely to be disregarded until there is an actual incident. The developer then prioritises protecting their business name against the consequences of not implementing security measures, practices, standards, or technologies.

Priority given to protecting the business versus users or clients relies on whoever is ready to cover the cost of security, and this may motivate developers to consider their security options during the development process. For small businesses, ensuring that the users or clients and the integrity of the business are protected requires more than just knowing about software security practices. It comes with understanding trade-offs, costs, and proper project management. In addition, protecting the organisation isn't only about implementing the right security measures, it also involves ensuring that the business stays afloat, and its reputation remains impeccable. So, developers must continuously find ways to fulfil both business and client's interests.

### 4.3.6 Motivation

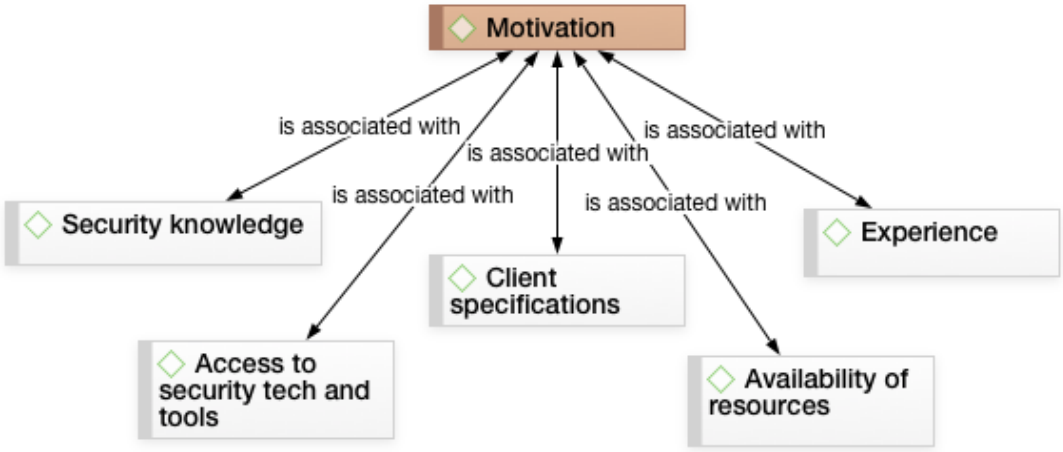



Figure 4.5– Security Practice Motivation

Motivation was another core concept that was identified during analysis. Developers in this study discussed various reasons for why they made certain security decisions and did some security activities. They explained that developers are mostly motivated to perform security activities such as code reviews, penetration testing or vulnerability assessments, based on their existing knowledge of security, access to the right technologies, client specifications, resources available and personal experiences with cyber-attacks or threats. In addition, participants also claim that these motivating factors provided good opportunities for them to learn about new security trends, practices, and standards. In the subsections below, more data from the participating developers were analysed to describe their prominent motivating factors.

### i. Personal Experiences and Background

One motivating factor for adopting security practices during the development process was developers' experiences, expressed by participants in different forms. Apart from academic and professional security training, experiences were described from personal encounters with security breaches and cyber-attacks. Participants also described the security experiences of other developers in their communities that motivated them. Academic background, professional security training and personal encounter serve as reference points for developers to adhere to certain levels of security in the software projects.

> "*I've got a master's in computer forensics, and I've got security experience and background. I kind of take security seriously and consider it. So, when I'm doing the software architecture, I'm thinking, "OK, how is that information going to be stored? Are we making sure that we're using encryption? What kind of encryption is being used?" I consider all that. I'm confident in deciding what needs to be done*." [P1]

Unlike participant P3's experience described below, developers such as P1 are motivated by the personal desire to implement various elements of their security knowledge gathered from training. This security knowledge motivates the willingness to apply security measures during the development process without necessarily considering the external rewards for their actions.

> "*From experience, I try every way I used to hack. So, I try it on my apps to see if they are vulnerable. Even while I'm still coding, I consider those things and I block them. "OK, can the user go in through this? Can a hacker go in through photo upload? OK, why would he be able to? Oh, yes, I left this extension. Let me add that*." [P3]

To give some context to P3's statement above, during the interview P3 had described themself as an unethical hacker before developing mobile apps full time. Their past experience as an unethical hacker motivates current security practices and activities as a mobile developer. Developers such as P3 rely on their instincts to practice software security effectively by ensuring that their apps are protected from potential attacks that are similar to experiences from their past. Participant P3 may have been driven to a certain degree because of the sense of self-guilt, personal fulfilment, or the need to outrun potential attackers, since they used to be a hacker in the past. This serves as motivation for implementing certain security measures during their software development process.

According to most participants, complex security guidelines and certification processes can be difficult, causing developers to ignore best practices or push security decisions further down the development process pipeline. This is to avoid spending extended periods of time trying to learn about new security measures for each unique software project, which could in turn demotivate the developer. P10 describes some form of complexity with security guidelines. This may be as a result of no security training and background, or that the existing security guidelines or standards are just difficult to implement.

> "*I haven't experienced any security issues in eight years of being a developer. So, no motivation to implement security. We don't have the time to read a bunch of security guidelines, I think security measures and guidelines need to be made easy*." [P10]

> "*Get your customers first! I recommend this because setting up proper security in a shared cloud environment, for example, is not straightforward. Okay, now imagine that process taking six to seven weeks of the app project time, just because you want to follow security best practices and your product is not even out yet. So, we deploy first then start fixing security issues later. Okay, I know this sounds bad when I say it like this but that's what it is*." [P12]

Participants P12 also described difficulty in setting up the right security measures and allocating the appropriate time for security between development and deployment. Even with security knowledge, P12 suggests that it is better to focus on building and deploying the features and functionalities of the software first before considering security based on past experiences. Developers may be motivated to adopt security practices based on their background or personal experiences but can also be demotivated by other factors due to their organisational context.

### ii. Security Knowledge and Awareness

Participants reported their various means of acquiring and sharing security knowledge within their teams and communities. Security knowledge was gathered from personal experiences, academic background, security training, developers' community discussions and online forums. For participant P3, acquiring knowledge from other developers was a good motivator for adopting certain security practices. However, the challenge was that software security topics were not often discussed as much as software scalability and performance within these communities.

> "*Another challenge I just remembered, you only stumble on security tips as a developer if you are following a lot of channels or a lot of fellow developers who are into security, that's what you see. Because I feel it's more about interest. What you see on the web most times will be how to speed things up or how to optimise that. Nobody's really talking about security … Most people don't really talk about security and it's a problem. Most times, they talk about performance and optimisation and scalability. We learn about scalability, scalability, and scalability! Most of the frameworks we have, focus mostly on scalability, rather than security*." [P3]

This is a particularly interesting situation because as identified earlier during the analysis process, developers in SMEs rely heavily on their communities to provide guidance for their security practices. When members of these developers' communities rarely discuss security topics, then individuals are likely to focus on the areas of development that are frequently discussed. These communities serve as learning platforms for developers in this study. Hence, a variety of security discussion topics can support the practices of developers. We

This is a key socio-technical factor that may influence how developers in SMEs consider security tools, technology, and practices.

> "*As a developer myself, there are some practices that may leave my apps vulnerable. So, while coding I have to think about, "Is this data coming from the user"? You should avoid things like script injections, know about the database security issues, know the libraries you use... that stuff. I think it's mostly knowledge and experience that makes developers better with handling security issues and also may be available resources*." [P6]

In addition to acquiring general security knowledge, participants also agreed that it is important to be well-rounded and learn about handling different cyber security incidents and implementing software security. According to participant P6, having knowledge of the types of security issues that you may encounter and ways to address these issues can motivate developers to practice secure software development. For all participants, acquiring security knowledge formally meant that developers needed to be trained to a certain level which requires extra time commitment and funds. When we compare the statements from participants P6 and P3 above, it gives insights into the challenges that developers in SMEs face. When developers are unable to acquire the right security knowledge, it could impact the outcomes of a software product. Developers in SMEs particularly will need additional financial resources to be able to train team members and developers running micro-businesses may need to pay a high price to get the software security information they need.

### iii. Availability of Resources

Resources have previously been described as a contextual factor that influence how developers in this study reported their security decisions during the development process. However, further analysis show that developers also classified the availability of both human and financial resources as motivation for adopting and implementing security practices. For example, being able to afford to hire a security expert to join the development team, or for training more developers in software security. Participants explained that limited resources can demotivate them from investing in security training, technologies or frameworks.

Referencing participant P5's statement again, the availability of financial resources motivates security implementation and in turn, effective security practices.

> "*...it depends on the number of resources you have, and I guess that's the motivation. Basically, your thing [app] is only as secure as how much money you can put behind it"*. [P5]

Other types of resources relevant to implementing software security practices include human expertise. Participants described that a considerable amount of human expertise in security and manpower are also needed to complete each software project. And so access to readily available resources motivates the developers to consider the importance of practising secure software development.

### iv. Client Requirement Specification

For most developers in this study, a software project typically starts with the presentation of a requirements specification brief from a potential paying client. Participant P9 briefly describes the process:

> "*Well, here is how it works, um, we get the feature lists from the business team, which are presented by the client and then our project manager presents the clients' requirements documents to the team. And then he outlines tasks, um, every Monday or so we do reviews to figure out how far we've gone. As developers, we are left with how and what we need to do to accomplish all the requirements and the timelines for it. So, we have no formal structure really. I think our current structure just supports personal creativity. Well, we have a level of control by the project manager, or the lead developer*." [P9]

Clients' requirement specifications may or may not include security requirements that developers need to implement. According to P9, the software project plan depends on what the client has stated and paid for, and each developer in the team works on their assigned tasks. If there are no security requirements, then team review meetings may not include software security discussions except the team already has security expert or developers with

security knowledge amongst them. Developers are expected to do exactly what the clients have presented in their requirements document.

> "*What I did was whatever my clients needed*." [P4]

At this stage, the study shows that the participating developers in SMEs and micro-businesses rely significantly on some form of payment from the clients prior to delivering their software products, so doing whatever the client needed may not include any security activity. However, clients can be charged additional fees that increase the cost of the project and resources needed to build a more secure software.

Participant P7 also explained that the choice of implementing security measures wasn't solely dependent on the money the client was willing to pay but also the type of client. For clients with some level of security knowledge who may have requested for certain security measures in their requirements specification document, P7 said they had to adhere to recommended security standards to fulfil the request of this client. But for clients who just wanted a working app, security standards may be initially ignored until after the first iteration of the product.

> "*The thing is, there are particular clients I've worked with that have certain standards that they want you as a developer to adopt. Some are non-technical people who don't know anything about any security standards, they just want you to build their apps as a solution, present results really. Those that have technical knowledge about what they're asking us to do, give you an idea of standards to follow*." [P7]

Implementing security measures in the development process based on client specifications or the client's willingness to cover any extra security-related costs, raises further questions about the role clients play in supporting and encouraging the security activities and practices of developers in SMEs and micro-businesses. Even with the knowledge and experience, developers may struggle to adopt security best practices if it means that they must use their personal funds, or in other cases, clients who do not state that they require security features in their software product.

### 4.3.7 Support

In further interview discussions, participants described support as security recommendations from other developers in their organisations or online community members they consider having more security expertise and experience. Other forms of support described by participants include platform-specific and SDK support and third-party security service providers.

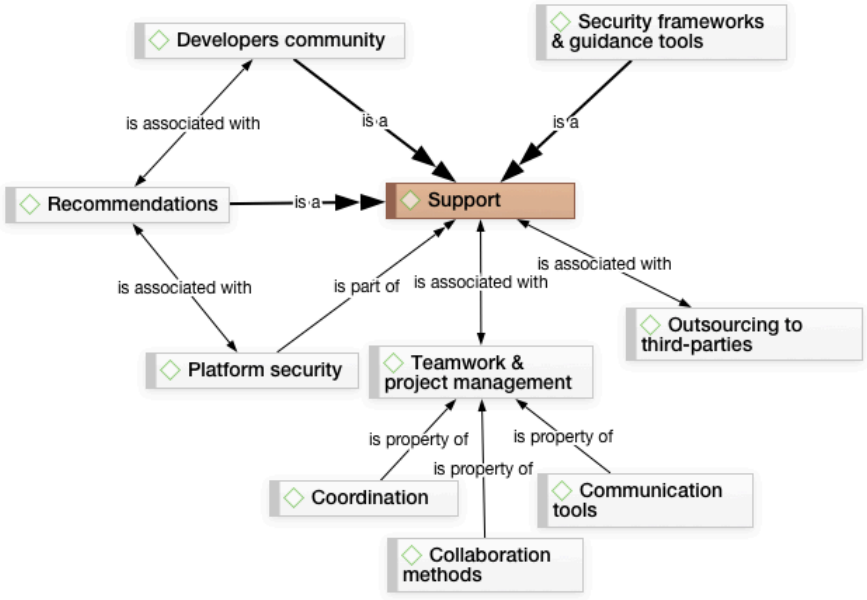



Figure 4.6– Developers' Security Support Network

#### i. Developers' Community of Practice

Most participants reported that although they can make decisions about implementing security measures themselves, they prefer to get some sort of approval about the situation from members of their communities. Interactions within these communities serve as a support system for developers in deciding what tools or technologies to use, the security practices they were necessary for their apps and how they were to be implemented.

> "*Developers actually follow recommendations from individuals [in the community] more than they follow institutions and companies*." *So, most of the time I go on Stack*

> *Overflow for example, when I am trying to fix a problem. And someone drops just a random answer that is not the best practice. You can see other people commenting, this is not the best way to be. As a developer, most times this is not the best way to do this, but I'm getting what I want from it.*" [P3]

Software development experiences, concerns, problems, ideas, skills, information about various topics of interest within their communities. Participants claim to be avid followers of recommendations shared by other members of the communities.

> "*I look up to mentors within my community and follow the norms.*" [P7]

Some participants explained that community members would meet online or in physical spaces to share their knowledge and expertise of different subject areas across the software application development landscape. For security topics in particular, participants say they relied on members they considered as 'mentors' within their communities, to provide security solutions by recommending software security tools, technologies, and frameworks. Active communication and collaborations within the communities seem especially important to developers in SMEs and micro-businesses because participants described that they rely on active responses to their questions as support for challenges they face during the development process. For developers such as P3, the developers' community seemed to be doing enough to support their security knowledge.

However, some participants such as P1 were concerned about the lack of cyber security and secure software development awareness within the developers' communities or at events for developers.

> "*I think awareness about security for developers is something that is a challenge. Awareness at developers' community events especially.*" [P1]

Similarly, participant P5 worries about implementing security ideas and decisions without verification from their community. Some developers in SMEs may rely on the consensus within their communities about every security measure for their software projects.

> "*It can feel dangerous if you are thinking of implementing your own security ideas that have not been tested or refined by the community. Basically, you have a false sense of security whereby, because you came up with this, you think, "Oh yes, this is good" but because nobody has tested it, nobody has verified it*." [P5]

### ii. Platform Built-in Security

Participants also described that they lean on built-in, automated security measures provided by their software development kits, platforms and third-party security services to support their security practices. For participants in this study, the term 'platforms' represented a variety of development environments, tools and frameworks used for software application development, deployment, and management. An example of support is for Android web and mobile app developers, they rely on the support, guidance and recommended security practices provided by Google Firebase platform. It was initially unclear from discussions with participants whether security mechanisms built into these platforms were enforced by default, or whether developers still had to decide if they were going to enable it based on fulfilling requirements.

> "*Yes, so we do think about security, but the good thing about software development nowadays is that if you choose to use a particular framework, most of the time, or more often than not, the security is built into it, so if you follow the dos and don'ts of the framework, then you can safely say your application will conform to certain standards*." [P4]

In addition to formal security training and knowledge, participants rely on their development platforms for security mechanisms that support their security decisions. Built-in security provided by these various development platforms and third-party security services can also support developers in the form of contact help centres, wiki pages dedicated to security-related information, recommendations, guidelines, tools, online forums for developers to discuss issues or default security plug-ins in the development framework.

> "*I work with MVC (Model-View-Controller) frameworks, and these are frameworks that have their own security, at least with CodeIgniter. When I'm using the form*

> *validation libraries, it does its own security checks, which sanitises the computer, filters, and scans for things. To that extent, I kind of feel my web app is secure*." [P1]

> "*I used a non-secure library once and GitHub started giving me these conditions that that particular plug-in to my app isn't secure. So, I had to change it. So that's something that GitHub does that is very good. So, they let you know if your plug-in is a threat to your project*." [P3]

The GitHub notification example described by participant P3 shows that software development platforms can enforce security mechanisms into the practices of developers. Developers may still need some form of support or guidance from team members in deciding what security mechanisms are necessary to implement on these platforms. In addition to depending on built-in platform security, P6 advised that developers choose platform providers that are considered to have advanced security standards.

> "*So, if you are a small team, don't try to build your infrastructure yourself. Put your stuff on maybe AWS, which is a secure platform, or put your stuff on maybe Google Cloud platform. They are platforms that I like, quite mature. When you try to build your infrastructure yourself or take care of your own security, then you end up getting it wrong. I think a mature environment means that they've been around for a while, they are enterprise ready. Enterprise-ready means that they've been used by different other companies that also have really good security, or set a good quality, high standard. Once those companies adopt them, more people in our community adopt them ... like Spring Framework is really big in the Java world*." [P6]

Platform providers such as Amazon Web Services (AWS), Google Cloud or Spring Framework, are described as "mature", according to further discussions with this participant, because they are owned by organisations who have been using the cyberspace for a long time and have extensive experience, resources, and tools to ensure that advanced security standards are maintained.

### iii. Outsourcing to Third-Party Security Services

Another group of developers in this study preferred to use the software and services of third-party security service providers. Participants described these third-party security services as a form of support, either by expertise from managed cybersecurity service companies, security libraries, security-as-a-service provider software, or penetration testing services by independent application security companies.

> "*Not all companies, but some small ones like ours, that have some extra funds, we get a penetration testing service company to go through our applications*." [P6]

For developers such as participant P5, thirty-party security services are also the immediate option for security support:

> "*When I was more of an independent developer, one of the things I tried to do was to piggyback on third-party services. So, instead of me managing my own server security, I would use a third-party service software*." [P5]

However, further analysis showed that deciding what third-party security service company or software to used depends on their popularity and credibility within the developers' communities. Participants explain that they are more likely to outsource security tasks to third-parties services if they come highly recommended within their communities. This presents another socio-technical factor that is pertinent to the security practices of developers in SMEs. This is because developers rely on their social groups and communities to support their decisions to implement third-party security services, tools, or technologies. These service providers and platforms also need to prove that they can handle everything security-related for software application infrastructure.

> "*So, it becomes a matter of credibility, so I try not to use third-party libraries that are not very popular. There are some de facto security libraries for different development environments. For IOS, for example, there is a networking library called ALAMOFIRE, and probably about 50% of us app developers use it. This is a very important layer regarding security, this is a network layer that answers all the communications*." [P8]

## 4.4 Conclusion

This chapter presents a number of factors that are relevant to the secure software development practices of developers in SMEs. It is the first contribution of this study, exploring the key socio-technical factors that influence the security practices of developers in SMEs. The CAMS taxonomy was created to visually represent these factors and details of each category was presented separately.

To further refine these categories, narrow the scope of this study and explore the social dynamics during the development process of developers in SMEs, another round of interviews was carried out using theoretical sampling techniques. The results are presented in the next chapter with a theory constructed to explain and provide more contextual information about developers' experiences and key social dynamics that exist within secure software development activities in SMEs.

# Chapter 5 The Role of Clients and Relationships in Secure Software Development for SMEs

## 5.1 Introduction

In this chapter, the findings from further investigation into the activities of developers in SMEs and the resulting construction of a theory of the *Socio-technical Security Practices of Developers in SMEs* are presented. The theory explains a social structure that explains how developers in SMEs and micro-businesses context interact and manage relationships with client to implement continuous security activities as practices in their development process. The theory is made up of theoretical categories that show how the security practices of developers are formed from the activities shaped by different human, social and organisational factors. The theory also draws insight from initial factors captured in the CAMS taxonomy presented in **Chapter 4**; refined to provide context for understanding the experiences of developers in SMEs. This phase of the study is guided by the research question: **How can we theorise the relationship between these socio-technical factors and the secure software development practices of developers in SMEs**?

## 5.2 Approach

The aim at this phase of the study was to further refine the conceptual categories that represent the broad socio-technical factors relevant to developers in SMEs, narrow the scope of study, and evaluate the social dynamics between developers and clients during software development in SMEs. To achieve this, another round of semi-structured interviews was carried out using theoretical sampling techniques with another set of 13 developers in SMEs across Europe, Africa, North and South America. Unlike the study approach in **Chapter 4**

where discussions with developers were exploratory, the interviews at the phase were more focused on participants' description of the factors identified in CAMS taxonomy and how these factors influence their security practices during software development. A theory is then constructed based on their experiences to explain the role of clients and relationships in secure software development for developers in SMEs. These interactions, dialogs, and negotiations between developers in SMEs and their clients, and communities, and the impact of these relationships on secure software development practice are described by the theory constructs. The theory is then evaluated by presenting it back to developers who have participated in the study through follow-up semi-structured interviews and scenario discussions. The evaluation process was to identify if the developers found the emergent theory to resonate with their experiences and was useful in explaining ways their security practices in software development can be supported.

## 5.3 Theory Overview and Scope

To better explain the security practices of developers in SMEs, considering the factors within the context of software development in SMEs, a theory was constructed. The theory of the *Socio-technical Security Practices of Developers in SMEs* represents the relationship between the individual actions of developers in this specific organisational context and the socio-technical factors that influence their actions. This gives insight into the activities that developers in SMEs describe to make up and influence their security practices during development process, thereby contributing to the security and software engineering bodies of knowledge.

Investigating the security practices in software development gives us a lens into the everyday encounters of developers attempting to implement security measures in their development process. The theory represents a dialectic between human activities (developers, clients, and potential end-users), contextual factors, a security knowledge base (a collection of security expert knowledge), and usable security technology within a specific organisational context. The dynamic relationship between these concepts working back and forth together

contributes to the process of how and why developers prioritise or deprioritise security practices during the application development process.

The theory of the *Socio-technical Security Practices of Developers in SMEs* explains the role of each identified socio-technical factors plays in the formation of security practices of developers in SMEs. The theory statement is as follows:

> *For developers in SMEs to effectively practice (form security practices) secure software development to produce (and deliver) a secured software product, they need to consider the **context** in which this product will be used, gather information from their individual or group security experience **knowledge base**, successfully navigate their relationships with **people** involved in the process (clients, end-users, and other developers) and have access to **usable security tools and technologies**.*

The software application development process for developers in SMEs and micro-businesses takes place within an organisational ecosystem where clients (and end-users) request for a software product to be made. The main goal of the transactions between people in this ecosystem is to produce a working software product. This software application can either be a secured product or a vulnerable and unsecured product depending on the *context* surrounding its development process, combined with the diversity of tools, technologies and methods used by the developer based of their security knowledge. The interplay between elements of this ecosystem forms the practices of developers and results in a software product that is presented to the client or deployed directly for end-users. If there are no successful transactions, negotiations, and dialogs between these elements for any reason, may result to the disregard for secure software development practices and a potentially vulnerable software product that is still been used by clients and end-users.

**Figure 5.1** summarises the interaction between each construct of the theory and the respective factors associated with them formed from further refinement of conceptual categories classified in the CAMS taxonomy and further data collection. As shown in **Figure 5.1**, the possible outcomes of these interactions and relationships are security practices that produce secured or unsecured or vulnerable software products. The relationship between

these constructs and the scope in which they apply or exist elements form the building blocks of the theory and are defined in subsequent sections.

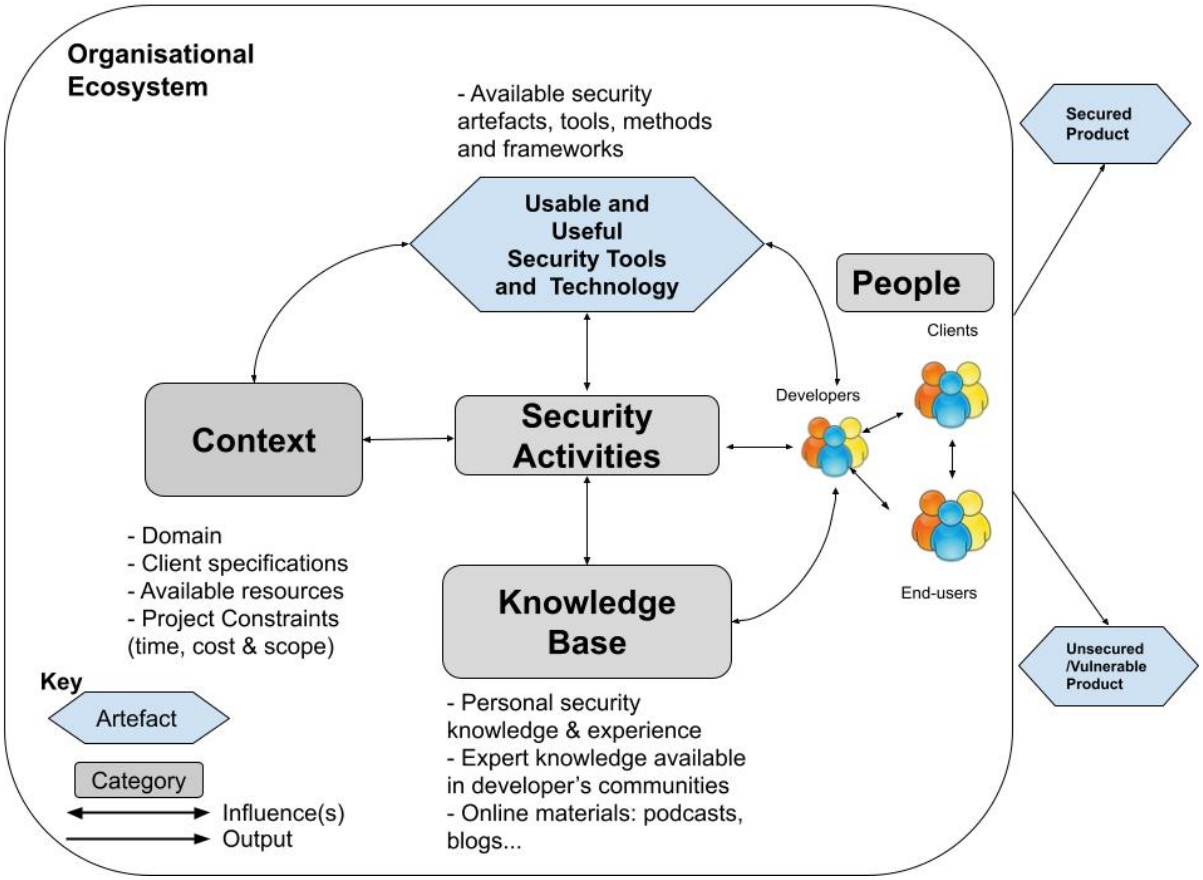



Figure 5.1– Diagrammatic Representation of the Theory

## 5.4 Theoretical Concepts

The components of a theory are conceptual elements that make up the building blocks of that theory and are used to explain the how and why of a certain phenomenon (Mitchell 2014). Bacharach also describes a theory as a "system of constructs (concepts) and propositions Relationships between those constructs) that represent a coherent explanation of a phenomenon of interest within specific boundaries."(Bacharach 1989) The theory of the *Socio-technical Security Practices of Developers in SMEs* is characterised by various conceptual elements defined below based on a subjective interpretation of the qualitative data

gathered. Each theoretical concept and the relationships between them are presented visually in **Figure 5.1** above. The relationship between people and the socio-technical factors identified from the data such as contextual factors, usable security technology and an expert knowledge base is determined by the flow of information within this SME organisational ecosystem.

### 5.4.1 Organisational Ecosystem

The software application development process also takes place within an organisational ecosystem where *clients* contact *developers* to build an application for their customers (*end-users*) or developers decide to build an application that is accessed directly by the end-users or sold via an application store such as the Apple App Store or Google Play Store. For most of the developers in this study, the clients and app stores represent distributors, they are the suppliers, and the end-users are consumers of the software product. Within these business ecosystems, the aim is for the developer to create a product that meets the client's requests or desired specifications and the end-users' immediate needs. Participants also explained that their ecosystem also includes relationships with larger organisations, which they referred to as clients. The need for the software product as well as the process of building this product emerges from interactions within this ecosystem.

> "*I'll give an example. Some people approached us in the past with their ideas. I want, like, Facebook, but for fishermen, and that's all they can say, and that they have funding. Or some people say, I want, like, Waze, but for families, so that's clear if they don't really know what they want. We want to get engaged with people that know their domain, or they're willing to spend time designing the product. Then the product discovery process; that is the next step. That needs to be in place. Product needs to be either fully or partially defined, or the founder, the client, must be willing to spend time in the product discovery stage.*" [P13]

The term *organisational ecosystem* represents an abstract level containing various elements involved in the process of developing and delivering a software product. The ecosystem may consist of a combination of software methodologies, organisation entities, personal

development styles, competing interests, corporate and government policies, company culture, a collection of expert knowledge, a variety of formal and informal practices, and supporting tools and technologies. Participants in this study agree that changes in their ecosystem consequently have an impact on their final products, particularly, changes occurring due to competing interests, government policies or changes in client specifications.

> "*I mean we've started developing the core and then the customer comes up to say, "Hang on, this UI that you gave me (designed) about three weeks ago, I want this to be changed to this, this and this*." [P1]

> *"Regarding codes and the software, it tends to change because of the client, because some of them like to use some type of tools, like using GitLab."* [P19]

Most participants who work directly with clients agree that changes during the development process occur due to various factors such as the introduction of new client requirement specifications or unexpected events that cause a change in the software delivery dates. These changes are expected but they are particularly very expensive for SMEs because they usually aren't any budgets for contingency in place. When asked about flexibility and reactions to these change during the development process, Participant P1 described how they would react to changes requested by clients versus changes introduced by themselves, highlighting that the main consequence of making any change is cost.

> "*It's more money for the business if the customer makes changes. But if it is my own cost going in, I might allow it once or twice. But definitely I would frown at it further down the line*." [P1]

Developers in this study shared similar company cultures with cross-functional teams made up of no more than ten people working on a single project and multi-disciplinary backgrounds. Some participants in micro-businesses described theirs as a "jack of all trades" situation with two or three team members playing all possible roles and taking on all responsibilities associated with the development process. Cross-functional and multi-disciplinary company cultures may exist in other types of organisations; however, it was particularly vital for the SMEs in this study for project completion.

> *"Our team is cross-functional. It's one team, no more than ten people. We don't start with ten people at the same time on that one day, one or two persons only. It could be the Scrum master that will drive the process; it could be a tech lead, a senior developer, that he can start understanding the domain, and then from there things are going…There is no fixed process that we apply to it (developing a software product)."* [P13]

In addition to cross-functional, multi-disciplinary teams, participants also described their business process as fluid with no fixed processes, being heavily dependent on the developer's interest, expertise, or client's requests. Participants agreed that building a company culture and structure for any SME is a continuous process that requires time, effort, and resources for the business to thrive.

> *"It has a lot to do with building the culture. It's, of course, the constant work that you do throughout the life of a start-up, but mainly you need to start right. You need to start with the right platform. What's the best platform at that moment? It doesn't mean that's set-in stone, but you need to have this discussion at the start."* [P13]

For most participants, an informal organisational structure with fluid company practices contribute to the issues around division of responsibilities and roles for each developer. For example, two participants P14 and P24, work for the same organisation - a small, fast-growing software development company that provides a range of services from mobile apps to cloud computing. P14 has about thirteen years of software development while P24 has about nine years of experience with four years dedicated to infrastructure engineering. When asked particularly about security responsibilities during the development process, P14 always referred to participant P24 as the person responsible for security matters. P24 was expected to play the roles of DevOps and Security Expert, apart from his assigned role as the Infrastructure Engineer. On the contrary, participant P24 argues that security should be a collective responsibility involving everyone on the team, and it wasn't their role or responsibility alone to address security issues during the development process. This reflects issues around assigned roles and responsibilities with a fluid informal organisation. It may also indicate the developers' level of confidence in their security knowledge.

> "*Everyone is responsible for handling security concerns, not just the infrastructure guy, but I can say for sure that we are not prepared for security. As a team, we are not prepared for making security decisions, and most of the decisions that we make come from me*." [P24]

Within various organisations, participants described their categories of clients. Clients were sometimes referred to as customers or end-users. However, clients were mostly other small businesses and larger organisations who sold the resulting software products their own customers. End-users were considered to be direct users of products developed by the participants. Some of the participants also explained that their businesses focused on developing and deploying products that was used internally by other employees who were also referred to as end-users. A consensus among participants is that developers attribute different levels of attention to each software project depending on the type of client and/or the developer's interest in their passion project.

> "*So, I started this company ten years ago with [co-founder name withheld], back then we built apps for customers, but since 2014 when I took over the company fully, I started to push the company towards more research, so most of the things we do are mostly for in-house products, so for in-house apps, so we've had less and less clients' work to do. So, nowadays, if we take on a job, it's because we just feel like, okay, we can flex our muscles with it. But most of our work is on maintaining and improving [or adding new] products to our internal portfolio."* [P4]

The different elements of the ecosystem of a software SME need to work together, interacting in one form or another as a system to produce a working software product. The dynamics and changes within this ecosystem present various possible outcomes of the software product and can possibly cause changes to the whole project. Changes in the *organisational ecosystem* for SMEs in this study is also reflected in the complexity surrounding the roles that developers play within these organisations. Participants explained that a developer can play the role of the project manager, client relationship officer or business development manager in addition to developing the software. Playing multiple roles in the organisation may influence how the developer interacts with the software project and other elements within their ecosystem. For

example, when asked about formal security training within development teams, Participant P4 described how switching from developer to project manager influences the decisions made during the development process. P4 started as the only developer in their company, and now as the business grows, their role has changed to being solely in charge of discussing directly with clients, bringing in business bids and making strategic decisions about projects with little time for writing codes.

> "*My role has changed a lot. I hardly have time to write codes anymore because someone else on my team deals with it. I now run the company and have to deal with customers, team members, setting them in the right direction and prioritising stuff. It will be silly of me to be investing in security personally. I would rather leverage on someone else's knowledge*." [P4]

Participant P4 is an example of a developer whose role switching has pushed their thoughts about security lower on the priority list expect when a client clearly specifies the need for security during the project planning and requirement specification phase. Then, they are willing to discuss it with other team members or employee a third-party security service.

### 5.4.2 Context

"*Without context, words and actions have no meaning at all.*" (Turner and Bateson 1980)

Contextual factors influencing the outcome of a software product are unique to the different stakeholders in the *organisational ecosystem*. Like every other individual, developers in this study require some context to make decisions about various activities during the development process of their software products. From design to development, launching and maintenance, developers must consider the consequences of deciding to use one tool over another or whether to implement a practice or not.

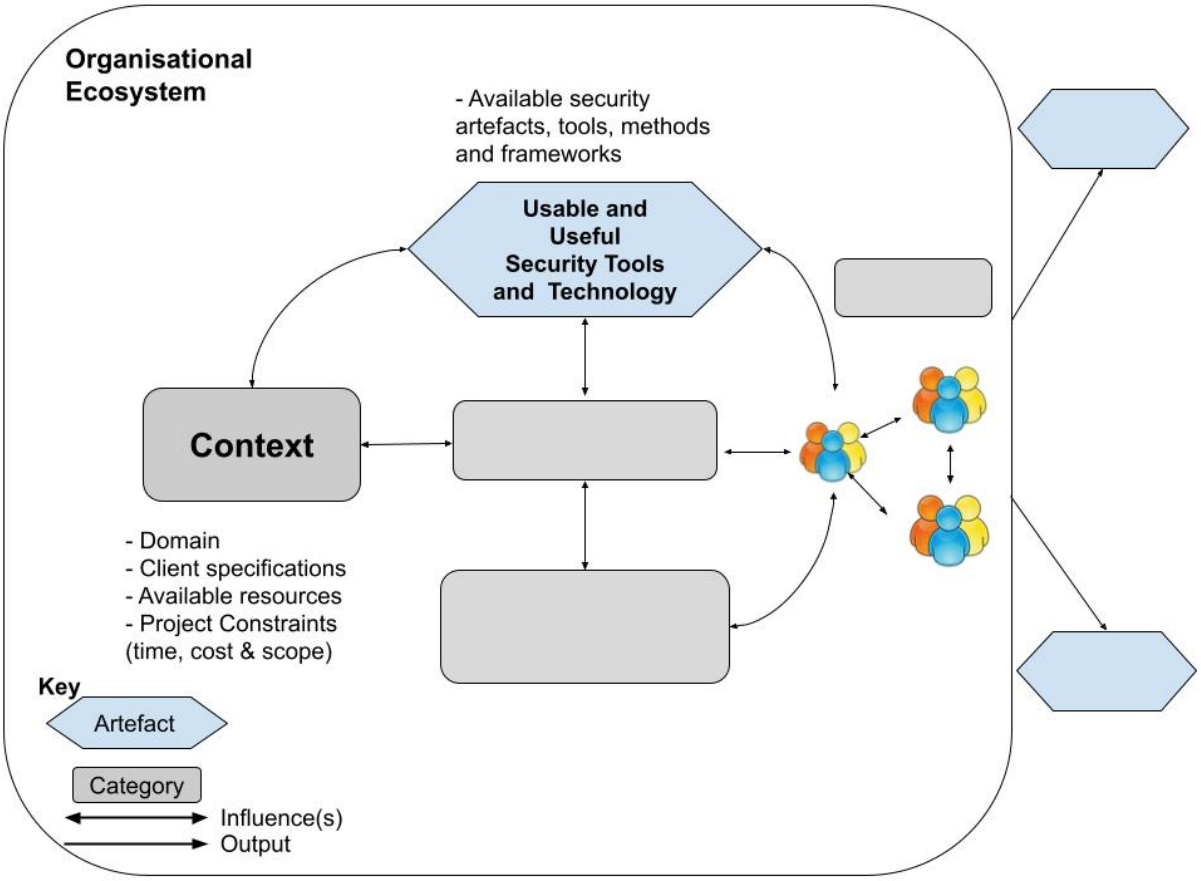



Figure 5.2– Context and the Need for Usable Security Technology

From the initial analysis described in **Chapter 4**, some contextual factors that influence the security practices and decisions of developers in SMEs were presented. Factors such as client specification, software requirements, availability of resources, access to security tools, cost of project, time constraints, and organisational roles and responsibilities were identified as key drivers of developer's security practices. Further analysis showed that these four primary categories were the most important contextual factors considered by participants:

**i. Domain**

This represents the various sectors where the software product is being used. For example, participants explained that an app created for a client in the financial sector may require more attention to security than an app for the gaming sector. The domain of software product or client's sector was a major factor that influenced the requirements for the project and in turn

affects how the developers implement any security practice during development. For participant P13, having a client from the financial sector required them to pay attention to the security aspects of the project from the beginning of the project. Their client was keen to understand their security knowledge levels and the types of measures that may be implemented even before the deal was finalised. However, when security requirements aren't specified by the client or do not involve financial transactions, they become less of a concern.

> "*It all depends on the type of project. We did a digital transformation project for a US bank, so they wanted to hear from the start how we were going to tackle security, what do we know about security before we even signed contracts because it's money. Everything that involves money, credit cards, tends to draw more attention, so when we don't deal with payment technologies then security is less of a concern, it can come later.*" [P13]

### ii. Client Specification

Participants describe this as the requests and needs of their client for the software product. As described in previous sections, clients of participants in this study ranged from individual or micro-businesses to large organisations. Sometimes the clients presents a document detailing their requirements for the software to the developers of a small software company. Alternatively, developers bid for software projects sponsored large organisations. For most of our participants, the client specification and documentation plays a critical role in determining the timeline, tools, methods, and practices that the developer must work with. When asked about making decisions about development tools, frameworks and security measures implemented during the development process, some participants expressed how dependent their decisions were on the client's specific requests for their project. Participants P1 starts by clearly stating that decisions for the project were client driven. Then P1 goes ahead to describe their role and responsibility to their client as a developer on the project.

> "*The client decides. My job is to initially provide technical guidance, develop the software and to help the client understand the length and breadth of what we're working on.*" [P1]

> "*Right now, we are building apps for some major big clients, so we can't really afford to make any mistakes.*" [P5]

In a similar manner to Participant P1, Participant P5 and P24 attributes their actions and decisions during the development process to be dependent on client's specifications. Participant P24 describes an example of a security standard that a client had specifically requested for, the PCI DSS, which stands for Payment Card Industry Data Security Standard. Specifying the security standard shows that the client is aware of the security needs of their software product and require the developers to implement. This client specification can also be seen as a motivator for secure software development for this particular software project, as participants are worried about many mistakes or not meeting the expectations of their clients. Participant 24 describes that the client's request had to be fulfilled even if it meant hiring external consultants.

> "*One thing to clarify, though, in one of our projects, the clients specifically told us, okay, "we need security first measures on everything that we do", so as developers we had to be PCI DSS compliant. That's okay. We understand that and we learned about it, and we thought about it, and we hired some consultants to help us on PCI compliance and everything else*." [P24]

### iii. Availability of Resources

To fulfil client's requests or software project requirements, all participants agree that resources need to be available. Participants described this as the presence of technological, human, and financial resources readily available to complete the software project. These resources need to be properly allocated in the best way possible to ensure that the expectations of the software product are met. For the participants who are the actual owners of the small business owners, the availability and allocation of these resources was a major challenge.

> *"I think the first thing that goes through my mind is it can be really expensive to maintain a security requirement because we can put a lot of tools and a lot of*

> *techniques around our application to make it secure that doesn't let anybody access things that they don't need to access, or they don't have authorisation to access.*
>
> *It's an expensive thing, it's important, but sometimes we have to bring a little thought about how much is enough because we can have sensitive data sometimes and we have to put a lot of effort to secure it, but sometimes there is no need for it because we don't have that much important thing to secure. It really depends on the context and how we approach the security. It's important but sometimes we have to put aside the correct amount of it."* [P22]

Resource estimations and allocations for a software project becomes an even more challenging task when the developers has to consider implementing or maintaining additional security measures. This is because participants explain that security requirement may require more manpower, security technology or services that are considered expensive. With the developers switching between various roles such as project managers, client manager or business owners, ensuring that available resources are properly distributed becomes a difficult task to execute. It might seem rather obvious that with more resources, software SMEs can employ more staff or acquire the security tools or technology needed. However, participants describe the decision-making process as the bottleneck.

> "*We are all developers, five, actually. Well, all tech backgrounds. Some, like one of our developers, he had to move away from development into a more managerial role. So, I would say four are now developing, but all of us have software engineering backgrounds*." [P4]

Participants explained that developers have to switch between multiple roles to fulfil the responsibilities required to complete the software project. In addition to developing the software applications, developers must decide on the right security tools or services to pay for and implement, they also have to keep track of team members, assess risks, and check client specifications to ensure that the project is within scope and meets the specified timeline with the resources available. Much of the concerns about *Availability of Resources* expressed by developers in this study was not only about having the resources readily available but also making sure that resources were used efficiently to meet the project goals.

### iv. Project Constraints

These are characteristics of any software project. With further analysis, the categories software requirements, cost of project, time constraints from CAMS taxonomy were abstracted into the concept of project constraints. As with managing resources, constraints such as timelines, cost, and scope of the project; affect security practices during the app development process. Time-to-launch for most apps are not enough for emphases on security best practices and changes in client's specifications can also add another layer of complexity to the app development, pushing the security concerns such as cyber threats further down the priority list. SMEs running on low budgets for projects cannot afford to hire a security consultant or pay any extra cost for security testing. Investing in security is not a top priority for the developer if it is not a top priority for the client. In addition, if security measures are not clearly defined as part of the project requirements or deliverables, then they are beyond the scope of the development process. Participant P3 reflects on his personal dilemma with deciding the implementation of better security measures versus handling the time pressure from clients. While P5 explained that time and cost were major factors that determined whether there would be thoughts of implementing security.

> "*When my client pushes me, I'm like, "I could have done this in a better way or more secure way, but who cares? This client wants it on time. When I'm upgrading, I would focus more on the security of the app. Well, that's one of the lies I tell myself.*" [P3]

> "*So, there is a lot of stuff, and, at the end of the day, I think that this is probably the biggest constraint, like how much time can you really afford to give towards thinking about the security of something*?" [P5]

> "*I think the answer depends on the type of client and on the project that we are working on. If it's a very small project and the income is not so good, I don't think anyone would improve this or pay for an external consultant or even hire someone as a security expert to help us because the trade-off is not really good.*" [P24]

It is not surprising that security isn't at the top of the budget list for projects, if clients and customers are not willing to pay for these measures, developers in SMEs are unable to get the funding they need to implement them. The security activities and practices of the developers are directly related to the actions and interests of their clients or customers. A great deal of the frustration expressed by the developers in our interviews was focused on the contextual factors that affected their ability to meet their own personal standards of security quality for their apps because they believed they could do more if they had more time, more money to purchase useable and useful security technology, and some flexibility to go beyond scope of the project. These factors were described as key drivers for developer's adherence to security best practices that lead to secured or unsecured app products.

### 5.4.3 Usable and Useful Software Security Tools and Technology

According to the participants in our study, security knowledge is only useful when you have easy-to-use security tools, technology, or frameworks available for implementation during and after app development process. To appropriately implement the security information they have gathered, participants explain that they need usable and useful security tools, frameworks and technologies that are readily available and easily accessible. In the theory, the *Usable and Useful Security Technology* construct represents available security technologies that have been designed to not only be usable but also to useful for the developer in terms of addressing the specific tasks and results they want to achieve. It focuses on the needs and perspective of developers in SMEs; taking into consideration challenges they face in order to have the desired product ready for clients and end-users in the agreed timeframe. In addition, this construct encompasses all tools, techniques, methods, artefacts, frameworks, and technologies that have been designed to enable developers to build systems that provide security to users. For participant P24, security tools and technologies are not independent of the tools he uses daily during app development and so, he depends on the security technological artefacts that are sold with the development platforms or tools he uses. P24 said:

> "*None of the tools I use are security tools. How can I explain this? I think none of them is really for security, but they may address some security concerns. Like if you*

> *work with Kubernetes, they have a whole lot of sessions talking about how you must secure Kubernetes clusters, how you should run it on prediction and all security-related issues. AWS platform also suggests things like what we should use for protecting our systems - using some security patterns and security tools that they offer. I don't use any specific tools for security, but the tools I use are related to or have something related to security. I don't know if you understand me*?" [P24]

Our interview with participant P6 gives a different perspective of the availability and use of security technologies.

> "*There are different tools, different languages, and different ways of doing security ... so I think it's a guide issue, especially for a new or junior developer. I don't know if there is one thing that brings everything together, because you have so many technologies. Although you make something secure in X application, it might be different from the way you make it secure in Y application. So, I don't know if there's one tool that brings it all together*." [P6]

Participant P6's perspective was common for some other participants. They agree that several security tools and technological artefacts already exist, but they struggle with having high-level guidance that supports their decision on when (during the development process) to apply what (security technology) in their context. So, the issue isn't necessarily with the availability of security tools but with better guidance on how, when and where these tools are applicable during the development process. Furthermore, it is obvious that developers will prefer the support of direct 'pointer' that matches available security technologies to the app development phases and platforms they are use. For participants such as P6, a useful security technology should be one that clearly guides the developer on what steps to take to build secure app products regardless of the development tool or programming language.

Participants in our study also expressed how much they relied on the companies that create software development kits and other platforms, to pre-set security measures and standards with the development tools. For participant P24, this made the app development process easier and consequently faster.

> "*Most of the tools that I use are related to cloud platforms. For example, Google Cloud has some security standards that are already on, I don't know how to say this, but they are already set, and I heavily rely on them because it helps my work to be easier*." [P24]

Beyond the availability of usable and useful security tools and technologies, our participants explained that the decisions they make about implementing security and adhering to best practices during the app development process is determined by how easy to use and fast these technological artefacts are. Majority of the participants also agreed that there is a vast number of security technologies available to develop secure app products, but they are difficult to find, complex with very steep learning curves, not 'new developer' friendly and require a lot of time and effort for implementation.

Current security research is evolving to include studies that try to find ways to simplify security tools and technologies with complex interfaces to improve how users interact with them. However, most of these studies largely focus on the design and use of security tools and technologies by end-users. It is important that we do not ignore the 'software developer as a user' too.

> "*You can't fully know the vulnerability because you don't know where. You might just know a few of them, "OK, this would cause an issue. This would cause an issue." But there are so many issues or so many vulnerable points that you don't know yourself. So, relying on the … I'll just say relying on the technology is good, but, at the same time, try as much to also secure in your own way too*." [P3]

> "*I think if we had support from specific consultants or people, even remote-based support I think would be awesome. Just one guy is saying, okay, guys, focus on this one, these security topics are good for this specific project, so that would be awesome*." [P24]

### 5.4.4 People

Software application development and the various security activities that may be involved in the process are enacted by *people* whose goals are to build products that meet the needs of other *people*. Security practices cannot be discussed without a good understanding of actions and activities of people. It is dependent on human actions and these actions are represented as social practices of these categories of people involved in the development process. The theory focuses on three main categories of people who are actors in this ecosystem: the *developers*, the *clients,* and the *end-users*. For simplicity, we decided not to break down these categories any further, but other actors exist in the software business.

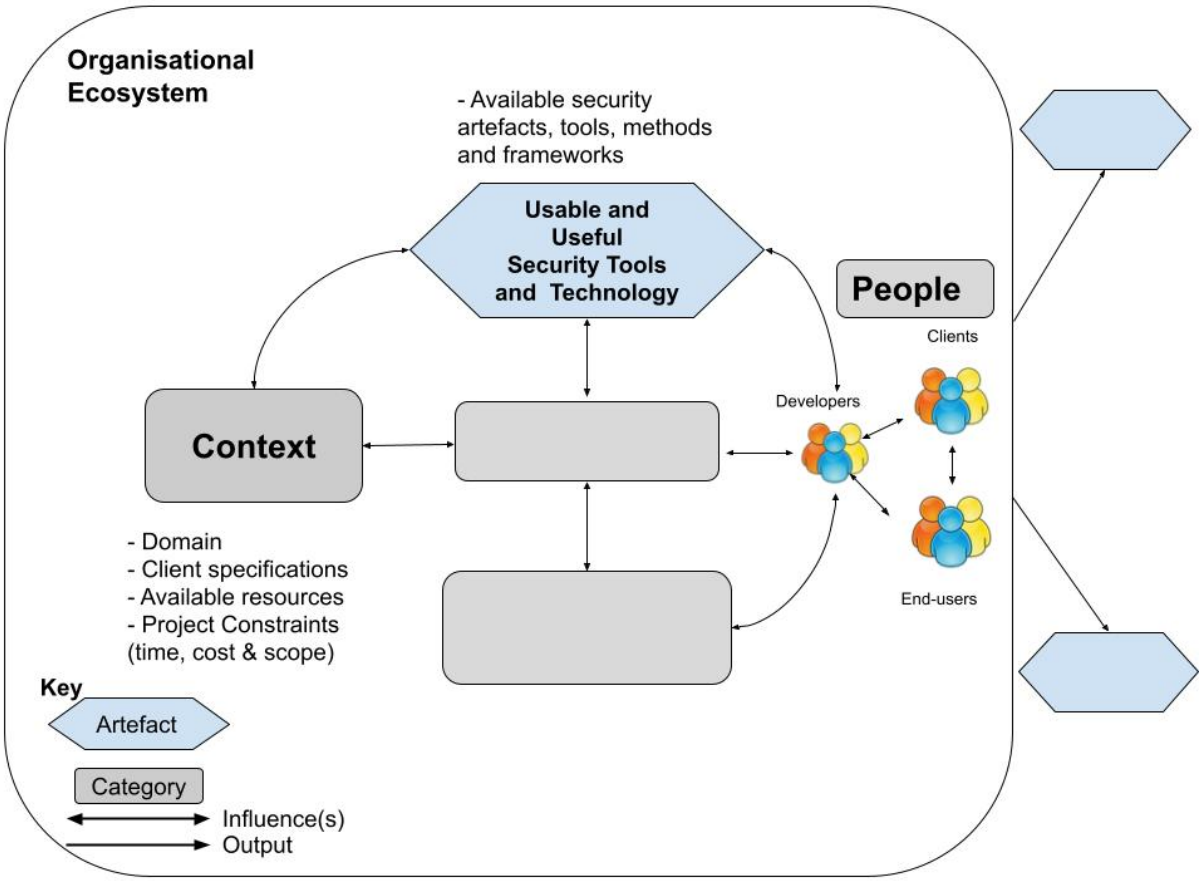



Figure 5.3– People: Human Elements of Security Practice

According to our analysis, we define each category of the people in the ecosystem as follows:

i. **The Developers**: are people who fulfil the requests of clients and build software application products to meet the needs of end-users.
ii. **The Clients:** are people who directly engage developers to create software application products that satisfy certain specifications needs of end-users.
iii. **The End-user:** are people who use the final software application products after it has been built by the developers and marketed by the clients or the developers themselves.

From the analysis, we identified a Developer-Client relationship on different levels within the organisational ecosystem of SMEs. Other relationships such as Client-End-user and Developer-End-user exist but fall outside the scope of this study. Developer-Client relationship is paramount in the development process of the software application product. This relationship contributes to decisions that influence the outcomes of a secured or unsecured product. The *Client* may be an individual or group representing a small or large organisation that requests the services of the developer to build a software system that they can implement in their own business. The *Developer* may be an individual or group who represent the software organisation or micro-business, capable of fulfilling the client's request. Participants described the Developer-Client relationship at their organisations, where representatives negotiate the features and functionalities of the software application.

> "*We want to get engaged with people that know their domain, or they're willing to spend time in designing the product…. You need to know why you want to build, otherwise, what are we going to write, what code are we going to do, and that could be a waste of money. If the product is not defined, people sit on the computers, let's write code, and after a month someone looks, that's not what I wanted. We want to avoid that*." [P13]

During these negotiations, clients may specify their security requirements based on their expertise in the domain apart from stating the features they want. Developers who are knowledgeable in security measures for apps pitch these to the clients who may not have requested security features, highlighting potential vulnerabilities and threats that the app may be exposed to. These negotiations also involve conversations about the cost of implementing

the security measures and clients may or may not be willing to include this cost in their budget.

> *"Sometimes, the client doesn't to invest money and time and resources in strengthening the security of the system. They just want basic security, and they want the software delivered quickly*." [P22]

Participants also spoke explicitly about another category of *People* that influence their security decision-making process and inform the set of cyber security practices they implement. These are the 'Developer's Communities", a group of developers geographically or virtually connected with shared objectives, to learn, share and work together to support each developer's development process. Apart from communities local to our participants, examples of the online communities include GitHub, Stack Overflow, iOS Developer, Android, and Google Developers Group. These communities play a key role in shaping the practices of developers, particularly for those in SMEs because of their limited access to human and financial resources.

Like every other human, developers also need a sense of belonging and benefits from social interactions. They get support from members to address software development issues, answer security questions, share experiences or discuss new technologies. It is important for us to understand the social activities that contribute to and support developer's security practices. We discuss further details of activities such as knowledge acquisition and sharing within these social groups of developers in the following sections.

### 5.4.5 Knowledge Base

Knowledge bases are known to be centralised repositories where structured and unstructured information is stored, organised, and shared by computers (Fagerberg, Fosaas, and Sapprasert 2018). In a social context, knowledge bases represent a selection of knowledge acquired, managed, and disseminated by individuals in different social structures (Carley 1986). Security knowledge bases for acquisition and exchange have become even more crucial as attackers continue to identify new ways to penetrate vulnerable systems. Due to the pervasiveness of apps in our everyday lives, the information and knowledge required to

secure apps and create app that are secured has become increasingly critical. In addition, the hostile characteristic of cyber security involves a constant competition between attackers, software development teams and end-users. This requires developers to be steps ahead in understanding the software security and cyber security landscapes. So, in order to stay abreast of vulnerabilities in the applications and possible security solutions, developers in this study explained that they draw on various sources of information to support their security decisions. The different sources of information are used to acquire and share security knowledge.

In this theory, a *Knowledge Base* represents a pool of information about secure software application development methods, developers' shared experiences and challenges of software engineering, individual and group expert security knowledge on different areas of the development process. Participants describe *Knowledge Bases* majorly as several social sources of security information such as from social networks, institutions, and developers' communities; and claim that information gathered from these sources greatly impact their security activities and decisions. This enables the conceptualisation of the social relationships, and the security learning processes of app developers in SMEs. For developers in this study, the process of acquiring and sharing security knowledge contributes to their security activities and overall practice during the app development process.

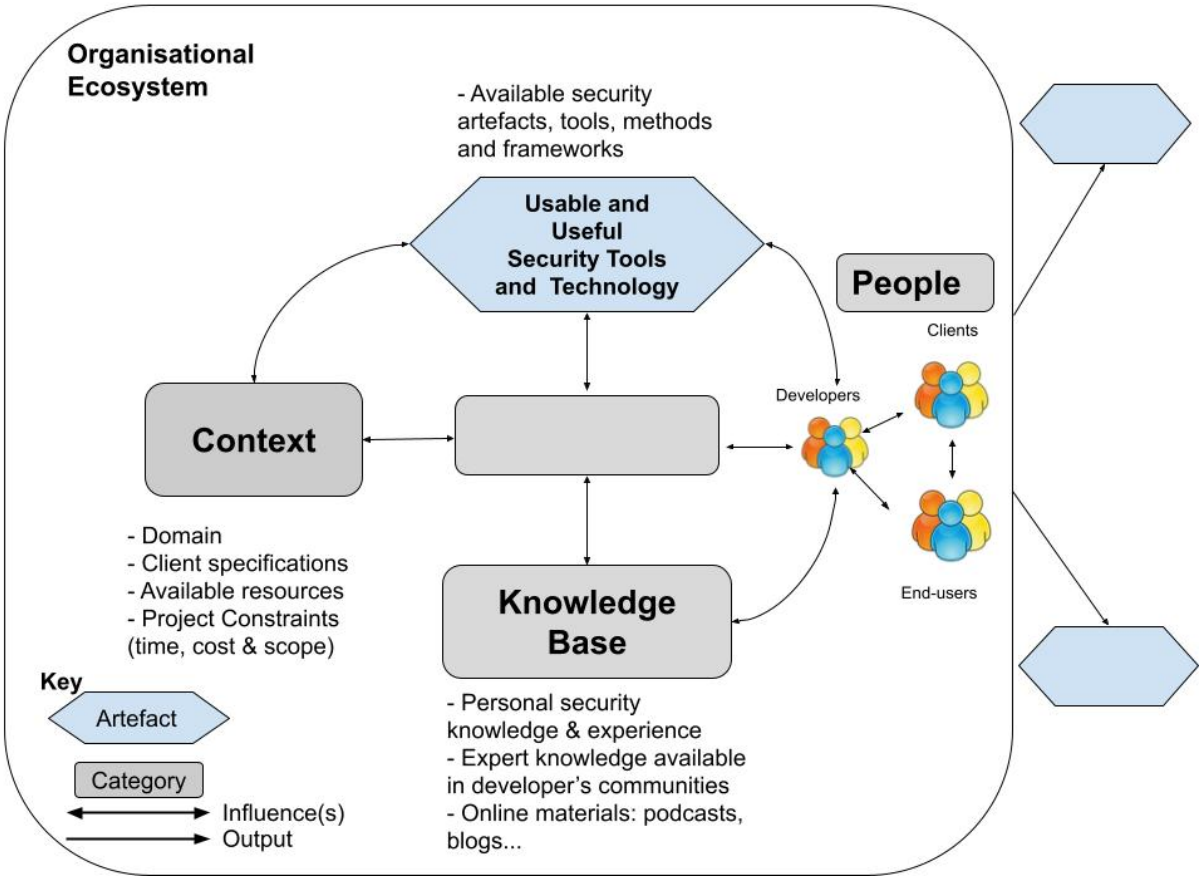



Figure 5.4– Security Knowledge Bases

### v. Security Knowledge Acquisition

From analysis, participants described various opportunities for acquiring security knowledge, from their daily and past experiences as developers, and shared experiences from other developers. This mode of acquisition was typically considered to be more informal and social, and the security knowledge more tacit. However, participants consider this informal or social mode of security knowledge acquisition as core methods for learning about cyber security and secure app development processes and procedures that inform best practices.

### vi. Tacit-to-Explicit Security Knowledge for App Developers

Tacit Knowledge according to (E. A. Smith 2001) is the skills, abilities, individuals gain through experiences and shared mainly through personal interactions, storytelling, analogies, or incident studies. Explicit knowledge is described as a verbally expressive form of the

output of activities and tasks in a more structured manner, with information being disseminated through formal techniques such as in schools, books, models, computer programs or algorithms (Semertzaki 2011). Although participants in this study did not directly call their methods of gathering information and knowledge of security processes and procedures, tacit, it was obvious from their explanations. Most participants depend on some form of conversion of tacit to explicit knowledge in security. That is, tacit knowledge presented and shared in a structured way through formal techniques such as social media, formal training, clearly written procedures, models, and tools, based on individual experiences. They rely on personal attack stories, incident reports and experiences and that of other developers within their communities or teams to be converted to explicit knowledge to make decisions about the type of security measures or tools needed for their apps. Others depend on security knowledge gained from formal institutions or cyber security training certifications.

> "*The truth is I rely on my own imagination and on other developer's experiences. I talk to people, yes. I ask questions. "OK, when you were hacked, how were you attacked? How did you fix it*?" [P3]

> "*I think it's mostly knowledge and experience that makes developers better with handling security issues and also available resources*." [P6]

Of all the participants in this study, only P1, P5, P12 and P24, had received what they consider to be formal security training and education from accredited institutions, self-paced online learning platforms and mandatory training sessions within organisations. Other participants described their security learning process as a more informal approach to acquiring security knowledge. The informal approaches were through social interactions with team members in their workplaces, online support from members of different developer's communities, podcasts of senior developers who were sometimes referred to as "celebrity developers" (based on their reputation in their communities) on platforms such as Stack Overflow, using security materials available online, reading articles and discussions in online forums. Furthermore, participants frequently relied on available and accessible online resources to solve specific security concerns during the development process. Both categories

of participants, those with formal security training and those without, still gather security information from free resources available online. The following interview excerpts gives a view of the security knowledge acquisition process of developers in SMEs:

> "*Yes, for me I think, aside from maybe one or two online courses that I've done in terms of security or more so like encryption, that's pretty much the level of what I have*." [P5]

> "*I don't know for sure if everyone in the team has had some sort of security training but yes, I have, personally. I did some of the PCI DSS training a while ago. In my last job, I did… I'm trying to remember the course... It's something like CompTIA Security+. I think that's the name of the certification course training, but that was a while ago. I'd like to say that I like a lot of security, talking about security and everything on software platforms and development or whatever, but I don't know as much as I want to. Most of the things I know are self-taught*." [P24]

A few participants did not describe the security knowledge acquisition as a technical practice beyond reviewing software requirements specification documents provided by their clients. To a significant extent, they described it as a social process of acquiring knowledge because of the social interactions and knowledge sharing within their teams and with a wider group of other developers in communities.

### vii. Security Knowledge Sharing

Developers in this research acknowledge that knowledge sharing is important for security awareness. Participants say that as developers they depended heavily on the shared experiences of other developers as their source of security knowledge. Security knowledge sharing for participants involves an active exchange of security information and knowledge, recommendations and/or tools among developers within their teams and communities physically or online via sites such as Stack Overflow and GitHub. At the core participants believe that sharing experiences (including security-related experiences) they face during software development with members of the community is a way of giving back to their

communities. Security knowledge exchange media extend from online websites to audio podcasts and physical meetups.

> "*So, for me personally, I go to Stack Overflow and look for information. I go to Google and type in whatever comes up. Oftentimes, when you're stuck on a particular thing, Stack Overflow will probably come up as one of the top reference sites*." [P1]

Generally, security knowledge for developers is shared socially, because like Participant 13 said: "*It is not feasible for all developers to gain formal knowledge of security."* They gather security information from various media, including face-to-face interactions with other developers is based on trust in the expertise and possibly the reputation and credibility of *People* (other developers within and outside their organisation). Once acquired, the information or knowledge gained is then implemented into their security activities during the development process.

### 5.4.6 Security Activities

Security practices refer to the combination of *Security Activities* that participants describe to have implemented repeatedly during the software development lifecycle in order to accomplish security goal(s) set by their clients or within the team. They are at the core of the theory because they form the actual application of the security knowledge gained alongside other socio-technical factors that influence the secure software development processes of participants. Participants majorly described their security activities from agile app development process viewpoint.

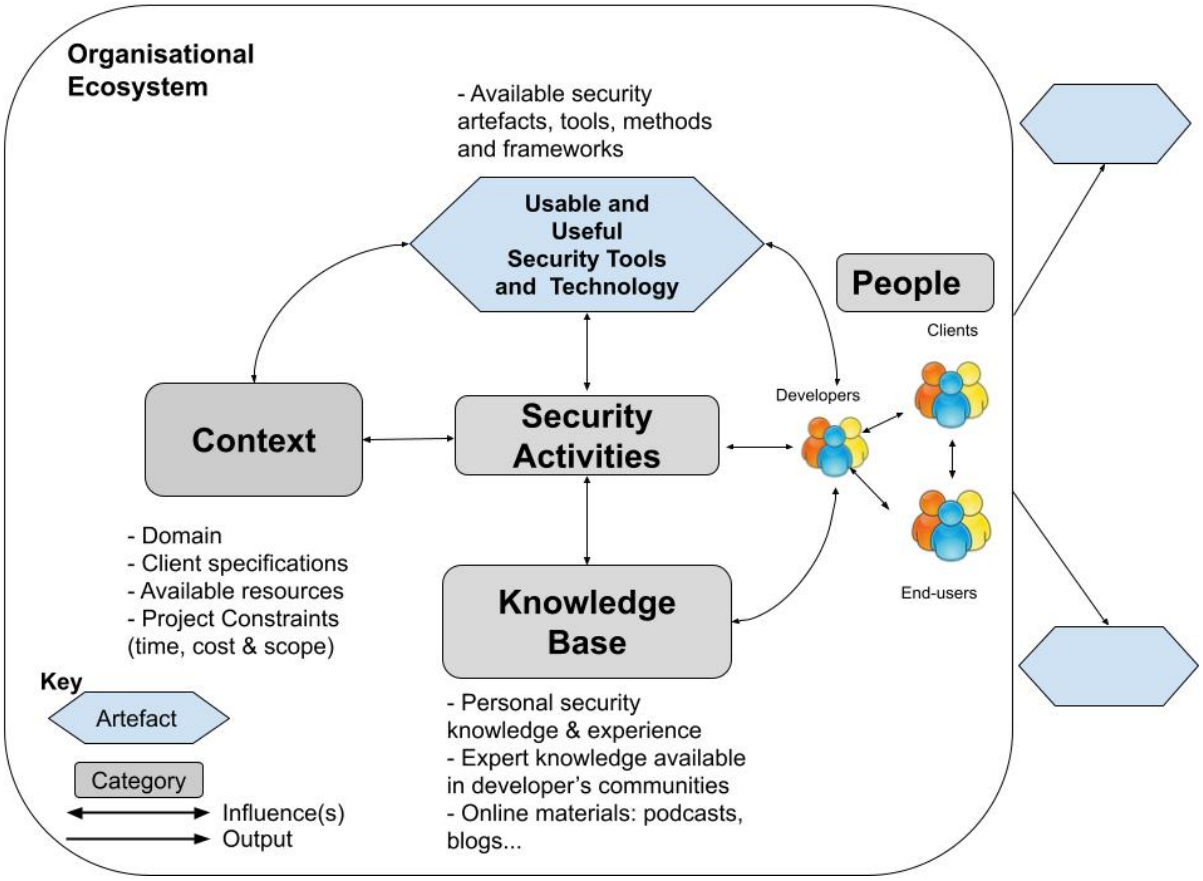



Figure 5.5– Security Activities that form Practices.

The extracted security activities are categorised and presented in below with some excerpts of interviews with developers to help understand how activities were initiated and implemented. The categories of security activities were created based on the descriptions that participants shared. The terms used to define Security Activities were based on 'in vivo' words directly coded from the interview data. Other words were based on secure software development concepts in literature.

**Requirements Analysis and Client's Specification** were described as the "talking stage", specifically involving conversations with clients about app project requirements and identifying the resources required. It was also described as discussions and analyses done within the team to ensure that the end-user or client specifications are clearly defined in the app. Participants also highlighted that communication barriers between clients and developers

during requirements specification exists and can sometimes hinder security considerations and measures.

> "*Most of the stages during development that we talk about security are when we have client-facing meetings, are with client info or client data. This is when we come together and talk about security, some of the security best practices, not all of them.*" [P24]

> *"Regarding handing the codes and the software, it tends to change or depend on the client, because some of them like to use some type of tools, like using GitLab or one client they are using the source internally. One of their requirements was that we use each package to store our code and Heroku to run the application online, but others use GitLab. It really depends on the client."* [P19]

**Identification of Roles and Access Control** for participants in this study, meant that not all developers had access to the database or source codes of the app being developed. Some teams had DevOps or an Infrastructure Engineer to handle accesses and privileges during the development process. This goes together with **Enforcing Least Privileges** to members of the development team depending on their roles and responsibilities. Giving each team member access to the only the data they require to complete their task and nothing more.

> *"So, when I am deploying an application to each environment, the actual keys that give the application access in those environments, are stored in a vault that someone else has access to [controls]. So, even though I am developing the application, we kind of break it down into roles so someone else puts in the keys. My application will just depend on requesting for the keys … "Oh, give me this key", but it won't be the actual developer who has possession of these keys. Even me as the developer, I do not have direct access to the key, the environment provides the key. We have someone in DevOps or Network Ops [they handle this]. That's the kind of security I do practically*." [P6]

> *"I don't work on security, it's not my role. I'm just interested in developing the app. DevOps team is in charge of security."* [P14]

> *"As a developer myself, there are some practices that may leave you unsecure (vulnerable). So, while coding you have to think about, "Is this data coming from the user"? If it is coming from the user, you have to treat everything as unsecure by default so you don't render [end up having] any data that's coming from the user, directly on your webpage because when you [render it] they could do a script injection to your webpage. So, these are things that come through experience. We have different environments as well. Where we are doing development, we have a development environment; and a test environment and each of them have different privileges. For example, I may not even have access to like production security keys, because we do things like [parameterised] environments."* [P6]

**Encryptions, Firewalls, Authorisation Methods, and Multi-factor Authentication** were some of the common cyber security practices highlighted by participants. Development platforms used by participants such as the iOS Software Development Kit (SDK), enforce various encryption and multi-factor authentication methods. Some participants also highlighted the importance using authorisation methods such as OAuth 2.

> *"We start from the basic, like having strong passwords. Strong passwords don't write down those passwords on any system. Do not put clear text passwords in property files; The architecture needs to be layered in between firewalls. You must have firewalls, and further down the firewall layers is what's most critical. Then use of different encryption algorithms throughout the stack, so I have symmetric and asymmetric encryption algorithms.* [P13]

> *"We always think about, okay, we have those things; we have to keep in mind that the data will have to be encrypted. We don't access the data without encryption. If it's an external web-facing app, we have to have SSL."* [P24]

> *"All lot of us don't focus on major security stuff at the beginning of the project. But we may consider some stuff like encrypting passwords and not storing it directly in the DB [database]; using OAuth 2 has our authentication method or JWT. We don't want anything to be delayed, so we just deploy."* [P12]

**Adhering to Built-In Development Platform Security Warnings** to developers in this study meant that they needed to follow the security practices recommended by the development tools and platforms they use.

*"I respect the warnings that my tools give me. I know some of the things by heart, and I tend to avoid them, but most of these things, my own tools warn me and sometimes fix it for me, so I don't have to deal with too much stuff. Like if I try to put a username and password inside a source code, usually my application will warn me, but this type of thing, we tend to avoid doing it, but if we do, the tool already knows about it, so it tells us that we are doing something wrong. Then I learn it and then I stop depending too much on this type of warnings because I already know about it."* [P19]

*"The big guys like Apple or Google, or probably other security vendors, try to make the life easier for the developer; they want to control that part of the logic, and provide a packaged function to the developers, so that developers don't have to implement security on their own. So, in our daily app development life, normally, we don't have to think so much about security, especially as an IOS developer, and as a backend developer, as well. If I recall about [Ruby on Rails] development, you have to definitely think about it [security], but it's pretty much standard procedure you have to follow. For example, when you create a [class], you have to add this line – it's kind of a magical line, but it is what you have to do for security's sake. You have to just remember to add that part, without really understanding what is going to happen if you don't."* [P8]

*"AWS [Amazon Web Service] will give you, for example, a PCI [Payment Card Industry Standard] compliant environment. If I wanted to build a data centre that was PCI compliant I would have to do a lot of things, so at the hardware level, at some service level, AWS is already PCI certified, but what I put on top, it needs to be PCI certified as well, so just by using AWS you already have this assurance that you know that the machines are in a safe place, in a safe physical place."* [P13]

**Vulnerability Assessment** involves automated checks for software application and computer network weaknesses. However, participants did not describe how the identified vulnerabilities were being handled or managed.

> "*So basically when I do the dependency check and analyse which is part of our [duty] process, every time I make a commit, or I add a new library, if I make a commit to the project, we run the CVEs (Common Vulnerability & Exposure database), run this dependencies check and that dependency check actually checks up, through the CVE and it tells you if there is a vulnerability in the library I'm using. We also have like some things like a PIC test that tells you if your code itself acts like a security loophole, it's a quality check. PI mutation test, tells you if you have a memory leak.*" [P6]

**Third-Party Quality Assurance Testing Services** were described by participants as a final step after deployment of the first iteration of the app product. Third-party testing services are employed to carry out exploratory, regression, user or penetration testing on the app and then the feedback is implemented in the next iteration of the project. These testing services were used in projects where clients specifically requested for it or the lead developer is security aware, and then authorises the test to be performed to evaluate a version of the app product.

> *"…then we quickly deploy, but there will be a lot of flaws. Right. Uh, and then we have the security service from XYZ would then do a regression test and all these other things on the API and then submit us a report of what we need to fix. Um, so we don't do any major security checks from the start."* [P12]

> *"…we move it to the QA. QA does exploratory testing and then that goes live like continuous delivery."* [P6]

**Other Testing Options** were alternative security tests done by the participant themselves or a close colleague to cut cost.

> *"…now when are in testing mode, try to test with all kinds of people. Not just with the normal, random user. Test with a fellow developer. And then, if you have a hacker friend, test with the hacker friend as well."* [P3]

For **Risk Identification**, respondents did not explicitly refer to security risk. However, risk identification and assessments were done with the aim to protect the business.

> *"I would not accept any decisions the customer takes that breaks GDPR rules because I have the risk to be considered a violator myself as the software developer."* [P18]

**Identification of Security Threats and Attacks** were described by participants as implementing lessons learnt from personal experiences and not rarely from penetration testing or automated monitoring systems report. These personal experiences also informed future security decisions and were informally added to their personal knowledge bases.

> "*There was a project I worked on last year, this time last year. It was around April, May. It was a real-time application. While I was monitoring, I just checked the access log, and I noticed someone was accessing a folder I never created. I started wondering, "How did this person create this folder?" It was in my password. It wasn't secure enough. I traced the folder, and I noticed that folder didn't only exist in one directory. It existed in several directories on my server. So, I continued to trace. I noticed the user uploaded it … He uploaded the script and then he ran the script, and the script created these folders, and it created another script in those folders. So that was how he got access. And then he continued to go into my database.*
>
> *When I checked, I noticed, "OK, that happened because I didn't block script extensions." And the upload is meant to just be photos. It's just meant to accept images, PNG, JPEG and GIF images. But because I was trying to be as fast as possible with the project, I ignored that fact and just made it opened to any file extension. That was why I got attacked. So, from that kind of thing, next time, I will never, ever make that kind of mistake again*." [P3]

**Code Reviews** were described by participants as manual or automated activities examining their source codes with the goal of checking requirements specification implementation, fixing bugs, and identifying security vulnerabilities.

> *"We usually use Git as the repository to hold the code. The code review is done on the pages of the tool that we choose. Sometimes we use Bitbucket, sometimes use GitLab,*

*and we use the pull requests review tools from those systems to do the code review."* [P18]

*"Pair programming, we basically use some IDEs that enable us to pair program. And other code reviews are done via a call using Meet and we share a screen and debate about the code."* [P22]

Participants made references to **Policy Reviews and Implementation** which meant that they needed to be informed about existing security policies such as the GDPR (General Data Protection Regulation) for developers within the United Kingdom and Europe. Participants outside the U.K and Europe acknowledge the GDPR but do not necessarily apply its principles to their development process. However, it wasn't clear if any cybersecurity policies have been developed or documented internally within the organisations.

*"We also take policies like GDPR seriously. So, in the GDPR we are involved that way, it has always been much of data protection, majorly. Everybody has to be aware of that and go through training, which is always renewed, I think, every six months or one year."* [P6]

*"Definitely, network related stuff is more important, because there will be a lot more at stake there. A lot of apps are sending a lot of information from every user, so in the server, on the server side, there are a lot more information compared to what is residing on the phone. Yes, so definitely, you want to send and store the information selectively. You don't want to send ... Whatever information you get from the app to the server, you want to select only information that you need. Especially with GDPR around, you have to explain what you use the data for, so yes, that will be pretty much enforced to think that way, now and in the near future."* [P8]

**Security Training and Awareness** was a mix of formal, informal, and social learning for participants in this study. It also involved accessing with the knowledge bases of team members and other developers to stay informed about software and cyber security trends.

*"We don't do not have any security training or something like that. Mostly, when we face them some security issues, we meet as a team, uh, it is our responsibilities and,*

*share information to see how we can do things in the better way. So, we don't have any, uh, training, but we, we try to share out the information and tools."* [P16]

*"When we are junior developers, we are kind of taught by the senior guys. They say, you have to do that, or you don't do this. I never got training on security or on software security, but when we are working it seems there are some kinds of practices that are generally accepted practices."* [P18]

Security practices are made up of activities conducted recurrently with the goal of producing secured products. Although, some security activities were done by the participants themselves, some are executed by the other team members such as the Infrastructure Engineer and third-party services in varying capacities.

### 5.4.7 Resulting Products – Secured vs. Unsecured Products

The theory has multiple instances that can produce two predictable outcomes, *Secured* or *Unsecured /Vulnerable* app product.

Many software development processes happen within *Organisational Ecosystems* where *Developers* supply products to *Clients* or directly to *End-users*. For developers particularly in SMEs and micro-business, the main goal is to deploy a decent product in good time, secured or not. However, the interplay between the stakeholders, social dynamics, and the different levels of expertise of individuals involved in the development process within a specific context can result in the prioritisation or deprioritisation of security *best* practices. The deprioritisation or prioritisation of security *best* practices then results in the production of insecure or secure software application products. For an individual to decide about prioritisation, there are several factors that contribute to the decision-making process. Likewise, for developers in SMEs and micro-business to consider prioritising security practices, various factors contribute to their decision-making process. The social dynamic relationships between people (developers, clients, end-users) and these factors present the predictable outcomes of their app products.

**i. Secured Software Products**

*Secured* apps products are as a result of a combination of finding ways to introduce, fix or enhance the security of the software application. A secured app can be achieved when developer understands the context in which the app is being built and used and can find ways from available security knowledge bases to implement standard security practices with usable tools that meet *Client* and *End-user* expectations. They were instances where participants described some client's expectations did not necessarily involving presenting a *Secured* app product – just "an app that works". However, *People* generally understand that the vulnerability in software application products and the damage that an attack can cause for organisations and individuals solidifies the importance having secured app products.

> *"Unconsciously implementing standard security practices because we've been doing it for a couple of years enables us build apps that are secure to some level."* [P21]

**ii. Unsecured or Vulnerable Software Products**

When developers are not well supported during the development process, particularly those in SMEs and micro-businesses, they struggle with factors that contribute to weak security practices. Participants described factors such as limited resources, project constraints, limited access to usable security technology even with a rich knowledge base as well as other socio-technical issues in affecting their security activities. The factors represent the various reasons for many developers to focus more features and functionality with little to no attention to security from the onset resulting in *Unsecured* or *Vulnerable* app products.

> *"Security is usually an afterthought for us. With the hope that no one would attack your server because you are a 'small developer' in a small or independent company."* [P5]

For some participants an app is to an extent safe from cyber-attacks when they avoid using public generic codes that they consider unsafe during the app development process. While for others, security measures are considered a choice depending on the type of client requesting for the app, the size of their users or the type of information gathered from their *knowledge Bases* (communities). However, developers need to prioritise and be proactive about app

security beyond the basics to prevent the unavoidable cost of damages that individuals and businesses can incur from attacks.

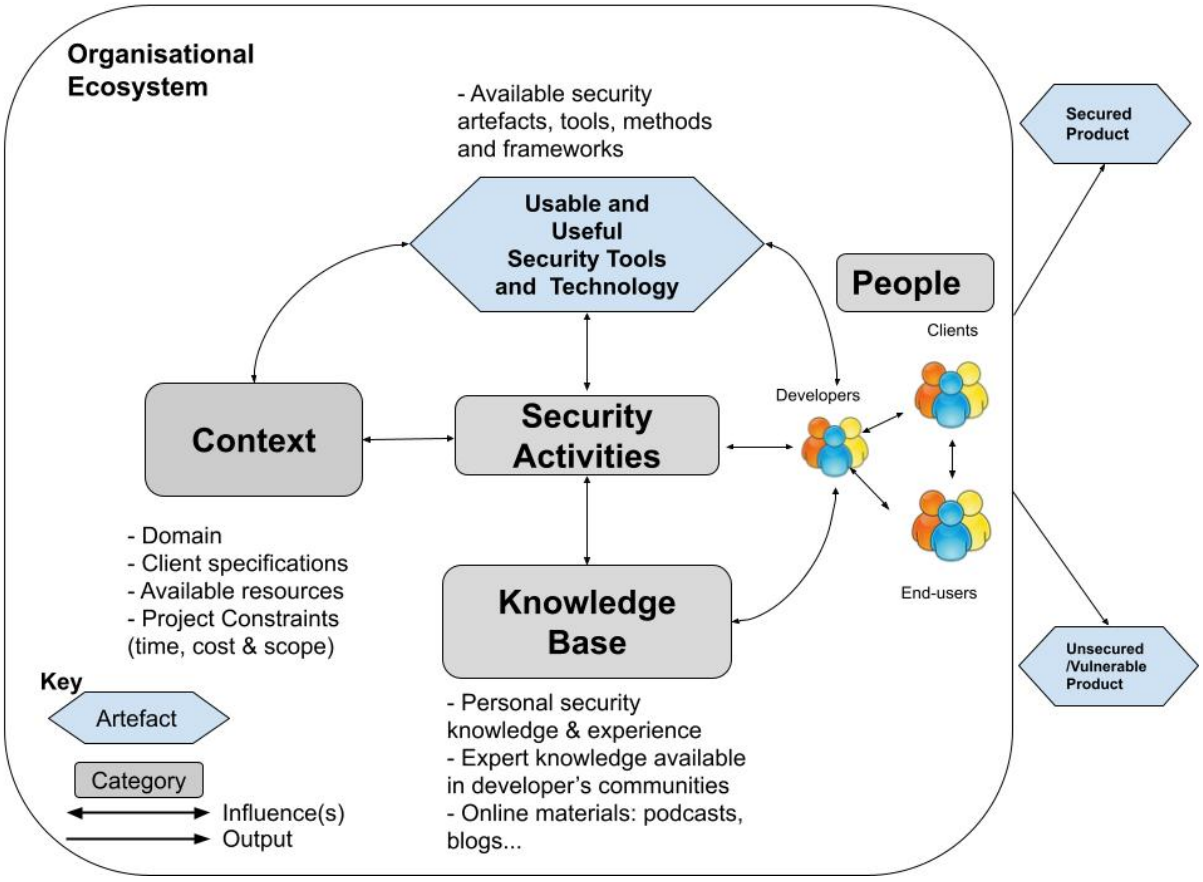



Figure 5.6– Outcomes of Ecosystem: Secure, Vulnerable and Unsecure Apps

## 5.5 Multiple Instances of the Theory

From communicating ideas to scoping and understanding the project specifications, and then identifying the technologies and expertise required in delivering an app product; the theory of the *Socio-technical Security Practices of Developers in SMEs* is not a single sequence of events. Individuals (developers, clients, and end-users) participate in various combination of interactions between elements of the theory make-up several instances of that result in the delivery of an app product. For example, **Figure 5.6** shows an instance when all elements are available, accessible, and work in the favour of the *People* in this ecosystem. That is, the

context of the app is clearly defined, there is limitless resources projects constraints are eased off, there is a wealth of security knowledge in the knowledge bases, usable and useful security technology is readily available, and the individuals are all knowledgeable about the different areas of security that pertains to the app project. Then, the choice is left to them to implement security practices and activities that will result in a secured software or do otherwise and cause the software to be vulnerable or less secure. The following sections below present other instances of the theory that describe the experiences of participants in this study.

### 5.5.1 Instance 1

**Figure 5.7**. present one example of an instance of the theory. In this instance, the client has no knowledge of software security, cyber security, or the implications of it on their final products, they just want an app that is functional and has all the features they need for their end-users. The developer(s) in this instance may have some basic security knowledge but limited or no access to usable and useful security technology and a rich security knowledge base. Other developers within his communities and team rarely discuss or share security tips, guidelines, or recommendations.

The context in which the app will be used has been clearly defined, for example, it can be a lifestyle and entertainment apps. A project plan has been drawn and constraints highlighted. Additionally, developer(s) have identified the available human and financial resources they require to get the job done. Without including standard security activities within this project, the outcome will be a vulnerable or unsecured app product that is delivered to the 'unknowing' client and then deployed to end-users. Some of the stories described by participants suggests this type of interaction is a common occurrence within software SMEs. Participant 1 gives an insight into how this instance of the theory may occur.

> *"So, say for example, in some of our projects, where we charge per hour or something, "Oh, definitely, why not? It's more money for the business." But if it is my own cost going in, I might allow changes once or twice. But definitely I would frown at it further down the line. So, if we're maybe wrapping up the application,*

*we've done maybe the first stage of testing and then you can't know if you don't know how that feels, but you want it to be done in a different way, one approach would be, "OK, you know what, let's finish what we're working on. Let's pack it as something we'll probably look at as maybe another sprint or another feature, another release." That's kind of the plan."* [P1]

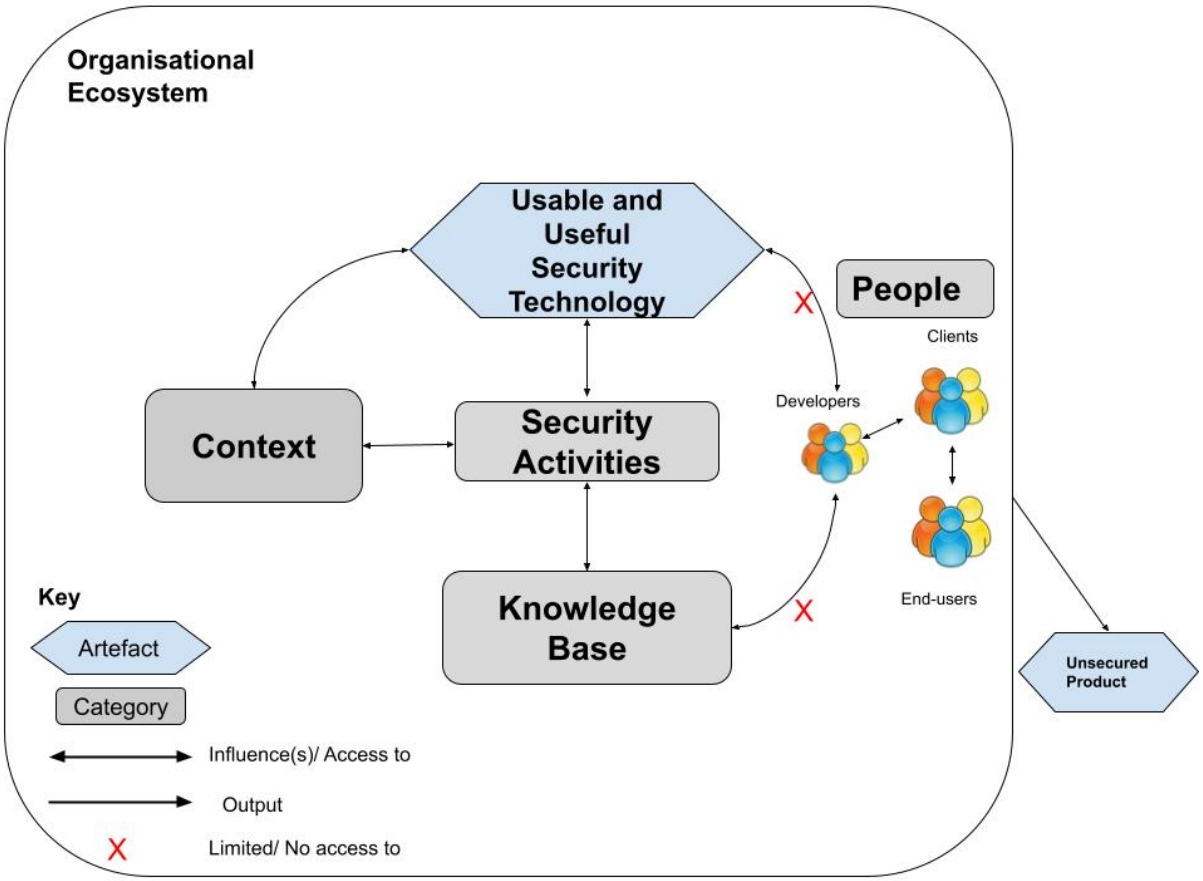



Figure 5.7– An Instance of the Theory Leading to an Unsecured App Product [P1]

### 5.5.2 Instance 2

Another instance coined from participant's experiences potentially produces a more secure software app product. Participant P13 describes their experience and stance in achieving a secured app product and **Figure 5.8** shows a diagrammatic representation of the app product outcome.

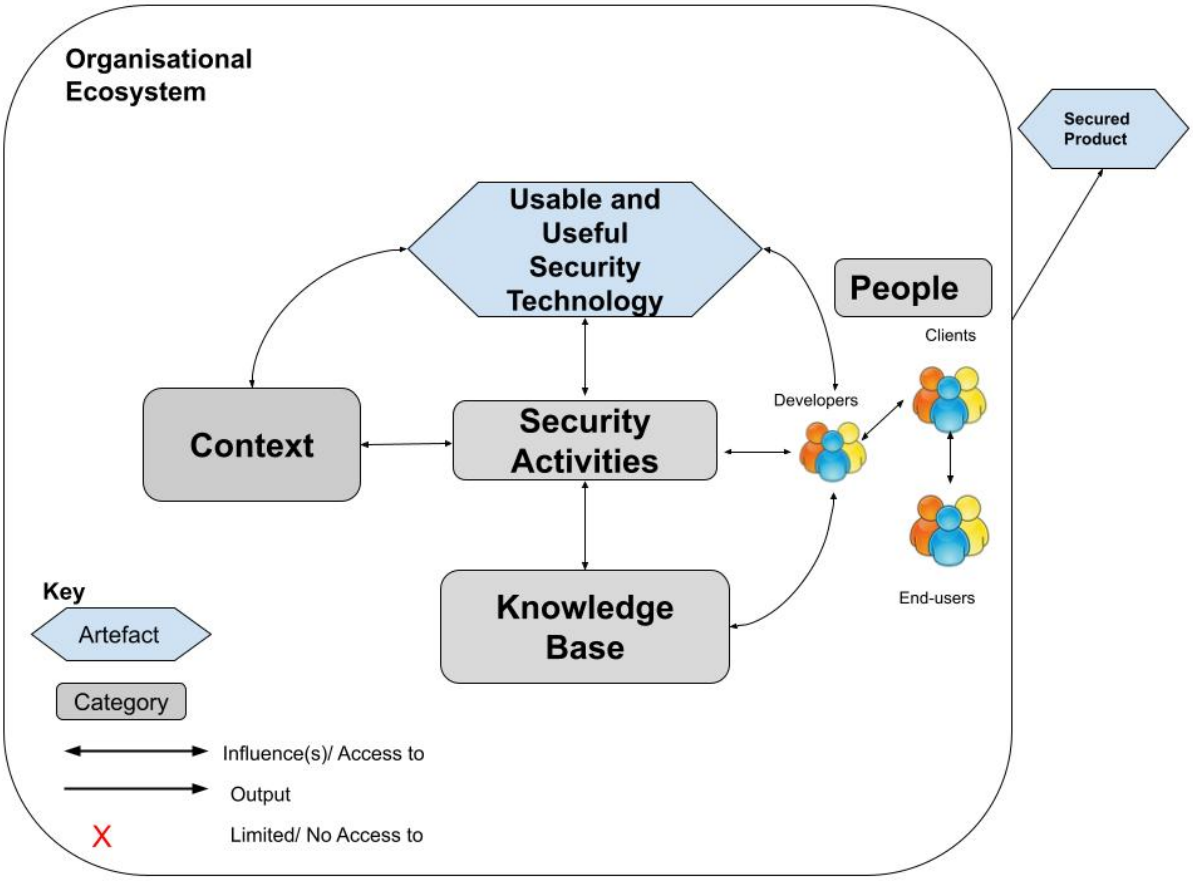



Figure 5.8– An Instance of the Theory Leading to a Secured App Product [P13]

*"Once I've got a client that wants work with me, the first step is to make sure one way or another, I mean from our side or their side, that they know clearly what they want us to build. That they are either domain experts, or he's passionate about what they want to change or do. If they're doing a solution that stores credit cards, we need to think about a lot more, but if we're developing a solution to find the best bus schedules the requirements are a lot less. What determines the security consideration for the app is what kind of data we will be storing.*

*We're not a cyber security company. We have one infrastructure engineer, he already worked out some security scenarios so, he'll know which to go for. That's the architecture we use. Then we start from the basic, like having strong passwords. Strong passwords don't write down those passwords on any system. Then use of different encryption algorithms throughout the stack, so I have symmetric and*

> *asymmetric encryption algorithms, we can apply well or not so well to different scenarios.*
>
> *To me it's a matter of philosophy and how you think. I think systems should be secure anyhow because what you don't want is data leaks that will then destroy the product, then the company."* [P13]

In this instance, the developers have sufficient knowledge of software security and cyber security measures. They also work with clients that are clear about their software requirements and security specifications. The domain of app use is defined to identify the type of data that will be collected and stored, in order to make specific security decisions. In addition, the developers have access to existing usable security technology, have the funds to acquire and know how to implement them. Although, the infrastructure engineer in this case plays the role of a security expert, P13 still describes how the team implements encryption algorithms and vulnerable assessments as part of their security activity during the development process. Further discussions with P13 showed that the team create security knowledge that contribute to their knowledge base. It is important to note that even though developers in this instance may tick all the boxes and follow security best practices, the final app product can still be attacked as hackers are constantly finding new ways to infiltrate systems.

### 5.5.3 Instance 3

In this instance an unsecured app was delivered, however the reports about security issues for the delivered product were sent back to the software company for review and upgrades. Then security measures are then taken, and a more secured app is re-delivered to the client. The cycle in this instance can be likened to that of sprints within an agile development process. Some participants explained that majority of the time, they consider implementing security measures at the end of the project or by the second sprint in their agile app development process.

In **Figure 5.9** we see a different interaction between elements of the theory. Developers have limited to no access to some level of security knowledge or tool and therefore failed to

include certain *Security Activities* or procedures in their development process. The client, however, has access to some security knowledge and tools that were used to scan the app product that was delivered. Although, we may expect that the client would have presented the development team with some clear security requirements at the initial phase of the development process to ensure that the final product is less vulnerable. But further discussions with this participant P4 did not address this and so it is unclear whether the client assumed that the developers would know what security tools to use, or the client just wanted a functioning app product delivered fast.

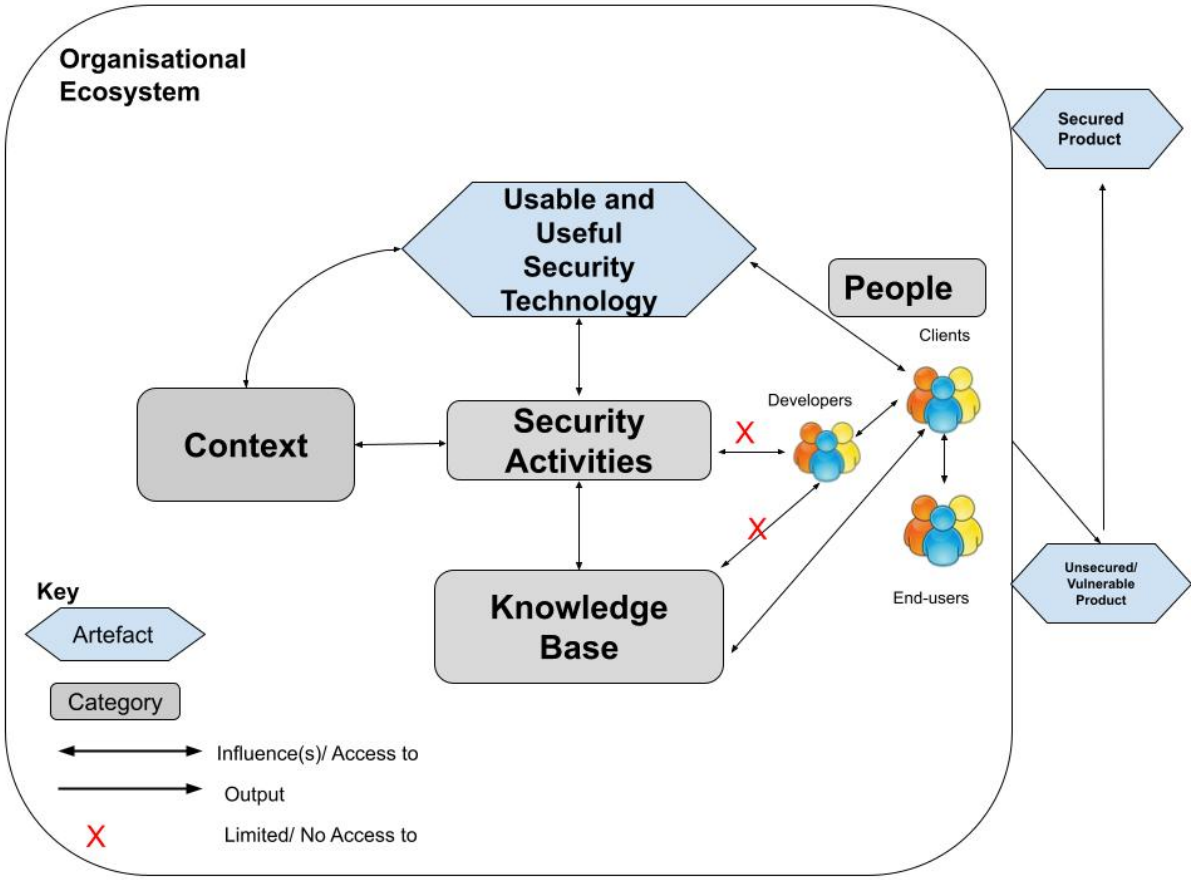



Figure 5.9– An Instance of the Theory Leading to an Unsecured App Product [P4]

*"For example, about two years ago, not exactly two years, just last year, (seems like this happens every year) some of our clients conduct security scans on their network and their internal apps. And every time they conduct the scans, they send us an update report of things that we need to fix and things we need to do, and we send the fixed software back. Usually, what I realised a while back, is that most of the time for our software, we don't have any high-risk stuff. Maybe medium to low-risk information that can be flagged. And, like I said, the reason is because we don't build from scratch; we rely on frameworks.*

*Yes, so I'm just trying to say that I'm aware of some standard security not because we follow it, but because when clients use their security scanning test tools, and they send feedback to us, we don't end up being flagged because usually it is not a lot of high-risk information. Obviously, if they are a lot, then our position with our clients comes under question. So, we don't get flagged. So, you may ask, why is the company that appears not to follow any security protocol not getting flagged for security issues? The answer to that the company understands that nobody here is a security expert, so we do the best we can as developers, and mitigate risk by making sure that we don't build anything from scratch, relying on platform security. That means we don't bear the responsibility for everything; it is shared with the platforms we use."* [P4]

For most of the study participants, security implementation, adoption or even practice, are centred around social interactions. These interactions are with clients, team members or other practitioners within their communities. Combined with personal experiences, the outcome of these social interactions encourages security adoption, implementation, and knowledge acquisitions.

## 5.6 From Theory to Practice: a theory evaluation approach

For a computer science dissertation, it is important that whatever is learnt can be applied to practical real-world situations of software developers. This study started with an overarching aim of understanding how developers in SMEs practice security effectively within their organisational context with the intention of informing the development of guidelines and recommendations that support secure software application development. Secure software

development does not mean that app products will completely be without vulnerabilities or be free of cyber threats. However, the elements of the theory of the *Socio-technical Security Practices of Developers in SMEs* exposes the social dynamics and negotiations that developers in micro-businesses, small and medium-size organisation navigate to ensure that they include security activities in the development process and deliver secure functioning software product.

Evaluating the emergent theory and demonstrates an application of to the real-world context using Charmaz's recommended criteria for evaluation. As described earlier in **Chapter 3**, Charmaz's recommendations for evaluating a constructivist grounded theory are based on the criteria of credibility, originality, resonance, and usefulness (Charmaz 2006). Feedback from follow-up interviews with participating developers is presented to check that the theory resonates with them and is useful, in that it supports their secure software development experiences and activities.

## 5.7 Credibility and Originality of Theory and Study

Evaluating grounded theory methodology in computer science, software engineering or cyber security disciplines can be difficult to deal with because of the expectations and reservations other researchers may have about qualitative studies. They impose dominant quantitative tenets on evaluating the quality of a qualitative research (Charmaz and Thornberg 2021). This study is backed by the argument that qualitative studies should be evaluated according to the principles of qualitative research traditions (Charmaz and Thornberg 2021; Glaser and Strauss 2017).

Evaluating credibility and originality in grounded theory from a constructivist point-of-view is subject to how much the researcher interprets and understands the actions and meanings of participants in their social, local, and interactional context (Charmaz and Thornberg 2021). Charmaz recommends that to evaluate the credibility and originality of a theory, it must sufficiently represent the data and extend, challenge, or refine current ideas, concepts or practices (Charmaz 2006).

The theory of the *Socio-technical Security Practices of Developers in SMEs* is specific to the social and organisational context of developers in small and medium-sized companies and its elements are familiar to practitioners and researchers in this subject area. A recap of the theoretical statement below shows that although the categories independently may not seem like novel insights, but together they describe the developers' perspective. The theory is original because it provides new conceptualisation of the secure software development problem for developers in SMEs and highlights various areas for future research.

> *For developers in SMEs to effectively practice (implement best practices) secure software app development to produce (deliver) a secured product, they need to consider the* ***context*** *in which this product will be used, gather information from their individual or group security experience* ***knowledge base****, clearly identify the* ***people*** *involved in the process (clients, end-users, and other team members) and have access to* ***usable and useful security tools and technologies****.*

*Context*, *People*, *Knowledge Bases*, and *Usable and Useful Security Tools and Technologies* are central to the continuous implementation of security activities which then become practices. These conceptual categories persuade researchers in software engineering and cyber security disciplines to go beyond investigating technical or behavioural aspects of cyber security and secure software development but considering other social aspects of the practices.

To further evaluate the theory, researchers must employ strategies that reveal more about their participants' experiences to participants and other researchers. In doing this, the theory resonates with and is useful to participants and the research community. The next section present an approach to gaining resonance and showing the usefulness of the theory in the activities of software application development and new areas of research.

## 5.8 Does the Theory Resonates with Developers in SMEs?

Additionally, grounded theorists suggest that the emergent theory should be evaluated based on it usefulness and resonance to the participants and offer them insights about their

experiences as developers in SMEs. The theory should also offer interpretations that are applicable in daily activities, contributing to knowledge and sparking further research (Charmaz 2006; Sikolia et al. 2013).

### 5.8.1 An Approach to Evaluating Resonance and Usefulness of Theory

To evaluate the theory of the *Socio-technical Security Practices of Developers in SMEs* and the overall value of this research, follow-up semi-structured interviews were done with 7 participants from previous pool of participants (P1*, P2*, P5*, P12*, P13*, P18*, and P22*), who were interested in getting an update on the outcomes of the study. Although the initial plan was to organise workshops with developers and security professionals for a more robust evaluation of the theory, plans were changed due to the impact of the global pandemic. Returning participants agreed to do virtual follow-up interviews. Materials used for the follow-up interviews are presented in **Appendix C.2**.

The interview sessions were divided into two parts for about 20 to 30 minutes each. In the first part of the session, participants were presented with multiple instances of the theory presented as scenarios to get feedback about how the theory applies to their daily activities. Scenarios and personas are used to present situations for solutions or identify alternative solutions or further problems (Bødker 2000). The interview session started with a description of the elements of the theory and then participants were presented personas and scenarios based on the instances of the theory to prompt discussions about how these elements resonates with their experiences. An example of the persona and scenario is shown below:

**Personas:** John is a 29-year-old employee at a digital health care start-up company and a freelance developer. He is a self-taught developer with no formal education in software engineering or cyber security practices. He is an iOS mobile app developer with eight years of experience.

**Scenario:** John recently joined the digital health care company less than three weeks ago. The company wants to develop an app to track patient's heart rates and other vital signs, alert medical personnel of possible high-risk activities of patients and report patient's behaviours. John recently got employed in this company to be a leader of a team of three developers for

the project. The CEO hopes that the app can be launched and released to users (private consultants and patients) in three months. Resources are limited as the company is in its first round of funding from investors.

With the scenario and persona example described above, the goal of the first part of the interview session was to evaluate if the theory resonates with participants. The interviews involved participants describing how the persona, 'John', would handle an app project considering the *context* in which the app will be made and used; the *tools and technologies* needed for the project; the type of *security activities* that may be introduced and how they would be introduced; and specifying the *knowledge bases* they would use to support their decisions.

> "*This is a typical situation for most developers including myself! Funny enough, I am currently in a similar situation for new mobile app project not health related though. However, this client has some knowledge about security, so we factor in discussions about app security into the process*." [P2*]

> *"Yes, the description of the theory and this scenario resonate with me. One major thing I would say about the situation is that developers such as John need to have had previous experience with a similar app. It will help push thoughts about the security activities early."* [P5*]

### 5.8.2 Lesson Learnt from the Theory Evaluation Process

Security issues, consequences, and cost of implementing specific security measures can be assessed and reflected on before implementation. It can also promote consistent involvement of (clients and probably end-users) throughout the secure software development lifecycle.

However, there are limitations associated with integrating new steps into an already time-constrained agile development environment:

i. The need may arise for product launch timelines to be extended if issues are identified during the prototyping and security tools and technology selection process.

Even though the processes encourage developers to stop and think about security, extended timelines can discourage its use.

ii. The analysis of data gathered during this evaluation process may be limited and so doesn't give a clear enough picture of developers' perspective of the processes, even though participants found it useful.

iii. Not evaluating the theory with new participants may also limit the scope, however it shows that the theory explains the experiences of the people within context.

iv. Finally, the theory was considered with the agile development methodology in mind This also limits the scope of application to other methodologies since developers in SMEs may be using a combination of various software development methodologies.

## 5.9 Comparison with Extant Theories

Theories of practice have their foundations rooted in philosophy and have developed across various disciplines such as organisation studies, anthropology, science, and technology studies, with no unified perspective. The theoretical perspectives of practice are relatively new to cyber security and security-related research.

Social practice theory opens with the premise that "people participate in multiple and variable social contexts" (Penuel et al. 2016) and thus offers a suitable perspective for explaining the complex social dynamics that developers experience daily and the impact it has on secure software development practices. Practices such as implementing security measures in software development depend on specific combinations of materials (*Usable Security Tools and Technologies*), shared meanings and competencies from the knowledge bases of people (*Developers and Clients*) in their specific context. Secure software development may evolve as the combination of these elements change over time. This study draws upon the definitions of social practices by (Shove, Pantzar, and Watson 2014; Reckwitz 2002) as a source of understanding the theory of the *Socio-technical Security Practices of Developers in SMEs*. Social practice theory positions this study in literature to explain the actions of participants (developers) paying attention to their ongoing struggles across the various activities involved in producing not just a functioning software product but one that is secure.

For developers in this study, the act of implementing security measures in app development is a practice that supports secure software development process. The theory of the *Socio-technical Security Practices of Developers in SMEs* explains the process of how developers in small or medium-sized organisations to effectively practice secure software development through social interactions, gathering information, identifying the people and tools they need to get the job done. The theory is built around the concept of a combination of various activities and actions that represent the developer's perspective of how security can be practiced effectively within their organisational context. Also, organisational structures such as SMEs are constantly changing and growing which in turn, shapes the security decisions and practices of developers.

Social practice theory concepts can be used as a base to explain secure software development as a social practice for developers, particularly for developers in micro-businesses and small and medium-size organisations. This makes an argument for the actions and activities of developers based on insights from the philosophy of sociology, to help explain the phenomenon of secure software development. And use Social Practice Theory as a base for explaining my theory. The theory of the *Socio-technical Security Practices of Developers in SMEs* provides conceptual base for understanding cyber security practices in software app development as social and epistemic practices within their social and organisational contexts. This contributes to the cyber security and software engineering bodies of knowledge.

The next section explains one of the core constructs of the theory – *Security Activities* – through the lens of social practice theory, as an example of how materials, meanings and competencies operate interpedently at an organisational level for developers in SMEs.

### 5.9.1 Security Activities Through the Lens of Social Practice

Shove et al. argument for practices, is that there are interdependent relationships between materials, meaning and competencies with the ability to mutually shape each other (Shove, Pantzar, and Watson 2014). This argument is particularly valid in cyber security and security activities involved in app development because materials such as the security tools and technologies used can shape the competences of individuals involved in the process, and the

shared meanings and ideas developers have about security. These elements are mutual shape and depend on each other for developers to be able to effectively implement cyber security practices. The concept of mutual shaping is frequently used in Science and Technology Studies to explain the mutually exclusive relationship between society and technology, that they shape and influence each other (Cole et al. 1995).

If we use **Vulnerability Assessment** as an example of one of the *Security Activities* described by participants in this study. The process of identifying, reviewing, quantifying and prioritising threats, weaknesses and vulnerability in the software product or computer networks can be broken down into the different required competencies of developers, clients and materials used. Although participants describe this process to be majorly automated, the decisions about what vulnerability assessment tools to use rests on the competency of the developer and sometimes, the client. Typically, the process can start with the developers' (with security knowledge and skills) review and understanding of security initiatives such as the OWASP Top 10 list of the most common and recent application vulnerabilities (OWASP 2021) or the client may request for an overall security analysis of their application. The developer then decides to run dynamic or static application security testing (penetration testing) for these app products based on their own understanding or recommendations from other developers in their communities. **Figure 5.10** shows an example of interactions for vulnerability assessment of an app product.

Vulnerability assessments and test activities can be done continuously as a practice involving interactions between developers, clients and other stakeholders who can participate in process. Finally, having a social practice of vulnerability assessment can support the cyber security practices of app developers in SMEs and this will be more helpful for developers in micro-businesses and SMEs with limited resources to practice secure software development.

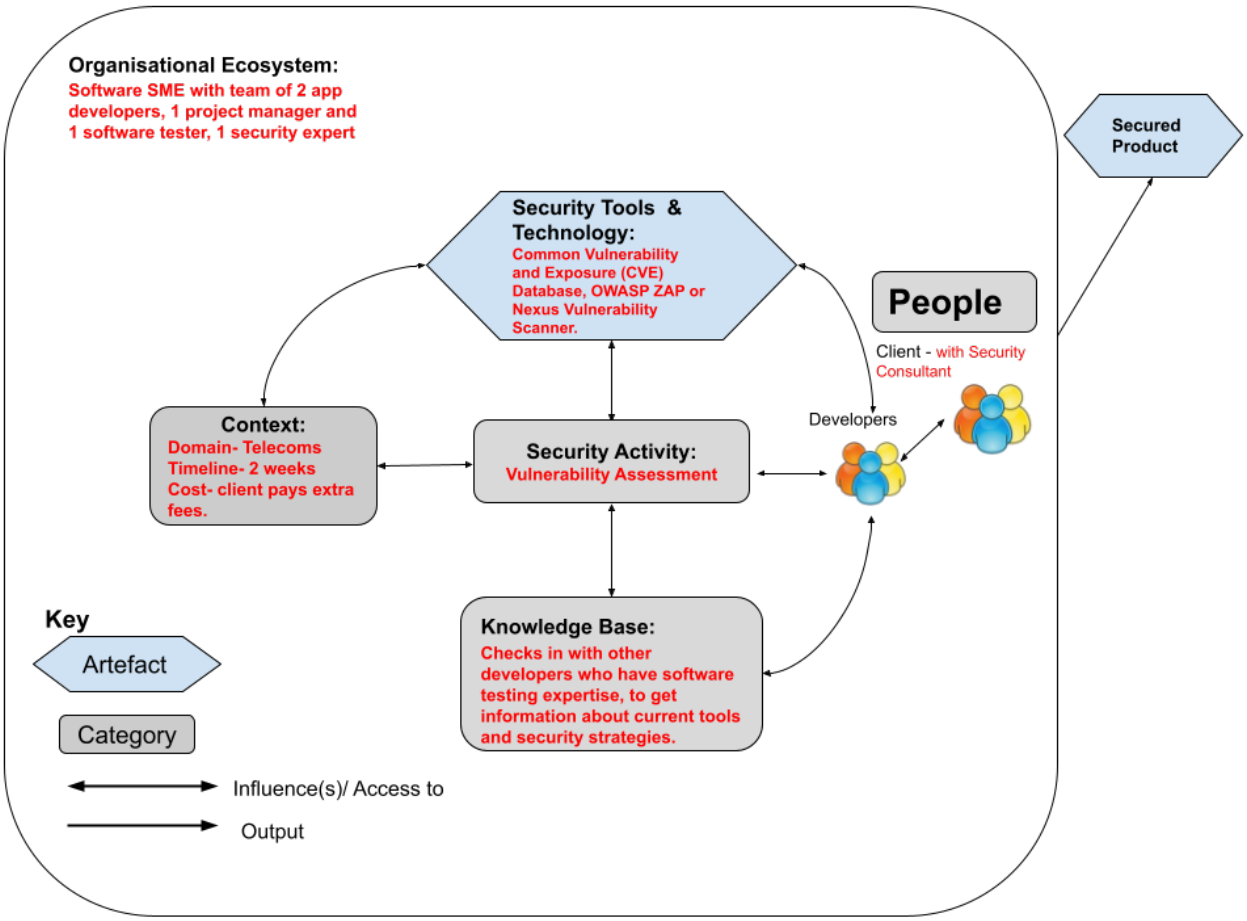



Figure 5.10– Vulnerability Assessment Process as a Social Practice [P1*]

### 5.9.2 Social Practice Theory and Secure Software Development

Based on the knowledge gained from this study, software security is considered a social practice from the perspective of Social Practice Theory, as a way of improving security activities and processes. This is because it helps consider the various connections between elements of security practices in an organisation and not just focus on behaviour change. The goal of Social Practice Theory is to move beyond focusing on ways to understand and influence the behaviours of individuals to understanding the dynamics between individuals and other elements that form a practice (Shove, Pantzar, and Watson 2014). Studying software security as a social practice enables us to ask more robust questions whose answers

can push the agenda for building more secure software products. Some examples of questions in this context would be:

i. Do the developers have the skills and competencies to perform the security practice?
ii. What impact do the materials being used have on developers' behaviours? Do materials inhibit or drive certain types of behaviours?
iii. What security practices are the most important? Do developers need new practices for secure software development, or can they leverage on existing materials?
iv. Is there a shared meaning amongst developers and security experts about performing certain security practices?
v. What constitute best practices in real-life for developers? Do they consider security practices as an inherent part of their jobs as developers? How do we think of competencies and materials that can help developers transition?
vi. Within the organisational context of SMEs, how do developers localise security practices that can be easily adopted by in their agile environments?
vii. What impact does the technology built have on society? Are developers being responsible about their innovations considering future impacts?

In practice-based fields such as security and software engineering, individuals are the carriers, performers, and sustainers of the practice through continuous performance (King 2019). Practitioners therefore must deal with constrains on available materials or the need to continuously learn about new security and software tools, technologies, or techniques. Furthermore, the range of competencies or skills needed to perform a practice can be from basic to expert level and individuals still need to engage at different levels.

In summary, the social practice theory may be limited in providing grounds for technical arguments, however, it supports our arguments of the importance of understanding the experiences of individuals - developers in SMEs - and how their social interactions and differences in security meanings influence their practices. In addition to understanding social practices, it is also imperative to understand the relationship between social and technical dimensions within an organisational context to further expand our knowledge of the conceptual categories of the theory.

Secure software development as a socio-technical practice is dynamic, constantly changing the entities and elements that engage in it. Therefore, to achieve more success in interventions and initiatives for secure software development, it is important to consider all other elements beyond isolated individual behaviours. In the next section, a well-established and frequently used Socio-Technical Systems theory is used to explain the interdependent relationship between the elements of my theory within the SME organisational context.

### 5.9.3 Security Practices and Socio-technical Systems Theory

The concept of socio-technical systems was first created by Emery and Trist (2005) to explain the synergy between the complexities of technology and social systems (Trist and Emery 2005; Trist 1981). Social systems in this context represent organisations that are made up of groups of individuals with some organisational arrangement, carrying out various work practices that achieve a purpose (Trist 1981). Whilst the use of tools, technology and the execution of other engineering activities represented the technical aspect of these systems (Trist 1981; Trist and Emery 2005). The Socio-technical Systems (STS) theory describes a synergistic relationship between "people, technology, organisational structures and processes, and the operating environment in which they occur" (Carayon et al. 2015; Malatji, Von Solms, and Marnewick 2019).

Perhaps a more appropriate definition and holistic view of the theory is one provided by Troyer (2016), that STS describes an interaction between the social aspects of development (including how teams and individual members perform tasks), and the technical systems that include other complex relationship throughout the development lifecycle (Troyer 2016). In this regard, recommendations for improving both social and technical aspects of an organisational system is key in achieving effective performance (Trist and Emery 2015; Miner 2020). This explains the relationship between core categories of the theory of the *Socio-technical Security Practices of Developers in SMEs*.

Researchers across various software engineering and cyber security disciplines have used the core elements of STS theory to analyse human, social and organisational aspects of the development lifecycle. Elements of the social dimension include organisational structures and

people (including members of an organisation and stakeholders); and the technical dimension constituting of technology, tools, and resources to carry out work activities in specific social infrastructures (Hester 2014). Kraemer and others have argued that human and organisational factors play a significant role in the development and analysis of security vulnerabilities with results from qualitative studies (Kraemer, Carayon, and Clem 2009; Dhillon and Backhouse 2001; Stedmon et al. 2016). Studying cyber security and software development practices through a socio-technical lens shows the interplay between technology and human and social dynamics (Fuggetta & Di Nitto 2014).

In order to understand how developers' practice cyber security effectively in organisational structures such as SMEs, we must understand the meanings of their development experiences and the social contexts in which they occur (Ralph, Chiasson, and Kelley 2016; Safa, Von Solms, and Futcher 2016). We cannot focus only on tools, systems performance, and software security while the social aspects are ignored. This study of cyber security as a social practice shows that technological and social elements should not be studied in isolation. It offers an integration of human aspects and technology to define the dependencies between the security activities of developers and their social context. Cyber security and software engineering are often considered a formal, technical domain, with security research focusing mainly on computer and information security infrastructures, however, security activities require social acceptance and compliance as well as the availability of usable and useful security technology (Albrechtsen 2007; Kraemer, Carayon, and Clem 2009; Casaca and Florentino 2014).

Studying security and software application development practices through a socio-technical lens shows the interplay between technology, humans, and social dynamics (Fuggetta and Di Nitto 2014). A research study that uses a technological approach tends to focus mainly on tools, systems performance and software security which is an approach used by most researchers in security-related studies such as cryptography, threat models, vulnerabilities, firewalls, multi-factor authentication, cloud, and network security. Research studies using the social approach have explored end-user security behaviours and factors that influence these behaviours, within their personal environment (Stanton et al. 2005; Albrechtsen 2007) or in an organisation (Abraham 2011; Balebako et al. 2014; Hasna and Mustapha 2016).

Regardless of the strength of access control policies or technical controls, if humans do not implement and use them appropriately, because of the possible influence of various human or organisational factors, the effect on security can be severe (Kraemer, Carayon, and Clem 2009). Security cannot be fully achieved through technology growth only; it must take into full consideration the experiences of people. Theories in secure software development research such as the one presented in this study, requires both technical and social approaches to ensure that security goals within organisations and disciplines are achieved (Dhillon and Backhouse 2001). Therefore, we must go beyond just building secured systems and tackling network threats, attacks, and vulnerabilities, to ensuring that other aspects of secure software development practices are better understood and addressed (Schlienger and Teufel 2002).

## 5.10 Usable and Useful Security Tools for Secure Software Development in SMEs

Several security-related research studies have integrated social and technological approaches to examine how end-users interact with and use complex of security tools and technologies. For example, studies by (Sasse and Flechais 2005) and (Joorabchi, Mesbah, and Kruchten 2013) have made efforts to better understanding of how people manage, understand, and use security tools within their specific context and environment. Other studies have attempted to analyse how developers The concept of usability as defined by ISO standards is the “extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use.” (ISO/IEC 2014) Usefulness, means that the product or tool creates value for the user by meeting the specific needs (Molich, Jeffries, and Dumas 2007).

The usability and usefulness of security tools and technologies are now basic determinants of the acceptance, adoption or and value of security measures in app development. It is critical that the research community pays more attention to understanding how developers use security tools and manage complexities. It is not beneficial or effective if developers struggle with use complex security initiatives, techniques, tools, or technologies, that may be classified as requirements for ‘best’ security practices. For example, if a developer in a small

software company with a few years of experience and no formal security training, encounters a complex security tools or framework, they are most likely to deprioritise security measures to focus on features and functionality. It is not always a case of a lack of motivation, for complex security tools and technology are simply unusable or require certain high-level expertise to implement or extended time to learn which can be discouraging for the developer.

According to the participants in this study, security knowledge is only useful when you have easy-to-use security tools, technology, or frameworks available for implementation during and after app development process. This means that even when developers are aware of what security measures to implement, the security tools or technology need to be usable and useful for developers to effectively carry out their work activities, particularly security activities. Just like end-users, developers need complex security tools, frameworks, and technologies to be easy-to-use, understandable, and useful to produce secure software. Furthermore, insights from the Socio-technical systems theory supports my claims for the theory that developers need access to usable and useful security tools and technology within their specific organisational context.

Defining project characteristics is common across various sectors and organisations including software companies, to predict and manage project challenges (Kumar et al. 2012). For smaller software companies, such as where the participants in this study worked, clearly defined project characteristics was particularly important if limited resources were to be distributed properly to complete a project. This includes but is not limited to the availability of resources needed for the project, clear functional and non-functional software requirements, the overall cost of the project, and time constraints set by clients. Software project characteristics estimation and management may not be novel concepts in software engineering research; however, studies on secure software development have focused on large projects and large organisations (Jones 2005; Chemuturi 2009) leaving the activities of developers in SMEs at a disadvantage.

# Chapter 6 A Framework of Secure Software Socio-Technical Practices for Developers in SMEs

## 6.1 Introduction

This chapter present the results of conducting an in-depth literature review with a more systematic approach to develop a conceptual framework, in addition to the grounded theory process presented in **Chapters 4 and 5**. The aim is to situate this study within the research landscape and strengthen the interpretation of conceptual categories presented in the findings, by systematically reviewing relevant existing literature and theories in the field and comparing data. This presents further discussions and insights into the key issues of secure software development practices of developers in SMEs.

## 6.2 Approach

"The logic of grounded theory differs from quantitative research logic that applies preconceived categories or codes to the data" from the inception of the study (Charmaz 2006). Charmaz argues that using a framework from the start, restricts how researchers learn about the actions, perceptions, and situations of participants within their settings (Charmaz 2006; Bryant and Charmaz 2010). However, to demonstrate an understanding of the state-of-the-art in literature related to secure software development, a conceptual framework analysis study was done to synthesise findings presented in **Chapter 4 and 5,** probe the data further in comparison to existing literature and to situate this study in the security and software engineering bodies of knowledge.

A systematic approach adopted at this stage of the study is presented visually in **Figure 6.1**. The process follows an adaptation and implementation of procedures for building a conceptual framework by (Jabareen 2009) and methods for rigorously reviewing literature using grounded theory by (Wolfswinkel, Furtmueller, and Wilderom 2013). and applied to publications in information systems field. search and review state-of-the-art literature in the secure software application development domain. This approach aims to review relevant related literature within the secure software development domain, compare findings in this study to construct relationships between any concepts and discuss the wider implications. In addition, the conceptual framework is proposed to support studies in secure software development and developers' experience in the SME context.

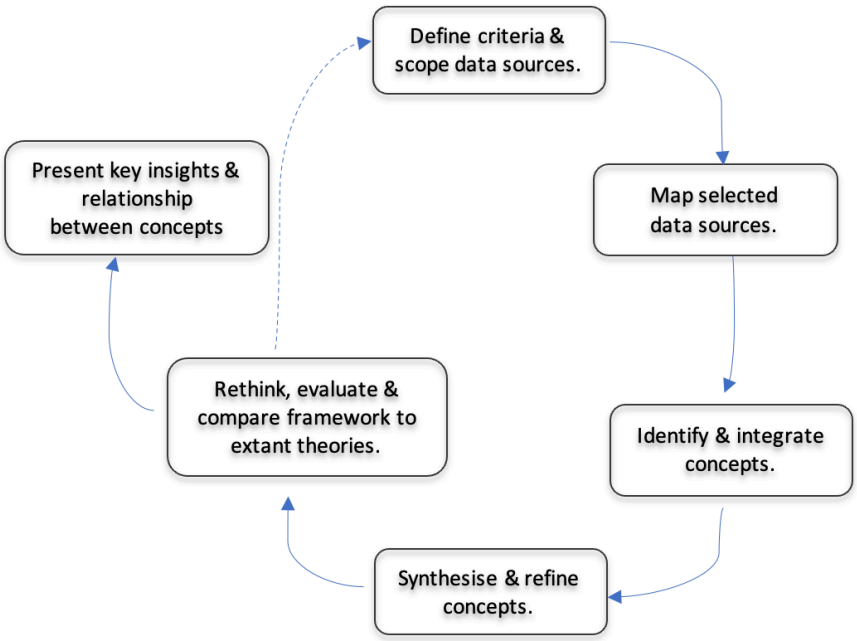



Figure 6.1– Adopted Conceptual Framework Analysis Process

Suitable publications and data for this work were selected based knowledge gained from the CAMS taxonomy and the theory of the *Socio-technical Security Practices of Developers in SMEs* presented in the previous chapters. The scope and literature search were from the following research domains - developer-centred security, software security, cyber security,

secure software development, software security engineering, security in agile development, agile software development, secure development practices, and software development and security in SMEs. The outcomes of previous related studies in comparison to factors identified in this study; the methodologies and methods used; theories and paradigms that these studies are based on; participant demographics, organisation size, structure and type were key search criteria. The multidisciplinary literature is then mapped to the phenomenon being study – the security practices of developers in SMEs. A total of 50 additional research articles focusing on this phenomenon including empirical data, articles on practices and findings from this study were selected.

The identification, integration and refinement of concepts was done using the focused and axial coding techniques described in **Section 3.4**. This coding process involved comparing incidents, contexts, and experiences in the data and extant theories of software security. The selected data sources and resulting conceptual categories are presented in **Table 6.1**. The outcome of the conceptual framework analysis process is a framework that represents the relationship between concepts identified in the data, diagrammatically presented in **Figure 6.2** to sufficiently represent the main factors and concepts identified in literature and findings in this study.

Table 6.1 – Conceptual Framework Analysis: Selected Data Sources and Identified Concepts

| | Selected Data Sources | Context / Contextual Factors | Attitude, Interests & Self Efficacy | Motivation | Support | Security Dialogues & Dialectic Interactions | Client Relationship & Stakeholder Management | Resource Negotiations | Personal & Community Security Knowledge | Organisation Size, Structure & Culture | Security Trainings, Interventions, Assurance & Adoption Techniques in SDLC | Communication & Collaboration Styles | Usable Security Tools & Technology | Programming language & Platform-specific Security | Security Standards, Guidelines & Initiatives | Theoretical Frameworks & Approaches |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | **Chapters 4 and 5 from this study** | ✓ | ✓ | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | ✓ | | | ✓ | ✓ |
| **2** | (Weir et al. 2018; 2019; 2020; Weir, Becker, and Blair 2021) | | | | | | | ✓ | | | ✓ | | | | | |
| **3** | (Chowdhury et al. 2021) | | | | | | | | | | | ✓ | | | | |
| **4** | (Poller et al. 2017) | | | | | | | | | ✓ | | | | | | |
| **5** | (Jøsang, Ødegaard, and Oftedal 2015) | | | | | | | | | | | | | | ✓ | |
| **6** | (Van Der Linden et al. 2020) | | | | | | | | | | ✓ | | | | | |
| **7** | (Mokhberi and Beznosov 2021) | | | | | | | | | | | | | | | ✓ |
| **8** | (Lopez, Sharp, et al. 2019b; 2019a) | ✓ | | | | | | | | | ✓ | | | | | |
| **9** | (Tahaei and Vaniea 2019) | | | | | | | | | | | | | | | ✓ |
| **10** | (Weir, Rashid, and Noble 2017; 2020) | | | | | ✓ | | | | | ✓ | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **11** | (Pieczul, Foley, and Zurko 2017) | | | | | | | | | | ✓ | | ✓ | | | |
| **12** | (Gutfleisch et al. 2022) | ✓ | | | | | ✓ | | | | | | ✓ | | | |
| **13** | (Weir, Rashid, and Noble 2016) | | | | ✓ | | | | | | | | | | | |
| **14** | (Ashenden and Ollis 2020) | | | | | | | | | | ✓ | ✓ | | | | ✓ |
| **15** | (Lopez, Tun, et al. 2019) | | | | | | | | ✓ | | | | | | | |
| **16** | (Ryan, Roedig, and Stol 2021) | | ✓ | | ✓ | | | | | | | | | | | |
| **17** | (Hall et al. 2008) | | | ✓ | | | | | | | | | | | | |
| **18** | (Assal and Chiasson 2018) | | | | | | | | ✓ | ✓ | ✓ | ✓ | | | | |
| **19** | (Hala, Assal, and Chiasson 2018; Assal and Chiasson 2019) | | | ✓ | ✓ | | | | | | | ✓ | | | | |
| **20** | (Acar, Stransky, et al. 2017) | | | | ✓ | | | | | | | | | | ✓ | |
| **21** | (Acar et al. 2016; Acar, Backes, et al. 2017) | | | | | | | | ✓ | | | | | | | |
| **22** | (Acar, Fahl, and Mazurek 2016) | | | | | | | | | | | | ✓ | | | |
| **23** | (Gorski et al. 2020) | | | | | | | | | | ✓ | | | | | |
| **24** | (Keramati and Mirian-Hosseinabadi 2008) | | | | | | | | | | | | | | | ✓ |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **25** | (McGraw 1998; 2004) | | | | | | | | | | ✓ | | | | ✓ | |
| **26** | (Rauf et al. 2020) | | | | | | | | | | | | | | | ✓ |
| **27** | (Rauf et al. 2021) | | | | | | | | | | ✓ | | | | | |
| **28** | (Maher et al. 2019) | ✓ | | | | | | | | | | | | | | ✓ |
| **29** | (Ashenden and Lawrence 2016) | | | | | ✓ | | | | | | | | | | |
| **30** | (Baca and Carlsson 2011) | | | | | | | | | | ✓ | | | | | |
| **31** | (Meng et al. 2018) | | | | | | | | | | | | | ✓ | | |
| **32** | (Bartsch 2011) | | | | | | ✓ | | | | ✓ | | | | | |
| **33** | (Caputo et al. 2016) | | | | | | | | | | | | ✓ | | | |
| **34** | (Chen et al. 2019) | | | | | | | | ✓ | | | | | | | |
| **35** | (Wijayarathna and Arachchilage 2019) | | | | | | | | | | | | | ✓ | | |
| **36** | (Palombo et al. 2020) | | | | | | | | | | ✓ | | | | | |
| **37** | (Votipka et al. 2020) | | | | | | | | | | ✓ | | | | | |
| **38** | (Parker et al. 2020) | | | | | | | | | | ✓ | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **39** | (Thomas et al. 2018) | | | | | | | | | ✓ | ✓ | ✓ | | | | |
| **40** | (Weir, Hermann, and Fahl 2020) | | | | | | | | | ✓ | ✓ | | | | | |
| **41** | (Aljedaani et al. 2020) | | | ✓ | | | ✓ | | | | | | | | ✓ | |
| **42** | (Witschey, Xiao, and Murphy-Hill 2014) | | | | | | | | | | ✓ | | | | | |
| **43** | (Xiao, Witschey, and Murphy-Hill 2014) | | | | | | | | | | ✓ | | | | | |
| **44** | (Jordan et al. 2014) | | | | | | | | | | ✓ | | | | | |
| **45** | (Witschey et al. 2015) | | | | | | | | | | ✓ | | | | | |
| **46** | (Zarour, Alenezi, and Alsarayrah 2020) | ✓ | | | | | | | | | ✓ | | | | | |
| **47** | (Votipka, Abrokwa, and Mazurek 2020) | | ✓ | | | | | | | | | | | | | |
| **48** | (Haney et al. 2019) | | | | | | | | | ✓ | | | ✓ | | | |
| **49** | (Sawaya et al. 2017) | | ✓ | | | | | | ✓ | | | | | | | |
| **50** | (Xie, Lipford, and Chu 2011; Xie et al. 2011; Xie, Lipford, and Chu 2012) | | | | ✓ | | | | | | ✓ | | | | | |

## 6.3 Framework Overview

The findings presented in **Chapters 4** and **5** show that the practices of developers in SMEs are formed by several factors that influence their software development process including but not limited to factors such as context, attitudes, motivation, support, the role of clients and navigating relationships within various social structure. However, it is important that secure software development efforts including initiatives, guidance, tools and technology providers and developers themselves, know how to assess factors that are particular to the SME development environment and identify what is needed to support the effective practice of secure software development in SMEs.

The conceptual framework in **Figure 6.3** presents an integration of core concepts from a comprehensive review of related research literature, extant theories, and findings from this study, to show what is needed to understand the complexities of developer's practices in SMEs. The framework describes three main categories of secure software development socio-technical practices (*individual*, *organisational* and *technological practices*) of developers supported by a list of fifteen supporting factors identified in this study and in literature. This gives a holistic view of the important areas to consider when describing and assessing the secure software development practices of developers. A holistic approach to viewing these practices provide the software engineering and security fields with a strategic frame of reference that considers all factors based on the experiences of developers, to support the creation of guidelines, procedures, tools, and policies for secure software development. Particularly for developers in SMEs, the framework sheds light on the experiences of these developers and how their practices are formed within their organisational context.

**Table 6.1** shows that a majority of the research efforts directed at understanding the activities and experiences of developers in secure software development have focused on improving security skills (Weir, Rashid, and Noble 2016); security trainings, warnings, interventions, adoption, and assurance techniques and in identifying developers' motivation for security (Weir, Becker, and Blair 2021; Acar, Stransky, et al. 2017; Hall et al. 2008). Majorly, these research studies have worked on understanding how industry software development teams have successfully achieved software security particularly in large organisations. These efforts

have been impactful in the creation of security guidance, frameworks, policies, and initiatives. However, further studies into the interactions between developers and counterparties and stakeholders have been encouraged to identify the best ways of leveraging human interactions to improve software security (Weir, Rashid, and Noble 2020). This study provides that contribution by presenting the experiences of developers in SMEs and identifying how the interactions between developers, clients, and communities impact secure software development practices.

**Figure 6.2** shows a high-level classification of socio-technical practices from selected relevant data sources including this study. This classification presents an overview of the different dimensions of how secure software development practices have been addressed in literature. Some studies have addressed individual security practices of developers based on their attitudes, behaviours, and motivations of security adoption (Maher et al. 2019; Witschey et al. 2015; Witschey, Xiao, and Murphy-Hill 2014). Other studies have looked into security motivations, deterrence and knowledge sharing activities of developers across various organisations and social structures (Herath and Rao 2009; Lopez, Tun, et al. 2019; Ashenden and Ollis 2020). Studies that focused on technological practices, explained the impact of security warnings, usable security APIs, issues around writing vulnerable codes and ways of improving security adoption during each phase of the software development lifecycle (Gorski et al. 2020; Oliveira et al. 2018; Green and Smith 2016; Assal and Chiasson 2018). This classification of data was a starting point for the framework development.

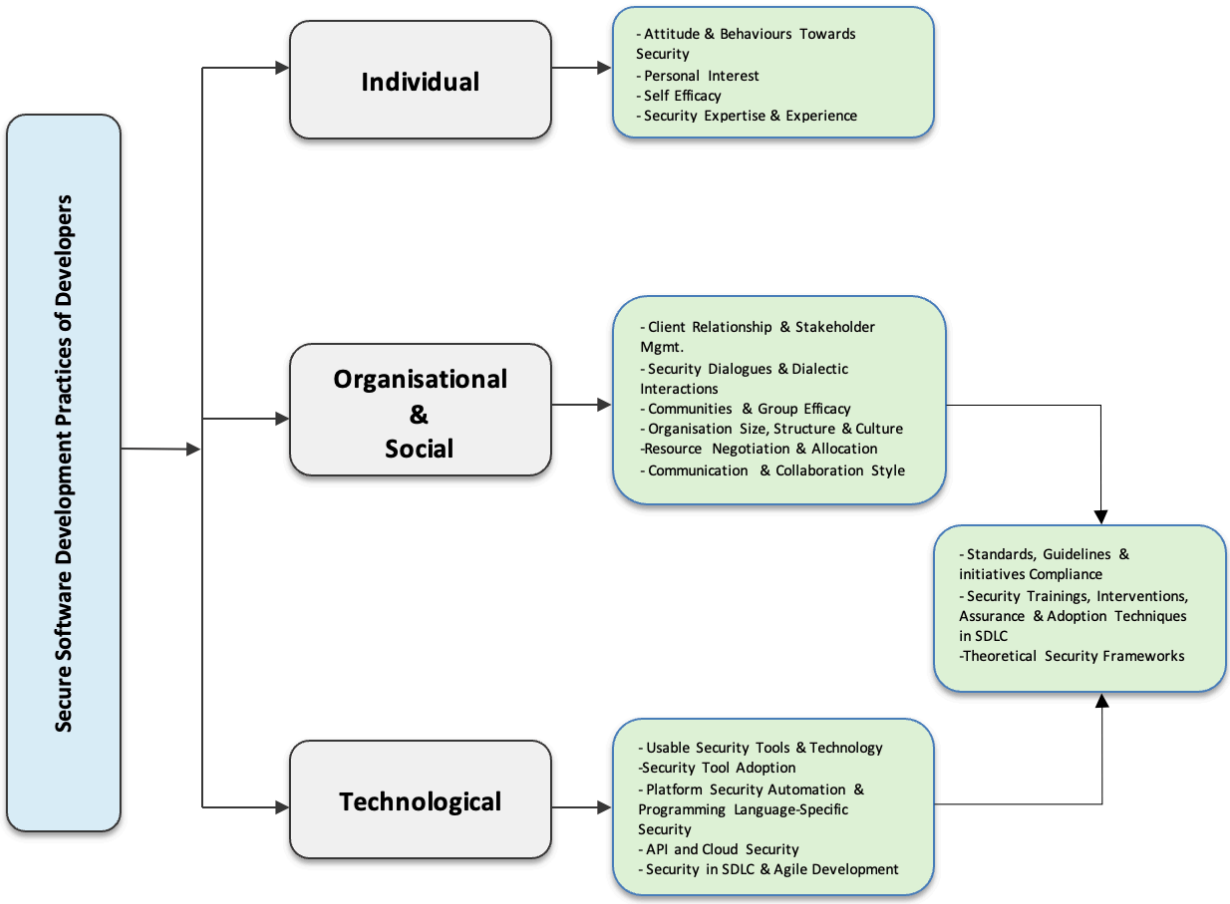



Figure 6.2– Classification of Conceptual Categories of Secure Software Practice from Literature

The framework in **Figure 6.3** presents a different perspective of factors that should be considered by the secure software development ecosystem in designing software security initiatives, new software security guidelines, policies, tools, or techniques for developers in SMEs. The main goal of the framework is to guide industry and academic research in secure software development efforts and draw attention to the particular needs of developers in SMEs. The scope of the framework is within the SME and micro-business context, to provide software security practitioners and researchers insights required in planning, designing, and evaluating secure software development practices for developers. While there are several studies describing and showing evidence of how developers are not properly adopting or

delivering security leading to vulnerable software product (Xie, Lipford, and Chu 2011; Oliveira et al. 2019; Wijayarathna and Arachchilage 2019), little evidence is shown to explain how developers in SMEs are navigating their day-to-day development and organisational activities to meet the expectations of software security standards. This study provides evidence of how the secure software development practices of developers in SMEs are formed. The framework doesn't point out security issues or behavioural flaws but provides a view into the factors that form and impact the security practices of developers based on their experiences particularly within the SME organisational context. The outcomes of studies such as the survey of the impact of software security practices on development efforts (Venson et al. 2019) and the need to understand dialectic interactions for improving the security of development activities (Weir, Rashid, and Noble 2020), are examples of studies that have presented areas where this framework can be applied. The CAMS taxonomy presented in **Chapter 4** is used to provide insights into the different dimensions of secure software development practices of developers in SMEs.

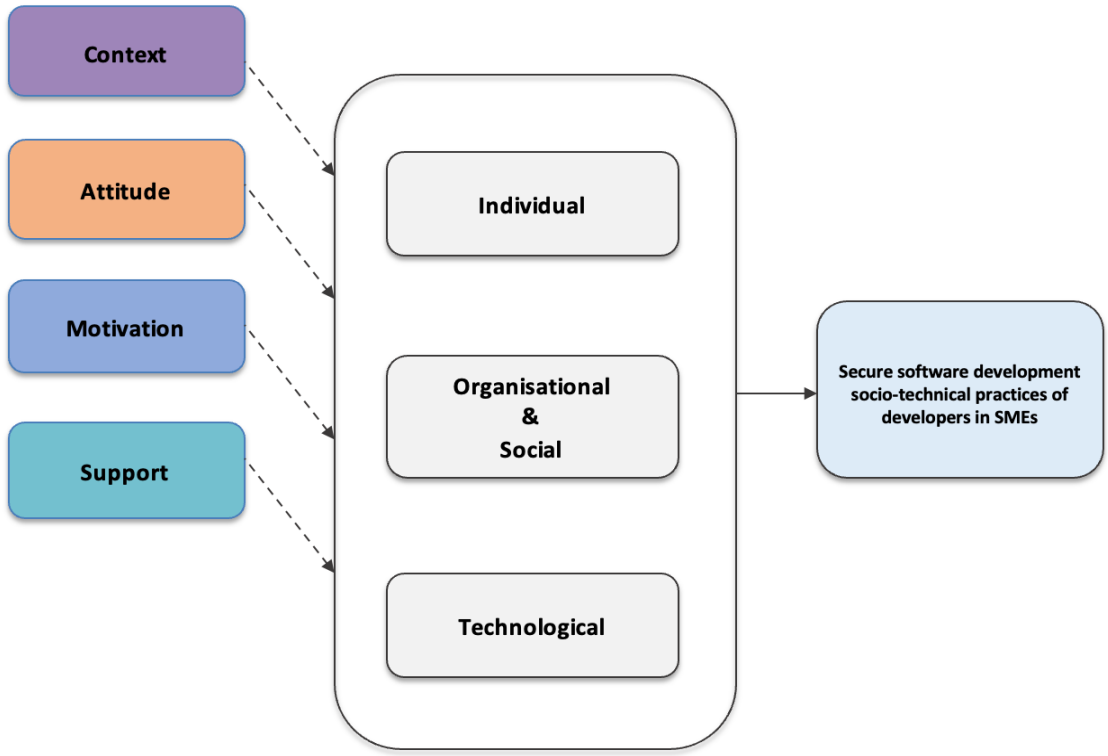



Figure 6.3– Conceptual Framework of Socio-technical Practices for Secure Development (CAMS- IOT Framework)

## 6.4 Individual Practices

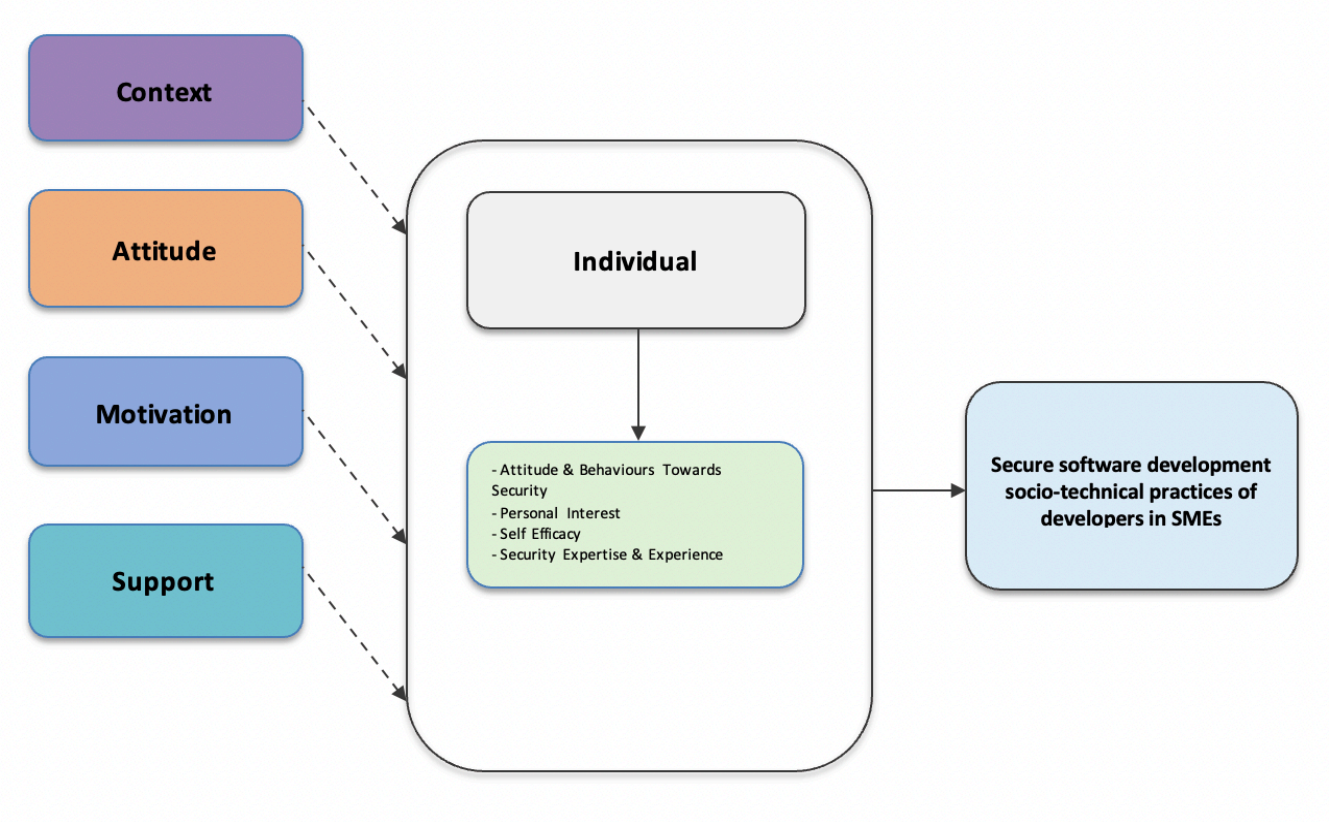



Figure 6.4– CAMS-IOT Framework in Individual Practices

### Context

Understanding the individual dimensions of secure software development practices of developers in SMEs requires a knowledge of the context surrounding why people do what they do. This enable us to explain how their practices are formed and the impact these practices have on the resulting software product. Merely identifying the security activities of developers and the potential threats to a software cannot describe fully the context of practices of secure software development. There is the need to understand the context in which these developers work. In an ethnographic study that involved using a discussion tool made up of security incidents from media and internet sources, showed that engaging with

developers based on the context of the experiences of others and their personal may improve security practice and technology adoption (Lopez, Sharp, et al. 2019b). The self-efficacy of developers in SMEs represented in this study are based mainly on their personal experiences with handling past security incidents, previous security training, being reformed hackers or security experts.

**Attitude**

Attitudes and behaviours towards security have been described in literature in various ways including using the Theory of Planned Behaviour or Reasoned Action (Leonard, Cronan, and Kreie 2004). The claim has been that developers in smaller companies are less likely to have positive security and privacy attitudes and behaviours (Balebako et al. 2014). However, attitude of individuals towards security can change based on experiences and information. Based on the outcomes of this study shows that for developers, attitude towards security are heavily dependent on their individual experiences with security incidents, client's specifications, and a shared security information within the developers' communities they are associated with.

**Motivation**

In a study by Assal, developers' software security practice motivators and deterrent factors were communication, collaboration, and a positive impact of a 'shared meaning' of security among all individuals involved in the development process (Assal 2018). In this study, communication and collaboration was also identified, however, these factors were not considered as motivators for implementing software security measures. Hall et al. reports that change, challenge, problem-solving skills, life cycle models and teamwork as some of the intrinsic motivators for software developers (Hall et al. 2008). Although these motivators are not specific to secure software development practices, they help us understand developers more. Developers in SMEs are motivated by the financial capacity of their clients to cover the cost of security and their experiences. Understanding these motivators, provides insights into how developers implement security practices in their development process.

**Support**

The core of this study is in understanding the practice of secure software development by developers in SMEs based on their experiences. Practices are organised or unorganised established activities habitually performed by individuals (Orlikowski 2002) or according to Bourdieu, they are "recognisable patterned actions" that both individuals and groups engage in (Theodore R. Schatzki 1997; Bourdieu 1977). In secure software development, this implies that practices may include organised development methodologies or emergent security activities that are repeatedly performed by individuals, teams or communities when developing a software. Support from developers' communities and development platform providers affect the developers' attitude and behaviour towards security. Security guidance and initiatives that improve the experiences and efficacy of developers in SMEs can contribute to improving secure software development practices.

## 6.5 Organisational and Social Practices

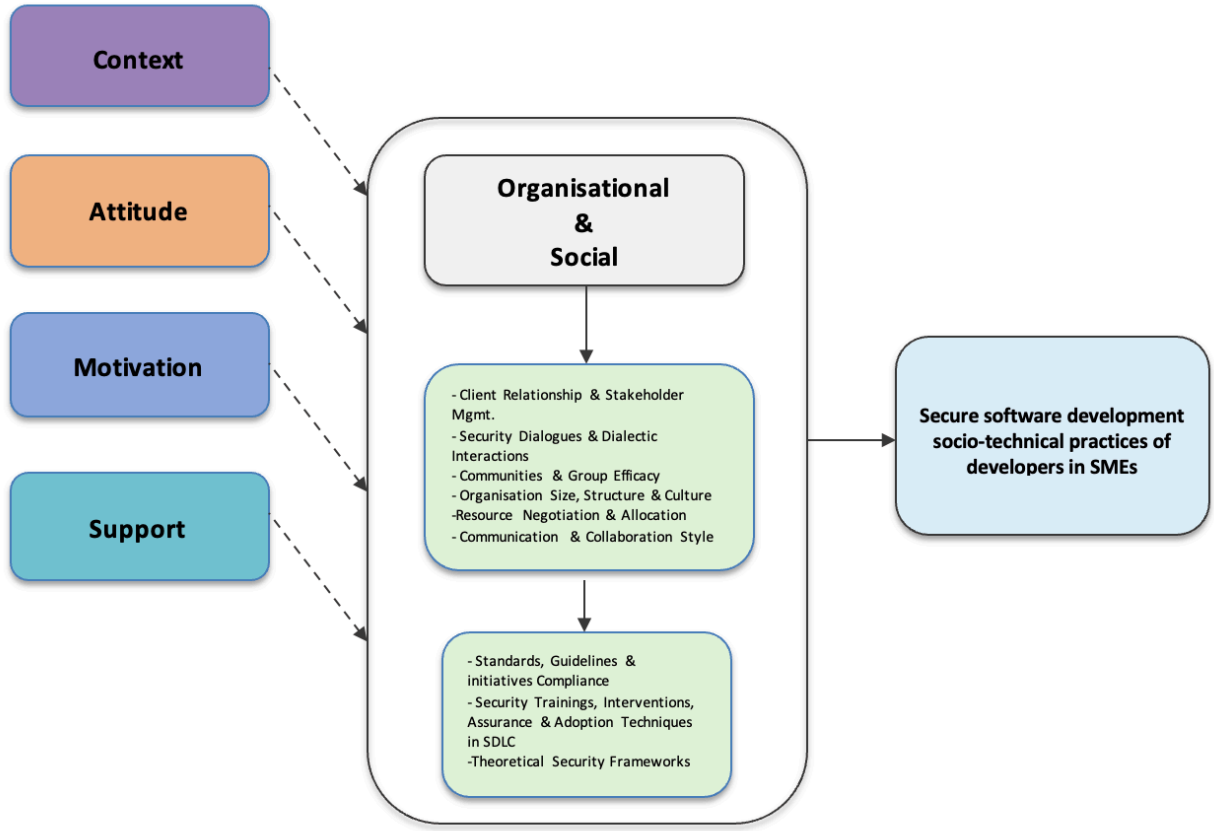



Figure 6.5– CAMS-IOT Framework in Organisational & Social Practices

### Context

Software engineering and security domains have focused on the various human actions and practices, in developing systems and securing information from unauthorised access (Clarke and Drake 2003).The software application development process involves procedures, practices, technologies, and policies that work coherently within different organisational structures with the aim of producing or maintaining a software product (Kraut and Streeter 1995). These processes entail various activities and phases where teams of developers, engineers and managers collaborate under various constraining company policies and structures as well as organisational norms and resources to achieve a common goal (Yilmaz

2007). It is important to understand the SME and micro-business context to provide the right security guidance, policies or initiatives that support the secure development practices of software developers in these types of organisational structure. For example, the role of client relationship management and involvement in the development process should be taken in consideration when explaining the practices of developers in SMEs.

Developers in SMEs also rely on the discussion within their social network and communities of collective shared security knowledge or experience to shape their secure software development practices. Similar to other social communities, developers' communities are made up of individuals who may look to others for directions, reassurance, and support. As described in **Chapter 4**, the celebrity developers represent a type of leadership within these communities, were developers rely on their expertise and knowledge on secure software practices.

**Attitude**

Studies in security research have also argued the importance of viewing cyber security practices as socially influenced processes that require us to understand human and social factors underlying people's security decisions and behaviours (Das et al. 2014). Das (Das 2017) used insights from a combination of interviews and surveys of users of a social network, Facebook, to build a theory of social cyber security. Das' argues that security attitudes and behaviours are strongly influenced by social influences in organisations and communities. Secure software development measures should be designed in a way that actively considers organisational and social factors to determine its adoption and encouragement of a better attitude to security.

**Motivation**

Developers are motivated to implement security practices based on their clients' specification, knowledge, and willingness to finance the cost of security. As described in **Chapter 4**, interactions, dialogues and negotiations between developers and their clients can drive the need for specific security measures to be implemented. In a similar study on

dialectic security done by Weir et al (2020) different stakeholders such as product managers, senior managements, and customers were identified as counterparties involved in the security costs and risk negotiations process for deciding what aspect of security to implement in the development process (Weir, Rashid, and Noble 2020). These negotiations were also identified in this study, however for developers in SMEs particularly where the developer is a sole proprietary owner of the business or working with a very lean team, these negotiations are make or break discussions that motivate how the developer thinks about software security and its implementation because developers are worried about losing business due to high cost of security. This study contributes to literature by identifying the impact of the client's specification and security knowledge on the motivation of developers in SMEs for a more effective secure software. A client with some level of security knowledge can specify the security features they require during the initial conversation with the developer. The developers can also recommend to clients the right security measures that can be implemented in the software product. These recommendations are usually based on their personal experience and expertise of the developer. The security features and cost of implementing these features are then outlined to clients. If the client is unwilling to pay any extra cost or cannot work with any extended timeframe, then the developers are likely to deprioritise security for that software project.

**Support**

Ortner's reports that practices are the continuously changing dialectical relationship between structures (social or organisational) and human actions (Ortner 1984). Social structures such as the developers' communities or organisational structures such as SMEs, have the capacity to direct the actions that developers take during the development process. Software developers in SMEs with flexible or no organisational and responsibility structure are more inclined to adhere to security trade-offs due to challenges such as the inability to afford standard security tools and standards, which are particular to SMEs (Osborn and Simpson 2015). It is therefore important to improve security guidance resources that support the developers in SMEs based on challenges particular to them.

## 6.6 Technological Practices

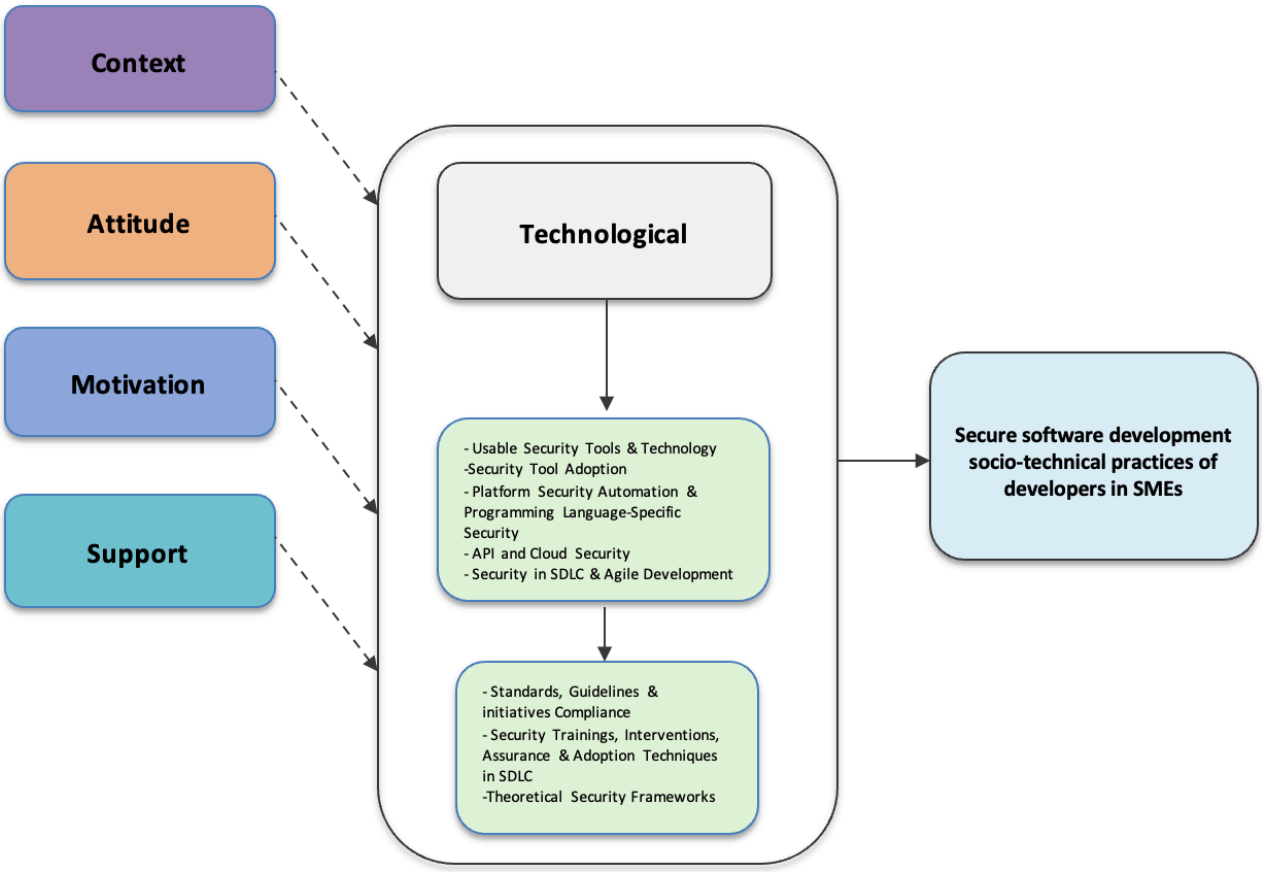



Figure 6.6– CAMS-IOT Framework in Technological Practices

**Context**

The practices of software development involve activities for defining the purpose of software, its domain, structure, behaviours, required computation and the interactions with its environment and users (Jackson 1994). Security measures may be a distant consideration for many developers particularly in SMEs who are justifiably focused on the functionality and utility of the software to meet the satisfaction of their customers (Wurster and van Oorschot 2009). But nowadays, with the web and cloud platforms being a software development environment for both large organisations and small- scale software development activities,

there is a need to devote attention, time, and effort to security considerations in this environment (Fischer 2016). Various technological trends contribute to security issues in software development for most organisations (Fuggetta and Di Nitto 2014). Some of these trends such as changes in development environments, communication, and collaboration methods require better understanding for improving guidance for secure software development practices.

For example, Microsoft developed (and implement) the Security Development Lifecycle process to minimise security vulnerabilities, develop policies, practices, and technologies to help developers build applications as secured as possible (Howard 2004; Lipner 2010). Microsoft represents some of the larger organisations with a standard organisational structure with the required resources needed to implementing security initiatives. These large organisations provide security awareness education for their developers, engineers, external security reviewers and SOC consultants. They also provide technologies for implementing good security design principles, organise security-focused events, have final security review teams and have built security resource centres (Howard and Lipner 2006). This presents developers in micro-businesses, small and medium-sized organisations with a structured set of practices to meet specific security standards with limited resources. Each phase in the development process requires the developer (and other operations member of the organisation involved in the project) to ensure that despite any challenges faced, the resultant software application should meet the client or end-users' expectations.

**Attitude**

Wurster & van Oorschot claim that developers ignore security practices by disabling tools that are designed to assist in creating secured code during compilation time for their codes to run more quickly in order to meet production deadlines (Wurster and van Oorschot 2009). Although developers in this study did not share any reports about this actions, they explained that security measures are considered after the deploying the first version of the software product to meet tight deadlines. Developers attitude towards secure software development development may be improved when the design of security measures account for client's specifications and project timelines.

**Motivation**

Trends such as open-source development, cloud computing and artificial intelligence services have changed the dynamics according to which software is being designed, developed, and distributed. (Feller and Fitzgerald 2002; A. Mishra et al. 2013).This transformation also poses another challenge to the management of development activities for developers in SMEs, with new, unconventional, and unpredictable environments and settings. Time and financial resources are required to keep up to date with these trends which can either deter or motivate developers in considering software security measures. It is therefore important to understand the implications of these trends on developers' practices in order to provide new approaches and guidelines for best practices for collaboration and communication during SDLC.

**Support**

Developers have a huge role to play in ensuring that applications and security holes do not expose important information to attackers via their various platforms, tools, and technologies by maintaining security practices that can be implemented around the development process of the software (Northrop and Lipford 2014). The fast-paced, resource constrained nature on SMEs necessitates the need to support new practices within different development environments and towards safer communication methods, more effective programming, and collaboration techniques. Developers in SMEs have to continuously evolve with the changing development environments and support technologies that operate over the Internet to produce quality products that meet the customers' requests and expectations. Hence, the need to support developers so they are more familiar with best security practices for internet-based software development.

## 6.7 Perspectives on Socio-Technical Practices in Secure Software Development

The CAMS-IOT framework is a conceptual framework that explains how context, attitude, motivation, and support describes the secure software development practices of developers in

SMEs. These practices are moderated by the socio-technical dynamics occurring between the social factors of individual and organisational activities, and the technological aspects of the development process. Other software engineering researchers have studies social aspects of secure software development and the influence on the practices of developers and a comparison of the framework with their results can help extend research insights.

Assal's study of how the security knowledge of developers influence security in the software development lifecycle (SDLC) (Assal 2018) describes an application of context, attitude, motivation, and support to the technological practices of developers. The study involved interviewing 13 developers to identify their tasks, priorities and tools used during the development process to understand how these affects the security of their applications and how well 'real-life' security practices compare to 'best practices'. Although the demographics of participants include developers from small, medium, and large organisations within North America only, the security attitude, approaches and priorities of developers where identified for the different SDLC stages which is similar to what the CAMS-IOT framework can be used to interpret.

The security attitudes and behaviours of developers in Assal's study were split into two extreme groups: security adopter and security inattentive (Assal 2018). Security adopters were classified as those developers who considered security in majority of the stages of the development lifecycle and being inattentive meant that the developer dismissed or poorly approached security. Emerging themes showed that to a security adopter, security is priority during the design, implementation, testing, code analysis, code review and post-development testing stages. However, for an inattentive developer, security is considered secondary in almost all stages of the development lifecycle. If developers in SMEs are to be described as either security adopters or inattentive, their development context, overall attitude, motivation, and the support available to them need to be understood. Developers in SMEs represented in this study whose client's specifications drove the security decisions of their software project, can be both security adopter or inattentive depending on the client's request or willingness to pay for the cost of security measures.

Understanding the secure software development practices of developers in SMEs through the context, attitude, motivation, and support for the developer can be used to provide insights into the adoption of various security techniques. In a study on developers' security assurance techniques for secure software development, a dialectical security theory was derived to explain how developers' thinking and six security techniques are being challenged by different counterparties such as the product managers, security experts, members of the development team, automated code analysis tools, operations staff, or end users (Weir, Rashid, and Noble 2020). From interviewing 16 software security industry experts, six most effective security assurance techniques were described – threat assessment, stakeholder negotiation, configuration review, source code review, vulnerability scan and penetration testing. Stakeholder negotiation was emphasised as an important requirement for software development security. This also applies to developers in SMEs, however the CAMS-IOT extends the dialectical theory by explaining how security techniques such as stakeholder negotiations become part of individual, organisational and technological practices through an understanding of the context, attitude, motivation, and support for developers. The context for developers in SME is not just identifying the list of possible attacks along with cost of security measures but in the client's willingness to pay. Expressing the risks and costs of software security to stakeholders in terms they understand may be beneficial, however, true implementation comes from the availability of funds to cover the cost of security. This in turn improves the attitude of the developer in SMEs towards security as well as motivates and support their security decisions.

In a study aimed at motivating developers to implement security in the development process through engagements, Lopez, Sharp and others (Lopez, Sharp, et al. 2019b) ran a series of engagement sessions with professional developers to develop a security discussion tool. The aim of the engagement activities was a provide developers with security incident scenarios to provoke discussions that support and encourage their security practices. This intervener approach is one of the major techniques implemented in industry to improve the security practices of development teams. Rather than applying intervention techniques that prove the inadequacy of the security practices of developers, this study suggests an alternative lens at

understanding and supporting the individual, organisational and technological practices of developers in SMEs. The CAMS-IOT framework suggests a socio-technical approach that explains the perspective of developers in SMEs regarding security efforts in software development. For example, understanding how the SME context - comprising of the client's specification and financial capabilities, availability of resources, software domain and level of experience – influences security decisions that can improve secure software development in smaller organisations. Surveys similar to the one done by Venson and others (Venson et al. 2019), support the contributions of this study, by highlighting the need to understand the experiences and perspective of software development professionals when planning secure software development initiatives.

## 6.8 Conclusion

If the secure software development practices of developers are to be explained and understood, it is imperative that organisations, development platforms, security practitioners and researcher are aware of the perspectives of security activities by developers in SMEs as socio-technical practices driven by various factors. It is also important that secure software development initiatives and technologies being created to support SME developers are directed at understanding how these developers' socio-technical practices are encouraged to produce results that are beneficial to the software industry. This chapter presents the CAMS-IOT framework, which represents the relationship and dynamics that influence developers' security practices in software development particularly in SMEs. The study recommends that context, attitudes, motivation, and support should be core concepts used to understand the individual, organisational and technological aspects of secure software development practices for developers in SMEs. The framework was used to discuss and extend current discussions within the software security landscape to provide insights that are applicable to developers in SMEs. The framework is limited by its scope which is the SME and micro-business context, however, can be extended to account for the perspective of larger organisations in further research.

# Chapter 7 Discussion

## 7.1 Introduction

This chapter further elaborates the wider implications of the theory of *Socio-technical Security Practices of Developers in SMEs* and the *CAMS-IOT* framework within software development and security research. The theory describes socio-technical factors relevant to the current security practices of developers in SMEs, and the framework reveals concepts that extends extant concepts and theories. The significance of this is to explain secure software development as a social practice for developers in SMEs - a social phenomenon from the perspective of the individuals who participate in this process within their specific organisational context.

## 7.2 Secure Software Development as a Social Practice for SME Developers

The theory of *Socio-technical Security Practices of Developers in SMEs* and the *CAMS-IOT* framework explains secure software development as a social interaction between individuals and groups of developers within their ecosystem. The key concerns and experiences of SME developers who participated in this study are mainly social issues. The outcome of this study supports the argument that secure software development is just as much a social practice as it is an organisational and technical practice (Weir, Rashid, and Noble 2020; Acar, Stransky, et al. 2017; Assal 2018; Lopez, Sharp, et al. 2019b). First, the theory of *Socio-technical Security Practices of Developers in SMEs* informs us that implementing security in software development is an ongoing negotiation and management process with clients particularly for

developers in SMEs. The *CAMS-IOT* framework then shows how concepts described in this study compare to other key studies in literature to describe how the secure software development practices of developers in SMEs are formed. To explain practices within different forms of social structure, MacIntyre defines practice as "a coherent and complex form of socially established co-operative human activities"(MacIntyre 1987). This definition supports Nicolini's argument that practice-based studies offer a new way of understanding human, social and organisational phenomena. Social approaches are hardly drawn on in generating theories in secure software development research, even though various social aspects make up an integral part of security practices in the software development process. Studies by Ralph et al. (Ralph, Chiasson, and Kelley 2016) or Button et al. (Button and Sharrock 1994) support this argument stating that many challenges in software development projects, including security challenges, are beyond technical, they can also be social, psychological, or managerial.

Several studies have developed guidelines, initiatives, and recommendations for implementing security practices from a psychological perspective with the focus being on the security behaviours of individuals – end-users and developers (Muhirwe and White 2016; Das 2017; Hasna and Mustapha 2016; Acar, Fahl, and Mazurek 2016). And although there are no shortages of experimental, hypothesis-driven, observational research studies addressing software security behavioural issues (Mashiane and Kritzinger 2020), developing fundamental social theoretical basis for security-related studies are beneficial for several reasons. To begin with, it allows us better explain concepts that have been divided in security research for a long time such as agency, responsibility, ideation, and practices (Bueger 2016). In addition, it boosts the empirical basis of security and secure software development studies by offering new methodological approaches. Furthermore, it contributes to the bodies of knowledge by giving us a broad view of understanding social dynamics in the technical aspects of secure software development.

The significance of the theory of the *Socio-technical Security Practices of Developers in SMEs* and the *CAMS-IOT* framework is that they offer an alternate sociological view of developers' experiences of security issues in software development within the SME context. This study explains how developers in SMEs navigate social and technical dynamics in

secure software development process and the impact it has on the software apps they produce. Other human centred research in security and software engineering disciplines have attempted to explain and improve secure software development practices (Apvrille and Pourzandi 2005; Howard 2004; Al-Amin et al. 2018), improve security awareness (Kritzinger and von Solms 2010; Muhirwe and White 2016), and impact security behaviours (Mashiane and Kritzinger 2020). This study develops theory and framework that identifies that individual, organisational and social and technological practices such as successful dialogues and negotiations between the developers in SMEs and their clients are the core drivers for secure software development drivers in SMEs.

Some studies have focused on software development practices from the perspective of larger organisations. For example, Adolph (Adolph 2013) developed a theory of how people manage the process of software development. The theory of reconciling perspectives describes software development as a social process that requires the individual abilities of each member of the team to constantly negotiate and agree on mismatched perspectives during the development process. These social dynamics and as well as other factors are major drivers for successful software engineering projects. Other researchers such as Fuggetta have also contributed to this notion, describing software development as a "social and creative process where the creativity, skills and co-operation of developers, users and procurers determine the quality of and effectiveness of the developed software product" (Fuggetta and Di Nitto 2014). In another qualitative study by LaToza and Myers (LaToza and Myers 2010), factors such as work style, social interactions and knowledge exchange between developers are some of the major factors that influence software developer's practices and the strategies they use day-to-day during the development process. Software engineering and security research are commonly grounded in formative empirical studies and are considered as technical fields. With humans involved it has become important to understand the intersection between people, systems, their actions, and practices from a multidisciplinary perspective. This study offers guidance to developing software security initiatives for developers particularly for practitioners in smaller organisations.

## 7.3 Supporting the Design of Secure Software Development Initiatives for Developers in SMEs

As described in Chapter 2, several studies have been done to produce a variety of processes, recommendations, and guidelines with the intent to support software companies and independent developers in implementing security best practices in software development lifecycle. Some of the common initiatives recognised in the software development landscape have focused on larger organisations. The outcomes of this study can support the design of these initiatives to account for the experiences of developers in SMEs.

The Microsoft's Security Development Lifecycle (SDL) initiative (Microsoft Corp. 2004) which consists of twelve recommended practices (such as providing core security training for everyone in the development team, establishing security and privacy requirements early and regularly etc.) should consider the context of the software project, the attitude, motivation and support available to the individual, organisational and technical practices of the developer. In addition, the role of the client in supporting the security requirements that enable developers build more secure software should be considered. The Microsoft SDL-Agile lightweight initiative also recommends that developers consider certain practices as part of every sprint requirement such as doing at least one security training and threat modelling for all new features. For developers in SMEs with limited resources and time constraints, these practices may be challenging to implement.

The Building Security in Maturity Model (BSIMM) framework (BSIMM by Synopsys 2022) should also consider incorporating concepts from the CAMS-IOT framework into its four main categories: Governance, Intelligence, Software Security Development Lifecycle (SSDL) Touchpoints and Deployment. Similar to the Microsoft SDL initiative, the BSIMM framework can be extended to consider the context, attitude, motivation, and support for implementing the security practices highlighted in each of the four categories.

The Open Web Application Security Project (OWASP) Top-10 is an initiative popular among participants in this study, focuses on creating awareness about the ten most critical security risks in web applications (OWASP 2017). This initiative provides free and open-source tools, standards, books, videos, and checklists based mainly on the technical practices of

developers. The OWASP initiative can be extended to promote the individual, organisational and technological practices of developers by providing insights into software security that considers the context, attitude, motivation, and support to give developers a starting point on the journey to building more secured apps.

Other initiates that can incorporate the concepts from this study include the Software Assurance Forum for Excellence in Code (SAFECode) Fundamental Practices for Secure Software Development initiative (SAFECode 2018), National Institute of Standards and Technology - Secure Software Development Framework (Souppaya, Scarfone, and Dodson 2021), SANS Institute Framework for Secure App Design and Development (McCown 2003), and the National Cyber Security Centre – Secure Development and Deployment Guidance (NCSC 2018) - government initiatives that claim to provide practitioners with a common language for describing secure software development practices. Beyond the unification of terminologies in secure software development, initiatives should consider the context, attitude, motivation, and support individual, organisational, and technological practices of developers in SMEs.

## 7.4 Responsible Innovation and Secure Software Development of Developers in SMEs

The concept of Responsible Innovation (RI) in secure software development is relatively new and still evolving, however, von Schomberg defines it as “a transparent, interactive process by which societal actors and innovators become mutually responsive to each other with a view to the (ethical) acceptability, sustainability and societal desirability of the innovation process and its marketable products (in order to allow a proper embedding of scientific and technological advances in our society)” (Von Schomberg 2012). In the software industry, responsible innovation is described as a “set of practices in software development that anticipate and address the potential negative impacts of technology on people” (Microsoft Corp. 2023). For secure software development practices, the responsible innovation paradigm supports the engagement of stakeholders in the process. Some studies have indicated that stakeholder engagement in the innovation process plays the main role in how responsible

innovation should be approached (Pavie, Carthy, and Scholten 2014; Koops 2015). Software SMEs should consider responsibility structuring, defining what roles are held by clients, developers, or end-users and modelling the relationships between these individuals in organisations in the secure software development process (Flechais 2005; Faily 2011). The impact of context, attitude, motivation and support on organisational structure, roles, and responsibilities for secure software development.

As software applications become more pervasive in our society, being developed, released to the public in short timeframes, with the ability to influence human activities, strengthen, or weaken democracy or even negatively impact people (Jirotka et al. 2017). It has become prevalent that software development innovators and researchers need to be exposed to more ways of ensuring that products are created in the society's interest. Responsible innovation frameworks and socio-technical frameworks such as the outcome of this study offer ways of enabling software development teams by incorporating social and ethical human-centred practices in the development process for the future of software engineering (Pavie, Carthy, and Scholten 2014; Von Schomberg 2019). With technological advancements in software security such as artificial intelligence driven security intelligence, threat detection and countermeasures (Hu et al. 2021; Sarker, Furhad, and Nowrozy 2021), it is important that developers in SMEs are kept in loop. Herrmann and Pfeiffer agree with the argument that the organisational practices of individuals are needed to shape the advancements in artificial intelligence (Herrmann and Pfeiffer 2023) which is beneficial to secure software development.

## 7.5 Agile Methodologies and Socio-Technical Practices of Developers in SMEs

Most participants in the study reported that their software projects are developed in agile environments. Agile development methodologies are considered to be more flexible, informal, high development speed, iterative and customer-centric processes to deliver software projects (Sharma and Bawa 2020). Extreme Programming (XP), Scrum, Dynamic System Development Method (DSDM), Adaptive Software, and Feature Driven

Development are examples of agile methodologies used to address different business needs (Cockburn 2002). **Figure 7.1** is representative of the iterative process involved in a typical agile development environment.

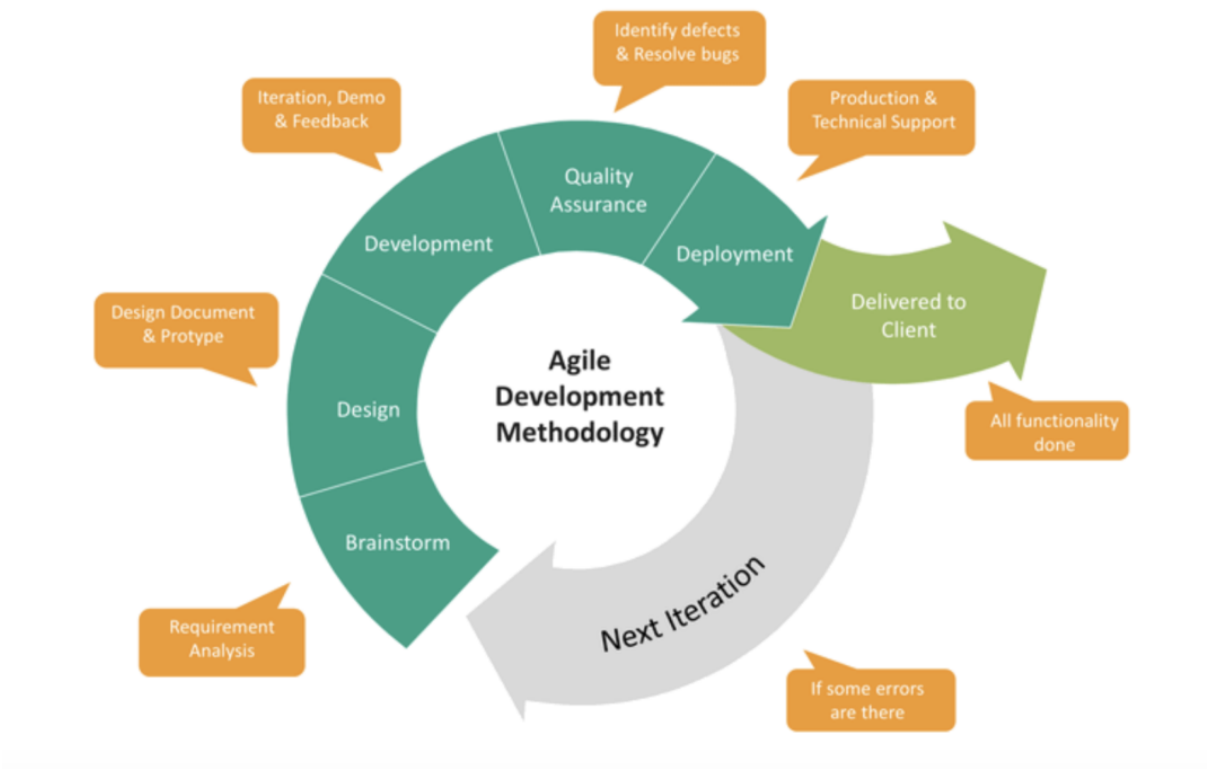



Figure 7.1– Agile Development Methodology (Hoffman 2020)

The theory of *Socio-technical Security Practices of Developers in SMEs* and the *CAMS-IOT* framework present a lens for interpreting secure software development in agile environments. Clients play a key role in driving how developers in SMEs implement security measures and the client's specifications and the context of the software project should be considered at the different stages of the development iteration. The development team need to identify and engage the people involved in and affected by the software project – team members, stakeholders, and end-users. Then, identify cost, available resources, context in which the software will be used and security requirements. Developers can acquire security knowledge from their developers' communities – for current security information that best fits the project.

Usable security tools, frameworks, and technologies required should then be selected and possible consequences of decisions made in tool selection should be considered and discussed with clients. Developers should implement agreed security tools, frameworks, and technologies according to security information gathered from community expertise, responses from stakeholder negotiations and personal experience. This process can be repeated for each iteration. Security in software development is a process rather than a product, and so every stage of the development process requires an integration of essential security measures. For each stage of the agile development process, from brainstorming to delivering the products to the client, the context, attitude, motivation and support for developers should be considered in ensuring that the security practices of developers are shaped throughout the process.

# Chapter 8 Conclusions

The aim of this study was to understand developers' experiences in micro-businesses and small and medium-sized organisations to explain how their secure software development practices are formed and implemented. Unlike many studies that claims that developers "are the weakest link" and require intervention for the adoption and implementation of security measures and initiatives, this study argues that the social dynamics that developers need to navigate influences their practices and craft. Beyond the adoption and motivation to use the many available security tools or security initiatives and sanctions for non-adoption lies the social issues that developers in SMEs need to address.

For example, this study spotlights the social interactions between developers and their clients. For a small software business that runs solely based on the number of clients they get, interactions with clients are critical and determine cash flow, project progression, security considerations and opportunities for future jobs. Developers are individuals operating in a world that expects them to develop flawless products in a bubble without reactions to various factors in the world they work. This study illuminates the experiences of developers in SMEs and their encounter various individual, organisational, social, and technical practices that shape their practice and how they approach security in software development.

Developers also rely on the shared meanings of security and quick conversations in their communities (virtually or in-person), in addition to their intuitions and personal experiences to make sense of potential security threats and countermeasures. For developers in SMEs, particularly, knowledge exchange in their communities may be the only accessible resource. Some categories of developers in these communities are called "celebrity developers", ambassadors or champions of software development tips and techniques. Recommendations from "celebrity developers" are trusted and can influence how developers approach secure software development practices. Consequently, "celebrity developers" may or may not

champion secure software development and security practices. The idea of celebrity developers may raise questions such as, "how are celebrity developers chosen or what is the process of becoming a celebrity developer?", "What happens if recommendations from community members have severe consequences for the software?" Further in-depth studies of social dynamics among developers' communities and their interactions with security communities are essential in identifying its impact on security practices in software application development.

This study is motivated by the question: '**How do software application developers in SMEs practice security effectively?'** The objective was to understand and explain the practices of developers in SMEs (and micro-businesses) by building a theory that explains the core concepts that shape these practices. The main research question is further broken into following sub-questions:

i. **RQ1:** What current key socio-technical factors are relevant to the security practices of software application developers in SMEs? – To scope and frame the study.
ii. **RQ2:** How can we theorise the relationship between these socio-technical factors and the security practices of app developers in SMEs? – To build and evaluate a theory of the security practices for developers in SMEs.
iii. **RQ3**: What recommendations can insights from the theory and study provide developers in SMEs? - To support the security practices of SME developers during the software development process.

The CAMS taxonomy is a classification of socio-technical factors found to be relevant to current security practices of developers in SMEs during the application development process. The taxonomy is an organisation system that shows the relationship between initial conceptual categories and their sub-categories resulting from coding phases in the grounded theory process. Four main conceptual categories were identified, Context, Attitude, Motivation and Support that are intertwined to reflect factors that may influence how secure software development is being practiced by developers in SMEs.

The grounded theory process was used to generate the theory of *Socio-technical Security Practices of Developers in SMEs* that explains the experiences and concerns of the participants in this study. And the *CAMS-IOT* conceptual framework was developed to enable an integration of core concepts and comparison with findings from existing related studies. Many studies done within security and software engineering research do not generate theories or conceptualise practices in a way that informs secure software development, particularly within smaller social and organisational environments. Therefore, generating theories that explain the individual, organisational and technical complexities of secure software development practices can serve as theoretical bases for more empirical research in security and software engineering fields. Other matured disciplines have established theories that support their studies and researchers in software engineering and security are now pushing for the creation and use of more theories, which is a contribution of this study

## 8.1 Summary of Theory and Conceptual Framework

**RQ2:** How can we theorise the relationship between these socio-technical factors and the security practices of app developers in SMEs? – To build and evaluate a theory of the security practices for developers in SMEs.

The theory of *Socio-technical Security Practices of Developers in SMEs* is specific to developers in small and medium-sized companies. It explains a phenomenon from the point of view of those experiencing the phenomena to show what is needed to understand the complexities of developer's practices in SMEs. However, its elements of this theory are familiar to readers in this subject area. A recap of the theoretical statement below shows that independently studying the conceptual categories (Context, People, Knowledge Bases, and Usable and Useful Security Tools and Technologies) may only give us parts of the complete picture of what developers face. Together, they aggregate the knowledge of developers' perspectives of security in software development. Furthermore, the theory is original because it provides a new conceptualisation for future study of SMEs' secure software development practices. The theoretical statement states that:

*For app developers in SMEs to effectively practice (implement best practices) secure*

> *software app development to produce (deliver) a secured product, they need to consider the **context** in which this product will be used, gather information from their individual or group security experience **knowledge base**, clearly identify the **people** involved in the process (clients, end-users, and other team members) and have access to **usable and useful security tools and technologies**.*

Context, People, Knowledge Bases, and Usable and Useful Security Tools and Technologies are interdependent elements central to the continuous implementation of security activities that become practices.

## 8.2 Using the Theory to Inform Secure Software Development Practices

The following recommendations from this study address the final research question:

**RQ3**: What recommendations can insights from the theory and study provide app developers in SMEs that supports their cyber security practices in the development lifecycle?

The recommendations from this study for app developers in SMEs to effectively practice security in software development is to move the focus from stiff technical aspects of cyber security and secure software development to a more socio-technical approach where social factors are considered just as important as the technical. The theory of *Socio-technical Security Practices of Developers in SMEs* and *CAMS-IOT* Framework offers a socio-technical lens for all individuals involved in the software development to understand that security practices and secure software development go beyond using the right tools, techniques, or technologies. Although, the theory only represents a partial description of from the developer's point-of-view, it still provides a frame for discussing the social dynamics of others involved in the development process.

## 8.3 Contributions

The theory of *Socio-technical Security Practices of Developers in SMEs* and CAMS-IOT Framework represents a dialectic between human activities (developers, clients, and end-users), contextual factors, a security knowledge base (a collection of security expert knowledge), and usable security technology within an organisational ecosystem. The dynamic relationship between these concepts working back and forth together contributes to the process of how developers prioritise or deprioritise security practices during the application development process.

i. The theory and framework provides the basis for explaining the activities of secure software development by examining the elements that keep developers from operationalising recommended measures. This can inform and support security initiatives to focus on secure software development within the SME context.
ii. It also provides developers with a frame for understanding the interactions between the technical aspects of secure app development and the social fabric that holds the way security activities are being implemented.
iii. The theory also supports the creation of future security practices that are more suitable for the context of developers in SMEs and informed by their current practices.

This study makes the following contributions:

- **Technical contribution**: the study produces the CAMS taxonomy, CAMS-IOT Framework and the theory of *Socio-technical Security Practices of Developers in SMEs* explains how context, attitudes, motivation, and support affects the formation of secure software development practices.
- **Methodological contribution**: the study demonstrates the usefulness and benefits of grounded theory methodology in secure software development research in broadening our knowledge through cross disciplinary research.
- **Academic contribution**: the findings of this study contribute to the software engineering and secure software development bodies of knowledge by aggregating

concepts in a theory that is a base for more empirical studies. In addition, generating theories that explain security practices in software development can serve as theoretical bases for more empirical research in security and software engineering disciplines. Other matured disciplines have established theories that support their studies and researchers in software engineering and security are now pushing for the creation and use of more theories, which is a motivation behind this study.

Other contributions of this study when compared to existing literature:

i. **Size of participating organisation** – some studies had no indication of the organisation sizes; some were majorly large organisations. This study focuses on the experiences of developers in SMEs.

ii. **Participant selection criteria** – this study involved developers only not other professionals within the development process. Other research studies include computer science students, consultants, security experts, software professionals (developers and non-developers in SDLC).

iii. No indication of the **Roles and responsibilities of developers in SMEs** and how this influences security decision-making in the development process in most studies. This study focuses on the experiences, roles, and responsibilities of developers in SMEs.

iv. Only a few studies have taken a deep dive to discuss the impact of **social interactions, dynamics and dialogues between developers,** API developers and security experts and clients. This study contributes to the limited studies around dialogues and negotiations between developers and clients (particularly for SMEs who have more interactions with their clients than in larger organisations). Understanding the prioritisation, trade-offs, and negotiations (resource and time) in security decision-making and software project management made by developers in SMEs in addition to their responsibilities as programmers.

v. **Partial implementation on grounded theory methodology** – particularly using the coding techniques only with no theory formation. The theory of *Socio-technical*

*Security Practices of Developers in SMEs* was generated based on the data gathered in this study.

vi. **The need for a more holistic approach** that considers context, stakeholders, and other factors in addition to the individual, organisational and technical aspects of secure software development.

vii. **Beyond security activities surrounding coding tasks** – majority of literature focuses on technological interventions, tool-based static analysis, or code reviews. However, this study argues that a socio-technical approach is required to better understand the secure software development practices of developers in SMEs.

## 8.4 Study Limitations

Conducting empirically grounded research in secure software application development and security are complex and come with some difficulties. One major difficulty of is that, not all developers are appropriate for taking part as a participant. Participants must be representative of experience, task domain and organisational context related to this study. Recruiting participants posed a challenge during the early part of this study. Probably because investigating security practices is also considered sensitive, it was difficult to gain access to developers or their work environments in order to observe their day-to day activities.

Another limitation of this study is the acceptance of the specific type of social research methods and methodologies in software engineering and security research disciplines. The interdisciplinary nature of this study means that it needed me to employ core sociological methods that help us better understand the experiences of developers, which goes beyond only understanding human interactions with technology but the context surrounding their activities and practices during the development lifecycle.

## 8.5 Considerations for Future Research

Fostering better cyber security practices depend on various activities, negotiations, communications, tools, and technologies, that happen within the software development process such as team collaboration and communication methods, knowledge sharing and learning in addition to complying with standard procedures or policies (Freed 2014; Wen 2018). This study explores various way developers in SMEs security practices within development activities, to better understand the influence of socio-technical factors on cyber security practices in the application development process. It focuses on human, social, and organisational research area for software engineering and cyber security.

Therefore, future research in these areas can:

i. Carry out ethnographic research studies that involve observing developers' daily activities during the development lifecycle.
ii. Effective mechanisms for communal security knowledge sharing among app developers was crucial for developers in SMEs. In-depth investigations of how security knowledge sharing in communities can optimise security practices will be useful.
iii. In addition, researchers should further investigate social interactions and dynamics between software developers' and security professionals' communities. This will contribute to finding new ways of properly disseminating information across communities.

# References


Abraham, Sherly. 2011. “Information Security Behavior: Factors and Research Directions.” In *AMCIS*.

Acar, Yasemin, Michael Backes, Sascha Fahl, Doowon Kim, Michelle L Mazurek, and Christian Stransky. 2016. “You Get Where You’re Looking for: The Impact of Information Sources on Code Security.” In *2016 IEEE Symposium on Security and Privacy (SP)*, 289–305. IEEE.

Acar, Yasemin, Michael Backes, Sascha Fahl, Doowon Kim, Michelle L. Mazurek, and Christian Stransky. 2017. “How Internet Resources Might Be Helping You Develop Faster but Less Securely.” In *IEEE Security and Privacy*. Vol. 15. https://doi.org/10.1109/MSP.2017.24.

Acar, Yasemin, Sascha Fahl, and Michelle L Mazurek. 2016. “You Are Not Your Developer, Either: A Research Agenda for Usable Security and Privacy Research beyond End Users.” In *2016 IEEE Cybersecurity Development (SecDev)*, 3–8. IEEE.

Acar, Yasemin, Christian Stransky, Dominik Wermke, Charles Weir, Michelle L Mazurek, and Sascha Fahl. 2017. “Developers Need Support, Too: A Survey of Security Advice for Software Developers.” In *Cybersecurity Development (SecDev), 2017 IEEE*, 22–26. IEEE.

Adams, William C. 2015. “Conducting Semi-structured Interviews.” *Handbook of Practical Program Evaluation*, 492–505.

Adolph, William Stephen. 2013. “Reconciling Perspectives: A Substantive Theory of How People Manage the Process of Software Development.”

Al-Amin, Shams, Nirav Ajmeri, Hongying Du, Emily Z Berglund, and Munindar P Singh. 2018. “Toward Effective Adoption of Secure Software Development Practices.” *Simulation Modelling Practice and Theory* 85: 33–46. https://doi.org/https://doi.org/10.1016/j.simpat.2018.03.006.

Albrechtsen, Eirik. 2007. “A Qualitative Study of Users’ View on Information Security.” *Computers & Security* 26 (4): 276–89. https://doi.org/http://dx.doi.org/10.1016/j.cose.2006.11.004.

Alexandre, Simon, Alain Renault, and Naji Habra. 2006. “OWPL: A Gradual Approach for Software Process Improvement in SMEs.” In *Software Engineering and Advanced Applications, 2006. SEAA’06. 32nd EUROMICRO Conference On*, 328–35. IEEE.

Aljedaani, Bakheet, Aakash Ahmad, Mansooreh Zahedi, and M Ali Babar. 2020. “An Empirical Study on Developing Secure Mobile Health Apps: The Developers’ Perspective.” In *2020 27th Asia-Pacific Software Engineering Conference (APSEC)*, 208–17. IEEE.

Alshenqeeti, Hamza. 2014. “Interviewing as a Data Collection Method: A Critical Review.” *English Linguistics Research* 3 (1). https://doi.org/10.5430/elr.v3n1p39.

Amrin, Nabila. 2014. “The Impact of Cyber Security on SMEs.” University of Twente.

Apvrille, Axelle, and Makan Pourzandi. 2005. “Secure Software Development by Example.” *IEEE Security & Privacy* 3 (4): 10–17.

Arabo, Abdullahi. 2016. “Mobile App Collusions and Its Cyber Security Implications.” In *Cyber Security and Cloud Computing (CSCloud), 2016 IEEE 3rd International Conference On*, 178–83. IEEE.

Ashenden, Debi, and Darren Lawrence. 2016. “Security Dialogues: Building Better Relationships between Security and Business.” *IEEE Security and Privacy* 14 (3). https://doi.org/10.1109/MSP.2016.57.

Ashenden, Debi, and Gail Ollis. 2020. “Putting the Sec in DevSecOps: Using Social Practice Theory to Improve Secure Software Development.” In *ACM International Conference Proceeding Series*. https://doi.org/10.1145/3442167.3442178.

Assal, Hala. 2018. “The Human Dimension of Software Security and Factors Affecting Security Processes.” Carleton University.

Assal, Hala, and Sonia Chiasson. 2018. “Security in the Software Development Lifecycle.” In *Fourteenth Symposium on Usable Privacy and Security ({SOUPS} 2018)*, 281–96.

———. 2019. “‘Think Secure from the Beginning’: A Survey with Software Developers.”

Baca, Dejan, and Bengt Carlsson. 2011. “Agile Development with Security Engineering Activities.” In *Proceedings - International Conference on Software Engineering*. https://doi.org/10.1145/1987875.1987900.

Bacharach, Samuel B. 1989. “Organizational Theories: Some Criteria for Evaluation.” *Academy of Management Review* 14 (4): 496–515.

Backhouse, James, and Gupreet Dhillon. 1996. “Structures of Responsibility and Security of Information Systems.” *European Journal of Information Systems* 5 (1): 2–9.

Bailey, Carol R, and Carol A Bailey. 2017. *A Guide to Qualitative Field Research*. Sage Publications.

Balebako, Rebecca, Abigail Marsh, Jialiu Lin, Jason I Hong, and Lorrie Faith Cranor. 2014. “The Privacy and Security Behaviors of Smartphone App Developers.”

Bartsch, S. 2011. “Practitioners’ Perspectives on Security in Agile Development.” In *2011 Sixth International Conference on Availability, Reliability and Security*, 479–84. https://doi.org/10.1109/ARES.2011.82.

Beck, Kent, Mike Beedle, Arie Van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, and Ron Jeffries. 2001. “Manifesto for Agile Software Development.”

Beckwith, Sue, Angela Dickinson, and Sally Kendall. 2008. “The ‘Con’ of Concept Analysis: A Discussion Paper Which Explores and Critiques the Ontological Focus, Reliability and Antecedents of Concept Analysis Frameworks.” *International Journal of Nursing Studies* 45 (12): 1831–41. https://doi.org/https://doi.org/10.1016/j.ijnurstu.2008.06.011.

Belgrave, Linda Liska, and Kapriskie Seide. 2019. “Coding for Grounded Theory.” *The SAGE Handbook of Current Developments in Grounded Theory*, 167–85.

Belk, Mark, M Coles, C Goldschmidt, M Howard, K Randolph, M Saario, R Sondhi, I Tarandach, A Vaha-Sipila, and Y Yonchev. 2014. “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*, 1. Springer.

Birrell, N D, and M A Ould. 1988. *A Practical Handbook for Software Development*. Cambridge University Press. https://books.google.co.uk/books?id=N2BItNyaRTYC.

Bjørner, Dines. 2006. *Software Engineering 3: Domains, Requirements, and Software Design*. Springer Science & Business Media.

Bødker, S. 2000. “Scenarios in User-Centred Design - Setting the Stage for Reflection and Action.” *Interacting with Computers* 13 (1). https://doi.org/10.1016/S0953-5438(00)00024-2.

Bourdieu, Pierre. 1977. “Outline Bourdieu, P. (1977). Outline of a Theory of Practice. (J. Goody, Ed.)Cambridge Studies in Social Anthropology (Vol. 16). Cambridge University Press. Doi:10.1590/S0103-20702013000100001of a Theory of Practice.” *Cambridge Studies in Social Anthropology* 16 (16).

Bruijn, Hans de, and Marijn Janssen. 2017. “Building Cybersecurity Awareness: The Need for Evidence-Based Framing Strategies.” *Government Information Quarterly* 34 (1). https://doi.org/10.1016/j.giq.2017.02.007.

Bryant, Antony, and Kathy Charmaz. 2010. “The SAGE Handbook of Grounded Theory: The Evolving Nature of Grounded Theory Method: The Case of the Information Systems Discipline.” *Forum Qualitative Sozialforschung/Forum: Qualitative Social Research* 11.

BSIMM by Synopsys. 2022. “BSIMM 13 Foundations Report.” https://www.synopsys.com/software-integrity/engage/bsimm-web/bsimm13-foundations.

Bueger, Christian. 2016. “Security as Practice.” In *Routledge Handbook of Security Studies: Second Edition*. https://doi.org/10.4324/9781315753393.

Burnett, Margaret. 2009. “What Is End-User Software Engineering and Why Does It Matter ?,” 15–28.

Button, Graham, and Wes Sharrock. 1994. “Occasioned Practises in the Work of Software Engineers.” *Requirements Engineering: Social and Technical Issues*.

Caputo, Deanna D., Shari Lawrence Pfleeger, M. Angela Sasse, Paul Ammann, Jeff Offutt, and Lin Deng. 2016. “Barriers to Usable Security? Three Organizational Case Studies.” *IEEE Security and Privacy* 14 (5). https://doi.org/10.1109/MSP.2016.95.

Carayon, Pascale, Peter Hancock, Nancy Leveson, Ian Noy, Laerte Sznelwar, and Geert van Hootegem. 2015. “Advancing a Sociotechnical Systems Approach to Workplace Safety – Developing the Conceptual Framework.” *Ergonomics* 58 (4). https://doi.org/10.1080/00140139.2015.1015623.

Carley, Kathleen. 1986. “Knowledge Acquisition as a Social Phenomenon.” *Instructional Science* 14 (3–4). https://doi.org/10.1007/BF00051829.

———. 1993. “Coding Choices for Textual Analysis: A Comparison of Content Analysis and Map Analysis.” *Sociological Methodology*, 75–126.

Casaca, Joaquim António, and T Florentino. 2014. “Information Security Research: Actual Trends and Directions.” *ICTIC 2014 (Conference of Informatics and Management Sciences)*, 251–56.

Chandra, Pravir, and Team OWASP. 2013. “Software Assurance Maturity Model., Version 1.0.” *Open Web Application Security Project*.

Charmaz, Kathy. 2006. *Constructing Grounded Theory: A Practical Guide through Qualitative Analysis*. Sage.

———. 2008. “Constructionism and the Grounded Theory Method.” *Handbook of Constructionist Research* 1: 397–412.

Charmaz, Kathy, and Robert Thornberg. 2021. “The Pursuit of Quality in Grounded Theory.” *Qualitative Research in Psychology* 18 (3). https://doi.org/10.1080/14780887.2020.1780357.

Chemuturi, Murali. 2009. *Software Estimation Best Practices, Tools & Techniques: A Complete Guide for Software Project Estimators*. J. Ross Publishing.

Chen, Mengsu, Felix Fischer, Na Meng, Xiaoyin Wang, and Jens Grossklags. 2019. “How Reliable Is the Crowdsourced Knowledge of Security Implementation.” In *Proceedings - International Conference on Software Engineering*. Vol. 2019-May. https://doi.org/10.1109/ICSE.2019.00065.

Chowdhury, Partha Das, Joseph Hallett, Nikhil Patnaik, Mohammad Tahaei, and Awais Rashid. 2021. “Developers Are Neither Enemies nor Users: They Are Collaborators.” In *2021 IEEE Secure Development Conference (SecDev)*, 47–55. IEEE.

Clarke, Steve, and Paul Drake. 2003. “A Social Perspective on Information Security: Theoretically Grounding the Domain.” *Socio-Technical and Human Cognition Elements of Information Systems*, 249–65.

Cockburn, Alistair. 2002. *Agile Software Development*. Vol. 177. Addison-Wesley Boston.

Cole, Stephen, Sheila Jasanoff, Gerald E. Markle, James C. Peterson, and Trevor Pinch. 1995. “Handbook of Science and Technology Studies.” *Contemporary Sociology* 24 (6). https://doi.org/10.2307/2076663.

Coleman, Gerry, and Rory O’Connor. 2008. “Investigating Software Process in Practice: A Grounded Theory Perspective.” *Journal of Systems and Software* 81 (5): 772–84. https://doi.org/https://doi.org/10.1016/j.jss.2007.07.027.

Colin Robson. 2002. *A Resource for Social Scientists and Practitioner-Researchers. Real World Research*.

Colwill, Carl. 2009. “Human Factors in Information Security: The Insider Threat–Who Can You Trust These Days?” *Information Security Technical Report* 14 (4): 186–96.

Corbin, Juliet M, and Anselm Strauss. 1990. “Grounded Theory Research: Procedures, Canons, and Evaluative Criteria.” *Qualitative Sociology* 13 (1): 3–21.

Corbin, Juliet, and Anselm Strauss. 2014. *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*. Sage publications.

Creswell, John W, and Abbas Tashakkori. 2007. “Differing Perspectives on Mixed Methods Research.” Sage Publications Sage CA: Los Angeles, CA.

Das, Sauvik. 2017. “Social Cybersecurity: Reshaping Security Through An Empirical Understanding of Human Social Behavior.”

Das, Sauvik, Tiffany Hyun-Jin Kim, Laura A Dabbish, and Jason I Hong. 2014. “The Effect of Social Influence on Security Sensitivity.” In *10th Symposium On Usable Privacy and Security ({SOUPS} 2014)*, 143–57.

DeMarco, Tom, and Tim Lister. 2013. *Waltzing with Bears: Managing Risk on Software Projects*. Addison-Wesley.

Dhillon, Gurpreet, and James Backhouse. 2001. “Current Directions in IS Security Research: Towards Socio-Organizational Perspectives.” *Information Systems Journal* 11 (2): 127–53. https://doi.org/10.1046/j.1365-2575.2001.00099.x.

Easterbrook, Steve, Janice Singer, Margaret-Anne Storey, and Daniela Damian. 2008. “Selecting Empirical Methods for Software Engineering Research.” In *Guide to Advanced Empirical Software Engineering*, 285–311. Springer.

EDC. 2021. “EDC Worldwide Developer Population and Demographic Study 2021 V1.”

EPSRC. 2013. “Framework for Responsible Innovation.” Https://Epsrc.Ukri.Org/Index.Cfm/Research/Framework/. 2013.

Fagerberg, Jan, Morten Fosaas, and Koson Sapprasert. 2018. “Innovation: Exploring the Knowledge Base.” In *Innovation, Economic Development and Policy: Selected Essays*. https://doi.org/10.1016/j.respol.2012.03.008.

Faily, Shamal. 2011. “A Framework for Usable and Secure System Design.” University of Oxford.

Feller, Joseph, and Brian Fitzgerald. 2002. *Understanding Open Source Software Development*. Addison-Wesley London.

Fischer, Eric A. 2005. "Creating a National Framework for Cybersecurity: An Analysis of Issues and Options." In . Library of Congress Washington DC Congressional Research Service.

———. 2016. "Cybersecurity Issues and Challenges: In Brief." *Congressional Research Service*.

Flechais, Ivan. 2005. "Designing Secure and Usable Systems." University College London.

Freed, Sarah Ellen. 2014. "Examination of Personality Characteristics Among Cybersecurity and Information Technology Professionals."

Fuggetta, Alfonso, and Elisabetta Di Nitto. 2014. "Software Process." In *Proceedings of the on Future of Software Engineering*, 1–12. ACM.

Gabriel, D. 2013. "Inductive and Deductive Approaches to Research." *Www.Deborahgabriel.Com*.

Geer, David. 2010. "Are Companies Actually Using Secure Development Life Cycles?" *Computer* 43 (6): 12–16.

Gerogiannis, Vassilis, George Kakarontzas, Leonidas Anthopoulos, Stamatia Bibi, and Ioannis Stamelos. 2013. "The SPRINT-SMEs Approach for Software Process Improvement in Small-Medium Sized Software Development Enterprises." *Proceedings of ARCHIMEDES III*.

Ghaoui, Claude. 2005. *Encyclopedia of Human Computer Interaction*. Idea Group Inc (IGI). https://books.google.com/books?id=h9iZh_I1YREC&pgis=1.

Gibbs, Graham. 2007. *Analysing Qualitative Data - The Sage Qualitative Research Kit. The SAGE Qualitative Reseach Kit*.

Glaser, Barney G. 1998. "Doing Grounded Theory: Issues and Discussion." *Recherche* 67.

———. 2001. “The Grounded Theory Perspective: Conceptualization Contrasted with Description.” *CA: Sociology Press*.

Glaser, Barney G, and Anselm L Strauss. 1967. “Discovery of Grounded Theory. Mill Valley.” *CA: Sociology*.

———. 2017. *Discovery of Grounded Theory: Strategies for Qualitative Research*. Routledge.

Gorski, Peter Leo, Yasemin Acar, Luigi Lo Iacono, and Sascha Fahl. 2020. “Listen to Developers! A Participatory Design Study on Security Warnings for Cryptographic APIs.” In *Conference on Human Factors in Computing Systems - Proceedings*. https://doi.org/10.1145/3313831.3376142.

Green, Matthew, and Matthew Smith. 2016. “Developers Are Not the Enemy!: The Need for Usable Security APIs.” *IEEE Security & Privacy* 14 (5): 40–46.

Guba, Egon G, and Yvonna S Lincoln. 2005. “Paradigmatic Controversies, Contradictions, and Emerging Confluences.”

Gutfleisch, Marco, Jan H Klemmer, Niklas Busch, Yasemin Acar, M Angela Sasse, and Sascha Fahl. 2022. “How Does Usable Security (Not) End up in Software Products? Results from a Qualitative Interview Study.” In *2022 IEEE Symposium on Security and Privacy (SP)*, 893–910. IEEE.

Hala, Hala Assal, and Sonia Chiasson. 2018. “Motivations and Amotivations for Software Security.” *WSIW’18 (SOUPS Workshop on Security Information Workers)*.

Hall, Tracy, Helen Sharp, Sarah Beecham, Nathan Baddoo, and Hugh Robinson. 2008. “What Do We Know about Developer Motivation?” *IEEE Software* 25 (4). https://doi.org/10.1109/MS.2008.105.

Haney, Julie M., Mary F. Theofanos, Yasemin Acar, and Sandra Spickard Prettyman. 2019. “‘We Make It a Big Deal in the Company’: Security Mindsets in Organizations That Develop Cryptographic Products.” In *Proceedings of the 14th Symposium on Usable Privacy and Security, SOUPS 2018*.

Hasna, Elkhannoubi, and Belaissaoui Mustapha. 2016. “User’s Behaviors Influence on Cybersecurity Strategy Effectiveness.” *International Journal of Advanced Engineering Research and Science* 3 (10): 188–96.

Henninger, Scott. 2003. “Tool Support for Experience-Based Software Development Methodologies.” *Advances in Computers* 59 (1): 29–82.

Herath, Tejaswini, and H Raghav Rao. 2009. “Protection Motivation and Deterrence: A Framework for Security Policy Compliance in Organisations.” *European Journal of Information Systems* 18 (2): 106–25. https://doi.org/10.1057/ejis.2009.6.

Herrmann, Thomas, and Sabine Pfeiffer. 2023. “Keeping the Organization in the Loop: A Socio-Technical Extension of Human-Centered Artificial Intelligence.” *AI & SOCIETY* 38 (4): 1523–42.

Hester, Andrea J. 2014. “Socio-Technical Systems Theory as a Diagnostic Tool for Examining Underutilization of Wiki Technology.” *Learning Organization* 21 (1). https://doi.org/10.1108/TLO-10-2012-0065.

Hodkinson, Paul. 2008. “Grounded Theory and Inductive Research.” In *Researching Social Life*.

Hoffman, Ana. 2020. “Four Advantages to Using Agile Processes in ECommerce and Web Development.” Https://Dandelife.Com/Advantages-to-Using-Agile-Processes-in-Ecommerce-and-Web-Development/. 2020.

Howard, Michael. 2004. “Building More Secure Software with Improved Development Processes.” *IEEE Security & Privacy* 2 (6): 63–65.

Howard, Michael, and Steve Lipner. 2006. *The Security Development Lifecycle*. Vol. 8. Microsoft Press Redmond.

Hu, Yupeng, Wenxin Kuang, Zheng Qin, Kenli Li, Jiliang Zhang, Yansong Gao, Wenjia Li, and Keqin Li. 2021. “Artificial Intelligence Security: Threats and Countermeasures.” *ACM Computing Surveys (CSUR)* 55 (1): 1–36.

Humphrey, Watts S, David H Kitson, and Tim C Kasse. 1989. “The State of Software Engineering Practice.” In *Proceedings of the 11th International Conference on Software Engineering*, 277–85.

ISO/IEC. 2014. “90003:2014, Software Engineering – Guidelines for the Application of ISO 9001:2008 to Computer Software.” In *Switzerland, Geneva*.

Jabareen, Yosef. 2009. “Building a Conceptual Framework: Philosophy, Definitions, and Procedure.” *International Journal of Qualitative Methods* 8 (4): 49–62.

Jackson, Michael. 1994. “Problems, Methods and Specialisation.” *Software Engineering Journal* 9 (6): 249–56.

Jarzębowicz, Aleksander, and Natalia Sitko. 2019. “Communication and Documentation Practices in Agile Requirements Engineering: A Survey in Polish Software Industry.” In *Information Systems: Research, Development, Applications, Education: 12th SIGSAND/PLAIS EuroSymposium 2019, Gdansk, Poland, September 19, 2019, Proceedings 12*, 147–58. Springer.

Jirotka, Marina, Barbara Grimpe, Bernd Stahl, Grace Eden, and Mark Hartswood. 2017. “Responsible Research and Innovation in the Digital Age.” *Communications of the ACM* 60 (5). https://doi.org/10.1145/3064940.

Jones, Capers. 2003. “Variations in Software Development Practices.” *IEEE Software* 20 (6): 22–27.

———. 2004. “Software Project Management Practices: Failure versus Success.” *CrossTalk: The Journal of Defense Software Engineering* 17 (10): 5–9.

———. 2005. “Software Cost Estimating Methods for Large Projects.” *CrossTalk: The Journal of Defense Software Engineering*, 8–12.

Joorabchi, Mona Erfani, Ali Mesbah, and Philippe Kruchten. 2013. “Real Challenges in Mobile App Development.” In *Empirical Software Engineering and Measurement, 2013 ACM/IEEE International Symposium On*, 15–24. IEEE.

Jordan, Tiffany Brooke, Brittany Johnson, Jim Witschey, and Emerson Murphy-Hill. 2014. "Designing Interventions to Persuade Software Developers to Adopt Security Tools." In *Proceedings of the 2014 ACM Workshop on Security Information Workers*, 35–38.

Jøsang, Audun, Marte Ødegaard, and Erlend Oftedal. 2015. "Cybersecurity through Secure Software Development." In *Information Security Education Across the Curriculum: 9th IFIP WG 11.8 World Conference, WISE9, Hamburg, Germany, May 26–28, 2015, Proceedings 9*, 53–63. Springer.

Kaboub, Fadhel. 2008. "Positivist Paradigm." *Encyclopaedia of Counselling* 2 (2): 343.

Kahkonen, Tuomo. 2004. "Agile Methods for Large Organizations-Building Communities of Practice." In *Agile Development Conference*, 2–10. IEEE.

Karin Knorr Cetina, T.R.S.E.S., T R Schatzki, K Knorr-Cetina, and E von Savigny. 2001. *The Practice Turn in Contemporary Theory*. Routledge. https://books.google.co.uk/books?id=RfaVpJBB5lgC.

Kazymyr, Volodymyr, and A Mokrohuz. 2016. "Information Technologies of Mobile Applications Development." *Технічні Науки Та Технології*, no. 2: 156–62.

Keramati, Hossein, and Seyed-Hassan Mirian-Hosseinabadi. 2008. "Integrating Software Development Security Activities with Agile Methodologies." In *2008 IEEE/ACS International Conference on Computer Systems and Applications*, 749–54. IEEE.

Khalid, Asra, Sobia Zahra, and Muhammad Fahad Khan. 2014. "Suitability and Contribution of Agile Methods in Mobile Software Development." *International Journal of Modern Education and Computer Science* 6 (2): 56.

King, Louise Margaret. 2019. "Applying Social Practice Theory to Contemporary Working Practices in Sustainable Office Buildings: Implications for the Performance Gap." University of the West of England.

Koops, Bert-Jaap. 2015. *The Concepts, Approaches, and Applications of Responsible Innovation: An Introduction*. Springer.

Koutsoumpos, Vasileios, and Iker Marinelarena. 2013. “Agile Methodologies and Software Process Improvement Maturity Models, Current State of Practice in Small and Medium Enterprises.”

Kraemer, S, P Carayon, and J Clem. 2009. “Human and Organizational Factors in Computer and Information Security: Pathways to Vulnerabilities.” *Computers & Security*.

Kraut, Robert E, and Lynn A Streeter. 1995. “Coordination in Software Development.” *Communications of the ACM* 38 (3): 69–81.

Kritzinger, E, and S H von Solms. 2010. “Cyber Security for Home Users: A New Way of Protection through Awareness Enforcement.” *Computers & Security* 29 (8): 840–47. https://doi.org/http://dx.doi.org/10.1016/j.cose.2010.08.001.

Kumar, Manish, Shilpi Jain, Sheetal Payyavula, Jude Fernandez, and Avitash Purohit. 2012. “Software Project Characteristics And Their Measures: Towards A Comprehensive Framework.” In *Proceedings of The*.

Kyophilavong, Phouphet. 2007. “SME Development in Lao PDR.” In *Third Workshop, the ERIA Related Joint Research of SME Project, IDE-JETRO*, 13–14.

LaToza, Thomas D, and Brad A Myers. 2010. “On the Importance of Understanding the Strategies That Developers Use.” In *Proceedings of the 2010 ICSE Workshop on Cooperative and Human Aspects of Software Engineering*, 72–75. ACM.

Leau, Yu Beng, Wooi Khong Loo, Wai Yip Tham, and Soo Fun Tan. 2012. “Software Development Life Cycle AGILE vs Traditional Approaches.” In *International Conference on Information and Network Technology*, 37:162–67.

Lebek, Benedikt, Jorg Uffen, Michael H Breitner, Markus Neumann, and Bernd Hohler. 2013. “Employees’ Information Security Awareness and Behavior: A Literature Review.” In *System Sciences (HICSS), 2013 46th Hawaii International Conference On*, 2978–87. IEEE.

Leblanc, Jeffrey. 2012. “Designing the Gender- Neutral User Experience.”

Lee, W, and S Jang. 2009. “A Study on Information Security Management System Model for Small and Medium Enterprises.” *Recent Advances in E-Activities, Information Security and Privacy*, 84–87.

Leonard, Lori N K, Timothy Paul Cronan, and Jennifer Kreie. 2004. “What Influences IT Ethical Behavior Intentions—Planned Behavior, Reasoned Action, Perceived Importance, or Individual Characteristics?” *Information & Management* 42 (1): 143–58.

Lethbridge, Timothy Christian, and Robert Laganiere. 2005. *Object-Oriented Software Engineering*. McGraw-Hill New York.

Linden, Dirk Van Der, Pauline Anthonysamy, Bashar Nuseibeh, Thein Than Tun, Marian Petre, Mark Levine, John Towse, and Awais Rashid. 2020. “Schrödinger’s Security: Opening the Box on App Developers’ Security Rationale.” In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, 149–60.

Lipner, Steve. 2010. “Security Development Lifecycle.” *Datenschutz Und Datensicherheit - DuD* 34 (3): 135–37. https://doi.org/10.1007/s11623-010-0021-7.

Lopez, Tamara, Helen Sharp, Thein Tun, Arosha Bandara, Mark Levine, and Bashar Nuseibeh. 2019a. “Hopefully We Are Mostly Secure’: Views on Secure Code in Professional Practice.” In *Proceedings - 2019 IEEE/ACM 12th International Workshop on Cooperative and Human Aspects of Software Engineering, CHASE 2019*. https://doi.org/10.1109/CHASE.2019.00023.

———. 2019b. “Talking about Security with Professional Developers.” In *2019 IEEE/ACM Joint 7th International Workshop on Conducting Empirical Studies in Industry (CESI) and 6th International Workshop on Software Engineering Research and Industrial Practice (SER&IP)*, 34–40. IEEE.

Lopez, Tamara, Thein T Tun, Arosha K Bandara, Mark Levine, Bashar Nuseibeh, and Helen Sharp. 2019. “Taking the Middle Path: Learning about Security through Online Social Interaction.” *IEEE Software* 37 (1): 25–30.

MacIntyre, Alasdair. 1987. “Practical Rationalities as Forms of Social Structure.” *Irish Philosophical Journal* 4 (1/2): 3–19.

Madden, Thomas J., Pamela Scholder Ellen, and Icek Ajzen. 1992. “A Comparison of the Theory of Planned Behavior and the Theory of Reasoned Action.” *Personality and Social Psychology Bulletin* 18 (1). https://doi.org/10.1177/0146167292181001.

Maher, Zulfikar Ahmed, Humaiz Shaikh, Mohammad Shadab Khan, Ammar Arbaaeen, and Asadullah Shah. 2019. “Factors Affecting Secure Software Development Practices among Developers-An Investigation.” In *2018 IEEE 5th International Conference on Engineering Technologies and Applied Sciences, ICETAS 2018*. https://doi.org/10.1109/ICETAS.2018.8629168.

Malatji, Masike, Sune Von Solms, and Annlizé Marnewick. 2019. “Socio-Technical Systems Cybersecurity Framework.” *Information and Computer Security* 27 (2). https://doi.org/10.1108/ICS-03-2018-0031.

Mashiane, Thulani, and Elamarie Kritzinger. 2020. “Theoretical Domains Framework Applied to Cybersecurity Behaviour.” In *Advances in Intelligent Systems and Computing*. Vol. 1225 AISC. https://doi.org/10.1007/978-3-030-51971-1_34.

McCallin, Antoinette M. 2003. “Designing a Grounded Theory Study: Some Practicalities.” *Nursing in Critical Care* 8 (5): 203–8.

McCaslin, Mark L, and Karen Wilson Scott. 2003. “The Five-Question Method for Framing a Qualitative Research Study.” *The Qualitative Report* 8 (3): 447–61.

McCown, Chris. 2003. “Framework for Secure Application Design and Development.” *SANS Reading Room, Http://Www. sans. Org/Rr/Paper. Php*. https://www.sans.org/white-papers/1481/.

McGraw, Gary. 1998. “Testing for Security during Development: Why We Should Scrap Penetrate-and-Patch.” *IEEE Aerospace and Electronic Systems Magazine* 13 (4): 13–15.

———. 2004. “Software Security.” *IEEE Security & Privacy* 2 (2): 80–83.

McGraw, Gary, Sammy Migues, and Jacob West. 2018. “Building Security In Maturity Model (BSIMM) Version 9.”

McManus, John. 2012. *Risk Management in Software Development Projects*. Routledge.

Meng, Na, Stefan Nagy, Danfeng (Daphne) Yao, Wenjie Zhuang, and Gustavo Arango Argoty. 2018. “Secure Coding Practices in Java.” In . https://doi.org/10.1145/3180155.3180201.

Microsoft Corp. 2004. “Microsoft Security Development Lifecycle.” Https://Www.Microsoft.Com/En-Us/Securityengineering/Sdl/Practices. 2004.

———. 2023. “Responsible Innovation: A Best Practices Toolkit.” 2023. https://learn.microsoft.com/en-us/azure/architecture/guide/responsible-innovation/.

Mills, Karen. 2015. “The 4 Types of Small Businesses, and Why Each One Matters.” *Harvard Business Review. April 2015*.

Miner, John B. 2020. “Sociotechnical Systems Theory: Eric Trist, Fred Emery.” In *Organizational Behavior 2*. https://doi.org/10.4324/9781315702001-21.

Mishra, Ankur, Ruchita Mathur, Shishir Jain, and Jitendra Singh Rathore. 2013. “Cloud Computing Security.” *International Journal on Recent and Innovation Trends in Computing and Communication* 1 (1): 36–39.

Mishra, Deepti, and Alok Mishra. 2009. “Software Process Improvement in SMEs: A Comparative View.” *Computer Science and Information Systems* 6 (1): 111–40.

Mitchell, Donald. 2014. “Advancing Grounded Theory: Using Theoretical Frameworks within Grounded Theory Studies.” *The Qualitative Report* 19 (36): 1–11.

Mokhberi, Azadeh, and Konstantin Beznosov. 2021. “SoK: Human, Organizational, and Technological Dimensions of Developers’ Challenges in Engineering Secure Software.” In *Proceedings of the 2021 European Symposium on Usable Security*, 59–75.

Molich, Rolf, Robin Jeffries, and Joseph Dumas. 2007. “Making Usability Recommendations Useful and Usable.” *Journal of Usability Studies* 2 (4).

Muhirwe, Jackson, and Nathan White. 2016. “Cybersecurity Awareness and Practice of Next Generation Corporate Technology Users.” *Issues in Information Systems* 17 (2).

Munro, David. 2013. “What Is an SME?” In *A Guide to SME Financing*, 7–13. Springer.

Naz, Riffat, and M N A Khan. 2015. “Rapid Applications Development Techniques: A Critical Review.” *International Journal of Software Engineering and Its Applications* 9 (11): 163–76.

NCSC. 2018. “National Cyber Security Centre - Secure Development and Deployment Guidance.” 2018. https://www.ncsc.gov.uk/collection/developers-collection.

Northrop, Erik E., and Heather Richter Lipford. 2014. “Exploring the Usability of Open Source Network Forensic Tools,” 1–8. https://doi.org/10.1145/2663887.2663903.

Observer, OECD. 2000. “Small and Medium-Sized Enterprises: Local Strength, Global Reach.” *Policy Brief*. http://www.oecd.org/cfe/leed/1918307.pdf.

Oliveira, Daniela Seabra, Tian Lin, Muhammad Sajidur Rahman, Rad Akefirad, Donovan Ellis, Eliany Perez, Rahul Bobhate, et al. 2019. “API Blindspots: Why Experienced Developers Write Vulnerable Code.” In *Proceedings of the 14th Symposium on Usable Privacy and Security, SOUPS 2018*.

Oliveira, Daniela Seabra, Tian Lin, Muhammad Sajidur Rahman, Donovan Ellis, Eliany Perez, Rahul Bobhate, Rad Akefirad, et al. 2018. *API Blindspots: Why Experienced Developers Write Vulnerable Code API Blindspots: Why Experienced Developers Write Vulnerable Code*. *Fourteenth Symposium on Usable Privacy and Security*.

Opdenakker, Raymond. 2006. “Advantages and Disadvantages of Four Interview Techniques in Qualitative Research [44 Paragraphs]. Forum Qualitative Sozialforschung / Forum: Qualitative Social Research, 7(4), Art. 11,.” *Qualitative Social Research* 7 (4).

Orlikowski, Wanda J. 2002. “Knowing in Practice: Enacting a Collective Capability in Distributed Organizing.” *Organization Science* 13 (3): 249–73.

Ortner, Sherry B. 1984. “Theory in Anthropology since the Sixties.” *Comparative Studies in Society and History* 26 (1): 126–66.

Osborn, E, and A Simpson. 2015. “Small-Scale Cyber Security.” *2015 IEEE 2nd International Conference on Cyber Security and Cloud Computing*. https://doi.org/10.1109/CSCloud.2015.12.

Otero, Carlos. 2012. *Software Engineering Design: Theory and Practice*. CRC Press.

OWASP. 2017. "OWASP Top 10 - 2017: The Ten Most Critical Web Application Security Vulnerabilities." *OWASP Top 17*.

———. 2021. "OWASP Top Ten Web Application Security Risks | OWASP." OWASP. 2021.

———. 2023. "The New OWASP Top 10 API Security 2023." *OWASP*. https://owasp.org/www-chapter-singapore/assets/presos/OWASP_SG_6_Sep_2023_The_new_OWASP_Top_10_API_Security_2023.pdf.

OWASP, ASVS. 2018. "Open Web Application Security Project--Application Security Verification Standard." *Open Web Application Security Project*. https://www.owasp.org/index.php/Category:OWASP_Application_Security_Verification_Standard_Project.

OWASP SAMM. 2019. "OWASP Software Assurance Maturity Model." 2019. https://drive.google.com/file/d/1cI3Qzfrly_X89z7StLWI5p_Jfqs0-OZv/view.

Paasivaara, Maria, and Casper Lassenius. 2014. "Communities of Practice in a Large Distributed Agile Software Development Organization–Case Ericsson." *Information and Software Technology* 56 (12): 1556–77.

Päivärinta, Tero, and Kari Smolander. 2015. "Theorizing about Software Development Practices." *Science of Computer Programming* 101: 124–35. https://doi.org/https://doi.org/10.1016/j.scico.2014.11.012.

Palmquist, M E, and K M Carley. 1997. "Applications of Computer-Aided Text Analysis: Analyzing Literary and Nonliterary Texts." *Text Analysis for the Social Sciences. Methods for Drawing Statistical Inferences from Texts and Transcripts*, 171–89.

Palombo, Hernan, Armin Ziaie Tabari, Daniel Lende, Jay Ligatti, and Xinming Ou. 2020. "An Ethnographic Understanding of Software (In)Security and a Co-Creation Model to Improve Secure Software Development." In *Proceedings of the 16th Symposium on Usable Privacy and Security, SOUPS 2020*.

Parker, James, Michael Hicks, Andrew Ruef, Michelle L. Mazurek, Dave Levin, Daniel Votipka, Piotr Mardziel, and Kelsey R. Fulton. 2020. “Build It, Break It, Fix It: Contesting Secure Development.” *ACM Transactions on Privacy and Security* 23 (2). https://doi.org/10.1145/3383773.

Pattinson, Steven. 2011. “Cultivating Communities of Practice for Innovation: What about SMEs?”

Pattinson, Steven, and David Preece. 2014. “Communities of Practice, Knowledge Acquisition and Innovation: A Case Study of Science-Based SMEs.” *Journal of Knowledge Management* 18 (1): 107–20.

Pavie, Xavier, Daphné Carthy, and Victor Scholten. 2014. *Responsible Innovation: From Concept to Practice*. World Scientific.

“Payment Card Industry Data Security Standard.” 2009. *Card Technology Today* 21 (4). https://doi.org/10.1016/s0965-2590(09)70094-5.

Penuel, William R., Daniela K. Digiacomo, Katie Van Horne, and Ben Kirshner. 2016. “A Social Practice Theory of Learning and Becoming across Contexts and Time.” *Frontline Learning Research* 4 (4). https://doi.org/10.14786/flr.v4i4.205.

Pham, Lan. 2018. “A Review of Key Paradigms: Positivism, Interpretivism and Critical Inquiry.” *University of Adelaide*.

Philipson, Graeme. 2004. “A Short History of Software.” *Management, Labour Process and Software Development: Reality Bites* 13.

Pieczul, Olgierd, Simon Foley, and Mary Ellen Zurko. 2017. “Developer-Centered Security and the Symmetry of Ignorance.” In *Proceedings of the 2017 New Security Paradigms Workshop*, 46–56. ACM.

Pikkarainen, Minna, Jukka Haikara, Outi Salo, Pekka Abrahamsson, and Jari Still. 2008. “The Impact of Agile Practices on Communication in Software Development.” *Empirical Software Engineering* 13: 303–37.

Poller, Andreas, Laura Kocksch, Sven Türpe, Felix Anand Epp, and Katharina Kinder-Kurlanda. 2017. "Can Security Become a Routine? A Study of Organizational Change in an Agile Software Development Group." In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*, 2489–2503.

Polya, George. 2004. *How to Solve It: A New Aspect of Mathematical Method*. Vol. 85. Princeton university press.

Ponsard, C, and J Deprez. 2018. "Helping SMEs to Better Develop Software: Experience Report and Challenges Ahead." In *2018 IEEE/ACM 40th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP)*, 213–14.

Pranam, Aswin. 2018. "Software Development Methodologies." In *Product Management Essentials*, 65–74. Springer.

Ralph, Paul, Mike Chiasson, and Helen Kelley. 2016. "Social Theory for Software Engineering Research." In *Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering*, 44. ACM.

Rauf, Irum, Dirk van der Linden, Mark Levine, John Towse, Bashar Nuseibeh, and Awais Rashid. 2020. "Security but Not for Security's Sake: The Impact of Social Considerations on App Developers' Choices." In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, 141–44.

Rauf, Irum, Marian Petre, Thein Tun, Tamara Lopez, Paul Lunn, Dirk Van Der Linden, John Towse, Helen Sharp, Mark Levine, and Awais Rashid. 2021. "The Case for Adaptive Security Interventions." *ACM Transactions on Software Engineering and Methodology (TOSEM)* 31 (1): 1–52.

Reckwitz, Andreas. 2002. "Toward a Theory of Social Practices: A Development in Culturalist Theorizing." *European Journal of Social Theory* 5 (2). https://doi.org/10.1177/13684310222225432.

Renaud, Karen. 2016. "How Smaller Businesses Struggle with Security Advice." *Computer Fraud & Security* 2016 (8): 10–18.

Rossignoli, Francesca, Andrea Lionzo, Thomas Henschel, and Börje Boers. 2023. "Knowledge Sharing in Family SMEs: The Role of Communities of Practice." *Journal of Family Business Management*.

Rouse, Joseph. 2007. "Practice Theory." In *Philosophy of Anthropology and Sociology*, 639–81. Elsevier.

Ryan, Ita, Utz Roedig, and Klaas-Jan Stol. 2021. "Understanding Developer Security Archetypes." In *2021 IEEE/ACM 2nd International Workshop on Engineering and Cybersecurity of Critical Systems (EnCyCriS)*, 37–40. IEEE.

Safa, Nader Sohrabi, Rossouw Von Solms, and Lynn Futcher. 2016. "Human Aspects of Information Security in Organisations." *Computer Fraud & Security* 2016 (2): 15–18.

SAFECode. 2018. "SAFECode: Fundamental Practices for Secure Software Development 3rd Edition." 2018. https://safecode.org/wp-content/uploads/2018/03/SAFECode_Fundamental_Practices_for_Secure_Software_Development_March_2018.pdf.

Sarker, Iqbal H, Md Hasan Furhad, and Raza Nowrozy. 2021. "Ai-Driven Cybersecurity: An Overview, Security Intelligence Modeling and Research Directions." *SN Computer Science* 2 (3): 173.

Sasse, M Angela, and Ivan Flechais. 2005. "Usable Security: Why Do We Need It? How Do We Get It?" In *Security and Usability: Designing Secure Systems That People Can Use*, edited by Garfinkel Lorrie, Faith Cranor; Simson. O'Reilly Media.

Sawaya, Yukiko, Mahmood Sharif, Nicolas Christin, Ayumu Kubota, Akihiro Nakarai, and Akira Yamada. 2017. "Self-Confidence Trumps Knowledge: A Cross-Cultural Study of Security Behavior." In *Conference on Human Factors in Computing Systems - Proceedings*. Vol. 2017-May. https://doi.org/10.1145/3025453.3025926.

Schatzki, Theodore R. 1993. "Wittgenstein 4- Heidegger on the Stream of Life." *Inquiry (United Kingdom)* 36 (3). https://doi.org/10.1080/00201749308602324.

———. 1997. "Practices and Actions: A Wittgensteinian Critique of Bourdieu and Giddens." *Philosophy of the Social Sciences* 27 (3). https://doi.org/10.1177/004839319702700301.

Schatzki, Theodore R, and Theodore R Schatzki. 1996. *Social Practices: A Wittgensteinian Approach to Human Activity and the Social*. Cambridge University Press.

Schatzki, T.R. 2005. "Peripheral Vision: The Sites of Organizations." *Organization Studies* 26 (3).

Schlienger, Thomas, and Stephanie Teufel. 2002. "Information Security Culture." *IFIP ATC11 International Conference on Information Security*, no. April: 191–202.

Schneier, Bruce. 2000. *Secrets & Lies: Digital Security in a Networked World*. John Wiley & Sons, Inc. http://dl.acm.org/citation.cfm?id=517959.

———. 2006. *Beyond Fear: Thinking Sensibly about Security in an Uncertain World*. Springer Science & Business Media.

Schomberg, Rene Von. 2012. "Prospects for Technology Assessment in a Framework of Responsible Research and Innovation." *Technikfolgen Abschätzen Lehren. VS Verlag Für Sozialwissenschaften*, 39–61. https://doi.org/10.2139/ssrn.2439112.

———. 2019. "Why Responsible Innovation." *International Handbook on Responsible Innovation: A Global Resource*, 12–32.

Schreiber, Rita Sara, and Phyllis Noerager Stern. 2001. *Using Grounded Theory in Nursing*. Springer Publishing Company.

Seale, Clive. 2004. "Generating Grounded Theory." *Researching Society and Culture*, 240–47.

Semertzaki, Eva. 2011. *Special Libraries as Knowledge Management Centres. Special Libraries As Knowledge Management Centres*. https://doi.org/10.1533/9781780632667.

Shahin, Mojtaba, Muhammad Ali Babar, and Liming Zhu. 2017. "Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices." *IEEE Access* 5: 3909–43.

Sharma, Amit, and R. K. Bawa. 2020. "Identification and Integration of Security Activities for Secure Agile Development." *International Journal of Information Technology (Singapore)*. https://doi.org/10.1007/s41870-020-00446-4.

Shaw, Mary, and David Garlan. 1996. *Software Architecture*. Vol. 101. Prentice Hall Englewood Cliffs.

Shoib, Gamila, Joe Nandhakumar, and Matthew Jones. 2006. "Using Social Theory in Information Systems Research: A Reflexive Account." *Quality and Impact of Qualitative Research*, 129.

Shove, Elizabeth, Mika Pantzar, and Matt Watson. 2012. *The Dynamics of Social Practice: Everyday Life and How It Changes*. *The Dynamics of Social Practice: Everyday Life and How It Changes*. https://doi.org/10.4135/9781446250655.

———. 2014. "The Dynamics of Social Practice." In *The Dynamics of Social Practice: Everyday Life and How It Changes*. https://doi.org/10.4135/9781446250655.n1.

Sikolia, David, David Biros, Marlys Mason, and Mark Weiser. 2013. "Trustworthiness of Grounded Theory Methodology Research in Information Systems."

Simpson, Stacy. 2014. "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*, 1–32. Springer.

Sjoberg, Dag I K, Tore Dyba, and Magne Jorgensen. 2007. "The Future of Empirical Methods in Software Engineering Research." In *2007 Future of Software Engineering*, 358–78. IEEE Computer Society.

Smith, Elizabeth A. 2001. "The Role of Tacit and Explicit Knowledge in the Workplace." *Journal of Knowledge Management* 5 (4). https://doi.org/10.1108/13673270110411733.

Smith, S. W. 2003. "Humans in the Loop." *IEEE Security & Privacy*, 75–79. https://doi.org/10.1109/MSECP.2003.1203228.

Souppaya, Murugiah, Karen Scarfone, and Donna Dodson. 2021. "Secure Software Development Framework (SSDF) Version 1.1: Recommendations for Mitigating the Risk of Software Vulnerabilities." National Institute of Standards and Technology. https://csrc.nist.gov/projects/ssdf.

Stanton, Jeffrey M, Kathryn R Stam, Paul Mastrangelo, and Jeffrey Jolton. 2005. "Analysis of End User Security Behaviors." *Computers & Security* 24 (2): 124–33.

Statista. 2021a. "Number of Software Developers Worldwide in 2018 to 2024." Https://Www.Statista.Com/Statistics/627312/Worldwide-Developer-Population/. 2021.

———. 2021b. "Total Numbers of Programmers and Software Development Professionals in the United Kingdom (UK) from 2011 to 2020." Https://Www.Statista.Com/Statistics/318818/Numbers-of-Programmers-and-Software-Development-Professionals-in-the-Uk/. 2021.

Stedmon, Alex, Dale Richards, Lara Frumkin, and Peter Fussey. 2016. "Human Factors in Security: User-Centred and Socio-Technical Perspectives." *Security Journal* 29 (1): 1–4. https://doi.org/10.1057/sj.2015.40.

Stilgoe, Jack, Richard Owen, and Phil Macnaghten. 2013. "Developing a Framework for Responsible Innovation." *Research Policy* 42 (9). https://doi.org/10.1016/j.respol.2013.05.008.

Stol, Klaas-Jan, Paul Ralph, and Brian Fitzgerald. 2016. "Grounded Theory in Software Engineering Research: A Critical Review and Guidelines." In *Proceedings of the 38th International Conference on Software Engineering*, 120–31.

Storey, David J. 2016. *Understanding the Small Business Sector*. Routledge.

Strauss, Anselm, and Juliet Corbin. 1994. "Grounded Theory Methodology." *Handbook of Qualitative Research* 17: 273–85.

Strauss, Anselm, and Juliet M Corbin. 1997. *Grounded Theory in Practice*. Sage.

Tahaei, Mohammad, and Kami Vaniea. 2019. "A Survey on Developer-Centred Security." In *2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, 129–38. IEEE.

Tamburri, Damian A, Patricia Lago, and Hans van Vliet. 2013. "Organizational Social Structures for Software Engineering." *ACM Computing Surveys (CSUR)* 46 (1): 3.

Thomas, Tyler W., Madiha Tabassum, Bill Chu, and Heather Lipford. 2018. "Security during Application Development: An Application Security Expert Perspective." In *Conference on Human Factors in Computing Systems - Proceedings*. Vol. 2018-April. https://doi.org/10.1145/3173574.3173836.

Thornberg, Robert, and Kathy Charmaz. 2014. "Grounded Theory and Theoretical Coding." In *The SAGE Handbook of Qualitative Data Analysis*. https://doi.org/10.4135/9781446282243.n11.

Trist, Eric. 1981. "The Evolution of Socio-Technical Systems." *Conference on Organizational Design and Performance*.

Trist, Eric, and Fred Emery. 2005. *Socio-Technical Systems Theory*.

———. 2015. "Sociotechnical Systems Theory." In *Organizational Behavior 2: Essential Theories of Process and Structure*. https://doi.org/10.4324/9781315701967-40.

Troyer, Lisa. 2016. "Expanding Sociotechnical Systems Theory through the Trans-Disciplinary Lens of Complexity Theory." In *Transdisciplinary Perspectives on Complex Systems: New Findings and Approaches*. https://doi.org/10.1007/978-3-319-38756-7_7.

Turner, Bob, and Gregory Bateson. 1980. "Mind and Nature." *RAIN*, no. 36. https://doi.org/10.2307/3032188.

US Department of Labor. 2017. "Occupational Outlook Handbook, 2016-17 Edition, Software Developers." https://www.bls.gov/ooh/computer-and-information-technology/software-developers.htm.

Venson, Elaine, Reem Alfayez, Marília M F Gomes, Rejane M C Figueiredo, and Barry Boehm. 2019. "The Impact of Software Security Practices on Development Effort: An Initial Survey." In *2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 1–12. IEEE.

Vidyaraman, S, M Chandrasekaran, and S Upadhyaya. 2008. "Position: The User Is the Enemy." In *Proceedings of the 2007 Workshop on New Security Paradigms*, 75–80. ACM.

Vliet, Hans Van. 2008. *Software Engineering: Principles and Practice*. Vol. 13. John Wiley & Sons Hoboken, NJ.

Vliet, Hans Van, and J C Van Vliet. 2008. *Software Engineering: Principles and Practice*. Vol. 13. Citeseer.

Votipka, Daniel, Desiree Abrokwa, and Michelle L. Mazurek. 2020. "Building and Validating a Scale for Secure Software Development Self-Efficacy." In *Conference on Human Factors in Computing Systems - Proceedings*. https://doi.org/10.1145/3313831.3376754.

Votipka, Daniel, Kelsey R. Fulton, James Parker, Matthew Hou, Michelle L. Mazurek, and Michael Hicks. 2020. "Understanding Security Mistakes Developers Make: Qualitative Analysis from Build It, Break It, Fix It." In *Proceedings of the 29th USENIX Security Symposium*.

Wamala, F. 2013. "The ITU National CyberSecurity Strategy Guide." *International Telecommunication Union* 53 (9): 1689–99. https://doi.org/10.1017/CBO9781107415324.004.

Wasserman, Anthony I. 2010. "Software Engineering Issues for Mobile Application Development." In *Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research*, 397–400. ACM.

Weir, Charles, Ingolf Becker, and Lynne Blair. 2021. "A Passion for Security: Intervening to Help Software Developers." In *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, 21–30. IEEE.

Weir, Charles, Ingolf Becker, James Noble, Lynne Blair, Angela Sasse, and Awais Rashid. 2019. "Interventions for Software Security: Creating a Lightweight Program of Assurance Techniques for Developers." In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, 41–50. IEEE.

Weir, Charles, Ingolf Becker, James Noble, Lynne Blair, M Angela Sasse, and Awais Rashid. 2020. "Interventions for Long-term Software Security: Creating a Lightweight Program

of Assurance Techniques for Developers." *Software: Practice and Experience* 50 (3): 275–98.

Weir, Charles, Lynne Blair, Ingolf Becker, Angela Sasse, and James Noble. 2018. "Light-Touch Interventions to Improve Software Development Security." In *2018 IEEE Cybersecurity Development (SecDev)*, 85–93. IEEE.

Weir, Charles, Ben Hermann, and Sascha Fahl. 2020. "From Needs to Actions to Secure Apps? The Effect of Requirements and Developer Practices on App Security." In *Proceedings of the 29th USENIX Security Symposium*.

Weir, Charles, Awais Rashid, and James Noble. 2016. "How to Improve the Security Skills of Mobile App Developers? Comparing and Contrasting Expert Views." In *Twelfth Symposium on Usable Privacy and Security (SOUPS 2016)*.

———. 2017. "I'd Like to Have an Argument, Please: Using Dialectic for Effective App Security." In . https://doi.org/10.14722/eurousec.2017.23002.

———. 2020. "Challenging Software Developers: Dialectic as a Foundation for Security Assurance Techniques." *Journal of Cybersecurity* 6 (1): tyaa007.

Wen, Shao-Fang. 2018. "An Empirical Study on Security Knowledge Sharing and Learning in Open Source Software Communities." *Computers* 7 (4): 49.

Wenger, Etienne. 1999. *Communities of Practice: Learning, Meaning, and Identity*. Cambridge university press.

———. 2011. "Communities of Practice: A Brief Introduction."

Whitehead, Jim. 2007. "Collaboration in Software Engineering: A Roadmap." In *Future of Software Engineering (FOSE'07)*, 214–25. IEEE.

Wijayarathna, Chamila, and Nalin Asanka Gamagedara Arachchilage. 2019. "Why Johnny Can't Develop a Secure Application? A Usability Analysis of Java Secure Socket Extension API." *Computers & Security* 80: 54–73.

Witschey, Jim, Shundan Xiao, and Emerson Murphy-Hill. 2014. "Technical and Personal Factors Influencing Developers' Adoption of Security Tools." In *Proceedings of the*

*ACM Conference on Computer and Communications Security*. Vol. 2014-November. https://doi.org/10.1145/2663887.2663898.

Witschey, Jim, Olga Zielinska, Allaire Welk, Emerson Murphy-Hill, Chris Mayhorn, and Thomas Zimmermann. 2015. "Quantifying Developers' Adoption of Security Tools." In *2015 10th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, ESEC/FSE 2015 - Proceedings*. https://doi.org/10.1145/2786805.2786816.

Wolfswinkel, Joost F., Elfi Furtmueller, and Celeste P.M. Wilderom. 2013. "Using Grounded Theory as a Method for Rigorously Reviewing Literature." *European Journal of Information Systems*. https://doi.org/10.1057/ejis.2011.51.

Woodhouse, E J. 2005. "(Re)Constructing Technological Society by Taking Social Construction Even More Seriously." *Social Epistemology* 19 (2–3): 199–223. https://doi.org/10.1080/02691720500145365.

Wurster, Glenn, and Paul C van Oorschot. 2009. "The Developer Is the Enemy." In *Proceedings of the 2008 Workshop on New Security Paradigms*, 89–97. ACM.

Wysocki, Robert K. 2010. *Effective Software Project Management*. Wiley+ ORM.

Xiao, Shundan, Jim Witschey, and Emerson Murphy-Hill. 2014. "Social Influences on Secure Development Tool Adoption: Why Security Tools Spread." In *Proceedings of the ACM Conference on Computer Supported Cooperative Work, CSCW*. https://doi.org/10.1145/2531602.2531722.

Xie, Jing, Bill Chu, Heather Richter Lipford, and John T. Melton. 2011. "ASIDE: IDE Support for Web Application Security." In *ACM International Conference Proceeding Series*. https://doi.org/10.1145/2076732.2076770.

Xie, Jing, Heather Richter Lipford, and Bill Chu. 2011. "Why Do Programmers Make Security Errors?" In *Proceedings - 2011 IEEE Symposium on Visual Languages and Human Centric Computing, VL/HCC 2011*. https://doi.org/10.1109/VLHCC.2011.6070393.

———. 2012. “Evaluating Interactive Support for Secure Programming.” In *Conference on Human Factors in Computing Systems - Proceedings*. https://doi.org/10.1145/2207676.2208665.

Yilmaz, Levent. 2007. “Modelling Software Processes as Human-Centered Adaptive Work Systems.” *Software Process Improvement*, 148–59.

Zarour, Mohammad, Mamdouh Alenezi, and Khalid Alsarayrah. 2020. “Software Security Specifications and Design: How Software Engineers and Practitioners Are Mixing Things Up.” In *ACM International Conference Proceeding Series*. https://doi.org/10.1145/3383219.3383284.

Zelkowitz, Marvin V, Raymond T Yeh, Richard G Hamlet, John D Gannon, and Victor R Basili. 1984. “Software Engineering Practices in the US and Japan.” *Computer* 17 (6): 57–66.

# Appendix A Contact Letter

Dear [Participant's Title and Name],

Further to our discussion at the Oxford [BCS Networking Event] on [Date], I am writing to schedule an interview with you for my research. OR

I would like to invite you to participate in an academic research.

[Just to remind you], I am a doctoral student studying at the University of Oxford and we briefly discussed my research project at the meeting. My research aims at investigating and understanding key factors that influence the cyber security practices and decisions of application software developers within small and medium-sized organisations. The research will involve studying factors that influence the cyber security activities and practices of these developers in SMEs because it has become a crucial step in understanding issues, identifying opportunities for improvement and supporting these businesses and individuals.

I have attached a **Participant Information Sheet and Consent form** to this e-mail. Please read these over and feel free to contact me by email or telephone (details below) if you have any further questions. We can find a suitable date, time and venue for the interview given your schedule.

Participation is totally voluntary and if this looks like something that you would be interested in participating in, all you need to do is sign the consent form and forward that to me in an e-mail or I can collect it on our meeting day. We do have the options of doing this interview face-to-face or via Skype.

I look forward to your response.

Yours sincerely

**Paula Fiddi (MSc. BTech.)**

# Appendix B Consent Form

DEPARTMENT OF COMPUTER SCIENCE
WOLFSON BUILDING, PARKS ROAD
OX1 3QD, OXFORD


Principal Investigator: Prof Marina Jirotka
Contact details: marina.jirotka@cs.ox.ac.uk
Primary researcher: Paula Fiddi, DPhil Candidate
Oxford telephone number: +44 (0) 7828012421
Oxford email: paula.fiddi@cs.ox.ac.uk

**PARTICIPANT CONSENT FORM**

**CUREC Approval Reference: R54135/RE001**

**Investigating the Dynamics of Key Socio-technical Factors in Cyber Security Practices of Software Developers in SMEs**

**Purpose of Study:** The purpose of this study is to investigate and understand key factors that influence cyber security practices and decisions of application software developers within small and medium-sized organisations. This consent form is to conduct a brief interview session to discuss questions related to security practices of developers during the application software development process.

| | | *Please initial each box* |
|---|---|---|
| 1 | I confirm that I have read and understand the information sheet for the above study. I have had the opportunity to consider the information, ask questions and have had these answered satisfactorily. | ☐ |
| 2 | I understand that my participation is voluntary and that I am free to withdraw at any time, without giving any reason, and without any adverse consequences or academic penalty. | ☐ |
| 3 | I understand that research data collected during the study may be looked at by designated individuals from the University of Oxford where it is relevant to my taking part in this study. I give permission for these individuals to access my data. | ☐ |
| 4 | I understand that this project has been reviewed by, and received ethics clearance through, the University of Oxford Central University Research Ethics Committee. | ☐ |
| 5 | I understand who will have access to personal data provided, how the data will be stored and what will happen to the data at the end of the project. | ☐ |
| 6 | I understand how this research will be written up and published. | ☐ |
| 7 | I understand how to raise a concern or make a complaint. | ☐ |
| 8 | I consent to being audio recorded. | ☐ |

| | | |
|---|---|---|
| 9 | I understand how audio recordings will be used in research outputs. | ☐ |
| 10 | I give permission to be quoted directly in the research publication. | ☐ |
| 11 | I agree to take part in the study. | ☐ |
| **Optional:** | I agree for research data collected in this study to be given to researchers, including those working outside of the EU, to be used in other research studies. I understand that any data that leave the research group will be fully anonymised so that I cannot be identified. | ☐ |
| **Optional:** | I agree for my personal data to be kept in a secure database for the purpose of contacting me about future studies. | ☐ |

| Name of Participant | Date | Signature |
|---|---|---|

| Name of person taking consent | Date | Signature |
|---|---|---|

# Appendix C Interview Guide and Worksheet

## C.1 Interview Guide

**Introductory Questions**

- How did you get into software or app development? What type of developer would you call yourself (native, cross platform, API, web app etc.)? What platforms do you work on majorly (iOS, Android, Windows, and Web etc.)?
- How long have you been a developer?
- How many apps/software do you have published? Are you currently working on any now?
- Do you work for a company (company size) or independently/ own a business or start-up?
- What types of apps/ software products do you develop?

**Developers' activities & tools used during application software development process.**

- What does your typical day at work look like? Or what does a typical day of a software developer look like for you?
- What are your go-to development tools for app/software development? (IDEs, SDKs, APIs or Open-source tools?
- Do you follow any methodologies (agile, scrum, RAD etc.) or a more customised approach?
- Where do you carry out the most work (in a team or individually) (personal systems (locally), public collaborative spaces like the cloud, GitHub etc.)? Do you develop or deploy your apps in the cloud?

**Cyber security knowledge, awareness & concerns**

- What does cyber security mean to you? Are there any standard cyber security frameworks or guidelines you use (such as the OWASP, BSIMMs, SANS etc.)? If no, why?
- Do you rely on in-built platform security for your apps or use any available security tool or service? What tools or resources do you used for guidance during the development process (such as web resources – stack overflow)?
- At what stage in the development process do you consider cyber security?
- How do you handle codes from 3rd parties, APIs, Library requirements or permissions etc.?
- Have you received any training on implementing cyber security practices for secure app development?
- What is the cost (time, resources) of implementing security in your apps (using security or encryption services for data)? Any other challenges faced?
- New General Data Protection Regulation (GDPR) laws and how do they affect you?

**Other security considerations during Application Software Development Process**

- What do you consider as good or bad security practices, particularly when using web-based technology for app/software development?
- Who do you think is responsible for implementing these practices (the end-user, developers, platform providers etc.)?
- (For group developments) Who makes the decisions about the standards security procedures, frameworks, or guidelines to follow and tools to use during the development of a software product?
- What do you think would provide the best support for developers to improve their security practices online (a tool that has all cyber security standards and frameworks in one place, educational training/awareness materials, more frameworks, and guidelines)?
- Do you have any questions or feedback on the interview?

# C.2 Follow-Up Interview Worksheet

DEPARTMENT OF COMPUTER SCIENCE
WOLFSON BUILDING, PARKS ROAD
OX1 3QD, OXFORD

Principal Investigator: Prof Marina Jirotka
Contact details: marina.jirotka@cs.ox.ac.uk
Primary researcher: Paula Fiddi, DPhil Candidate paula.fiddi@cs.ox.ac.uk
Oxford telephone number: +44 (0) 7828012421
Oxford email: ethics@socsci.ox.ac.uk

**Cyber Security as a Social Practice: investigating the dynamics of socio-technical factors in and security practices of developers in SMEs**

**PARTICIPANT THEORY EVALUATION WORKSHEET**

**Ethics Approval Reference: CS_C1A_20_009**

## Background

While there is research evidence that highlights existing cyber security practices designed to cater to app developers within large and structured environments, little attention has been given to practices adopted by developers in smaller organisations. While both large and small organisations might delegate security responsibility to service providers, the very heterogeneous nature of SMEs increases the potential for incompatibilities between security practices and responsibilities.

To address these concerns, we performed a study to understand the experiences and challenges of app developers in SMEs, their perspectives of existing security practices and activities, and factors that influence these practices during the development process.

From this study we constructed a theory represent diagrammatically in **Figure 1** below. Our theory describes key socio-technical factors that we identified are relevant to the current security practices of developers in SMEs. These follow-up questions and tasks are to help us evaluate our theory.

**The organisational ecosystem** represents the environment in which all these factors interact.
**Usable Security Technology**: access to available security technologies such as
**Context**: the context in which the software product will be used
**Security Activities**: Security education and awareness, attack and resource identification, threat modelling, security requirement analysis, building security teams and assigning roles, security designs review, static code analysis, penetration testing, and incident response planning.
**Knowledge Base**: containing information about security experiences, and approach to security incidents from individual and community expertise.
**People**: actors involved in the development process and those affected by its outcome

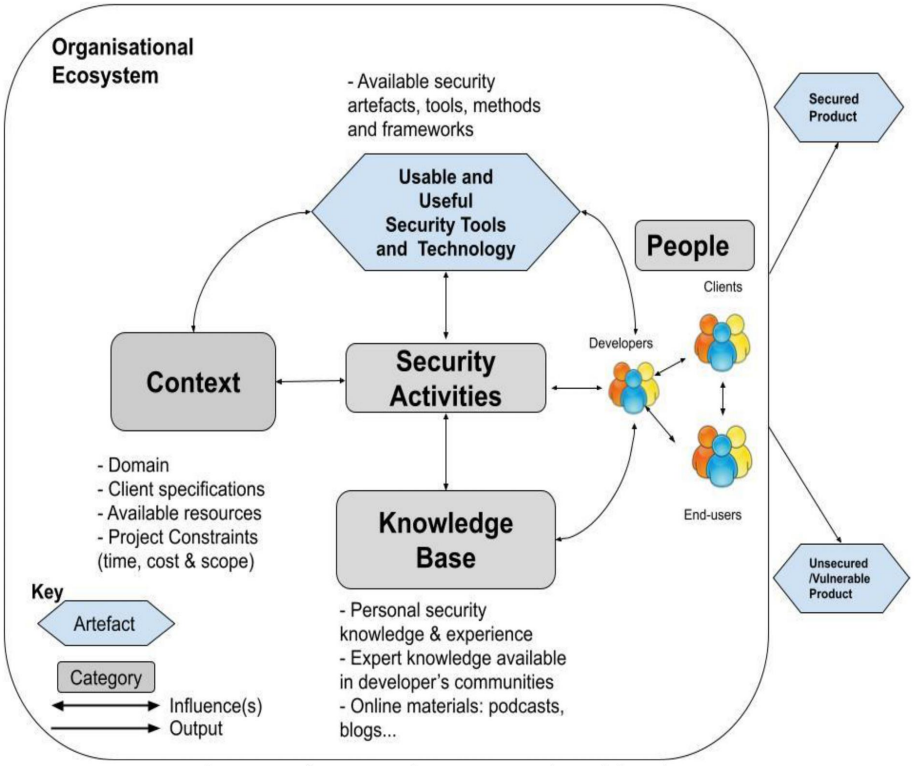



**Figure 1**: Diagrammatic Representation of the Theory

## Task 1

Your task is to think about how to adopt and incorporate secure activities and technologies in your app development process using the scenario and activities described below.

**Personas:**
John is a 29-year-old employee at a digital health care start-up company and a freelance developer. He is a self-taught developer with no formal education in software engineering or cyber security practices. He is an iOS mobile app developer with eight years of experience.

**Scenario:**
John recently joined the digital health care company less than three weeks ago. The company wants to develop an app to track patient's heart rates and other vital signs, alert medical personnel of possible high-risk activities of patients and report patient's behaviours. John recently got employed in this company to be a leader of a team of three developers for the project. The CEO hopes that the app can be launched and released to users (private consultants and patients) in three months. Resources are limited as the company is in its first round of funding from investors.

**Some guiding questions**:

1. What stands out to you in the scenario described above?
2. How much does it resonate with you? Can you describe your experience with a similar or different example?
3. How should John introduce security activities in his app project?
4. Reflect on any other possible scenarios during the development process of an app project in your experience that may have influence security decisions.

5. What are some strategies for incorporating security practices in the development process?

## Task 2

**Implementing 4-Step Iterative Process**

We propose an iterative four-step plan that can be incorporated at different stages of an agile development process:

> Step 1: **Assess,** identify and engage the people involved in and affected by the project - your team, stakeholders and end-users. Identify cost, available resources, context in which your app will be used and security requirements. Access knowledge base – security or developers' communities – for current security information that best fits the project.
>
> Step 2: **Select** the useful and usable security tools, frameworks, and technologies required.
>
> Step 3: **Reflect** on possible consequences of decisions made in Step 2 such as checking that the security tool or technology is fit for purpose. Identify possible alternatives.
>
> Step 4: **Act** on implementing the best security tools, frameworks and technologies according to security information gathered from community expertise, responses from stakeholder engagement and personal experience. Repeat from Step 1 for each sprint.

**Some guiding questions**:

1. How realistic is it to introduce these processes during development lifecycle?
2. How would you incorporate these processes in your development process?
3. In what format (workshops, group discussions, stand-up meetings)?
4. Do you find these processes useful, and do they support your considerations for software security?
5. What else would you consider useful?

# Appendix D Codes, Categories and Memos

| Sub-categories | Categories | Memos |
|---|---|---|
| -Hybrid apps<br>-Native apps<br>-Web apps<br>-Progressive web apps | Application software type | Participants developed either mobile or web or progressive web apps. |
| -Web app development tools.<br>-Mobile app tools.<br>-Hybrid app tools<br>-Cross platforms | Tools (Platforms, SDKs, Frameworks and Programming Languages) | Developers are experts in using one or more tools. Examples of platforms include iOS, Android, Ionic framework, Database technologies (MongoDB, Firebase, MySQL). Cross-platforms technologies such as Reactive Native etc. Different platforms, frameworks and languages have different security needs. |
| -Sole proprietorship businesses (one-man businesses, Independent)<br>-Micro businesses (2-5)<br>-Small businesses<br>-Medium sized companies | Type (or size) of organisation | Represents the staff size of the organisation developers belong to. |
| -Business owner<br>-Project manager<br>-Support developer | Roles within organization/Level of responsibility in decision-making process | Developers play different roles within their organization. |

| | | |
|---|---|---|
| (external) | | |
| -Web app, Mobile app or Desktop developer<br>-API | Type of developer (can be as a freelance/independent developer or a company staff) | The types of apps developers build represent the titles they give themselves. |
| -Size<br>-Cost<br>-Project Uniqueness<br>-Specific security needs<br>-Methodology<br>-Complexity<br>-Timeline | App project characteristics | - Project sizes differ between clients and determine the type of tools or technologies used.<br>-The type of app project affects the approach to security.<br>- Developers relate the cost of an app project to its flexibility.<br>-Size of the project determines the team size. |
| -Agile scrum<br>-Waterfall<br>-Custom (combination of other methods) | Development methodologies | Application of different types of software development methodologies. |
| -Large customer (user) base<br>-Small customer (user) base<br>-International exposure to larger customer base. | User base | Represents the users currently using or predicted to use apps developed by developers. |
| -Application software development training<br>-Security training | Training | Formally trained or self-taught |
| -Remote app development<br>-Physical<br>-Cloud development | Types of app development | Apps are developed with the development team in the same space or remotely.<br>Cloud-based development can be shared and tracked globally. |
| -Group chats (on WhatsApp)<br>-Skype or Google hangout | Communication tools | Common tools for communicating with team members. |
| -Team work distribution<br>-Remote collaborations<br>- Across-desk/ shared | Collaboration and coordination | Collaboration and coordination approach depend on the type and size of the project. |

| | | |
|---|---|---|
| space | | |
| -Project management tools (E.g., Slack, Asana)<br>-Code reviews (Git Bitbucket) | Collaboration tools | Tools for planning & managing app development projects; for reviewing and sharing codes. Most participants worked independently alongside their day jobs with a small or medium company<br>Tools are project dependent. |
| Experienced app developer (4 years and above) | Level of experience in application software development | Developers who have being actively developing for four years or more consider themselves as experienced developers. Less than 3 years consider themselves as newbies. |
| -Cyber security knowledge and awareness<br>-Previous unethical hacker | Developer's personal knowledge of information & cyber security practices | Indicates cyber security topics developers are aware of or activities they've done. |
| Previous security-related training or activities. | Developer's security experience | Developers have received formal training on security related topics in software development or are self-taught. |
| Client's requirement specification | Requirement elicitation. | Client's specification for apps. |
| Outsourcing various parts of the app project including security checks. | Outsourcing | App projects that are not internally developed or have support developers in collaborations. |
| -Non-disclosure agreements with external developers | Organisation structure and/or policies | Policies put in place by the company to restrict information sharing between developers. |
| -Secure/encrypt sensitive data<br>-Sanitise app information<br>-Encrypt passwords<br>-Restrict (remote) access to servers.<br>-Database security constraints enforced | Standard security practices/perception of general knowledge of security practices. | Practices that a participant expect to be obvious to any other developer. |
| -Unwritten security rules<br>-Use of code extracts from | Personal security practice | Practices that participants report are not standard but they do personally (either due to personal experience or knowledge of advanced security techniques). |

| | | |
|---|---|---|
| web resources.<br>-Security as an after-thought.<br>-Pays no attention to security guidelines mentioned in podcasts or blogs. | | |
| -OWASP<br>-Stack overflow<br>-Podcasts<br>-Blogs | Knowledge of security guidance tools, guidelines, frameworks and other advice resources. | Developers refer to these guidance avenues and tools |
| Applies security guidelines | Application of security guidelines, tools or advice | Reports application of guidelines/ instructions suggested in security guidance materials found. |
| Depends on built-in security | Confidence in brand security measures | |
| Custodian of information | Custodian of information | A developer in charge of information flow and distribution (restricting and granting access to information to other developers) during development process. |
| Features and functionalities of apps | Focused mainly on features and functionalities of app | Developers talk about focusing majorly on the features of the apps created and not security issues. |
| -Trust in app developer's communities<br>-Trust in certain brands of tools | Trust | The developer depends on the approval/recommendations of other developers within the app developer's communities. |
| User's privacy | Privacy | Developers talk about the importance of user's privacy while using their product. |

| | | |
|---|---|---|
| -Developer's responsibilities<br>-Third-party libraries<br>-API developers | Responsibility | Developer's talk about who is responsible for ensuring app security. |
| -Security-led briefs for each project<br>-Policies per app project<br>-Third-party security libraries<br>-Recommendations from other developers | Support for developers | Developers suggest ways they think they can be supported. |

# Appendix E The First Emergent Theory

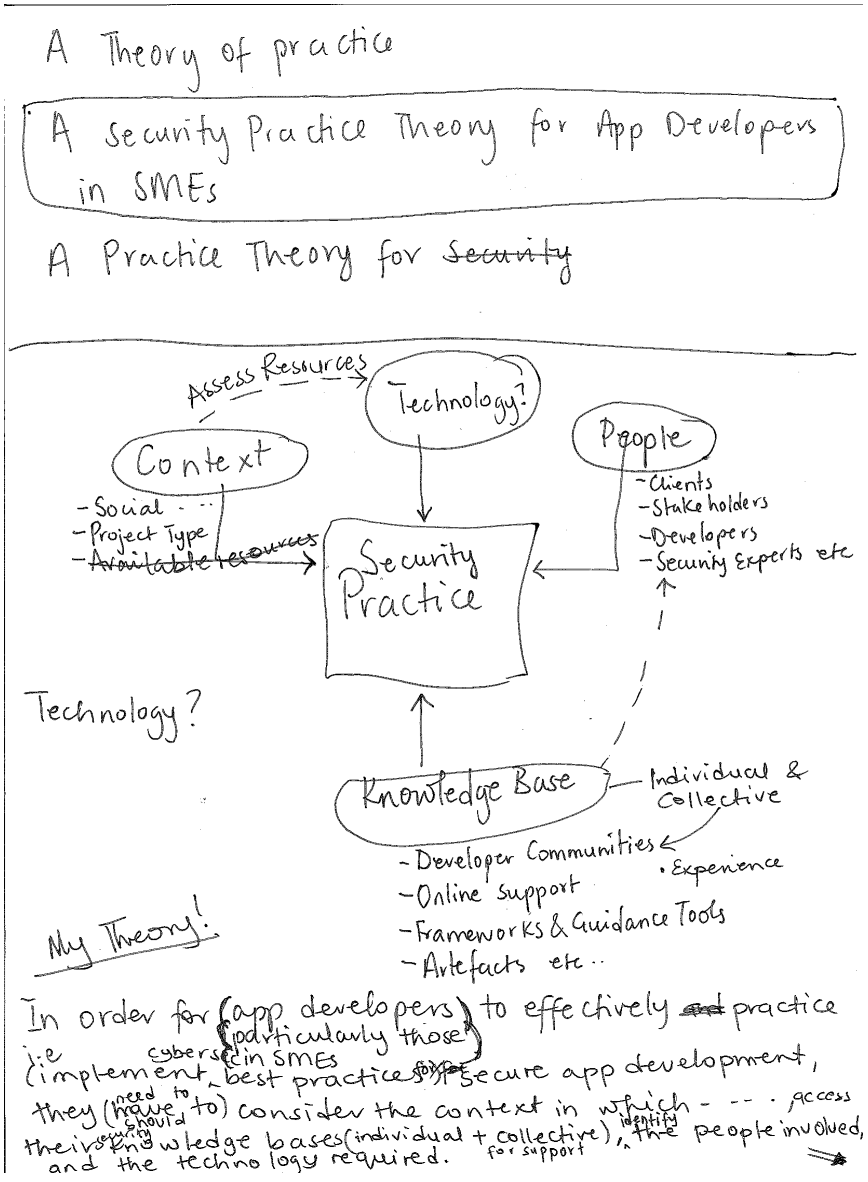



Fig. E.1 The Theory Emerging for the First Time