Elina Räisänen & Aino-Maria Väyrynen

SECURE SOFTWARE DESIGN AND DEVELOPMENT –

TOWARDS PRACTICAL MODELS FOR IMPLEMENTING INFORMATION SECURITY INTO THE REQUIREMENTS ENGINEERING PROCESS

UNIVERSITY OF JYVÄSKYLÄ
FACULTY OF INFORMATION TECHNOLOGY
2020

ABSTRACT

Räisänen, Elina & Väyrynen, Aino-Maria
Secure software design and development – towards practical models for implementing information security into the requirements engineering process
Jyväskylä: University of Jyväskylä, 2020, 125 p.
Cyber Security, Master's Thesis
Supervisor: Siponen, Mikko

The aim of the Requirements Engineering (RE) process is to elicit and refine into a solution the ideas and needs from identified stakeholders of a product or a service. These solve problems in customer business while bringing added value. Software development's central theme is software's security. It has been studied abundantly but its usage and implementation are often problematic and deficient. Software threats and risks evolve continuously, and vulnerabilities from software's development are discovered and exploited in new ways. Software development should invest into information security as a part of requirements engineering processes' continuous development. This process should encompass the entire product lifecycle and consider post-launch phases where the on-market product is further developed. Requirements should be reviewed iteratively to keep current and adapt to the changing threats and risks in the software. The research objective was to create a suitable model for the commissioner (a large manufacturer of physical security products in Finland) which would adapt information security as an integral part of the software development and thus produce more secure software. Two stages of action research were applied to problem solving. The first step was to create the theoretical background for requirements engineering and information security. After that, the current situation analysis was initiated, and document analysis was used to map out the organizational operating environment with a focus on the requirements engineering process model and the stakeholders utilizing it. These results formed the foundation for the interviews, where the problems of the requirements engineering process were surveyed. Results were analyzed with coding and categorizing. A second part of the diagnosis was a comparative study, which was utilized to discover suitable practices to form the needed elements for the model. The resulting change recommendations from the interviews were combined with suitable practices from the field. This combination formed a model for information security in RE process and it will be later implemented by the commissioner. The model has a novelty value because it merges agile development practices with the idea of threat and risk modelling, which is still an understudied subject. Additionally, both components work as a part of a linear RE process.

Keywords: information security, software development, requirements engineering

TIIVISTELMÄ

Räisänen, Elina & Väyrynen, Aino-Maria
Turvallinen ohjelmistosuunnittelu ja -kehitys – kohti konkreettisia metodeja tietoturvallisuuden implementoimiseksi osaksi vaatimusmäärittelyprosessia
Jyväskylä: Jyväskylän yliopisto, 2020, 125 s.
Kyberturvallisuus, pro gradu -tutkielma
Ohjaaja: Siponen, Mikko

Vaatimusmäärittelyprosessin tavoitteena on kerätä ja jalostaa ratkaisuiksi tuotteen tai palvelun sidosryhmiksi tunnistettujen osapuolten ajatuksia ja tarpeita. Näiden ratkaisujen avulla poistetaan asiakkaan liiketoiminnassa olevia ongelmia ja tuotetaan lisäarvoa. Ohjelmistokehityksessä on tällä hetkellä keskeistä erityisesti ohjelmistojen turvallisuus. Sitä on tutkittu paljon, mutta sen käytäntöön vieminen on usein ongelmallista ja puutteellista. Ohjelmistojen tietoturvallisuusuhkat ja -riskit lisääntyvät jatkuvasti ja ohjelmistojen kehityksessä muodostuneita haavoittuvuuksia paikallistetaan sekä hyväksi käytetään uusin tavoin. Ohjelmistokehityksen tulisi panostaa tietoturvallisuuden osalta vaatimusmäärittelyprosessin jatkuvaan kehittämiseen. Prosessin tulee kattaa koko tuotteen elinkaari, huomioiden myös lanseerauksen jälkeiset vaiheet, joissa markkinoilla olevaa tuotetta kehitetään. Vaatimuksia on kyettävä tarkentamaan iteratiivisesti, jolloin ne pysyvät ajantasaisina ja huomioivat muutokset ohjelmiston uhkissa ja riskeissä. Tutkimustehtävänä oli luoda toimeksiantajan (iso Suomalainen fyysisten turvallisuustuotteiden valmistaja) tarpeisiin sopiva malli, jonka avulla on mahdollista implementoida tietoturvallisuus kiinteäksi osaksi ohjelmistokehitystä ja turvallisempaa ohjelmiston tuottamista. Tutkimusongelman ratkaisussa hyödynnettiin käytännönläheisen toimintatutkimusmallin kahta ensimmäistä vaihetta. Tutkimuksen aluksi luotiin työn teoreettinen perusta vaatimusmäärittelystä ja tietoturvallisuudesta, sitten aloitettiin nykytilaanalyysi. Siinä selvitettiin dokumentti analyysillä toimeksiantajan organisatorista toimintaympäristöä: keskittymällä vaatimusmäärittelyn prosessimalliin ja sitä hyödyntäviin sidosryhmiin. Saatujen tietojen pohjalta laadittiin suunnitelma haastatteluun, jonka avulla kartoitettiin vaatimusmäärittelyprosessin ongelmakohtia. Saadut tulokset analysoitiin codingilla ja teemoittelemalla. Toinen osa diagnoosia oli vertailututkimus, jota hyödynnettiin parhaiden käytänteiden selvittämiseen ja oikeiden elementtien muodostamiseen. Saadut muutosideat yhdistettiin kirjallisuuskatsauksesta nousseisiin, kohdeyrityksen liiketoimintaan sopiviin käytänteisiin. Tämä kombinaatio muodosti mallin tietoturvallisempaan vaatimusmäärittelyprosessiin, joka jalkautetaan kohdeorganisaatioon. Työn uutuusarvo on se, että malli yhdistää ketterää ohjelmistokehitystä riski- ja uhkamallinnus pohjaiseen ajatteluun, jota on tutkittu vielä vähän. Lisäksi molemmat komponentit toimivat lineaarisessa vaatimusmäärittelyprosessissa.

Avainsanat: tietoturvallisuus, ohjelmistokehitys, vaatimusmäärittely

FIGURES

TABLES

TABLE OF CONTENTS

1 INTRODUCTION

Alexander and Beus-Dukic (2009, p. 217) represent Howard Chieves's statement in their book Discovering requirements, which describes the special characteristics of secure software development: "You can't calculate the probability that a system is secure based on the risks it handles, if it's certain that insecure humans will form a part of it."

Nowadays, software products perform everyday tasks and ensure that the most critical applications operate uninterrupted. These systems include critical infrastructure, banking, transportation, and many others. This means that security has become one of the most critical aspects of reliable software product development (Barabas et al., 2019, p. 1).

Software development means problem solving, customer's problem is identified and possibly solved with a suitable software (Aitken & Ilango, 2013, p. 4752). Software is executed based on the stakeholder minimum requirements where the software security and information security requirements are emphasized. These requirements compiled into the requirements engineering process. It aims to refine the process inputs - ideas and thoughts of the product and service's recognized stakeholder needs – into solutions. The process emphasizes documentation based on it the process can be well planned and managed, the change and risk management in addition to product acceptance is possible.

This thesis was commissioned by a large manufacturer of physical security products in Finland. The aim was to produce a model for the commissioner to implement the information security as an integral part to the company's software development and its practices. The commissioner wants to examine and further develop the software development's current situation so that the company could produce high security software in even quality; to better respond to inner and outer stakeholder needs and expectations.

Security is a vital part of the commissioner's brand and the company is known for its secure products. The shift of the market to a more digital environment requires that the company's core values are transferred to modern products. It is vital that information security requirements are identified and addressed as early as possible in the development process. It was agreed in co-

operation with the commissioner that the most effective means to respond to this need is to implement information security into the software development's requirements engineering process.

Based on the commission the aim of this research is to answer the following question: "What is the best model for secure software development for the commissioner, to implement information security requirements into the requirements engineering process, in order to produce more secure software?". This question is the main research question. Taken apart the question includes three subtopics: A Secure Software Development (SSD) method, Requirements Engineering (RE) and Information Security (IS).

This thesis was concluded as a pair project. The theoretical section was divided into two parts, other parts were done in cohesion. During the writing, progress was constantly monitored, evaluated and peer reviewed.

This research utilizes the action research methodology. Baskerville has authored many research papers about action research and its usage in information system context. In one of these papers Baskerville and Woodharper (1998, pp. 96–97) have concluded that the aim is to increase research comprehension while solving a real-world problem. Davison, Kock and Martinsons (2004, p. 73) specify that an action research involves two parties: the commissioner and the researcher. The commissioner receives aid in problem-solving and the researcher discovers a practical problem that can further develop an existing theory. An action research is focused on a specific need or a problem and it is very pragmatic.

This is an empirical and qualitative study that intends to construct an understanding about the phenomenon of RE in the context of software development. The semi-structured interview presents the empirical part, where the problems of the requirement engineering process of the commissioner where investigated. Thus, the comparative study, in turn, examines the best models and practices used in the field of SSD. These two parts were used to diagnose the current situation and based on them to create a combination model for implementing information security into requirements engineering process of the commissioner in action planning stage.

Software development changes to more agile practices and thus the emerging research focuses on added information production to further develop agile practices. Butler and Vijayasarathy (2016, p. 90) have researched different software development approaches and methodologies. They compared their usage during software development projects. The most used approach was a hybrid 45,3 % and the second was an agile 33,1 %. However, most frequently companies used Waterfall methodology (32 %). This thesis aims to produce a combination of these models by bringing agile practices to linear software development model.

Security is an even more integral part of software development and thus, security requirements engineering's role is highlighted. Software products, critical especially, require that the security requirements mitigate the identified threats and risks. Therefore, requirements engineering should be based on

threat and risk modelling. Bernsmed et al. (2019, p. 2) have concluded that there is relatively little research on implementing threat and risk- based requirement engineering into agile development. This thesis brings a threat and risk- based, iterative requirements engineering process to linear framework where the actual work is done with agile practices. This means that the resulting Threat and Risk Driven Software Gateway (TRD-SGW) -model is a hybrid which focuses on information security perspective.

There are copious amounts of research related to the field, from widely different perspectives these focus areas can roughly be divided into thirds. First third focuses on generic models of software development process, and one third surveys the widely used practices, compares, and combines them. And the remaining third concentrates on specific aspects, features or process sections and defines them in minor detail. Software develops into agile direction which means that the research concentrates on new knowledge producing. However, the Waterfall software development model is still the most extensively used.

This thesis contains six chapters from which the first is the introduction. In the introduction the background, purpose, progression, and the preliminary results are presented. After which the chapter two encompasses the theoretical background for this work based on literature review. This chapter delivers a comprehensive understanding of the research subject and introduces the background for development ideas which are reflected in the analysis sections for requirements engineering process and information security requirements. In chapter three the research methodology of this study is presented. In chapter four the results of both current situation diagnosis and action planning stage are presented. The chapter five, in turn, includes discussions and chapter six conclusions.

2 THEORETICAL BACKROUND

The theoretical background and starting point for the research was limited to the secure software development and requirements engineering process in the part of information security. The most meaningful theoretical subjects were requirements engineering, information security and secure software development. These theories were used as a foundation for the analysis and interpretation of the data that was gathered from semi-structured interviews and a comparative study. The formulation of the combination model will also lean on this theory and enable an interesting discussion about the different models used in secure software development.

2.1 Requirements engineering

Zave (1997, p. 315) has given a widely respected definition about the requirements engineering. She claimed that requirements engineering is a branch of software engineering concerned with the real-world goals of software system functions and constraints. Requirements engineering is also focused on to the relationship of these factors, their evolution over time and across software families. It examines how the precise specifications of software behavior compare to real-world goals.

This definition is a foundation to many other writers and influencers of this topic like for Laplante (2017, p. 3). He modified this universal definition and included the complexity of modern technology into it, be it hardware, software, a combination of these or something even more complex. He observed that software should be used instead of “software engineering”, he altered all “software systems” terms into “systems” and added “of related systems” after the “software families”. He continued to investigate the term in its new definition and all the related activities involved with the subject in detail throughout the book.

According to Abran, Kotonya, Moore and Sawyer (2001, p. 9) the main reason for the emergence of the term requirements engineering has been a need to express systematic handling of requirements. It is a widely spread belief in

software industry field that the software projects perform poorly when the requirements process activities such as acquisition, analysis, specification, validation, and management are done insufficiently. These activities are widely approved as the most essential steps in successful requirements engineering.

Eberlein et al. (2003, p. 1) have a more concise definition for requirements engineering objective believing it to be a conventional software engineering process. Which objective is to detect, assess, document then confirm requirements for the system that is being developed. They also state in their research that requirements engineering must be done before the actual system development begins to prevent mistakes and aid in requirements discovery.

Easterbrook and Nuseibeh (2000, p. 37) survey requirements engineering from the stakeholder perspective. They defined the requirements engineering process's aim as a process which establishes the purpose for the software or a product by discovering the correct stakeholders and their needs. Those needs are then documented into a form that can be easily analyzed, communicated, and eventually implemented to use. These activities produce the requirements to which the software development activities are then founded on. Requirements establish the foundation for project planning, risk management, change control, acceptance testing and trade-offs (Dick, Hull & Jackson, 2005, p. 2).

2.1.1 Requirement types

A requirement is a feature which must be displayed with the intention of resolving a conundrum of the real world (Abran et al., 2001, p. 4). Requirements can be anything from a desire expressed in a natural language, a sketch on a sticky note or a formal mathematical statement (Laplante, 2017, p. 3). There are various classes and categories for different requirements types and Laplante (2017) concisely dictates that the types can be explained by the different stakeholders that give and read the requirements. Stakeholders view the software, or a product from their own perspective and reflect their individual desires on to the design.

According to Laplante (2017) requirement types can be subdivided into: domain, non-functional and functional requirements and on the requirements level Laplante (2017) divides requirements into design, system and user levels. Beatty and Wiegers (2013, p. 10) disagree and divide software product requirements only into functional or non-functional requirements.

Functional requirements describe "what the software intends to do" and Non-Functional Requirements (NFRs) determine "how to accomplish that". Functional requirements describe circumstances that generate certain behavior from a product. NFRs are added features on the requirements document for instance security, quality and resilience. (Beatty & Wiegers, 2013, p. 7; Merkow & Raghavan, 2010, p. 14). Various researchers have observed that non-functional requirements such as safety, security and reliability are often disregarded during the software development. The process naturally focuses on functional requirements rather than non-functional. This leads to the situation

where these non-functional requirements are easily overlooked or forgotten. To maintain security's high level the security related issues require a high priority and security requirement elicitation must be done comprehensively (Beg, Khan & Parveen, 2014, p. 11).

Beatty and Wiegers (2013, pp. 7–9) agree with Laplante (2017), and state that there are several forms of different requirements. They regard user requirements as something that a user wants to have or be able to do with a certain product. They describe business requirements as high-level business objectives and functional requirements as a behavior that the system needs to perform. They conclude that these three also function as requirements levels. Based on another interpretation by Dick et al. (2005, p. 23) requirement levels are divided into five categories: needs statement, stakeholder, system, system component and subsystem component requirements. The combination of various representations of distinct levels and types can be seen in the FIGURE 1 below.

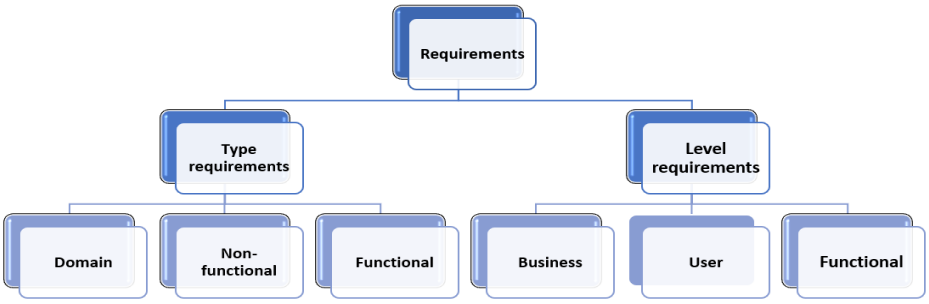

FIGURE 1 Requirement types and levels

The difference between a goal and requirement should be kept evident on the customer and on the engineer's part. A high-level objective that a business, an organization or a system has is a goal and requirements determine how a goal ought to be reached by the intended system. (Laplante, 2017, p. 4). Dick et al. (2005, p. 21) agree that before any system or product can end-up in development the need for such a system has to be established, the reason and the eventual use must be made clear. Thus, the product has a reasonable chance to reach for that conclusion. Without this there is a real change that the production will eventually lead into failure if this phase is not done thoroughly.

Dick et al. (2005, p. 85) and Laplante (2017, p. 21) state that there are "obligations" for requirements. They have concluded that natural language expressions and desires do not translate well into requirements. Those expressions or ideas can be too vague, there can be ambiguity, inadequateness, wrongness, or requirements can be too open for interpretation. Thus, requirements that are chosen must be written down objectively, consistently and chosen requirements must have clear metrics. Criteria for requirements obligations is listed below, divided into six subcategories (FIGURE 2) by Dick et al. (2005, p. 85). There is therefore a need to adhere to a process or utilize a unified form in the company to effectively conclude that "obligations" have been fulfilled.

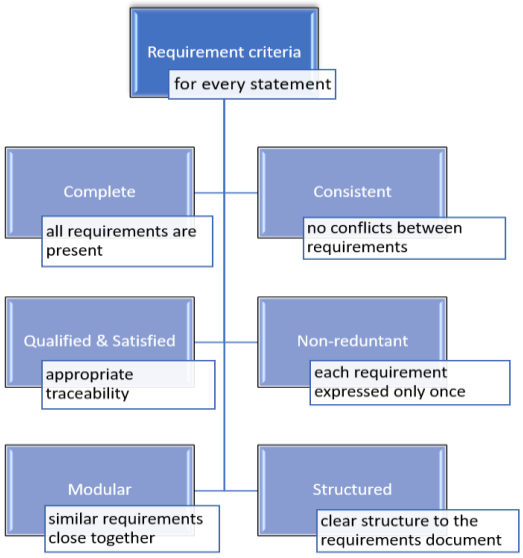

FIGURE 2 Criteria for requirements

The criteria shown in FIGURE 2 are universally applicable, not specific to any project or product. There can be more specified and elaborate criteria for requirements, but these six subcategories provide a good baseline for any requirement set.

2.1.2 Stakeholders

A stakeholder is an individual, a group or an organization which has a stake in the project. They are the ones benefitting from the service or a product - the product or a service is meant for them. Stakeholder is actively part of the project and affects its outcome. There are internal as well as external stakeholder groups. Choosing meaningful stakeholders to a project is a vital phase of the requirements engineering process. At the beginning it is beneficial to include a large number of groups to ensure that no group is accidentally overlooked, participants can be reduced from there. (Beatty & Wiegers, 2013, p. 27). There are several ways and models that aid in the choosing process, most basic are the viewing of an organizational chart and having a conversation with a client.

Beatty and Wiegers (2013, pp. 22, 25) state that there is no substitute for a real customer opinion. A seller or a developer might think that perceived understanding of the customer will suffice when trying to understand their needs. These needs are understood differently with the various levels of involvement of the customer in different development approaches. Good customer relations, extensive customer engagements and co-operation from the start of the project will most likely provide the best results and make the expectation gap narrower between what the customer wants and what the developer delivers.

Beatty & Wiegers (2013, p. 4) comment that stakeholders include project customers, users, developers, inner stakeholder groups and many additional ones. They add that though they all can be involved with a same project they most likely do not want the same thing out of the project or a product. One

stakeholder – like a user might think that a feature is essential to a product and a developer might see it as unnecessary and time consuming to build. Alexander & Beus-Dukic (2009, p. 31) remind that these differences of opinion must be acknowledged, analyzed and negotiated on to discover common ground, this is one of the most critical parts of requirements engineering.

There are various hardships related to stakeholders and their understanding. Lauesen (2002, p. 4) writes that stakeholders may express themselves unclearly. They might have conflicting demands, completion of written and agreed upon requirements. Furthermore, it does not guarantee that the customer is satisfied with the end-result. The product might have a new niche on the market, and it is hard to find initial users. Demands also evolve over time, changing the desire that the customer has originally expressed, this must be monitored.

2.1.3 Requirements engineering model

According to Beatty and Wiegers (2013, p. 4) various problems for software development ascend from the deficiencies that involve learning, documenting, agreeing upon and modifying product's requirements. Requirements engineering model outlines what the development team is trying to produce and aids in establishing mutual understanding on the abstract level, about the solution that has been planned. Dick et al. (2005, pp. 22–23) remind that it can also be used to assure stakeholders about the direction the process is heading to and it documents the system requirements in a structured manner.

Most researchers divide the requirements engineering process (FIGURE 3) to five phases from elicitation, analysis as well as negotiation, documentation, validation to management. Some draw the first four phases on the same level while the "management" phase encompasses the entire process. However, all agree to the number of named process activities which is five (Beatty & Wiegers, 2013, p. 15; Dorfman & Thayer, 2000, p. 1; Eberlein et al., 2003, p. 1; Kotonya & Sommerville, 1998, p. 32; Laplante, 2017, p. 12).

Beatty and Wiegers (2013, p. 45) explain that these phases are interwoven, incremental and iterative. They add that roles cannot be identified because the duties and responsibilities change according to the needs of different companies and products. However, according to Kotonya and Sommerville (1998, p. 36) it is a good practice to identify the roles that ordinarily are associated with the process actions while modelling a process.

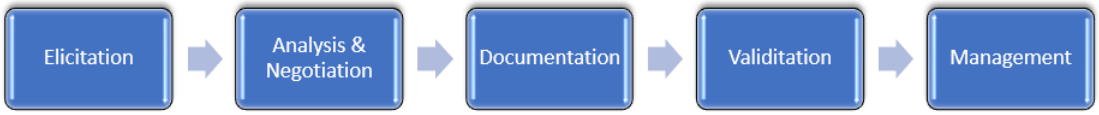

FIGURE 3 Requirements engineering process phases

Next five sections condense the essential idea behind every phase, describe activities related to it and common ways of establishing these required activities.

There are general forms, practices and standards that need to be mentioned in relation to the requirements engineering process model that ensure its successful completion. These principles and theory were also behind the interview form and its drafting.

Elicitation

Requirements elicitation is frequently seen as the first step in requirements engineering process and all other phases follow what has been ascertained during it (Easterbrook & Nuseibeh, 2000, p. 39). Abran et al. (2001, p. 4) conclude that elicitation specifies how the requirements are gathered and where they emerge from, requirement sources and techniques for elicitation. Eberlein et al. (2003, p. 1) view elicitation as a way to establish the requirements, system context and identify the system boundaries. They present various techniques how it might be established one of these techniques is an interview. They state that the goal is to discover facts and opinions that stakeholders have about the developed system.

Easterbrook and Nuseibeh (2000, p. 40) write that some commonly used techniques for requirements elicitation are surveys, interviews, focus groups, prototyping and participant observations. Kotonya and Sommerville (1998, p. 63) add that if interviews are part of elicitation process as they should be in their opinion, they should always be combined with other elicitation techniques. Lauesen (2002, p. 338) reminds that stakeholder analysis as well as supplier and domain-requirements analysis are also used for elicitation. All these methods are relevant, but the appropriateness of a certain measure must be considered project-specifically.

Analysis and negotiation

Kotonya and Sommerville (1998, p. 57) remark that while elicitation and analysis are separate phases of the requirements engineering process they are still closely related and tightly interwoven. Abran et al. (2001, p. 4) write that requirements analyzing is done to detect and resolve problems between different requirements. System limits and desired interaction with the environment must be discovered and these must be translated into intricate system requirements. Classification, conceptual modelling, architectural design, and requirements allocation as well as requirements negotiation is also done in the analyzing phase. Dick, Hull and Jackson (2011, p. 79) add that all requirements need to be identified, classified, elaborated on and their status must be trackable. Also, tracing, placing them into a context and retrieval must be accomplished during the analysis phase.

Easterbrook and Nuseibeh (2000, p. 41) and Eberlein, Maurer and Paetsch (2003, p. 2) agree that conflicts between requirements are solved with negotiation and prioritization with stakeholders and compromises must be made. Easterbrook and Nuseibeh (2000, p. 41) write that a common technique for requirements analysis is customers made requirements prioritization. Eberlein et

al. (2003, p. 2) add that modelling like data-flow models and object-oriented approaches are also common, models provide a way to create abstract descriptions that are open to interpretation.

Documentation

Documentation aids the future maintenance, explains choices and ensures that data is not lost during time or with the loss of key personnel (Eberlein et al., 2003, p. 6). According to Parnas (2000, pp. 3–4) a document is a written description that has an official status or authority and may be used as a legal document. If deviations from the document must be made those changes must be written down and approved by an appropriate role. Code in itself is not a document; it can falsely be thought as a document but in practice programs are so intricate that thinking code functions as documentation is naïve and misleading.

Beatty and Wiegers (2013, p. 19) elaborate that writing and documenting requirements simply means the documentation process about the things that have been learned from the customers or other stakeholders. Clarifying, elaborating, and recording what has been learned ensures that the team works towards the right goal and tries to solve the same problem. Without knowing and comprehending the requirements it cannot be gleamed in any certain manner that the project has been completed or has it been done successfully.

Parnas (2000, p. 1) writes extensively about documentation. He states that documentation is an essential step of requirements engineering process, but it carries a negative label in most people's eyes. Program developers do not want to do it, user documentation is left to technical writers who often do not have the big picture. Thus, it easily leads to incorrect, inconsistent, and incomplete documents that must be revised when the user complains about them. Intended readers prefer not to read the documentation, because they have experienced it to be poorly organized and unreliable. "Help" systems have begun to replace documentation. This is not a sustainable replacement because often these systems can only answer frequently asked questions, and this means that answers can be incomplete and redundant.

Laplante (2017, pp. 107–108) has detailed demands to the writing of the document. He states that it should be clearly written, the writing should be reviewed by other people, there should be a clear structure to the requirement numbering from the first-level 1.0 to the fourth level 1.1.1.1 and the format should be clear, concise, consistent and precise. The positive form and imperatives should be used when shaping requirements such as "email shall be sent" not "email will not be sent".

Dick et al. (2011, p. 77) state that writing down requirements is a technical process, which involves two aspects that must be balanced. Requirements must be processable and their document readable. They state that the document should be well organized, and it should set the requirements into context. Statements should be organized clearly, precisely and be traceable into singular items. Beatty and Wiegers (2013, p. 4) add that comprehensively documenting requirements prevents problems that arise from inadequate user input and in-

formation gathering. Misunderstanding and mismanagement of customer requirements, miscommunicated assumptions, implied functionality, badly specified requirements, and an informal change process can also cause difficulties.

Easterbrook and Nuseibeh (2000, p. 41) write that the way and form to which requirements are documented steers the process forward. It ensures that the requirements are readable, can be analyzed, rewritten if necessary and validated. They state that requirements documentation aids communication between stakeholders and developers.

Laplante (2017, pp. 31–32) writes that before initiating a new development or redesigning it should be described what the desired end-result should do and this is often called a product mission statement or Conops. A product mission statement can be used to gather stakeholder needs and aid in problem understanding as well as the product definition. Product mission statement provides the input for the list of features in the product. It is a short descriptive summary of the product containing the information of the intended users, product purpose and what problem the product will solve. It describes the expected functionalities for the stakeholders and acts as the input for the non-functional requirements identification. Agile methodologies employ a "system metaphor" which can be seen to fulfil the same role to some extent.

ISO/IEC/IEEE (ISO, 2011b) has constructed a structure that aids in documentation. They provide a model like IEEE 29148 launched jointly by the IEEE, IEC, and ISO in 2011 and updated in 2018. Laplante (2017, p. 96) and Parnas (2000, p. 9) write that this model provides an understanding about the software's purpose and framework for requirements assessment. It also provides the means for risk and cost evaluation and helps in verification and validation of plans. Furthermore, it aids in deployment of the product or service to inexperienced users or environments and provides the structure for product improvement. Functional and non-functional requirements can be managed easily with the aid of a document. Eberlein et al. (2003, p. 3) add that the requirements document acts as a foundation for evaluation of the processes such as design and testing of resulting products.

Laplante (2017, p. 97) also recommends a form for System Requirements Specification (SRS) document. It includes the main and subheadings to which the information can be collected. The form is reminiscent of an academic article starting from introduction, scope definition, references, a chapter for specific requirements – including subchapters like functions, design constraints and usability requirements, the last two chapters are verification and appendices.

Laplante (2017, pp. 102–104) reminds that requirements document is intended to be used by multiple users in diverse ways. The document provides information to the customer, aids maintenance and even acts as a legal document and so on. The document should, regardless of the form have a consistent modelling approach and separate operational specifications from descriptive behavior. It should also use consistent levels of abstraction and conformance within the models, include non-functional requirements and omit hardware and software assignments in the specification. Parnas (2000, p. 1) motivates

documenting activities by concluding that without a proper documentation and a model of the system environment, inconsistencies and incompleteness cannot be reliably detected.

Validation

Requirements validation aims to ensure that requirements are correct, whole and consistent. It also ensures that requirements can really be met and a resulting product completes the requirements satisfactorily can be built from them (Bahill & Henderson, 2005, p. 2). Eberlein et al. (2003, p. 3) clarify that requirements validation certifies that the chosen requirements are acceptable and accurately represent the system that is to be implemented. Validation requires multiple iteration rounds to fully develop requirements into "good enough", "perfect" is unrealistic, but a mutual understanding must be reached. This agreement is according to most done in cohesion with the customer.

Beatty and Wiegers (2013, p. 17) elaborate that validation is accomplished with reviews of the documented requirements and based on those reviews' acceptance tests are developed. Kotonya and Sommerville (1998, pp. 87–90) insert that these requirement reviews are the most common technique for validation and validation should answer the question "do we have the right requirements, and did we understand them correctly". In the validation phase the customer is heard and a confirmation about the needs of the customer and achievable business objectives must be charted.

Management

Requirements management does what the term implies, it helps to manage information and its changes, in this case the altering requirements. Eberlein et al. (2003, p. 3) specify that management means capturing, storing and dissemination of information. Kotonya and Sommerville (1998, p. 117) dictate that the most essential responsibility of requirements managements is to ensure that all the requirements have a unique identifier. They elaborate that this is an apt way to measure the effectiveness of requirements management.

Easterbrook and Nuseibeh (2000, pp. 41–42) state that managing the evolution of requirements is essential and ability to trace requirements to their origin is important. Tracing provides reason for the requirements inclusion as well as sheds light to the impact of the specific requirement. This provides integrity and completeness to the documentation which is integral in change management. Dick et al. (2011, p. 182) highlight the stakeholder perspective. They state that requirements management means the capturing, tracing and management of stakeholder needs and inspecting their changes throughout the process lifecycle.

Kassab, Laplante and Neill (2014, pp. 5, 8) investigated requirements engineering practices in 2013 among 119 interviewees from 23 countries. When asked about the requirements review and inspection 53 % of respondents answered that they used some methods. On average there were 2.29 various an-

swers per individual. These researchers listed techniques such as team review, ad hoc walk-through, checklists and formal walk-through, scenario and others.

2.1.4 Requirements engineering in a software development process

Dale and Saiedian (2000, p. 419) write that communication and co-operation are key components in successful requirements engineering process, like they are in many other instances. When developing a new product, technical, cultural, interpersonal, and organizational factors must be considered. These factors form the context of the software product and affect its design and features.

Requirements engineering for a software development process covers a wide spectrum of viewpoints, roles, responsibilities, and objectives. Software can be developed traditionally or with agile practices. Requirements engineering is perceived to be a traditional tool. Traditional development is often structured into strict phases and has a lot of documentation. Agile methods are code- and people oriented and perceived to be less process and documentation centric. Because of this difference and the need to document less and do more, requirements documentation process can be left wanting with agile methods. The five phases involved with requirements engineering process are present in the agile methods to some capacity (Eberlein et al., 2003, p. 6).

The development life cycle that the organization has chosen be it a waterfall, iterative, incremental, phased, agile or a combination model, must complete the requirements model activities, this is an easy way to improve customer satisfaction. (Beatty & Wiegers, 2013, p. 15).

All the components that form the unified whole of this chapter inspect requirements engineering as a process model with certain activities. These activities can be systematically completed with the aid of a similar model and the activities can be combined with agile practices. Agile embraces change and the customer wants to know her requirements are met in the resulting software. Reassuring the customer does not mean that the process cannot be agile at the same time. According to Beatty and Wiegers (2013, p. 41) a client can sign-off on the requirements based on the user stories, this can be an acknowledgement a ”we are here” conversation today, it doesn’t mean that tomorrow the process cannot be somewhere else. This sign-off would simply ensure a mutual understanding and function as a point of reference.

2.2 Information security

In the 1980s computers entered to the field of commerce and the cheap software and hardware spread widely to consumers both in business and private sector. This expansion of information and communication technology increased data invasion and thus shifted the focus of security from hardware to data and information. (Kamkarhaghighi, Moghaddasi & Sajjadj, 2016, p. 5).

Before this era, machines and computers were limited in number and used mostly in military environments, where the information was secured and supported by the military. This quick shift caused a need to set up new priorities for information security in commercial settings. New unaccustomed commercial users lacked data security, strict physical data support as well as initiated unintentional and intentional cyberattacks. This decade (1980) started intensive discussions about security of data and information. (Kamkarhaghighi et al., 2016, p. 5).

After 40 years of study, information security is a widely researched field. Therefore, information security has many definitions. These definitions can be technical, behavioral, philosophical, managerial, or organizational, depending on one's viewpoint. In this case, this research focuses on managerial point of view, representing the elements needed for secure software development from the perspective of information security.

The first subchapter contains some of these definitions used in the field of information security, describing both words separately, which together form the whole of information security. Therefore, the intent of the first chapter is to answer the question; why information security is important.

The second subchapter, however, will represent several models, which have been composed to explain the key attributes of information security. To accomplish information security in a software, these attributes must be met. Therefore, they also serve as information security goals, contriving the first element: security objectives of the software. They provide an answer to the question; what are the objectives that the software development organization must establish with the software to make it secure.

In the third subchapter present are the components of information security: computer and data security, network security, policy, and information security management. All these parts are essential for achieving information security in the software. However, when the scope of this thesis is limited to the creation of a model used in secure software development, the focus is on both information security management and its policies. The aim of this thesis is to aid the thesis's commissioner in his endeavors to implement information security into the requirements engineering process. This requires a comprehensive understanding of information security management and its components in requirements engineering context. Therefore, this chapter answers questions; what components are critical in information security management in the context of secure software development and therefore critical for further investigation.

The fourth, and the last subchapter of information security entity, focuses on the critical components presented in the third chapter. It represents threat modelling and risk assessment as crucial components for secure software development, which will also play a vital role when the final, combination model is composed.

2.2.1 Security, information, and information security

Security is defined as "the state of being or feeling secure". Secure alternatively, is defined as "free from danger, damage etc.; in safe custody; not likely to fail; able to be relied on". (Collins English Dictionary, 2019). In a general sense security signifies protecting our assets.

Information is defined as a representation of knowledge in a stored form or as data in the phenomenon's environment – data in its context (Madden, 2000). Van Niekerk and von Solms (2013, p. 100) infer, that the stored form of data and a possibility to transmit it, leads to a conclusion that information is also a possessable asset to a user or an organization. Therefore, information security as an entity, simply denotes all aspects of protecting information and business through it. Mattord and Whitman (2017, p. 10) agree with Niekerk and von Solms, clarifying that the type of security is determined by the ultimate objective of it. Information security's objective is logically information, through all the stages of its life cycle, from the creation until the eventual end-of-life.

Peltier's paper (2013, p. 15) considers information (as an objective) even closer. It divides it into two distinct parts: 1) information assets not using information and communication technology (ICT) and 2) information assets using ICT. This division is also part of the fundamental idea in van Niekerk and von Solms's paper (2013, p. 101), which aims to clarify the relationship between information and communication-, information- and cyber security (FIGURE 4). This paper focuses on the context of information security, where information assets are using ICT.

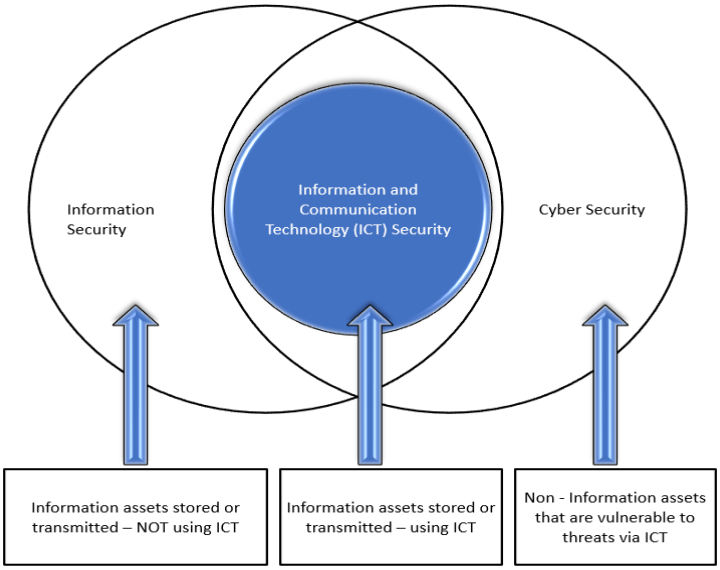

FIGURE 4 Relations between information security, ICT-security, and cyber security

Von Solms and van Niekerk (2013, p. 98) explain that the intent of information security is to guarantee business continuity and to reduce business damage by confining the security incidents' impact. Therefore, as the Finnish security committee (2018, p. 15) defines in its vocabulary of cyber security that infor-

mation security could be conceived as a state, where information security risks are under control, but also as an umbrella term encasing all the arrangements aiming to ensure it.

National Institute of Standards and Technology (NIST) (2017, p. 2) in turn, describes information security through the means of protection of information and information systems. They clarify that information security is protection from unauthorized actions such as access, use, disclosure, modification, disruption, or destruction to ensure availability, confidentiality, and integrity.

Over the years several models of information security have been presented. These models are performing attributes that the organization is required to meet to attain information security. These attributes or features are continuously growing in number with greater capabilities to achieve information security but also emphasizing the role of control and assurance needed for information security management. Following sections will present three models of information security to gain a deeper understanding about the attributes that have an effect to a secure state of information. These attributes can also be defined as objectives of information security.

2.2.2 Models of information security

Availability, integrity, and confidentiality are three of the primary concepts of information security. The collection of these concepts, as shown in the FIGURE 5 below, is commonly known as the CIA- triad and was first presented in 1987 by Clark and Whilson (Kamkarhaghighi et al., 2016, p. 5).

FIGURE 5 The CIA- triad

In CIA- triad model confidentiality means the access to information. Clark and Whilson state that the access should be allowed only for those, who have legal disclosure and for that reason authorized restrictions should be preserved. The second concept, availability, means that the access to the information should be timely and reliably ensured. Lastly, the third concept, integrity means guarding against improper modification and destruction of information. (Kamkarhaghighi et al., 2016, p. 2).

The CIA- triad might be seen as a too restrictive with its definition of information security. In 1998, in his book "Fighting computer crime: a new framework for protecting information" Donn Parker (1998, p. 85) proposed an alternative and more extensive model. It later gained a title: The Parkerian hexad (Andress, 2011, p. 6). The Parkerian hexad (FIGURE 6) is a variation of the classic CIA- triad. It represents a set of six atomic elements of information including the elements presented in CIA- triad (confidentiality, integrity and availability). Parker (1998, p. 85) adds three new elements to the classic combination; possession, authenticity and utility.

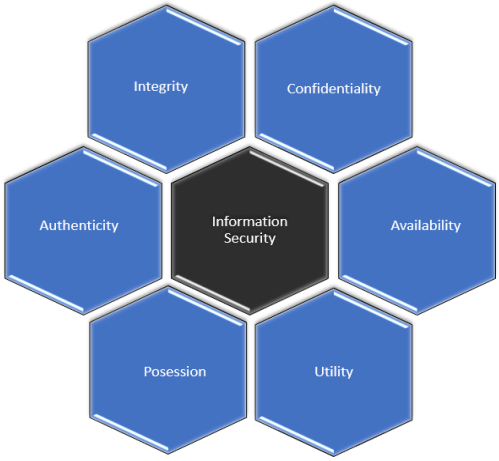

FIGURE 6 The Parkerian hexad- model

The Parkerian hexad- model represents the possession of information as a quality or state of ownership or control of an object or an item. Parker (1998, p. 85) highlights that possession of information should be one of the core attributes and protected against theft. Andress (2011, p. 7) notes that in case of information, it is in one's possession if it is independent of format, other characteristics and obtained by the individual. Therefore, he states that it refers to a physical tendency of the media on which the data is stored. Mattord and Whitman (2009, p. 13) add that by removing the data from its secured environment - its store, is consequently a breach of possession.

Parker (1998, p. 85) specifies that authenticity conforms reality. Andress (2011, p. 7) clarifies that authenticity is necessary for ensuring that the data, documents, transactions, communications and parties involved with the action are genuine or original. This requires that the data, for example, can be verified and therefore trusted. It allows for a discussion about the appropriate attribution as to the proprietor or author of the data in question.

Parker (1998, p. 85) describes utility as the measure of how useful data is in the hands of its user. Andress (2011, p. 8) adds that a user could be an attacker having unauthorized access to encrypted backup tape, when the utility is little compared to authorized users with the encryption keys. Mattord and Whitman (2009, p. 12) summarize the utility of information as a value to a particular purpose or an end that it can serve. Available information needs to meet user requirements to be useful to the user otherwise it is rendered useless.

In agreement with Donn Parker, also Ross Anderson (2001, p. 7) corroborates that information security is not covered entirely by the CIA- triad. He declares that the approach to information security is multidimensional and presents the idea that people, are not less essential than the technical features. He claims that a solely technical approach to information security is not effective.

Anderson's (2001, p. 7) general view on the economic incentives behind information security point out that collaboration between lawyers, economics and managers is necessary to solve the problems of information security. However, Gordon and Loeb (2002) took a deeper look to Anderson's economical approach and created a model, which aims to aid in determining the optimal amount of investment in information security. The work was based on the idea of information security, with goals of confidentiality, availability, integrity, authenticity, and non-repudiation. This model is generally known as information assurance model.

The term Information Assurance (IA) was invented in 1998 by the US Joint Staff. It was released for the first time in Joint Doctrine for Information Operations (1998, p. 51). The term itself has been formulated from two parts, where the first part - information - was earlier defined as a representation of knowledge in a stored form. The second part – assurance – stands for the state of being assured, such as being secured (Merriam-Webster Dictionary, 2020).

NIST (2020) defines IA measures as a protection and defense of information and information systems by assuring their availability, integrity, authentication, confidentiality and non-repudiation. IA measures consist of incorporated protection, detection, and reaction capabilities to provide restoration of information systems. IA was originally retrieved from the concept of information security and its definitions. It incorporated the CIA -triad into a definition of five pillars of information assurance. (Dardick, 2010, p. 3). As presented below in FIGURE 7 IA includes four familiar attributes; availability, integrity, confidentiality and authentication (authenticity), but also represents a new attribute called non-repudiation (Joint Pub, 1998, p. 51).

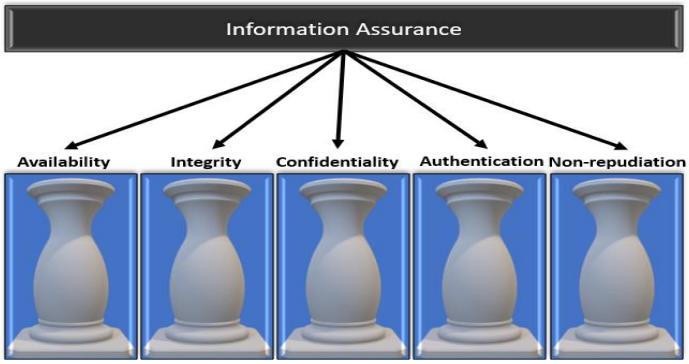

FIGURE 7 The model of the five pillars of IA

In the Joint pub's (1998, p. 51) first publication of IA, non-repudiation was described shortly as; undeniable proof of participation. Later Committee on National Security Systems (CNSS, 2010, p. 50) opened this term in more detail. Their instruction No. 4009 described that non-repudiation of the information

assures, that the sender is provided with proof of delivery and the recipient receives proof of the sender's identity. After that, neither party could deny completed actions like creating information, sending a message, approving information and receiving a message.

Joint Task Force Transformation Initiative (2013, p. 50) completes these two definitions by stating that the role of non-repudiation is to protect individuals against later false claims such as denying actions made by different parties. Also, the authors behind of the authorized documents, senders that have transmitted messages, receivers that have received messages, or signatories that have signed documents.

All previously presented models; the Five Pillars of Information Assurance, the Parkerian Hexad as well as the CIA -triad, included confidentiality, integrity and availability (TABLE 1). Derived from that fact, these three attributes form the fundamental core of information security.

TABLE 1 Core attributes of information security

Attribute/Model	**The CIA – triad**	**The Parkerian hexad**	**The Five pillars of IA**
Confidentiality	X	X	X
Integrity	X	X	X
Availability	X	X	X
Possession		X	
Authenticity		X	X
Utility		X	
Non-repudiation			X

Campbell (2016, p. 5) claims that these three fundamental attributes of information security are also special security properties. They are attached to every security action, such as risk mitigation or security control implementation that is done and there is always one or more of these properties covered from this perspective. As described earlier in this chapter, security actions protect assets. Therefore, these three attributes apply to every asset that we protect (FIGURE 8).

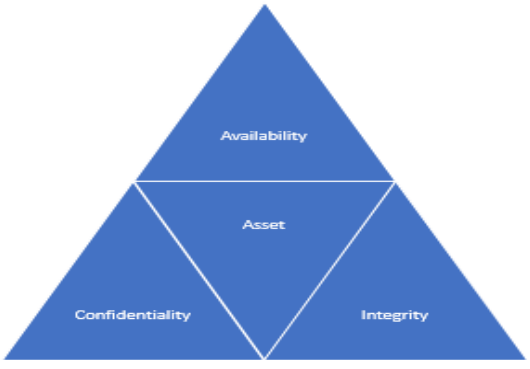

FIGURE 8 Relationship between security properties of IS and asset

In the case of information security, all the protection measures secure these attributes and therefore protect assets. Campbell (2016, p. 6) writes that when the

organization is designing solutions to improve their security, they must analyze all the threats affecting these security properties: confidentiality, integrity and availability. Campbell (2016, p. 98) also presents that security controls implemented to mitigate those threats should be matched against the security classification schemes defined by the business. Security classification should be established in the preliminary stages of information security implementation project.

2.2.3 Components of information security

Whitman and Mattord (2013, pp. 4–5) call information security with a term Infosec and represent it as a combination of three main components: management, computer and data security and network security. These three main components have a common overlapping area a policy shown in the FIGURE 9 below.

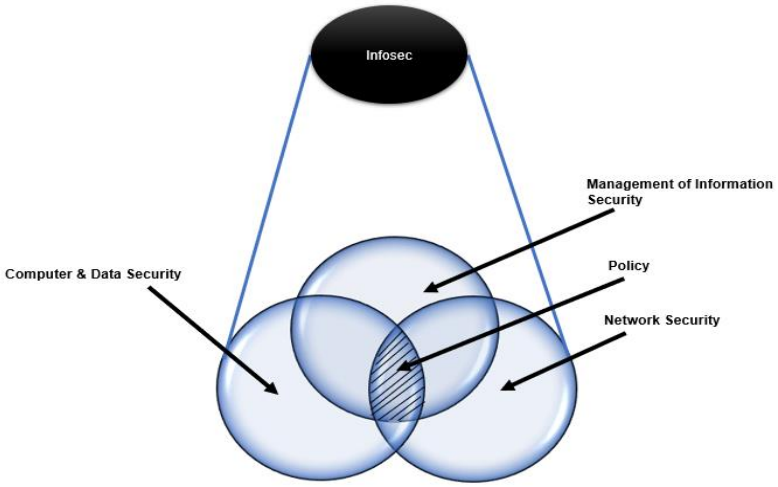

FIGURE 9 Information security components

Mattord and Whitman (2011, p. 177) state that information security policy forms the basis for all information security planning, design and deployment. Those policies direct how issues should be addressed, and technologies used. Accordingly, information security policy is a management tool that obligates personnel to operate in a manner that protects the security of information assets.

According to Mattord and Whitman (2013, p. 4) network security focuses on protecting data networking devices as well as connections and their contents. Anuradha and Pawar (2015, p. 504) summarized this in their paper by stating, that network security means that message sent from one nod to another as well as computers at the end of the communication chain, are secured. However, Pandey (2011, p. 4351) depicts the objective of the network security from user-perspective. He states that the purpose of network security is to assure that the network performs in critical situations and it has no damaging effects for user or employee.

Computer and data security, in turn, include protection of all the systems and hardware that are applied to using, storing or transmitting information (Mattord & Whitman, 2013, p. 4). According to Ahmad, Horne and Maynard

(2016, p. 3) computer security is also known as Information and Communication Technology (ICT) security. Data security, in turn, is defined by Consortium of European Social Science Data Archives (2017, p. 1). It defines data security as data protection from accidental or malicious damage.

As defined earlier in this paper, information is defined as a representation of knowledge in a stored form. In order to understand the difference between information and data security, closer look at the Data-Information-Knowledge-Wisdom (DIKW) -hierarchy specified by R.L. Ackoff (1988, p. 1) presented in FIGURE 10, might be in order.

FIGURE 10 Hierarchy of data, information, knowledge & wisdom

According to Ackoff (1988, p. 1) data symbols represent the properties of both events and objects. Information, in turn, consists the processed data, which increases its usefulness. It is contained in descriptions and it can provide answers to questions like who, what, where, when and how. Therefore, data security is a component of information security. As described earlier, both computer and data security function on the same level of the DIKW -hierarchy. Thus, they can be discussed as a united entirety: computer and data security.

Information security management can be seen as one of the most essential components of information security. Mattord and Whitman (2011, p. 176) express that far too often information security is considered as a technical concern, when it is, in reality, a management issue. To tackle these issues, information security management should meet the goals of information security governance. Mattord and Whitman (2011, p. 177) conclude that firstly, information security should be in alignment with the business strategy to aid organizational objectives. Secondly, it should include risk management, which executes appropriate measures to manage and mitigate threats related to information resources. Thirdly, information security knowledge and infrastructure should be utilized efficiently and effectively by the resource management, and fourthly information security performance should be measured, monitored, and reported to ensure that the objectives of the organization have achieved. Lastly, Mattord and Whitman (2011, p. 177) suggest that information security investments should be optimized in order to support organizational objectives.

However, Raggard (2010, p. 7) highlights that there are no off-the-self solutions on information security management, because security requirements always vary depending on the vulnerabilities and threats associated with the environment in question. That is also why the effects and consequences of similar

security incidents vary from one environment to another. Thus, information security management, as well as security investigation, must be risk driven.

According to Alexander, Finch, Sutton and Taylor (2013, p. 6) Information Security Management System (ISMS) concept is part of an overall management system of the organization, based on a business risk approach. It is used for establishing, implementing, operating, monitoring, reviewing, maintaining, and improving information security.

International Organization for Standardization has created a standard model for information security management (Calder & Watkins, 2010, p. 11). It is based on risk management, which is divided into two phases: 1) Risk assessment and 2) Risk treatment. The first phase, risk assessment, is a process that is used to identify threats and assess their likelihood for exploitation of a vulnerability (FIGURE 11). This phase also evaluates the prospective impact of such an incident transpiring (Calder & Watkins, 2010, p. 17).

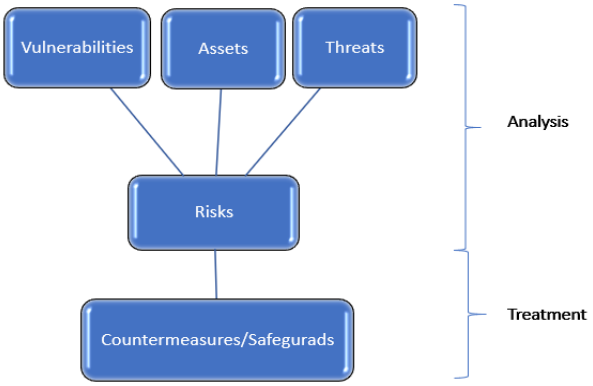

FIGURE 11 Risk management phases in ISO/IEC 27001

The second phase, risk treatment, takes estimations about threats and risks as well as impact as an input. It aids the organization to mitigate risks with proper countermeasures and safeguards. (Calder & Watkins, 2010, p. 18).

The objective of this model is to create and implement a risk management strategy into organization to reduce undesirable impacts. Additionally, it also delivers a structured and consistent basis for deciding among the risk mitigation options. (Calder & Watkins, 2010, p. 17).

Mattord and Whitman (2017, p. 255) also presented risk management as an integral principle of information security management, when the organization wants to maintain objectives of information security. In their publication (2017, p. 256) risk management included three parts named as "three major undertakings" and therefore, the model was named here accordingly. These undertakings were risk identification, risk assessment and risk control as presented in FIGURE 12.

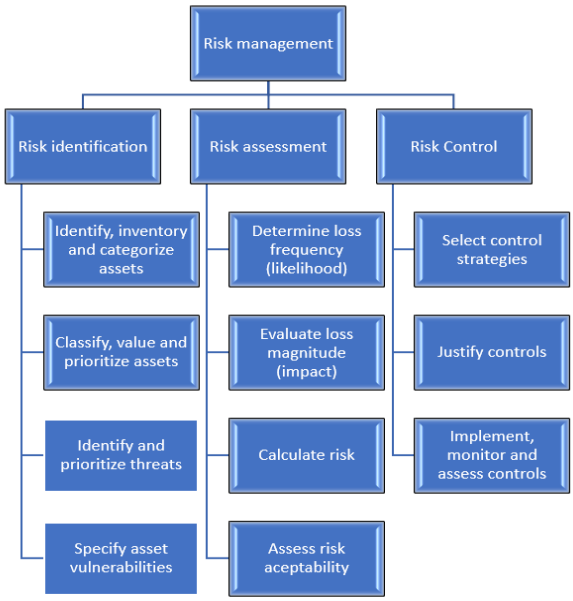

FIGURE 12 Three major undertakings of risk management

Mattord and Whitman's (2017, p. 256) model is also a high level graph, but it differs from the ISO/IEC 27001 in analysis, splitting it into two separate phases: risk identification and risk assessment. It also offers a more detailed information on how to manage assets, threats, and risks as well as how to use this information to utilize risk controls. Compared to the first high level model, Mattord and Whitman's presentation also pays more attention to control monitoring. While ISO/IEC 27001 focuses more on the existence of the process, Mattord and Whitman's model describes its content in detail.

Siponen (2006, p. 97) supports this conclusion by stating that information security management standards, like all standards, have a certain feature: they are process oriented and more concerned about the existence of a process rather than the content of it. This produces a two-folded problem. First, standards are more concerned about ensuring that particular security activities exist in the organization but disregards evaluation of how well those activities are conducted. Secondly, standards provide processes, guidelines and principles that are simple and abstract and provide no instructions on how desired end-results are to be reached in practice.

Therefore, it can be stated that even if the organization has an ISO/IEC 27001 standard, it only guarantees certain process activities existence from the information security perspective, but not the efficiency of those activities. To implement information security effectively into requirements engineering process, the process itself should be investigated from information security management point of view.

As presented earlier through the examples of ISO/IEC 2001 standard and Mattord and Whitman's model of three major undertakings, the most meaningful part of the information security management is risk management. Risk management contains asset, threat and risk identification and modelling as well as

security control creation, through the understanding of possible vulnerabilities related to the software. Next subchapter presents these terms shortly in the context of software development.

2.2.4 Information security controls in a software product

As stated earlier, confidentiality, integrity and availability (CIA), are the goals of information security. These goals must be met in order to provide information security in a system and protect its assets (Haley, Laney, Moffett & Nuseibeh, 2008, p. 138).

Havadi et al. (2008, p. 5) state that assets are the abstract and innate resource of the system. Alexander et al. (2013, p. 21) add that assets vary in form from tangible to intangible, but when consequences of the security incident are examined, assets are always impacted. If the asset is stolen, lost or damaged in any way, the organization will suffer from the result. In case of severe damage, organization might never recover.

Impacts are consequences of realized threats (Alexander et al., 2013, p. 21). According Alexander et al. (2013, p. 2) threat is a potential cause of an incident, which may result in harm to a system or an organization. It depends on the perspective, environment, and situation that it is being considered. Haley et al. (2008, p. 135) add that threats, that might violate assets, can be constructed by enumerating the assets of the system and then estimating all those actions that would violate the security concerns of them. Bernsmed et al. (2019, p. 2) call the process as threat modelling.

Threat modelling is a requirements engineering approach, which is used for specification of security requirements (Hadavi, Hamishagi & Sangchi 2008, p. 5). Bernsmed et al. (2019, p. 2) write that a well-defined threat model aids organization in threat identification related to assets of the system. This identification is done through well-founded assumptions of the capabilities of an attacker, who might be interested in system exploitation. Threat modelling also enables the development teams to discover the most crucial areas of the system design, which must be protected. Through this process the mitigation strategies can also be easily determined (see the FIGURE 13).

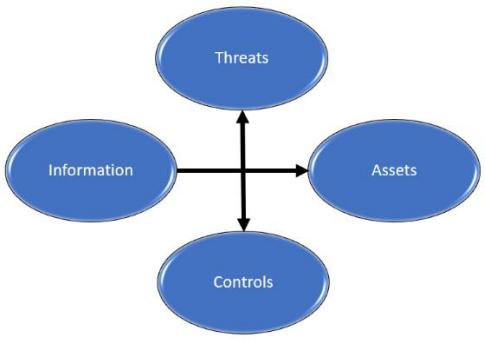

FIGURE 13 Purpose of security requirements

According to Ansari et al. (2018, p. 4) security requirements engineering considers security interests of all the stakeholders of the software product. In addition

to threat modelling, it also takes risk analysis into account. Alexander et al. (2013, p. 22) explain that a risk is considered as combination of the impact and the likelihood that the threat can be realized. Therefore, when the overall risk is calculated, both factors must be estimated and considered threat by threat (Alexander et al., 2013, p. 23). Ansari et al. (2018, p. 7) continue that the typical risk calculation formula used in industry is; "Risk = probability x damage potential". This formula aids in evaluating and prioritizing threats according to their potentiality, after which threat mitigation decisions can be made.

Security requirement is a countermeasure to a threat. Ansari et al. (2018, p. 7) write that security requirements are security controls of the software, created to mitigate the identified threat. However, unmitigated security threats are vulnerabilities, that the attacker might exploit, to damage the system. Additionally, every threat is not detrimental to the system and thus, mitigation for every threat is not necessary.

Alexander and Beus-Dukic (2009, p. 217) remind of the main hardship with security requirements; it's not possible to have absolute security against anything, guarantees cannot be given nor can there, truly, be an honest probability that a specific threat shall be defeated. Actions taken or defenses constructed cannot guarantee security or give an estimation of the time that they will provide protection for. Security is a trade-off that fact must be accepted and responsibility for those choices must be taken. There it is a choice between those design steps that will be sufficient yet affordable.

Abraham et al. (2016, p. 18) note that once the list of essential security requirements has been established, continuous risk monitoring should be organized. This means that the organization must make sure that the agreed security requirements are implemented and that the measures taken have a desired effect on security. Additionally, other new requirements or technological choices will add new threats to the list and therefore iterations are needed. McGraw (2006, p. 88) adds that keeping security requirements up to date and even identifying the appropriate ones is an intricate undertaking deserving proper attention. In his opinion software security aims to build software that can endure attacks proactively.

2.3 Software development and secure development models

This chapter combines the ideas of requirements engineering, software development and information security that were represented during the previous subchapters. It concludes the relations between these theories and formulates the foundation for the second research.

2.3.1 Software development

According to Boehm (2006, p. 13) in the 1950's software was developed like hardware and the development process followed this trend. In 1984 Zave (1984, p. 104) wrote that the conventional life cycle has experienced chronic problems in software development. Development processes have a long history on the mechanical side and even the software development has had time to gain a relatively long history. However, this hardly compares to the mechanical and hardware development history.

Zahran (1998, pp. 389–390) states that a software process simply means that procedures that ultimately precedes to the development of a software product. Aitken and Ilango (2013, p. 4752) on the other hand have more encompassing definition and they conclude that software development is in its basic essence the art of problem solving, no matter how it is performed or what is the process used, the goal is to solve a problem. This requires an understanding about the problem. Its requirements must be elicited, analysis must be done to ensure correct problem comprehension and design the best solution with specification and implementation – using the resulting solution.

Maciaszek (2007, p. 3) agrees with this definition and remarks that software is fundamentally complex, modern systems even more so, their size, interdependencies between components and the amount of data the system has to process make these system intricate. This sets some requirements to the model which often must adapt to various process sizes and provide a framework to various projects. A certain model size might be suitable to a large company when a smaller counterpart requires a less cumbersome model. Zahran (1998, p. 78) concludes that software process improvement helps the development of new products and their revision. These are also improvement objectives that this thesis aims to enhance.

Kotonya and Sommerville (1998, p. 30) define process models as simplified descriptions of a process, one model views the process from its perspective so one process might be described with multiple modes such as fine- and coarse-grain activity-, role-action- and entity-relation models. The coarse-grain activity model is used in this thesis to encompass the sequencing of the requirements engineering actions. These activities are elicitation, analysis and negotiation, documentation, and validation, additionally the model used in this thesis involves requirements management.

Kotonya and Sommerville (1998, p. 9) also state that there is not a single process that suits all organizations, every organization chooses its own process which are appropriate for the type of systems that is developed, fit the organizational culture and the expertise level and abilities of the people working with requirements engineering. Easterbrook and Nuseibeh (2000, p. 37) conclude that the most prominent mark of success for a software system is the fulfilment of the purpose for which it was intended and designed.

Qadir and Quadri (2016, p. 189) write about the organizational reliance to the use of information and communications technology. The have gathered that

the most meaningful resource are the information system and the network to which it is connected to. They have divided the information system components that affect the information availability into three classes: software, hardware and network (FIGURE 14). From which the software is the most critical of the three in their opinion. Hardware and network run on their operating codes and the code is the factor that gets under the attack. All the security attacks and the solutions addressed to those attacks are addressed via software or through the operating code. To secure the information system, the software must be the priority. Pressman (2005, p. 5) specifies that hardware is physical: wires, circuits or chips, but software is non-physical, the code that is running on the machine. Everything from data to web sites to different apps can be software. Software is something that a human or engineers develop, hardware is manufactured.

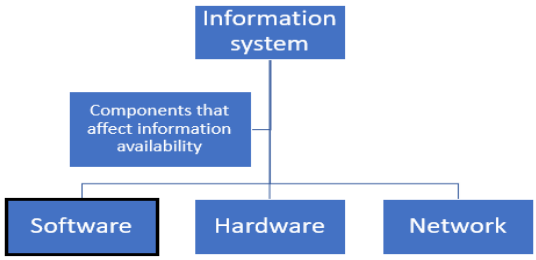

FIGURE 14 Information availability component classes

Buyens, De Win, Grégoire, Joosen and Scandariato (2009) states that typical software development process is divided into nine phases: education and awareness, project inception, analysis and requirements, architectural design, detailed design, implementation, testing, release and deployment and support. Apvrille and Pourzandi (2005) on the other hand state that project lifecycle includes an iterative process of analysis, design, implementation, testing and maintenance.

2.3.2 Various levels and types of software

Qadir and Quadri (2016, pp. 189–190) divide software architecture into three distinct levels: service, component or object/class level. Service level is the highest level of application software architecture and the external view of the system, it also is the level that hackers or attackers exploit open services if these are found. Component level is the second, components are accessed via interfaces and provide service to client programs that produce the correct interface, because of this strong authentication and access control mechanisms must be planted. Object and class levels are the last and most fine-grained level of software architecture, it ensures efficient performance and secure functioning. If a secure system is the aim of a development process all these levels must be considered and secured.

Doyle (2000, p. 111) divides software into two categories: systems software or operating software and applications software. Bizzell, Clinton, Prentice and

Stone (2017, p. 321) disagrees and gives a three-way divide into systems software, programming languages and application software. Doyle (2000, p. 111) defines systems software as a program that directly controls the computer and ensures the usability of the hardware. The systems software can be described as the bridge that connects the hardware and the application software together. Application software must obey certain rules that the software and hardware platforms demand and integrate to existing information systems.

Doyle (2000, pp. 111–112) dictates that systems software must also ensure that the resources of the computer like internal memory or input and output devices are effectively managed. Systems software includes are operating systems, utility programs, file management programs and virus detection software. Systems software enables the user to multitask while using the computer.

Maciaszek (2007, p. 3) remark that business processes and requirements change continuously and application software must be built to accommodate change. According to Kotonya and Sommerville (1998, pp. 12–13) there are three main classes of systems that are developed for the customers; information systems, embedded systems as well as command and control systems. Apvrille and Pourzandi (2005) remind that every development process should be initialized with clearly defining the behavior that is expected of the software. In their opinion security development must be done with the same rules in mind, concepts must be defined for the security environment and its purposes in the development process's initial stages.

2.3.3 Software development models

In 1984 Zave (1984, p. 104) defined "software development" as an effort to solve a problem with a computer system and spoke about the deficiencies of a conventional model in software development. Software development methodologies offer a context for planning, executing, managing, and controlling the process of software system development. Ruparelia (2010, p. 8) lists multiple models that are used from traditional and agile sides. Three of the listed models are Waterfall, Kanban and Scrum. This listing is continued by Ghilic-Micu, Mircea and Stoica (2013, pp. 72–73) and still further by Butler and Vijayasarathy (2016, pp. 86–89). Bassil (2012, p. 1) writes that all models have the same principle, they have steps or phases that must be completed to have results and produce a product. All methodologies that came up during the literature perusal are shown in

TABLE 2, agile practices are used as a foundation for the model that is produced for the commissioner and some parts follow Scrum. That is why Agile principles and Scrum are described in more detail.

TABLE 2 Traditional and agile methodologies

Traditional		Agile		
Spiral	SDL	ASD	JAD	RUP
V-model		Crystal	Kanban	Scrum
Waterfall		DSDM	Lean	TDD
		FDD	Prototyping	Unified
		Incremental	RAD	XP
		Iterative	Rapid	

Maciaszek (2007, p. 6) claims that most contemporary software development processes are consistently iterative and incremental. Babar, Liming, Ming and Verner (2004, p. 520) state that on the abstract level Waterfall and Agile are very different but their practices in the development cycle do share parallels. Hossain and Moniruzzaman (2013, p. 5) confirm that traditional and agile methodologies have different characteristics. They write that traditional development trusts in predictability, specificity and extensive planning. Agile development relies to small teams, continuous design improvements and feedback. Jain and Patel (2013, p. 1386) specify that in traditional methodologies the process in plan driven and the process is initiated with requirements elicitation and documentation, after which architectural design and design development and inspection follow.

Balaji and Murugaiyan (2012, p. 26) write that iterative and incremental development form a base for agile software development which is a group of software development methodologies where requirements and resolutions develop through cooperation amid self-organizing cross-functional teams. Cho (2008, p. 188) confirms this and adds that agile methods do emphasize iterative and incremental development and also focus on customer satisfaction, frequent and fast delivery including quicker adaptation on requirements changes.

Babar et al. (2004, p. 523) specify that customers support the development teams through the whole development process in agile models. In waterfall model the customer is typically involved during the requirements definition phase and sometimes during system or software design. They do not however contribute significantly and are not as involved as the customer is in agile models.

Balaji and Murugaiyan (2012, pp. 28–29) define Agile as "moving quickly", an adaptive team can respond to changing requirements swiftly and changes are welcomed. It has iterations instead of phases. Rapid delivery at short intervals and keeping the customer satisfied are the most important principles, these are achieved through continuous communication with the client and involving the client to the process.

Cho (2008, p. 191) tells that Scrum is an agile process that operates an empirical process control with three points in all its implementations; transparency, inspection and adaptation. Transparency implies that all facets of the process that influence the outcome must be kept evident. Inspection entails that the aspects of the process are examined periodically to detect any undesirable variances in the process. Adaptation means that if the inspection discovers any undesirable aspects the process will be adjusted accordingly.

Ghilic-Micu et al. (2013, p. 74) write that Scrum is centered on two aspects: team autonomy and adaptability. Scrum does not focus on implantation level practices but rather on how the members of a development team should co-operate to produce a flexible, adaptive, and productive system in a continually transforming environment. Cho (2008, pp. 191–192) concurs and adds that Scrum process consists of responsibilities, meetings and texts. How the work is divided and what roles do the team members have, how are the meeting organized and when and ultimately, what text material does the process produce.

Next the traditional methodologies are represented. Those have also been collected to TABLE 2 and Waterfall is described in more detail because its principle forms the foundation for the model that is being produced. Ruparelia (2010, p. 8) remarks that Waterfall or a cascade model relies firmly on requirements definition and analysis before development initiation Babar et al. (2004, p. 521) write that waterfall model is divided into five consecutive phases, each phase results in well-defined deliverables. Every phase requires the deliverables of a previous phase as an input so, no subsequent phase can commence before its predecessor has produced its deliverables and they have been signed.

Balaji and Murugaiyan (2012, p. 27) specify that Waterfall model has sequential steps that must be completed before the next one can be initialized, there is no overlap between the phases and because it is a linear model it is easy to implement. Documentation and testing are conducted after every phase to maintain high quality of the project. Requirements are frozen from the very beginning of the project and changes are not considered; this means that requirements are clear before development starts. Waterfall does not consider changes well, if a stakeholder changes their mind or a new need arises it will not be taken as a part of the current development process.

Lauesen (2002, pp. 3–4) reminds that Waterfall model is an ideal, one phase is not always completed before the developer embarks onto the next one, something must be redone, iterative analysis, design and programming take place and then several phases are repeated more than once. Analysis, design and programming happen, but often these actions take place iteratively and concurrently. This in turn leads to altered requirements when missing, wrong and unrealistic requirements are spotted, this is where requirements management is needed. Cho (2008, p. 189) remarks that Waterfall model has drawbacks, it is inflexible and it is often not completed on-time or on-budget, rather it is often finished with less features and functions than intended and one third of the projects get cancelled altogether.

Babar et al. (2004, p. 525) argue that software quality cannot be compared realistically or reliably between waterfall model and agile methods because their initial development conditions are not equal particularly concerning to cost. Mitchell and Seaman (2009, p. 514) add that there is hardly any empirical evidence of one model's superiority compared to other models. There are a lot of opinions and anecdotes but proof of advantages on one model over others in regards of quality, cost and duration are minimal. It is vital that the team or a company chooses the best suited method for a project, every method has its

drawbacks and advantages. Bhatia and Kumar (2014, p. 196) remark that traditional and modern models are suited to different projects and the project type affects the choice; whether the project is critical or not so critical or are the requirements dynamic or are they stationary.

Software development processes need to consider a varying number of requirements that must be included in the process, depending on the product that is developed or the commissioner for which the product is meant. Some requirements are such that all the projects need to consider them. Some requirements are so specific that only a product or two must take them into account. Including information security into the process model of software development helps to enhance and maintain the high quality of the product.

2.3.4 Traditional versus agile security principles and aspects

Baskerville, Kuivalainen and Siponen (2005, p. 6) note that agile methods typically lack precise software security features, several separate methods can be added like checklists and management standards. However, they add that only a few can be integrated effortlessly to the traditional software development methods, making implementation to agile software development even more arduous. Baskerville et al. (2005, p. 2) have defined agile development model's security features into four phases: requirements analysis, design, implementation and testing and they add that these phases are not sequential and each phase is optional (FIGURE 15).

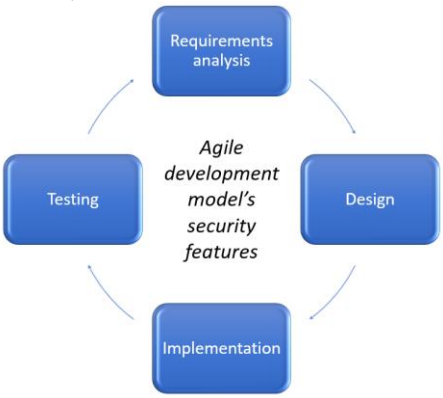

FIGURE 15 Agile development model's four security features

Beckers, Bruegge, Klepper, Lachberger and Moyon (2018, p. 31) agrees with Baskerville et al. and proclaims that security engineering is often planned for linear rather than iterative and incremental development that by their nature convolute risk analysis and assurance practices. The focus is often on functionality; security requirements and traceability are neglected, and dynamically changing processes hinder audits.

Vuori (2011, pp. 23–25) explains this further and states that agile development has renounced the traditional requirement specification and presentation and replaced them with user cases which have in turn been replaced with

user stories. These agile practices are insufficient to safety critical projects. User stories are non-systematic and subjective, and most safety features are objective demanding standard-defined design and implementation requirements. Thus, the process of requirements management cannot depend on agile culture and the traditional techniques should be used instead.

Vuori (2011, pp. 24–25) continues that utilizing threat and risk analysis for user behavior study can be applied to systemize descriptions and to elicit safety requirements, such a practice is typical for agile development. The non-commitment to concrete planning as well as evolving feature list during development complicates safety feature planning. If agile methods are to be used for this type of development some form of more detailed up-front design is needed. These details are utilized as a foundation for the safety argument building process (Abdelaziz, El-Tahir & Osman, 2015, p. 43).

2.3.5 Secure software development

As early as 1986 Rice and Tompkins (1986) have examined security in relation to development process. They have concluded that SDLC methodology provides a structure that aids in security safeguard planning, designing, developing, and testing with a manner consistent to sensitivity of information. McGraw (2006, p. 20) agrees and defines software security as the process of designing, building and testing, where it discovers and obliterates problems in the software itself. McGraw (2006, p. 5) continues and clarifies that good software security practices consider security from the first stages of the software lifecycle, know and comprehend the basic (including language- based) problems, design for security and expose all products to impartial risk analysis and testing.

Ajayi, Onashoga and Sodiya (2006, p. 638) write that to produce software its engineering process has to be combined with security engineering. This will require a detailed understanding about the software development process. Flechais and Sasse (2005, p. 15) dictate that secure software development process is basically a mandatory requirement if a company wants to provide secure products to the customer. They remark that security cannot be a patch that gets added on later by security experts. Security must be included into the development process. Security is frequently thought as a non-functional requirement that gets included into the process when functionality is addressed, but often not before this. Several other researcher concur and Howard (2004, p. 63) writes that security should never be thought at the end stages of the software development process rather it should be an integral part of the whole development. However, Koskinen (2020, p. 36) who wrote her thesis about implementing security into the core of DevOps- model. Based on a literature review the thesis concluded that the biggest challenge is still how to ensure software development pipeline security.

According to Howard (2004, pp. 63–64) there are several ways to ensure that development process considers all the best ways to include security throughout the development process. He suggests several means of achieving

this. Organization can create a team that the development team can consult and receive training on security matters and an external reviewer for the design and code security review is also invaluable. The executive level must be made to understand and accept the cost of security their advantages. The developer team must understand the threats and vulnerabilities that a process and coding without security produces, education is the key in raising awareness.

Howard continues that (2004, pp. 64–65) understanding the attackers and one's own product helps to mitigate the threats that the product is facing. Secure design practices must be part of the process including least privilege principle, simplicity and fail-closed defaults. He advises that process of coding should be secured, all code must be security reviewed not only by tools but by a human eye. Holding checkpoints and security-focused events aids in finding security vulnerabilities these events include several activities: responsible parties re-review the threat models and design documents to ensure that the most current threats are presented. Before the product is released it should be reviewed for security for the last time including but not limited to penetration testing, bug analysis and fuzz testing. Keeping-up with current development, updates and maintenance aids with the future and on-going product security. There are two review types: One occurs at a precise point during the process like prior to a phase completion and the other type is the final review.

Security can be implemented into separate phases of the process, different models implement security in distinct phases like requirements, design or coding phase or security can be a part of the whole development process. It all depends on the perspective and need the company has. According to a literature review made by Alshayeb, Mahmood, Mohammed and Niazi (2017, pp. 110–111) most researchers consider security practices to be most meaningful in the coding phase 41 %, 29 % in design and 19 % in the requirements phase, the whole development lifecycle is considered in 11 % of the 118 papers that were studied in that review.

Various researchers see the number of phases in Secure Software Development Lifecycle (SSDLC) differently, four views are presented next. Baskerville et al. (2005, p. 2) list four phases, Futcher and von Solms (2007, p. 43) list the five phases for SSDLC, Goertzel & Jarzombek (2006, p. 5) disagree and conclude that there are four phases and Higuera, Mohino and Montalvo (2019, p. 4) again list four phases that they consider relevant. These differences can be seen in TABLE 3.

TABLE 3 Various perspectives on SSDLC

Studies	Year	Phases	SSDLC				
Higuera et al.	2019	4 (+1)	Identification of requirements		Design	Implementation	Verification & Validitation
Futcher and von Solms	2007	5	Investigation (determine requirements)	Analysis (how to satisfy requirements)	Design (how to implement services)	Implementation (ident./impl. software security tool & components)	Maintenance
Goertzel & Jarzombek	2006	4	Requirements		Design & implementation	Reviewing, evaluation & testing	Distribution, deployment & support
Baskerville et al.	2005	4	Requirement analysis		Design	Implementation	Testing

Goertzel and Jarzombek (2006, p. 5) perceive the division as a way to enhance security in the development life cycle. The also state that threat modelling is a key factor of risk-driven software development and must occur in the initiation phases of the process. Futcher and von Solms (2007, p. 43) considers their divination from their SecSDM model's perspective which has five phases and divided between them 10 steps to ensure that security concerns are addressed during each phase. Higuera et al. (2019, pp. 3–4) indicate that especially agile principles are perceived to be fast paced because security concerns such as security impact analysis, verification and validation tests are disregarded during development. Thus, they conclude that verification should be included as one of the phases so that no matter what SSDLC model is chosen the security testing is accomplished during it. Baskerville et al. (2005, p. 2) concur but speak about testing instead of verification. This last division is used during the comparative research in fifth iteration, because it was the most suitable for the organizational context and included agile methods in its perspective of security.

2.3.6 Quality in software development

Alexander and Beus-Dukic (2009, p. 138) state that quality can be measured with the aid of standards that list the most important or vital "ilities" that a software product should have so it could be considered as of high-quality. A standard ISO/IEC 25010:2011 Software Product Quality and ISO/IEC/IEEE 29148:2011 Recommended Practice for Software Requirements Specifications are two of the available standards that could be used to measure and inspect the quality of the software product and the process leading to it (*ISO/IEC 25010:2011.*; *ISO/IEC/IEEE 29148-2011*). Siponen and Willison (2009, p. 1) note that if a company is using ISO27001, it guarantees a base-level for the information security environment. Every company needs to have a method for software development tailored to their needs, even though the company may use a generic model like an ISO standard-family as a basic model for operations.

These previously mentioned standards are meant for software system development and are suited only for that purpose. They provide software requirements specification (SRS) document forms to guide the quality measurements of the process and products. Alexander and Beus-Dukic (2009, pp. 138–139) write that qualities can be anything from usability, reliability and maintainability to security, flexibility and portability. The idea is to use a checklist to cover those requirements that are expected from the project. For every item on the checklist it should be asked what kind of requirements the project should have. Then a brainstorming or a workshop session should be held to identify relevant goals for the project, those goal are often "ilities" and then those goals and "ilities" are to be analyzed to create measurable and realistic requirements (Alexander & Beus-Dukic, 2009, p. 141).

Gupta (2014, pp. 145–147) defines quality management's purpose for software and its development processes. She declares that an effective system reduces IT risk by averting problems and spotting defects where they appear.

She mentions three activities which produce quality management. These are quality assurance, - control and – planning. She mentions that for software product the quality assurance has two section which are process- and product assurance. Product assurance ensures that the resulting product will meet its specification and this assurance is achieved via testing. Process assurance evaluated the process that was used to design the product. It is important to maintain a high-level of process quality because software must be used for a while before its maintainability can be measured. Although she notes that the change of the process does not always lead to an improved product quality.

2.3.7 Models for secure software development

Ruparelia (2010, p. 8) writes that a model describes what to do and methodology describes what and how to do it. SSDL models can be categorized under three wide categories: linear, iterative and a combination model of the precious two categories. Linear model is sequential meaning that one phase leads to another phase. An iterative model sees development as a constant process, where all phases are repeated multiple times. A combination of linear and iterative endeavors to end the repletion of the iterative model at some point.

Baskerville (1993, p. 411) tells that the initial security methods concentrated on checklists and simple risk analysis to support decision making. Those methods evolved a focus on mechanistic partitioning of intricacy in the coveted system. They entailed critical control checks which offered the barest acceptable protection for the comprehensive information system. Later the interest in development methods focused to abstract models. The key feature of this kind of an abstract model was to comprehend the information system's diverse security requirements.

Ruparelia (2010, p. 8) adds that software development lifecycle (SDLC) model considers all the phases of software from the initiation; requirements engineering phase, all the way through to maintenance. McGraw (2006, p. 34) adds that a company can create its own secure development lifecycle by implementing security touchpoints to the existing software development lifecycle.

McGraw (2006, p. 35) dictates that software security's main pillars are knowledge, software security touchpoints and risk management. He highlights the need for prescriptive, diagnostic and historical knowledge about software security, current research and best practices for a stable foundation of software security practices. If these pillars are applied gradually, in an evolutionary manner and equally the resulting software is cost-effective and secure.

Various secure software development models exist, and the most suitable models were chosen for further inspection during the literature review (annex 4). This forms the foundation for the comparative study, so the six models out of 41 that were chosen in co-operation with the commissioner are briefly represented here.

McGraw (2006, 83-84) has developed a Touchpoints model which examines an assortment of software security best practices which McGraw has de-

termined. He states that integration into existing software development is possible and this forms one of the center pillars of software security. Touchpoints are organized into a liner model but can be applied to any existing model and done iteratively as presented in the FIGURE 16.

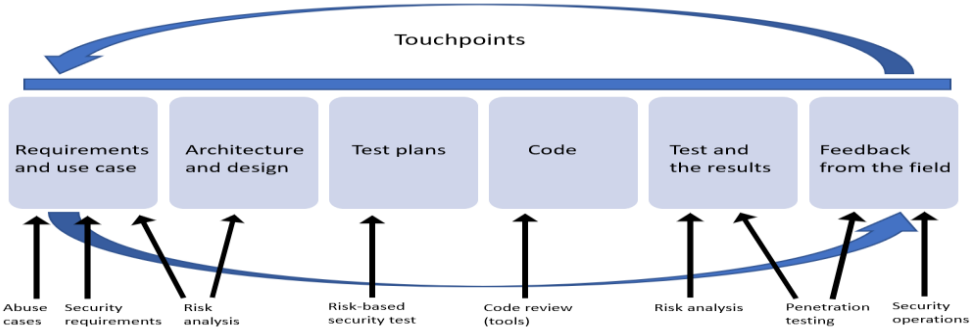

FIGURE 16 Touchpoints- model

McGraw (2017, p. 1) has also been a central influencer to the development of BSIMM model. BSIMM stands for the building security in maturity model and it is the result of multiyear study. Over 100 firms were included to compile the BSIMM version eight, which entails 113 real-world software security initiatives. (McGraw et al., 2017, p. 5). McGraw et al. (2017, p. 8) write that BSIMM is divided into four domains which are formed from 12 main activities (FIGURE 17).

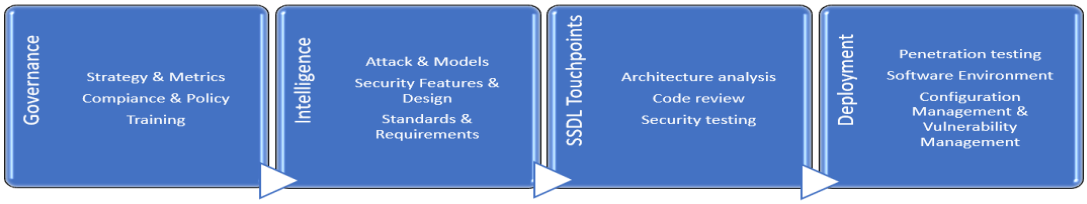

FIGURE 17 BSIMM- model

The next model focuses especially to the requirements engineering process. SQUARE means secure quality requirements engineering (Gedam & Meshram, 2019, p. 3) and it was developed by Mead, Padmanabhan, Raveendran and Viswanathan (2008) as a part of CERT program in 2008. Therefore, it is a secure software development model, which focuses purely on integrating security into requirements process. It is based on coordination between stakeholders and requirements engineers and it contains nine process steps (FIGURE 18) (Gedam & Meshram, 2019, pp. 1–3).

Agree definitions	• Structured interviews with a focus group
Identify assets and security goals	• Facilitated work sessions, surveys and interviews
Develop artifacts to support security requirements definitions	• Work sessions
Perform risk assessment	• Risk assessment method, threat and risk analysis
Select elicitation techniques	• Work sessions
Elicit security requirements	• Joint spplication development, interviews, surveys, model-based analysis, checklists, list of reusable requirement types and document reviews
Categorize requirements	• Work session using a standard set of categories
Prioritize requirements	• Prioritization methods such as analytical hierarchy process (AHP), Triage or Win-win
Inspect requirements	• Inspection method such as Fagan, peer reviews

FIGURE 18 SQUARE- model

SQUARE considers all the software's life-cycle development phases from the initial phases to the end-of-life. Thus, it is a security requirements engineering model and a model for SDLC improvements (Gedam & Meshram, 2019, p. 1).

Another maturity model besides BSIMM is a SAMM model. Shoemaker and Sigler (2014, p. 224) have described SAMM as a benchmark to evaluate the progress of its security assurance initiatives and create a scorecard. These scorecards provide a way to trace and demonstrate organization's improvements where an iterative software assurance integration process into existing policies and procedures is evaluated. SAMM can also be used as a map to aid in building or improving a security assurance initiative. SAMM has 12 security practices (TABLE 16) with three maturity levels and each level has a criterion that specifies the critical success factors to implement and assess to reach the desired level. Those levels have an assigned objective and it is a general statement of goals for achieving the desired level.

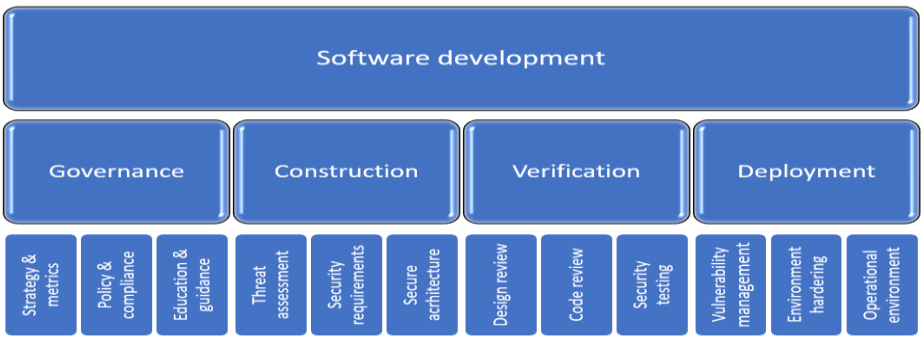

FIGURE 19 SAMM- model

SAFe is the scaled agile framework established by Dean Leffingwell and his collaborators. It combines agile and lean practices through four levels of organization including team, program, value stream and portfolio. Every level contains its own activities and is tied together. (Alqudah & Razali, 2016, p. 830). SAFe's activities are a mix of Scrum, Lean, DevOps, Kanban and XP (FIGURE 20 SAFe- model). It supports especially large enterprises confronting difficulties

in Agile practice adopting by offering a structure that eases the transition from traditional framework to agile. (Alqudah & Razali, 2016, p. 835).

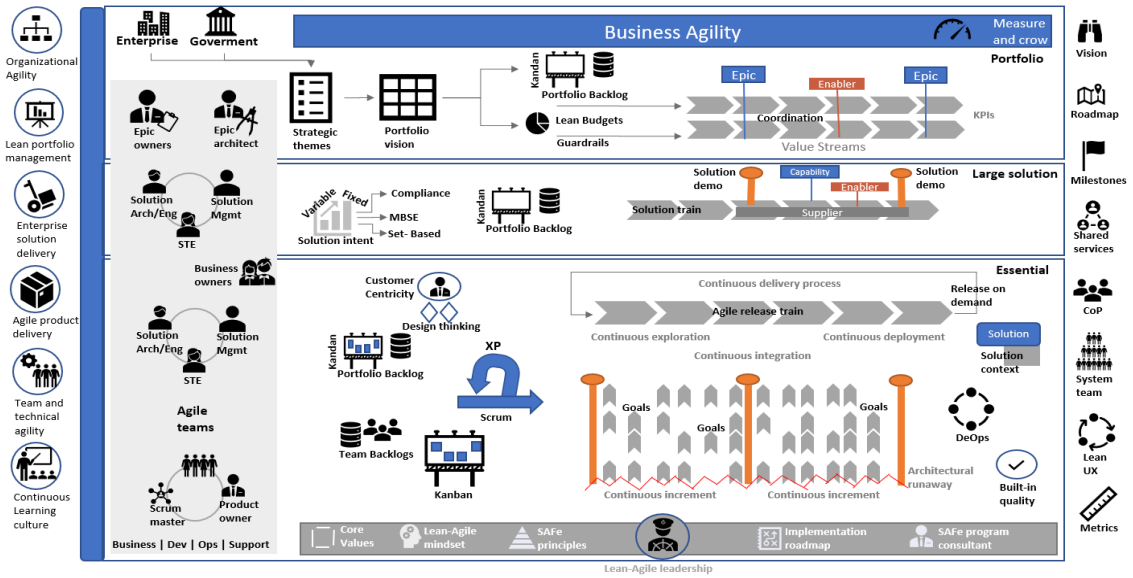

FIGURE 20 SAFe- model

Stage-Gate as it was originally called by its developer Cooper (1990) defined a framework which is applied to an existing development process. It aids the process and ensures that the new product proceeds without difficulty through idea phase to launch. It combines project management disciplines and those processes that are needed for new product realization. It is often implemented to aid in problems related to product performance, cost increases and time slippage during the development and is thus, a tool for risk reduction. Each stage has a product development element which is often a set of activities. Each gate stands for a review point for the preceding stage and as a decision point based on the conclusions of the previous phase's activities. (Broughton, Neailey & Phillips 1999).

In his conference publication Thamhain (2000) presents a Stage-gate based Phase-gate model that proceeds step-by-step through the five process phases (FIGURE 21). Each phase is outlined with principle scope, objectives, activities, deliverables and functional responsibilities. After this each phase ends up in a gate which defines the exact criteria and mandatory outcomes for success in the next phase and beyond. When accomplished and designed correctly gates validate with multifunctional reviews all success conditions.

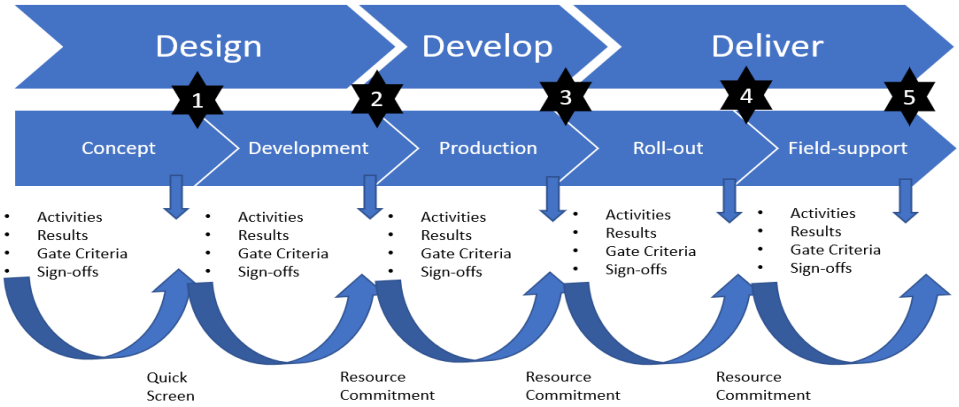

FIGURE 21 Phase-Gate process

3 RESEARCH METHODOLOGY

This chapter describes the combination of research problems, methods, and the gathering of empirical research material. First subchapter describes the aim and scope of the research, what is being examined, what are the research questions and material to which the research is founded on. Second subchapter focuses on methodological choices, how the material was gathered, analyzed, and refined.

3.1 Aim and scope of the research

In this thesis the commissioner wants to maintain a high-quality and expertise in software products. Mutual conversations led the parties to the conclusion that this aim would be best addressed through the further development of requirements engineering process from the information security perspective.

These conversations led to the main research question: ”What is the best model for software development in the commissioner, to implement information security requirements into the requirements engineering process, in order to produce more secure software?”. Davison, Martinsons and Ou (2012, p. 766) state that diagnosis, planning and assessment are crucial stages in action-change process and this process is connected to the main research theory. Davison et al. (2012, p. 767) dictate that after the prevalent situation diagnosis is finalized an intervention plan that addresses the problem or problems must be created.

This divided the research into two stages; 1) Current situation diagnosis, chapter 4.1 and 2) Action planning, chapter 4.1.6. The current situation diagnosis maps the prevalent situation of the requirements engineering process as well as concludes possible practices for secure software development to be able to represent intervention plan to the customer organization. Thus, the question for the diagnosis stage is: “What are the current problems of requirements engineering process of the commissioner and what practices in the field of secure software development, would best solve them?”. The question maps out an un-

derstanding about the current practices and those process phases that should be further developed to improve process quality. Baskerville (1993, p. 377) notes that by modelling the prevalent requirements engineering process as a component of the company specific product development framework it is possible to survey and evaluate numerous current tools to accumulate information security requirements and their design methods.

A systematic way to follow process steps creates process quality and that produces security. Therefore, a well-defined process including both requirements engineering and information security practices, results in a more secure software product. Siponen and Willison (2009, p. 3) add that the problems and implications of applying guidelines are to be examined. The key findings should be abstracted to identify proper improvement actions.

A document analysis provided the answer to the first sub-question of the diagnosis. This document analysis provided an understanding about the current model of requirement engineering process used in the commissioner organization and mapped the stakeholder groups participating in its main functions. The main stakeholder groups were later interviewed to gain an understanding of the current state of the requirements engineering process.

The aim of the second sub-question of diagnosis stage was to find out which practices and methods of the secure software development were widely used in the studied field and answer the second sub-question of the stage: "What are the widely used practices befitting the commissioner's needs for software development which implement information security into requirements engineering process?". Shortly, the goal was to identify the most suitable practices to implement them into a model of requirements engineering process. The second sub-question partly forms the scope for this section limiting the focus area to the secure software development's practices. These practices will be examined especially through the commissioner's business perspective.

The aim of an action planning stage was to combine the final model from the ideas emerging from the literature review, stakeholder needs gathered through the semi-structured interviews and the widely used practices on the field derived from the comparative study. This combination was adjusted according to the commissioner's needs. Thus, a created model is unique, and not directly applicable for other business contexts. The software development organization – as a customer of this work - together with an advisor from the commissioner and thesis steering group defined the goals that the end-result should fulfil.

Goals for the model:

a. The model must provide the means to define project-specific security requirements and concrete security measures to various stages of the software development process.
b. Model must be easily implemented to the commissioner's software development and its Gateway- process model.

c. The model must be founded on widely used software development practices on the field and it must support in a concrete manner the development of high-quality and secure product and service development and life-cycle management.

According to Davison et al. (2004, p. 73) diagnosis and action planning are the first and second stage of the Canonical Action Research (CAR)- model respectively (FIGURE 22). The rest three stages are not in scope, although some of their activities have been accomplished through evaluation sessions and research result representation to the commissioner during the writing process. Their feedback has influenced and developed the outcome of the intervention, evaluation and reflecting stages have thus been observed to some degree. With high probability the intervention action taking and change management will take months for the commissioner. Thus, the restricting time limit for this thesis confined thesis activities to the first two stages and other stages of Davison et al. (2004) model were not observed in their entirety.

3.2 Research methods

The research methodology applied in this study is action research, where the empirical part of it was structured though the semi-structured interviews. The selection of the research methodology was justified by the characteristics of the study, where the aim was to both create a new research information as well as solve a real-world problem in the commissioner's organizational environment. This solution will also initialize a change process for the commissioner in the future. According to Baskerville and Meyers (2004, p. 329) the action research does not simply investigate a phenomenon but seeks an organizational change. It develops and alters practices with pragmatic research and provides useful information about the objective of the investigation (Baskerville & Wood-Harper, 1998, p. 96).

In this research, the real-world problem was that the information security had not been properly implemented to the requirements engineering process for the commissioner. Therefore, the software's security level was unclear because it was not certain what requirements the software should fulfil. Davison et al. (2004, pp. 72–73) has written extensively about Canonical Action Research (CAR) in information system discipline. He has defined for it a five-stage cyclical process model (FIGURE 22). The process establishes the best rigor when it is conducted in sequential fashion and it often requires multiple rounds of iterations.

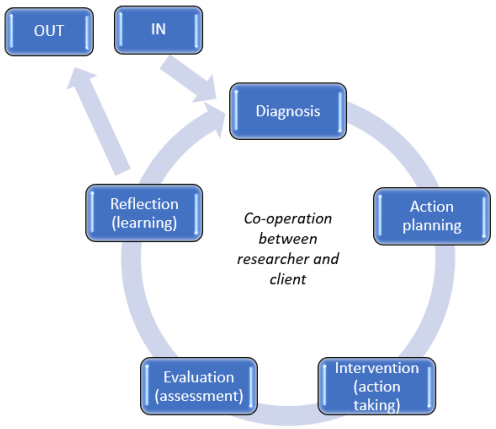

FIGURE 22 Canonical action research (CAR)- model

Davison et al. (2004, pp. 72–73) write that the first stage is diagnosis and it aims to comprehend the prevalent situation in the organization. The researcher has a duty to conduct their own diagnosis on the problems and their cause, but the client may also present deficiencies and problems. Intervention cannot be applied if the environment is not understood and the diagnosis will directly influence the planning of actions. Action plan reflects the underlying theory and these plans are implemented and evaluated after diagnosis stage.

Action research is separated from other research methods by its active involvement of the practitioners to study their own practices, it is an exception where research is not done on but rather with the practitioner. The researcher is inside the study, a part of the context that is under scrutiny not outside observing the situation. (McNiff, 2017, p. 10). The researchers of this thesis participated intensively to the commissioner's operations during the research process. Conversations with the stakeholders were had outside the interviews. The researchers familiarized themselves with different departments, their main functionalities, and the key personnel responsibilities. These conversations gave the researchers additional development ideas for the entire process.

Action research assumes that all the stakeholders affected by the change should be engaged with the investigation (Stringer, 2013, p. 15). It enables everyone to evaluate, investigate, improve and develop their practices and habits at work and thus it is a practical form of enquiry (McNiff, 2017, p. 9). This perspective will also give the interviewees a chance to comprehend the reasons behind their actions. An interview is thus not only a conversation where information is traded between the interviewee and the researcher but a development opportunity. (Nielsen, 2007, p. 219). The stakeholders were identified and included to the process and key personnel was interviewed for the prevalent situation comprehension. The key personnel provided a vital input to the requirements engineering process's change investigation.

3.2.1 Literature review

Literature review aimed to provide the answer for thesis's main research question. Which was divided into three main themes: a secure software development method, requirements engineering and information security. These themes were analyzed closely from the perspective of earlier research. First the material for literature review was collected and this functioned as the foundation for theoretical frame of reference. As Ellis and Levis (2006, p. 183) stated one major justification for a literature review is to examine what has been previously understood. According to Watson and Webster (2002, p. 13) it is essential to review prior and significant literature for every academic project. It generates a foundation for evolving knowledge and enables theory development. It also aids the researcher to discover among this prior research those gaps that would benefit from further research.

Earlier research sources were examined with search engines and appropriate library publications in university of Jyväskylä were studied. Specific searchers were concluded to digital databases such as Google scholar, ACM - digital library, IEEE Xplore, ScienceDirect, Springer, Taylor and Francis Online as well as ResearchGate. The electronic search was conducted with English search terms such as: " requirements engineering", "requirements engineering models", "requirements engineering process", "information security", "information security management", "principles of information security", "software development", "secure software development" and "SDLC". These terms were combined to different combinations and, with more detailed field specific terms such as "agile" or "agile development".

The aim was to gather research articles published in scientific conferences, publications of the industry and instructive material from authorities such as standards and glossaries to establish the foundation of the research material. Essential influencers of this study are Mattord and Whitman, Beatty and Wiegers and McGraw. Mattord and Whitman have studied extensively IS management and IS principles. Beatty and Wiegers are known influencers in the field of requirements engineering. McGraw influenced the secure software development section but is also used widely throughout this thesis. McGraw for is the creator of Touchpoints, which is one of the comparative study models.

The literature review was a part of thesis's mandatory structure, and its usage was preordained. Copious amounts of software development research material created difficulties for inclusion and exclusion choices. Forming a coherent synthesis from the material was challenging and the scope had to be narrowed based on the initial material search. In addition to the literature review, an empirical interview material was gathered. This was compared to the observations gained from the literature. According to Byrne, Keary and Lawton (2012, p. 239) a literature review forms the foundation for merging research findings of a subject matter into a cohesive unity and it indicates the current progress, constraints and potential trends for the research. The next subchapter presents

the semi-structured interview and its results and contains the described comparison.

3.2.2 Document analysis

A document analysis systematically reviews and evaluates documents. Like all analytical methods in qualitative research, document analysis entails examination of the data and interpretation of it to gather meaning and develop empirical knowledge. The analysis of the data involves discovering, selecting, appraising and synthesizing it into major themes and categories. (Bowen, 2009, pp. 27–28).

Through the document analysis existing documentation, files and archives for the commissioner were studied and mapped thoroughly. There were also multiple occasions of familiarization to the used documents, platforms, services, applications and current practices of the software development organization and the security aspects that had already been considered. This familiarization involved occasions where the interviewers had an opportunity to learn about the Company Specific Software Development Process (CSSDP) and ask clarifying question from company experts.

The information provided by the documents analysis was supported by the open, non-structured interviews. These were conducted with a product development manager of traditional product development as well as with a software development manager. This provided a managerial understanding about the CSSDP, its phases, inputs, and outputs, and gave the researches an estimation about its usage in commissioner's software development site. Identifying the stakeholder groups was done after the interviewers had familiarized themselves with the process. Process comprehension enabled interviewer participation to the selection process. The CSSDP has many participants from inner as well as outer stakeholder groups.

Lamsweerde (2009, p. 62) writes about stakeholder analysis. He states that effectiveness of their role for the system-to-be, domain expertise level, exposure to the alleged problems, impact in system acceptance and individual objectives as well as conflicts of interest need to be considered. He continues that the group most likely must be revised during the process because new relevant viewpoints are usually uncovered. During this research, a stakeholder analysis was done to reach an understanding about the problem that had to be solved. It was utilized in identifying the most critical groups for this project. The correct groups were chosen based on their roles, stakes, interests, and knowledge they could contribute.

The document analysis was a preparatory and necessary phase before the semi-structured interview process enabling its conclusion, so its extent was not as profound compared to the interviews. The familiarization based on the document analysis would have benefitted from more instructed and managed process on the commissioner's side. All the archives were not accessible to the researchers so some of the material was not accessed simply because its existence

was not known. If this research would be repeated the results of the document analysis would be quite different. There was not clearly defined or even drafted material for the process and the initial perusal was done with scatted document drafts and initial "to-do-lists". All these factors affected the reliability and validity of the document analysis.

3.2.3 Semi-structured interview

Subjective experiences can best be collected through an interview process. This method is very usable and provides a good technique for system level requirements extraction from stakeholders and stakeholder groups particularly in the case of usability requirements. (Laplante, 2017, p. 64). In this part system level meant the company's requirements engineering process from which this interview aimed to identify the problems for further process improvement. Process improvement recommendations were closely related to usability of the current requirements engineering process.

There were three choices for the interview type (structured, semi-structured and unstructured interview) and the choice resulted in a semi-structured interview. A semi-structured interview combines the best aspects of the structured and unstructured interviews and it is especially well-suited to a process-oriented organization. It also provides a carefully thought out list of the questions, but allows spontaneous questions to creep in during the interview (Laplante, 2017, p. 65). It suited the main research question and provided a way to have a structured form for the interview, but also left room for improvisation and additional questions that proved to be an asset.

It was possible to divide the interviews questions in accordance with the suitable role perspective. The interviews lasted anywhere from 30 minutes to 1 hour and 20 minutes in one sitting, the medium was 43 minutes. 13 participants were interviewed face to face and seven were done remotely via Skype. Interviews were conducted between 25.10.2019-5.12.2019, sixteen were interviewed once and five were interviewed twice. Nuances and subtle aspects of the responses can be lost if the interviews are done remotely like through a video conference (Laplante, 2017, p. 65). Face-to-face was the preferred method for the reactions of the people were more discernible as was noticed from the very beginning, due to scheduling difficulties some still had to be conducted remotely.

The timeframe was flexible which is why some were interviewed twice. They often had interesting viewpoints and additional information that they provided. Interview questions were done beforehand to produce a structured way to conduct the interview. Some questions included example answers, not to lead the answers but to provide a direction and reduce uncomfortableness. This subject was perceived as a difficult one because the process documentation and the supporting material was severely lacking.

Interviews were done individually, and all were recorded to have the opportunity to store all the information gained from the interview and not lose any due to slowness of writing or misunderstandings in the moment. Permis-

sion for the recording was asked and all the participants agreed to the request. Reliability was affected by the fact that the participants of the interview were from various departments and department levels. Thus, their perspective to the subject matter varied and they chose distinct terms typical to their own department which was not always unified throughout the company. Researchers had to make interpretations from the transcripts which affected the results.

3.2.4 Comparative study

The widely used secure software development practices that fit the commissioner's context had to be identified and compared. This comparative study was concluded to determine models and their feature suitability for the business context. This affected the choice for the method which is a qualitative comparative study.

Pickvance (2005, p. 2) writes that a principal rationale for a comparative analysis is the explanatory curiosity of achieving an improved grasp of the causal processes engaged in the creation of an event, feature or relationship. In comparative study differences between the cases are mapped and data is collected from two or more cases according to a shared framework. Pickvance also cites Tilly (1984, p. 82) who has defined four types of comparative analysis. This thesis utilizes the variation-finding comparison which tries to establish a principle of variation in the character or intensity of a phenomenon by examining systematic distinctions among instances. This comparative study aims to detect differences and similarities between the models. Because this is a qualitative study the focus was on multiple features which were compared between six models. They were given a certain criterion to fulfil and these criteria was adapted to the commissioner's goals:

a. Generic (software development model)
b. Traceability of information requirements
c. Adaptability to linear software development
d. Process accommodates iterations
e. Widely used in real-life
f. Founded on threat- and risk principles

The listed criteria are elaborated here. The first criterion is a generic software development model which implies that the model acts as the foundation for the whole software development. The second, information security requirement traceability entails that the commissioner wants to systematically trace information security requirement implementation into usage. This encompasses requirement status, owner, category, and risk- based prioritization and this combination aids in requirement implementation decisions. It is vital for traceability to justify the decisions accurately and document them comprehensively. It means that the decision making can be traced to its origin during the development. The after-launch changes for improved traceability must also be included.

Linear software development adaptability simply means that the represented practices can be implemented to linear software development. This process must enable iterations. In this context enabling process iterations means that the practices that are implemented to as part of the requirements engineering process will not disable the iterations between phases. Thus, agile practices are also supported.

Widely used entails the model recognizability and usability in large scale by the industry. These criteria ensure its easily accessible and there are enough experiences of its usage on expertise and developer levels. A widely used model is also better maintained and further developed.

Foundation on threat and risk modelling entails initiation of risk management. This idea is founded on the presumption that to protect critical software assets, their threats and risks must be identified, and their probability and effect evaluated. Through this the prioritization of security requirements and refinement implementation to software development can be achieved.

Comparative study included five iterations (TABLE 4) and the first iteration was already outlined during the literature review. This was done by listing frequently mentioned secure software development models from industry's research and literature. The exception being the "Phase-Gate". It is the original version of the commissioner's Gateway- model to which these practices are to be implemented and was added per the commissioner's requests. There were 41 models that emerged, and they are listed in their entirety to the annex 4. The initial listing was examined in co-operation with the commissioner during the second iteration and the most suitable models for third iteration were selected:

a. Phase-Gate (GateWay)
b. SAFe
c. BSIMM (by OWASP
d. SAMM (by OWASP)
e. Touchpoints (by McGraw)
f. Square

Third iteration included a comparison against the chosen criteria and the fourth iteration was done to exclude some unsuitable models. The fifth and final iteration compared the process with an existing research paper's results to assess perspectives with another set of criteria.

TABLE 4 Iterations in comparative study

Iteration	**Purpose**
1.	Discover and list initial SSDL- models found through literature review
2.	Select the most suitable models to third iteration with the commissioner
3.	Evaluate models to selection criteria
4.	Comparison between models
5.	Comparison of the model content between Higuera et al. and this thesis

It cannot be concluded to any degree of certainty that all suitable models for the comparison were found during the literature review. Time limits for this research did not permit a full investigation into every available model. Search

results were logically restricted for example if the model was specific, unknown or brand new then the probability of its exclusion was high despite the best efforts of the researchers. These restrictions affected the first iteration, its scope and steered its direction thus, the reliability of the research was negatively affected. The five iterations ensure that the issues have been considered from various perspectives and several times. Furthermore, it guarantees high-quality results and fulfils the research directives.

4 APPLYING ACTION RESEARCH AND EVALUATION OF RESULTS

This chapter consists the two phases referred as stages according to Davidson's study presented earlier. The first stage 4.1 is a current situation diagnosis, which consists of the document analysis-, interview- and comparative study material as part of action research. The document analysis is compact and functioned as the familiarization phase for the researchers. Semi-structured material is also introduced, and its material was processed through coding and categorization. The material from the interviews formed the empirical foundation of this research. Comparative study is based on the literature review where comparisons are done on the suitable models and their practices for the commissioner. The second stage 4.1.6 is action planning, which ties together all the results of previous mentioned material. These materials as well as literature review together are used to form the plan for intervention.

4.1 Current situation diagnosis

This section contains results of document analysis, semi-structured interview, and comparative study. It also provides analysis and conclusions related to them. The intent of this section is to generate a diagnosis of the prevalent situation equally from commissioner's requirement engineering process as well as secure software development practices used widely in the field of software development and befitting to the commissioner's needs. The diagnosis works as the input to the planning of the intervention stage.

4.1.1 Familiarization of the commissioner

Document analysis results established the foundation for the semi-structured interview that is why its results are the first ones presented. Its aim was to ascertain what kind of model is currently used in the requirements engineering

process. Additionally, it sorted the most critical stakeholder groups for that process to interview to find out the prevalent situation.

What kind of requirements engineering process is used by the commissioner?

Like told in introduction the commissioner utilizes a company specific process model for new product development. This model has been used in the traditional mechanical business side and it is transitioning to the software developments side. Meaning that the process is linear and has been created to accommodate the mechanical software development's needs. The model's phases are consecutive and there are checkpoints that must be completed after every phase these checkpoints provide the model its name: Gateway.

This model (annex 2 picture 1) has seven phases respectively; InnoStream (-1), Concept (1), Product specification (2), Planning (3), Product Design (4), Ramp-up (5) and Launching (6). Each phase has phase specific actions and practices that advance the practices defined for it, the new product development and produce the documentation needed for the product and its features during its lifecycle. Concisely the idea of this process is to refine a product from an idea to the market.

The InnoStream phase (-1) channels the ideas from inner and outer stakeholder groups into one forum where an initial business case is formed. This case is evaluated twice: it receives the initial evaluation and the second evaluation in a meeting where the potential business ideas are transferred to the concept phase with the product council's approval. The initial business case evaluation is founded on the estimated profits and the possibility to actualize the ideas.

The Concept phase (1) is where the potential business idea is analyzed again. This analysis is performed with the aid of QFD (Quality Function Deployment) – matrix where the correlation between customer requests and quality is inspected. The result is the understanding about the features that the product should have and what the customer wants. This understanding provides the foundation for the calculation of resource consumption for these most meaningful features. Essentially this phase evaluates the product's business profitability. Corporate model initializes "agile development" from this phase to fourth phase (1-4) (annex 2 FIGURE 30).

If product development is worthwhile the actual business case becomes a project suggestion. This suggestion includes the requirements book where the product's requirements are gathered. This document enables the creation of a concept from the business case. Concept examination helps the executing decision and how and what is the specific product that would fulfil customer's needs. When the concept has been created the project progression can be preplanned and its risks and resources inspected.

Product specification (2) is founded to the concept understanding and from it the product specification and launching are planned. When product understanding increases the project's risk and threat documentation can be clarified and updated. The product undergoes a failure mode and effect analysis (FMEA) which identifies the product's feature malfunctions and problems

caused by these. Analysis forms the base for concept description which can be updated, risks re-evaluated and the resources that are required can be clarified. Corporate level dictates that the alpha product should be accomplished between these phases 2-3 (annex 2 FIGURE 30).

After product specification a planning phase (3) is initiated. This phase is intended for the planning of the project like the name of the phase indicates. The progression requires plans like the project plan, initial manufacturing plan, testing plan and the update for the precious phase's launching plan. Additionally, the technical specification is drafted, and the first prototypes can be created based on this. Corporate level dictates that the beta product should be accomplished between these phases 3-4 (annex 2 FIGURE 30).

After the planning phase the product design (4) is initiated. The central idea is to produce the product plan and review and their documentation such as product drawings. While product understanding increases in the third and fourth phase, the FMEA analysis is updated on the malfunctions of the product features and the problems they cause. Corporate level dictates that the MVP should be accomplished between these phases 4-5 (annex 2 FIGURE 30).

Fifth phase leads form planning to execution, the phase is named as Ramp-up (5). This essential idea is to produce the first production patch and compare the execution to the requirements that have been set for it. The comparison acts as the foundation for the launching decision and prepares the product for market. If the product is market suitable is will move to Launching (6) and with this shift the responsibility of the product will move to software development to production.

What stakeholder groups participate in requirements engineering process?

To map out the prevalent situation of the requirements engineering process the most critical stakeholder groups were selected with the guidance of the project steering group. The most critical stakeholder group was identified as the product development from where the software development section was especially critical. In addition to software development business development and product management are linked to the development process. All these three groups were added to the list of critical stakeholder groups.

Requirements and stakeholder's needs expertise roles focus to sales, service center and law department, which is why all of these were added to the list of possible stakeholders. On the part on information security the expertise lay with the IT-department and security organization, which is why these were added. The final decision privileges were reserved for the project steering group.

4.1.2 Identifying the problems of requirements engineering process

This section will provide the results of the first part of the diagnosis – problems of requirements engineering process. The results will be provided question by

question in the order they were presented to the participants of the interview process.

In total 23 interview requests were sent, only one individual was unable to participate to the interview process. The most critical stakeholder groups that were interviewed were business development, software development, product management, legal, sales and service center (in the FIGURE 23).

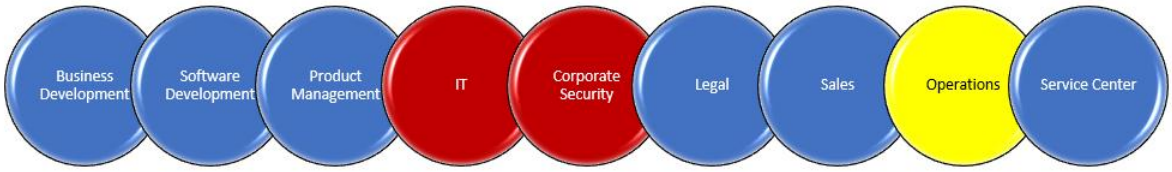

FIGURE 23 The most critical stakeholder groups

From every group the aim was to interview at least one individual, but preferably two or three. The distribution of the groups went according to annex 3 TABLE 12.

Three of the most critical groups that were not interviewed on the requirements engineering's prevalent situation were IT-department, security organization and the Operations-unit. At the time, the security organization did not have a specific person to point as suitable and knowledgeable for the interview. IT was as a more consultative than anything else and its role centered more to the information security side, this was a part of the second research instead of the first, which this interview- process was for. Operations-unit was not identified as a critical stakeholder group in the pre-study phase and its role was specified during the interviews. This resulted to the lack of an interview or interviews from this unit, and it was only later recognized as one of the most crucial stakeholder groups by the project steering group.

Two of the 22 interviews were shorter in duration, and most of the 18 questions were not asked at all from these two individuals. These two interviews were done differently because the aim was to clarify and gain a deeper understanding about certain aspects, practices and partners used during the requirements engineering process of the commissioner. These two were not as actively involved with the requirements engineering process thus it was decided to use them only as a reference and clarify certain aspects of the interviews with their responses. These responses also easily reveal the identity of the respondents so only the data was collected. This is reflected on the results, were the response rate is often 20 instead of 22.

There were 18 questions in total and most questions had specifications or clarifications that are not included into the heading of the question. Complete questions and their possible clarifying subparagraphs can be seen in annex 1.

Questions 3 and 4 as well as 7 and 8 from the form (see annex 1), have been combined so there are 16 subchapters instead of 18 (the number of questions) to this chapter. This merging was done because the question 3 defined the phase to which the interviewee participated on and after that knowledge was gained, the most essential phase to that individual's workload was mapped in

the question 4. These were asked separately but the question 4 can be perceived as a clarification to the question 3. This same reasoning holds true for the question 7 and 8, question 8 is a clarification to question 7, so these questions have also been merged in the answer section.

Could you provide your personal information?

Results relating this question are not included here, this was done because the information provided for this question was personally identifiable. This information included titles, roles, or more specific job descriptions and these have been erased to preserve individual privacy.

How do you see your role in the requirements engineering process?

Participants received a list of the most typical roles in the requirements engineering process to assist them in their responses and gain a usable material to analyze. These five role models were: subject matter expert, business process expert, software systems engineer, architect and hybrid role, the question in its entirety can be seen in the annex 1. These role models were chosen based on the role model distribution done by Laplante (2017, p. 18). Respondents saw their own role in the requirements engineering process as a hybrid role meaning a combination of some of the five roles presented previously, after this the options were elaborated on orally. Following this discussion most chose a more specific role shown in the FIGURE 24.

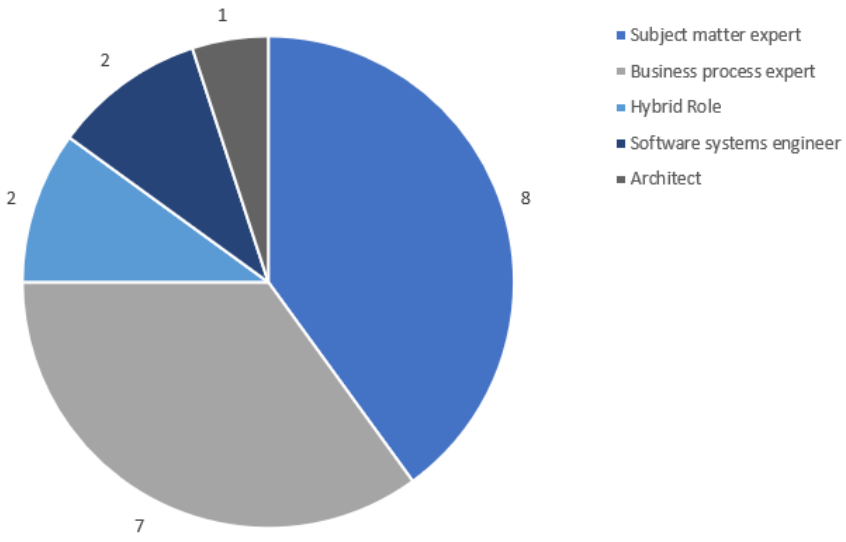

FIGURE 24 Interviewee's role in the requirements engineering process

As shown in the FIGURE 25 most (three out of four) felt that their role in the requirements engineering process in either a subject matter expert or a business process expert. This being so, the remaining quarter is formed from the roles of: software systems engineer, hybrid and architect. After the question most of the

participants indicated that in their experience the RE was not part of their main work duties.

Which phase/phases of the Company Specific Software Development Process (CSSDP) are you involved with? In what phase is your role most involved with the process?

Questions and 3 and 4 aim to map out the participant collaboration into the Gateway- process and their attitudes to the model usage in the software development process. The results for these questions are presented in unison in this subchapter.

Question 3: "Which phase/phases of the company specific software development process (CSSDP) are you involved with?" tried to direct the interviewee to think their own role in the CSSDP and cover all the areas of the process that the interviewee is participates in. Along with the question the respondents were shown a picture of the CSSDP (see annex 2), which is used to develop new products to aid their thinking.

The question was leading, and it was only intended to make the participant to consider his/her role in the software development process and thus the answers were not recorded. The intention was not to gain any data from this question, but the answers provided valuable input as to the attitude of the interviewee towards the CSSDP.

After the picture was shown most of the participants told that they did not consider the current model to be suitable to the software development's agile principles, because the model is fundamentally linear and thus too inflexible. They also mentioned that they had understood that currently there were multiple different methodologies used it software development. When they were asked, how had this come to be, interviewees adduced two things; diverse teams work differently and traditional mechanical product development versus software development process are quite different. Most also expressed their wish about more unified work practices between the teams.

Question 4; "In what phase is your role most involved with the process?", mapped the specific phases of the software development phases that the participants saw more vital than the other phases in their own perspective or to which they used more resources.

Along with the question the respondents were again shown a picture of the CSSDP (see annex 2), which is used to develop new products. Participants were asked to name a singular phase or limit their involvement to its most meaningful place in the process, this could mean two distinct process phases.

These mentions were scored in such a manner that every mention was counted individually and if the mention was made it received a point, most interviewees gave two or three mentions and thus the mention count was greater than one. The mentions counted to one phase do not then reflect the number of participants but rather the number of mentions. The mentions were divided according to FIGURE 25.

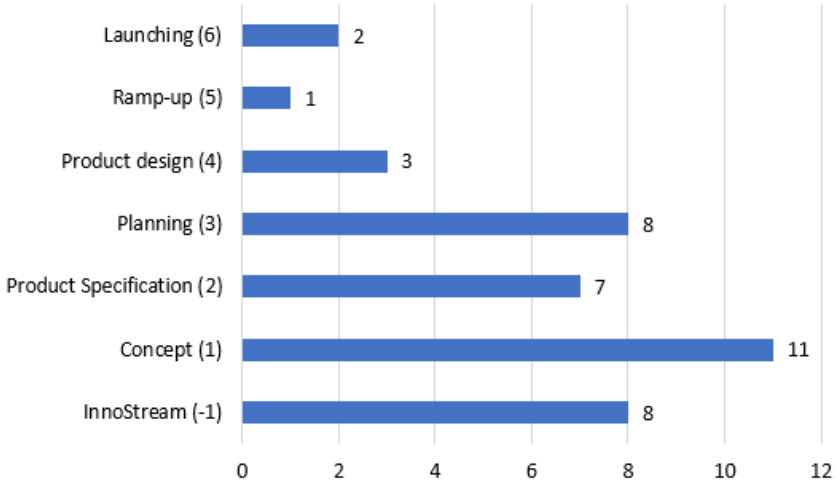

FIGURE 25 Process phases where the role of the interviewee is emphasized

There were 40 mentions all together. Most mentions were given to the concept phase, which gathered over one quarter of all given mentions. Additionally, respondents highlighted the planning, InnoStream and product specification phases, where many felt that their role was essential.

Is the company using a product mission statement (PMS) or any document that would provide that information?

Fifth question inspected the PMS document and who is responsible about its approval. This document (described in more detail in the chapter 2.1.3) in used to store the product description and the most essential functions. This document clearly states why the product has been developed and what is the need it answers to. Answers to the question: "…using PMS?" were divided according the annex 3 TABLE 13.

From twenty participants 19 answered this question. The most mentions, seven, were either "I can't answer, I don't know, I'm not familiar with this kind of a document". Two answered that they have not seen any documents of this kind, but they felt that it could be useful. Especially the sales – people felt that it would be a useful referral point when customers asked after a certain functionality or inquired about the products and service available.

Other answer can be divided into three categories, 1) respondents who believed that the commissioner used such a document but could not directly name such a document, 2) respondents who admitted that they did not believe that such documentation exists with the commissioner, but provided suggestions about other documents and 3) respondents who mentioned another documentation that in their opinion acts as product description document. Every group was represented by four answers.

The respondents were next asked who approves the product description document. This question was answered by eight participants, whose answers can be seen in the annex 3 TABLE 14.

This question was not asked of all the participants, because they had told that there was not such a document in use, to their knowledge. Question was provided only if the respondents believed that the company used some product description documentation or directly suggested another choice for the company's document. Participants did not have a unified perception about the person who approves such a document. All the interviewees, who answered to this question, gave a different role of responsibility as a result.

Why are requirements collected?

This question ensured that the process meaning in software development and in business perspective has been correctly understood. All 20 participants responded the question and provided 25 answers. Answers were grouped into the annex 3 TABLE 15, showing the mentions in every group, in the table one mention means a mention made by the respondent.

There were 25 responses in total and the most mentions were given to "…produce best product, software or service to fulfil customer needs", customer was overall seen as the most meaningful factor for elicitation and this was reflected in the answers overall. Mentions relating to business growth and customers collected 18 responses out of 25 in total. There was dispersion in the answers in the departments, some took the perspective of the customer, some business, and others a product point of view.

When and how are requirements collected?

The 7th and 8th question both cover requirements collection, so the results of both questions will be presented in this subchapter. 7th question: "when are requirements collected?", surveyed how long does the collection last and the question was specified with a clarification; "is the collection iterative or relating to a specific phase in the CSSDP?". This clarification was provided for the interviewees as a frame of reference to the answer they were to provide. All the 20 participants provided an answer and the distribution can be seen in the annex 3 TABLE 16.

Out of the 20 respondents nine respondents thought that requirements engineering is a continuous process and requirements should be collected throughout the product lifecycle. One participant specified that elicitation and re-review of the requirements must be done multiple times. Besides that, two interviewees mentioned that elicitation should be a continuous process. These 12 responses can be merged, and then every respondent represents continuous collection.

Eight of the responses are divided into various groups. Two participants responded that they did not know when the requirements are to be elicited. Additionally, one participant mentioned that requirement elicitation depends wholly on what process will be followed and what kind of process model is used. With this the respondent meant, that some projects follow a so-called hardware specific development process, and some do not. All four replies re-

flected the uncertainty on how to elicit the requirements in software development.

The remaining four responses represent the beginning stages of the requirements engineering process. Two replied that requirements are elicited immediately at the beginning of the process. The third respondent felt the same and he considered the questions from the hardware point-of-view and he felt that this process is specifically and inclusively meant for hardware development and requirements are thus elicited before phase 1. The fourth respondent notes that requirements are elicited when a development for a new product is begun. This means that all these four responses consider the initial phases of the process and consider the current model as a linear process.

The 8th question: ”how are requirements elicited (collected)?” examined the methods that were used during the elicitation. All 20 participants responded to this question and everyone gave two to three different mentions to this question. As previously told, all mentions were counted separately, so the final count of answers to this question was 51 meaning the number of distinct mentions. After initial grouping the answers were categorized to the table x (annex 3, table 1).

Provided responses were not process related but rather separate tools and methods to be used in elicitation. These initial grouping results of methods and tools were further grouped to annex 3 TABLE 18 into eight categories, to create a clearer picture of the used tools and methods. This table was created from these elicitation methods and tools that received three or more mentions. Other individual groups form one bigger category “other tools and methods”.

Most mentions were given to discussions that was mentioned six times, but the responses provided additional variations. In must be noted that in discussions the opposite side -from whom the requirements are elicited, like the stakeholder group change and the interviewees did not specify the opposite side who participated to the discussion.

Different forms of discussion that were provided were discussions with the development team, where the experience of the team was utilized, this has two mentions. Customer meetings (one mention) or regular customer meetings, where customer needs are discussed with the customer gained two mentions and where these needs can be written down on notes gained one mention. These different variations of discussion gained six mentions.

Another theme strongly associated with discussions was sales events. This was mentioned three times, but the method was not defined. It was left unclear was the method a discussion with the customer or are there structured forms that are filled or are there some other methods in use.

Third theme was sales sparring that was mentioned twice. In this case the discussion takes place inside the organization and between the software development and sales (more specifically export) employees these discussions aim to gather knowledge to product development from the customer needs. All together there were 17 discussion related mentions.

Interviews were mentioned as an elicitation method three times, but the respondents did not specify with whom the interviews were made with. Interviews can be utilized in market research as a data elicitation method. Market research was mentioned as requirements elicitations method four times. Market research can be executed with various kinds of questionnaires, questionnaires were mentioned three times.

Regularly field observation was mentioned in connection with interviews and questionnaires. Observation was mentioned five times, respondents also specified that end-customer was observed in their workplace and after that observation their needs are noted down. The end-customer ja end-user perspective was used by four participants. The fifth mention came from supplier observation and their need observation, which was documented for later use.

Suppliers and end-customers have needs that must be elicited and additionally work practice observation was mentioned as an area where requirements should be elicited from. Three additional mentions were made about this area and from this area competitor analysis was mentioned as an elicitation method. One participant mentioned a central pool on collected competitor analysis lists and this knowledge is then shared with the commissioner. These listings provide knowledge on how large companies have overseen, won or lost these competitive tendering processes. From this knowledge a further analysis on what lead to the competitor success or to the choice of their service instead of the commissioner's comparable product or service.

Participants told that requirements elicitation is not structured and because of this everyone does things in their own way. Three respondents expressed an opinion that there are no methods for requirements elicitation or if there is it has not been implemented to the whole organization.

These three respondents expressed a need for a structured process for requirements elicitation. In these conversations respondents also reasoned that a structured process would aid them in simultaneous elicitation between the teams and iron out the quality differences of the work and ease resourcing. Additionally, a structured process produced uniform documentation that was mentioned as a development point. Documents used for the elicitation were told to be different even in the team not to mention between separate teams. This was a source of frustration and confusion for the participants.

"Other tools and methods for elicitation" consist of 12 mentions from which singular tool like email, user story mapping and voice of the customer (VOC) -analysis. These have been listed more in detail in annex 3.

Overall, the participants told the experience helped with requirements elicitation and that it is done in co-operation with the development team. Elicitation can be aimed to the right place if the market understanding is good and comprehensive. While interview was ongoing with the system developer, he mentioned that he is not usually a part of the elicitation of the software requirements. He felt that he could benefit from participation, to gain an understanding of the bigger picture and context for the code and he saw a way to provide insight that could be useful for elicitation.

Requirements elicitation brought out responses of singular tools and methods, which were known to the participants from those projects they had participated on. This was also the explanation behind the fact that the respondent did not know how to describe company level practices and processes for requirements elicitation.

How are the "raw" requirements analyzed?

This question surveyed how requirements are first analyzed after initial elicitation of them. All requirements are not valid, so the objective of the analysis phase is to figure out the first group of applicable requirements for software development and its later iterations. Altogether, 19 participants answered the question producing 38 mentions about analysis related to collected raw requirements. This means, that each interviewee mentioned two possible ways or methods for requirements analysis.

The analyzing method was the same in this question as in all previous questions. Answers were grouped into the annex 3 TABLE 20, showing the mentions separately in the table and one respondent could make several mentions. Like with the previous questions the answers were put together to categories and mentions were counted, one mention is represented by one point.

The most points were given to the choices that produce the most business potential. Meaning that the biggest customers will receive what they want and need, this was mentioned four times. Four mentions were given to requirements prioritization in a workshop with the development team and this can be combined with a conversation with the development team, which received three mentions.

The responses underlined two analysis priorities: the monetary gain produced by the requirements in business potential and the prioritization of requirements through importance and executability. In its entirety the responses were divided to very few responses that defined specific analysis tools, that they had experienced useful when analyzing requirements, there were three from the whole group of 20. Most focused on their descriptions to what, how and when the right circumstances were to analyze requirements.

It should be noted that most of these mentions are based on conversation. This was essential when mapping and specifying the concept and researching the bigger picture the requirements create. This is not a structured or a precise analysis technique for the elicited requirements.

Centralized database aids concept correcting through customer need understanding. Previously collected information would also be more easily utilized and additional stakeholder group's needs better realized. Especially the sales organization experienced that there were plenty of customer needs that have been collected but their full potential is not realized.

How are requirements documented?

All 20 participants answered the question producing 59 mentions about the documentation, this meant nearly three distinct document suggestions to one interviewee. The method was the same in this question as in all previous questions.

Like with the preceding questions the answers were sorted to categories and mentions were counted, one mention is represented by one point. There were 26 groups, in which PowerPoint and Jira received six mentions each and Confluence had five mentions. Every other category received 3 or less mentions. TABLE 19 with groups and over all point score can be found in annex 3.

It is not necessary to present all the possible groups that were categorized, or list all given mentions, but rather place weight to group's total count of mentions which was 26. In practice this means that requirements documentation and storing is done with 26 diverse ways.

Every documenter has had the means to choose a form according to their fancy and craft the content as they desire. This has contributed to the situation where the data is fragmented and scattered to many different documents which do not have a consistent content and thus comparison of these documents is difficult.

There were 16 different document form mentions, this result confirms and underlines the conclusions of the previous question that there needs to be a systematic documentation and a structured process. The commissioner does not utilize a standardized model for requirements documentation, which has led to the state where the amount of used document form is varied. Every individual doing documenting can basically choose a form to which document the requirements. This leads to a situation where the information has spread to different documents, in non-uniform manner and comparison between the documents is nearly impossible.

Besides the document forms also the database count was large. Ten different databases were mentioned for requirements documentation storage or direct listing. Used databases varied from personal laptops to several different company databases. Just the sheer number of different databases leads to challenges, because the knowledge cannot be utilized effectively because it is spread virtually to multiple locations.

How do you utilize the requirements?

This question surveyed how requirements are utilized after elicitation, analysis, and documentation phases. The purpose of this question was to check how the organization of software development sees the role of requirements after they have been identified. What happens to requirements then? 19 participants answered the question producing 30 mentions about the requirement utilization.

The method was identical in this question as in all questions. Answers were put together into the annex 3 TABLE 21, showing the mentions individually in the table and one respondent could make several mentions. Like with the previous questions the answers were categorized and mentions were counted, one mention is represented by one point.

The initial presumption was that the most critical stakeholder groups for the software development would have described how these elicited requirements give an opportunity to create the software, the coding, testing and after these activities the eventual launch. Basically, to answer stakeholder needs.

Responses revealed two bigger group categories: business development and increasing sales and software development responses. These categories can be linked to interviewee roles and responsibilities.

Most mentions were given to the group that responded that the requirements are utilized to fulfil business potential and for building a portfolio for the future. This group gained three mentions, additionally requirements are utilized in sales cases, which received two mentions and one mention was given to aiding in strategic decision making. Requirements are utilized in software development planning like allocating resources and prioritizing what must be done and what is cost-effective, both received a mention.

Altogether 14 mentions in separate groups were directly involved with software development. The interviewees mentioned that based on the requirements, planning, designing, completing the program and testing for it can be done. For organization purposes based on the requirements sprints, tasks and the documentation they require can be completed. Additionally, interviewees have experienced that the requirements are needed for technical specification and for minimum viable product creation.

The interviewees provided direct development suggestions for the process. One mentioned that requirement documentation and writing does not have a specific tool and highlighted the need for more systematic documentation. Two interviewees told that forwarding ideas and requirements needs to the software development stakeholder groups is not clear. They specified that the problem is that they do not receive confirmation about the development phase, or has it even been begun on part of their idea. They wanted a confirmation when the idea has been implemented to software.

How is it supervised that the requirements get implemented?

This question surveyed how requirements implementation is supervised. The purpose of this question was to gain an understanding about methods or roles involved with requirements implementation. This question surveyed the responsibility for requirements implementation, monitoring and controlling process phases after requirements elicitation, analysis, and documentation. By asking “how” instead of “who”, the possibility was left to the respondents to name methods or tools used in implementation supervising.

In total 17 participants answered the question producing altogether 29 mentions about the implementation supervising. The method was the same in this question as in all previous questions. Answers were grouped into the annex 3 TABLE 22, showing the mentions separately in the table, where respondent could make several mentions. Like with the earlier questions the responses were categorized and mentions were calculated, one mention is represented by one point.

Most of the mentions related to requirements implementation supervising are associated with software testing. Software testing was mentioned 13 times and is composed of alfa and beta testing (3 mentions), usability testing (4 mentions), pen-testing (1 mention), in-house testing (1 mention; "we are testing the software ourselves") and automatized testing (4 mentions). Therefore, it can be generalized that the tool used in requirements implementation supervising is testing. It aids in finding out if requirements have been met as they should.

Another significant theme related to requirements implementation supervising, as mentioned earlier, are the roles and responsibilities related to it. 4 mentions were given to theme, which proposes that customer has a crucial role in supervision. These mentions outline the reality that requirements implementation is supervised by involving the customer with the process.

Other role centered mentions brought up the responsible person in-house. One interviewee suggests that the responsibility lies with the product manager, where another underlines that the responsibility belongs to product owner. The third role centered mention proposed that the responsibility for implementation supervision belongs to the software development team.

Two interviewees questioned the supervising process of requirements implementation and gave interviewers direct improvement ideas. The first one stated that the company lacks a systematic process and hopes to gain improvements via this thesis project. Another one noted that the process itself is neither scheduled nor included in someone's responsibilities. These both statements and the earlier uncertainty about the responsible person, indicates that there is no shared understanding between stakeholders about the roles and responsibilities related to requirements implementation supervision. Also, interviewees, who did not mention a person but preferred mentioning tools or ways for supervising, support this conclusion.

What are the different kinds of requirements you see the most?

This question examined how different types of requirements are ranked by the interviewed stakeholder groups, what they emphasize and when they are compared to each other. The aim of this question was to gain an understanding about the role and significance of information security requirements among the most crucial stakeholder groups in the company's software development.

In total 19 participants answered the question producing 62 mentions about the different types of requirements that they saw meaningful from their own point of view. The technique was the identical in this question as in all prior questions. Answers were grouped into the annex 3 TABLE 23, showing the mentions separately in the table and one respondent could make several mentions. Like with the preceding questions the answers were clustered to categories and mentions were counted, one mention is represented by one point.

Software development requirement types concentrated to three requirements groups primarily. These were user, functional and business requirements (money, time, and resources). User and functional requirements were both mentioned seven times each and business requirements gained five mentions.

It can be discerned from the responses that the people participating and doing the developing in the software process, co-operate, emphasize user friendliness and ease of usability. It is logical to consider the business potential of a certain product or a service. This means that the leadership and corporation have their own requirements for the development process, money, time and resources are evaluated and defined for the development project.

It should be noted that information security requirements were explicitly mentioned only three times. Additionally, respondents mentioned other information security related requirements such as protection policies, legal requirements, data protection and GDPR, the law, VAHTI-instructions and KATAKRI -recommendations.

In the part of requirements types especially the legal requirements were mentioned as a development target. Stakeholder's mentioned that they had no clear practices in place to indicate the correct actions to be taken if a law change occurred affected the company's software and systems. One respondent mentioned that he would not know the correct stakeholder groups to inform in-house or how to proceed with the change. Checklists were provided as a suggestion to solve the murkiness and unclear practices relating to legal requirements.

The legal department shared this view. It was their perception that the legal requirements defined by the law are currently considered in late stages of the development process. This means that privacy by design does not get actualized in the best possible manner. Legal requirements should be integrated as a part of the company's software development process from initiation phase forward to be considered from the same point on as other requirements. Integration would be easier to accomplish by fortifying legal's involvement to the software development process but simultaneously keeping the role consultative.

From what stakeholder groups are requirements collected from?

This question examined from what stakeholder groups requirements are collected this aimed to gain an understanding about the stakeholder groups that are the focus of the collection and are there deficiencies. All 20 participants answered the question, but one response was not included because it did not respond to the question. So, the 19 respondents produced 82 mentions that were divided into 36 groups. The method was the same in this question as in all previous questions. Answers were grouped into the annex 3 TABLE 24, showing the mentions separately in the table and one respondent could make several mentions. Like with the earlier questions the answers were categorized and mentions were counted, one mention is represented by one point.

Most mentions were given to the groups on various levels that form the client base of the commissioner. Various levels mean in this context customer's distributors and distributor's customers, basically the end-customers of the company. In places it was hard to comprehend the meaning of "customer" during an interview session, all the various levels for customer were not defined clearly and the term "customer" was used about both the retailer of the end-

customer and the end-customer itself. Combined all the mentions relating to ”customer” received 23 mentions, meaning over a quarter of the mentions for this question. Customer voice in its entirety is well considered in the company’s requirements engineering process.

The interviewees mentioned that customer stakeholder group and end user requirements are usually elicited through company’s retailers. This was seen as an effective collection method but at the same time the interviewees told that a secondhand knowledge did not entirely meet customer needs. There was a consensus about the opinion that the commissioner should focus more resources on stakeholder requirements elicitation and elicit them itself not operate through a third party.

Another development target for requirements elicitation was the commissioner groups of this action, the biggest and the best customers. It is a wasted opportunity that the failed, lost or transferred customers are not systematically interviewed. This would provide the commissioner more information about the development needs to further improve the business.

Another meaningful finding was that the inner stakeholder groups such as legal, marketing and operations were all mentioned only once. However, outer stakeholder groups such as the law and legal requirements were mentioned six times making it the second biggest group.

What are the models for requirements presentation?

This question surveyed how requirements are presented to stakeholder groups that participate to the requirements engineering process. The question was asked: “what forms or models are used to present stakeholder needs?”.

The purpose of this question was to examine how stakeholder requirements information is usually presented. This is meaningful, because the accuracy of the requirements information in the used forms or models, affects the need for extra clarifications and the workload of the programmers. The systematic way to gather this information and write it up, helps programmers to fulfil their duties more efficiently and therefore also have an influence on software security.

Altogether 20 participants answered the question producing 39 mentions in total about the requirements presentation. The system was constant in this question as in all preceding questions. Answers were grouped into the TABLE 25, showing the mentions separately in the table and one respondent could make several mentions. Like with the previous questions the answers were grouped to categories and mentions were counted, one mention is represented by one point.

From all the answers plain text got most mentions. It was mentioned as a requirements presentation form 11 times. After plain text, pictures got 6 mentions and verbal form 5 points. All the rest got only from 1 to 2 mentions, so these 3 groups are most common forms or models for requirements presentation of the commissioner.

As described earlier, commissioner is eliciting requirements mostly from its retailers. One matter of improvement related both to requirements elicitation and presentation with retailers, mentioned by the software development team. Retailers should be informed and guided to correct requirements presentation. This guidance should be developed in cooperation between operations and software development departments, to create a form which helps programmers to fully understand what the customer truly needs and wants. Currently the form and accuracy of the information is free, and therefore programmers must often perform extra clarifications with the customer, through the operations department.

One interviewee explained that he often gets requirements as a picture attached to an email and according to the picture, he should be able to program. This interviewee highlighted the significance of accurate requirement specification, which helps to understand how the software should be functioning and performing.

How is it confirmed that the market/user is understood correctly (from the business point of view)?

This question provided a better understanding for the market and customer understanding. The better these are understood the better different customer needs and reasons behind these needs can be comprehended. This leads to improved collection, higher quality requirements and better security.

18 interviewees responded to this question and provided 25 mentions, two did not provide a response. The technique was consistent in this question as in all preceding questions. Answers were grouped into the annex 3 TABLE 26, showing the mentions separately in the table and one respondent could make several mentions, these 25 mentions were divided into 21 categories. Like with the previous questions the answers were grouped to categories and mentions were counted, one mention is represented by one point.

This means that the mentions were divided to diverse ways to gain market and customer understanding and it is not possible to highlight a single or even a few main techniques or means.

How the system context is understood?

This question examined how system context is built, four sub-questions were presented; “System boundaries”, “Who are involved with system usage”, “How does the system converse with other systems” and “What is the (business and usage) environment like”. The aim of this question was to gain an understanding about how the system context is understood in its entirety among the most crucial stakeholder groups of the company’s software development.

All 20 participants answered the question, producing 24 mentions divided into 19 categories. The method was similar in this question as in all earlier questions. Answers were grouped into the annex 3 TABLE 27, showing the mentions separately in the table and one respondent could make several mentions. Like

with the previous questions the answers were grouped to categories and mentions were counted, one mention is represented by one point.

In this question, like with the previous one the mentions had a wide variance and it is not possible to highlight a single or even a few main techniques or means to ensure that the system context is correctly understood during the requirements engineering process. One anecdote from an interviewee; "there are as many ways as there are people doing it".

Compared to the previous question this question produced most mentions that belong to the "I don't know" – group. This response was directly dictated twice and additionally three participants mentioned that "not part of my job description", which is seen as a part of the group in this context. This made the "I don't know" – mentions the largest group with five mentions.

Who takes responsibility if the requirements engineering process fails?

The final question in addition to; "Do you have any questions about the interview or its subject" was to map out who is responsible if the process fails or who has the final responsibility for the requirements engineering.

All 20 participants answered the question, but two responses were not included because they did not respond to the question. So, the 18 respondents produced 23 mentions that were divided into 18 groups. The method was the same in this question as in all previous questions. Answers were grouped into the annex 3 TABLE 28 showing the mentions separately in the table and one respondent could make several mentions. Like with the prior questions the answers were sorted into categories and mentions were counted, one mention is represented by one point

Based on the variance the responsibility is not clear in the commissioner organization's requirements engineering process. Most mentioned first that they themselves were responsible if the process fails. After additional questions respondents categorically provided another person's name.

4.1.3 Analyzing the identified problems

This thesis explored the problems of the requirements engineering process by interviewing its most essential stakeholder groups. There were 22 interviewees and two of them were more specific and concentrated on few topic clarifications that emerged during the interview process. The results were analyzed with categorizing and coding. The analysis of the results provided three crucial themes: various requirements engineering models, methods and tools, roles and responsibilities as well as requirements management. This chapter is structured according to these themes and the progress follows their presented order.

Various models, methods, and tools

It came apparent during the interviews that the requirements engineering process is concluded with several various models and a unified model has not been implemented. Some teams operate through agile principles and others according to mechanical side's company specific development process. This process is thought to be unsuitable for software development because the model is fundamentally linear and thus thought too inflexible for agile software development purposes. It is often seen as a traditional product development tool including a lot of documentation. However, a combination model, utilizing both linear and agile methods, can be created. This was also mentioned by the interviewees as an improvement idea for systematic process usage. The resulting process model should be business specific and include only the least amount of documentation needed for tracing requirements through their life cycle.

A lack of a unified model means that teams have various operating practices during the process and thus produce divergent documentation. Various operating practices between the teams occur from agile practices and the developer's need to have the freedom to select the most appropriate and suitable tools for their work. However, from administrative perspective the requirements engineering process needs its own process model to evaluate, monitor, guide, and control activities.

In addition to various operating practices the teams utilize various tools for requirement elicitation, analysis, documentation, storage, and utilization, this was perceived to be a positive thing. It would still be beneficial for the commissioner to ensure that the necessary tools are available, the usage has been instructed and the teams utilize the available tools comprehensively when refining the requirements.

Requirements create the foundation for project planning, management, risk and change management and eventually to approval and trade-offs. Functional, user and business requirements were mentioned multiple times. Security requirements like information security were mentioned only thrice. It might be that they were mentioned so rarely because the roles are not defined concisely and the responsibility for software security is not assigned to anyone. The threats and risks are not on the table during software development but rather they are dealt with among other nonfunctional requirements. The commissioner should ponder how much it wants to invest to software's information security and what role is should have during the development process.

The commissioner should also ensure that the process produces at least the unified quality ensured by standardized tools. Deficiencies in tools emerged in the analysis phase because the interviewees could not name specific tools to use in this phase. Instead they described what, how and when should analysis be done in their opinion.

Requirements implementation supervision was said to be completed with various testing methods, so it is a logical deduction that testing is the current method for supervision. Most interviewees had a hesitant voice tone when mentioning or maybe suggesting testing methods. This led the researchers to

believe that they were not sure about their suggestions or they were unsure as to how and with what is the testing accomplished.

Like all processes the requirements engineering process relies to the previous phase and the results that it has produced. This means that the foundation for the process is laid during its initial phases and these either guarantee a success or ensure a failure. Development needs for these phases should be considered seriously because problems accumulate during the process.

Roles and responsibilities

Confusion relating to the roles and responsibilities came apparent during the interview questions. The interviewees could not name their own role in the process even though they represented critical roles in it. Like told in the results the interviewees thought that requirements engineering is not an essential work task they should be concerned about. This leads to the conclusion that the interviewees did not completely understand what the requirements engineering process in its entirety covers and through this their own meaning to the process.

Role and responsibility divide and the confusion relating to it continued during the product mission statement question and the responsibility for its approval, also they were not knowledgeable about the requirements implementation approval and does the product fulfil the requirements in the testing phase. Lastly, they did not have a specific person to name as the responsible person for requirements failure or as the proprietor of the requirements engineering process. All these express deficiencies in role and their responsibility assignment, where one critical deficiency is the lack of a process owner's definition. If the process owner is not defined, logically no one oversees the process. Then the process will not evolve with the business and its operational environment.

The confusion related to roles and responsibilities was reflected across the requirements engineering process. The practices and activities have not been assigned to those groups that participate to it and this makes the completion erratic. Which means that when the requirements engineering process model is generated, it is a good practice to identify the roles that ordinarily are associated with actions in the process.

Defining the roles presumes that the process stakeholders - inner as well as outer have been identified. This is one of the defining phases of the requirements engineering process. Stakeholder identification must be examined from their perspective and their desires must be mapped and expectations for the product managed. This does not affect only the outer stakeholders but also the inner groups.

Inner stakeholder input in software development processes needs to be improved upon. Various inner stakeholder groups, and their voice should be considered more thoroughly during the development process. These inner stakeholder groups include the law, which was a meaningful outer stakeholder representative. The legal department does not have a specific role or responsi-

bilities in current software development projects, and it is worth considering should it be capitalized more efficiently as an inner stakeholder group.

Also, software engineer's role and responsibilities should be clarified. According one of the interviewees, software engineers are not participating in requirements engineering process from its initial phases. This leads to situation, where software engineer has no comprehensive context understanding about the software to be developed.

Requirements management

The role of requirement management, as described more in detail in chapter 2, is to aid in information management, its capture, storage, and dissemination. It also permits requirement traceability, which was mentioned by one of the software engineers as an area for further improvement.

Like Beatty and Wiegers (2013, p. 13) state, requirements cannot be managed if they are not well documented. Documentation aims to maximize the benefits of the elicited information and through that the understanding of the software and at the same time ensures that if key personnel changes the data loss is minimized. Additionally, documentation aids maintenance and development choices can also be justified legally. This was also mentioned as an improvement point by the company's legal representative.

Demands evolve over time, and therefore requirements engineering process must be flexible adapting to changes. Requirements should be open for re-evaluation and review. This is accomplished by validating documented requirements during the occurring changes.

Before requirements can be elicited, stakeholders must have a mutual understanding about the software and its primary functions. This information is usually presented in product mission statement document, which according to the interviewees is lacking from the commissioner's software development. Because the product mission statement is not used, there cannot be a person that would approve such a document.

Requirements documentation, analysis and utilization all suffer from similar challenges. Requirements documentation method steers the process forward and ensures that requirements can be read, analyzed, rewritten, and validated. The commissioner utilizes multiple requirements documentation and storage solutions which weakens requirement utilization in software development processes as well as during development lifecycle.

Centralized requirements database was suggested as a solution for the scattered information. It might also help with the utilization of the elicited information to its full potential. Especially sales representatives mentioned a need for further investigation of the gathered requirements. They often base their forecasts of the market development and customer need changes on this material.

Interviewees did not mention any specific tools for requirements analysis, and they seemed confused about the subject of the question. Because there is no

structured method for the analysis phase, the analysis quality is diverse and same for the results. The quality depends on the individual who forms the analysis.

Requirements analysis should be a continuous process. After every change, the effect of this modification to the product and its most important assets must be recognized. This analysis provides an understanding on how the change shall affect the product, its safety and requirements can then be specified. Thus, making it possible to update risk and threat evaluations and their mitigating factors, from where the requirements have originally been created.

4.1.4 Concluding the prevalent situation of requirements engineering process

Interviews provided a good perspective to the commissioner's current situation and the development needs for the requirements engineering process as well as provided a part of the answer for the question; "What are the current problems of requirements engineering process for the commissioner and what practices in the field of secure software development, would best solve them?".

The commissioner did not have an applied model for requirements engineering, thus its phases could not be compared across various projects. Typical requirements engineering actions and practices were done but the lack of a defined process caused these actions serious deficiencies. During the research three crucial themes emerged: various requirements engineering models, methods and tools, roles, and responsibilities as well as requirements management. There same themes were present in the analysis chapter and they also formed the structure for this chapter. The four recommendations founded on these themes are represented here.

The first recommendation was that a model for requirements engineering should be implemented. This model should be a combination of linear and agile practices. The commissioner's corporation applies a linear model and its use is mandatory, but the commissioner wants to maintain a partly agile procedures in its practices.

A second recommendation was that there should be a standardized custom for method and tool usage, while leaving the developers a choice as to the most appropriate tool or method. This standardized custom would ensure a homogeneous quality foundation for the products, software and services generated throughout the process.

A third recommendation was that the stakeholder groups should be identified, and their responsibilities in the process must be both examined and defined. Additionally, stakeholder interests for the software should be mapped, to understand the basic framework of the requirements related to the development projects.

The fourth recommendation was that a central database should be established for the requirements to ensure their usability and traceability. This database should be used across projects and locations. The database could be an ex-

isting one, but it should be assigned as the official and mandatory destination for this information.

All these recommendations (TABLE 5) were important themes for further development of requirements engineering process. However, the main purpose of this thesis, was to produce a model for requirements engineering. This frames the focus area to the first recommendation and its further investigation. This means, that the researchers first identified the widely used practices for implementing information security into the software development process and secondly unified these identified practices to a combination model.

TABLE 5 Recommendations for requirements engineering

Recommendation	Issue	Content
1.	Model	Implement a RE- model
2.	Methods & tools	Customs for tools & methods usage
3.	Roles & responsibilities	Identify stakeholders and their roles
4.	Requirements management	Establish a central database

4.1.5 Comparing secure software development - practices

The literature review can be perceived as the first iteration of this study where 41 secure software models and frameworks were discovered and listed (see annex 4). Additionally, the commissioner requested that the CSSDP would be included into initial comparison to evaluate its success compared to other software development models. This listing was represented to the commissioner: the models, their central idea, features and emphasis and the most suitable models were selected to iteration three. This second iteration resulted into models in TABLE 6 and they are presented in more detail at the literature review section 2.3.7.

TABLE 6 Secure software development models presented in literature review

Name of the model	Content
Touchpoint	7 security software development practices, called touchpoints developed by McGraw.
BSIMM	12 secure software development practices gathered from 109 companies, divided into 4 domains.
SQUARE	Security quality requirements engineering model (S-RE) concluded from 9 phases considering the whole SDLC.
SAMM	12 secure software development practices. All have three different maturity levels and every level has its own criteria in order to fulfil level objectives.
SAFe	Framework used to ease the transition from traditional development methods to agile. It combines agile with lean practices.
Phase-Gate	Framework for existing software development process to improve its management and effectiveness. Reduces project related risks by improving performance.

The third iteration included an evaluation with the commissioner on the model criteria and its results can also be seen in previously mentioned section 3.2.1. The condensed selection criteria of desired features:

a. Enables documentation and its traceability
b. Enables practice implementation into a linear model
c. Enables iterations between phases
d. Enables risk- based security requirement prioritization
e. Enables risk- based decision making

Based on the third iteration it was concluded that none of the models fulfils the criterion by itself, as presented in TABLE 7 . Therefore, a combination model should be drafted. The third iteration excluded the Phase-gate and SAFe models. Phase-Gate model was excluded because it was already used by the commissioner, so its features and characteristics had nothing new to provide in regards of information security. SAFe did not adapt to linear software development foundation, which was a mandatory and critical requirement, so it was also excluded. So, the third iteration resulted into four models: BSIMM, SAMM, SQUARE and Touchpoints.

TABLE 7 Model comparison according to commissioner's business goals

Model	**Emphasis**	Generic software dev. model	Traceabilty of IS re-quirements	Adapts to linear soft-ware dev.	Enables process iterations	Commonly used	Based on threat& risk modelling
BSIMM	SDLC/ Practice		x	x	x	x	x
Phase-Gate	Model	x		x	x	x	
SAFe	Model	x			x	x	
SAMM (OWASP)	SDLC/ Practice		x	x	x	x	x
Square	RE-Model		x	x	x		x
Touch-points/ McGraw	SDLC/ Practice		x	x	x	x	x

The fourth iteration was the comparison of the models, previous iterations were done to exclude the unsuitable and undesirable models out of the comparison. The fact that the SQUARE model did not fulfil the evaluation criteria completely on the part of "commonly used" was in this case disregarded. The model was a more specific one than the others, focusing on requirements engineering and its quality. Thus, it was concluded that its value as a requirement engineering based model would exceed this one shortcoming and it was included to the next iteration.

This comparative study utilized an existing research paper from 2019 were Higuera et al. (2019, pp. 4–7) compared SAMM, BSIMM, SQUARE and Touchpoint models among others. They had made a comparative analysis of the SSDLC and evaluated the security actions which were offered for each phase.

They considered the four main phases of SSDLC: identification of requirements, design, implementation, and verification as well as validation. In their study all four phases were considered by all, but the SQUARE framework was the only one of the four that was not reported to be used in the software industry.

This research utilized four phases of SSDLC which are requirements (analysis), specification, implementation, and testing. This follows the categorization of Baskerville et al. (2005, p. 2). Their division is represented for the first time in the subchapter 2.3.4. However, in the research of Higuera et al. (2019, p. 4) they used verification instead of testing but the content is compatible to Baskerville et al. (2005, p. 2) categorization choice thus it is treated as such.

The fifth iteration included the comparison of research results by Higuera et al. (2019) and this thesis's views on the practical implications of the models in the commissioner's context. The benefit of each model for the company was represented in phases and can be viewed from the SSDLC- process as well as CSSDP perspective. The results have been gathered in to the FIGURE 26.

The target company's DP	CSSDP Phases					
	Pre-study	Requirements specification	Specification	Product & process design	Industrialization & market preparation	Launch
	SSDLC Phases					
Model	**Requriements**			**Design**	**Implementation**	**Testing**
SQUARE	Product definition, security goals, asset identification, risk assessment, security requirements creation, security requirements categorization & prioritization.			-	Documentation of decision-making process & rationale related to requirement implementation.	Documentation of decision-making process & rationale related to requirement testing.
BSIMM	Strategy, compliance, policy & standards related to requirements creation.			Attack models & definition of secure design though threat & risk modelling.	Security testing & architecture analysis based on the discovered threats & risks.	Security testing & vulnerability management based on discovered threats & risks.
SAMM	Strategy, compliance, policy & standards related to requirements creation. Security requirements & threat assessment.			Security architecture.	Vulnerability management based on discovered threats & risks.	Design review & security testing based on discovered threats & risks.
Touchpoints	Security requirements creation & risk analysis.			Risk analysis.	Risk analysis & risk based security testing.	Risk analysis & risk based security testing such as penetration testing guided by security requirements and abuse cases.

FIGURE 26 Comparison of models in the fifth iteration

SQUARE provided the most comprehensive practices to the requirements phase where they were founded on the understanding of the most vital features of the product. This also provided a way to identify the most important assets, prioritize threats and risks related to them and formulate the security requirements. Additionally, Touchpoints supported this view with its risk analysis-based practices.

BSIMM and SAMM had a more thorough inclusion of product related legal requirements, recommendations, and standards than SQUARE. These should be considered during the requirements identification phase. The output

of requirements phase into action planning stage was a product mission statement (PMS), security goals, asset identification, threat and risk- modelling, requirement elicitation where BSIMM and SAMM model brought the components of strategy, compliance, policy and standards as well as requirement prioritization and categorization.

BSIMM provided the most suitable practices for the design phase where threat and risk modelling for security design definition and establishment. These activities were commenced after initial confirmation of software's design and architecture. Secure architecture production was established with the practices from BSIMM while SAMM provides additional resources. The output of design phase into action planning stage was threat and risk modelling, which is used to clarify and confirm security design as well as security requirements. Creation of security architecture was supported with SAMM practices.

The practices in implementation phase were divided between various model perspectives. BSIMM and Touchpoints highlighted threat and risk-based security testing. SAMM emphasized vulnerability management through threat and risk identification and SQUARE emphasized the decision-making process of requirement implementation and the encompassing documentation of their rationale. The output of implementation phase into action planning stage was threat and risk modelling based security testing offered by BSIMM. Additionally, decision-making process and rationale (a security report), which were related to security requirements implementation phase and were considered as the output.

All the phases emphasized security testing and especially ideas from Touchpoint fit well to the commissioner's context. SQUARE included befitting practices that emphasized documentation and decision-making process and rationale (a security report), which are related to security requirements testing phase that was considered as the output. All the iteration outputs were gathered to annex 5. Outputs of the iteration five are shown in TABLE 8 and they acted as the inputs for the action plan.

TABLE 8 Outputs of the comparative study

Phase (SDLC)	**Output**
Requirements	PMS, security goals, asset identification, threat and risk modelling, requirements elicitation, prioritization and categorization
	Strategy, compliance, policy and standards, which also form requirements
Design	Threat and risk modelling, which clarifies and confirms security design and requirements
	Security architecture can be supported with SAMM practices
Implementation	Threat and risk modelling based security testing (round 1)
	Security report (report 1)
Testing	Threat and risk modelling based security testing of finalized software (round 2)
	Security report of finalized software (report 2)

4.1.6 Conclusions of the current situation diagnosis

Current situation was formed out of two parts: interviews and a comparison between practices. These two parts together provided the answer to the research question; "What are the current problems of requirements engineering process for the commissioner and what practices in the field of secure software development, would best solve them?".

The interviews revealed problems with the whole requirements engineering process and provided the first part of diagnosis of the prevalent situation. The prevalent situation was that there is no process model currently used in requirements engineering. This diagnosis is the foundation for the second stage of the action research, where the possible intervention for it is planned.

The second part of the diagnosis formed from the comparative study, which concluded the practices needed for the final model creation. These practices were inspected through four phases of SDLC and compared against each other, after which the most suitable ones refined elements required for the new requirements engineering model.

4.2 Action planning for implementation phase

Action planning stage considered the first recommendation of the semi-structured interview, structuring a model for requirement engineering. The framework of the model is CSSDP and its main added elements were formed through literature review, document analysis (subchapter 2.1.3), semi-structured interview and comparative study and also justified by them. These elements were presented in in more detail with justifications (annex 6) and a condensed version (TABLE 9) can be seen below.

TABLE 9 Elements for the final model

Elements	Literature	Document analysis	Semi-structured interview	Comparative study
1. Product mission statement (PMS)	x		x	x
2. Security classification	x			x
3. Requirements document	x	x	x	x
4. Technical design plan		x		x
5. Test plan		x		
6. Threat modelling & risk analysis -Security requirements -Privacy requirements	x	x		x
7. Beta security report	x	x		x
8. MVP security report	x	x		x
9. Requirement changes	x			x

The model which was drafted from CSSDP as well as from the elements presented above is named as the Threat and Risk Based Software Gateway (TRB-SGW) (FIGURE 27). Gateway- process (the foundation for this model) and its phases were marked with dark blue, thick arrows and two lowest rectangles in the picture. Process deliverables are light blue rectangles (TABLE 10) and two-sided arrows and relationships were marked with black arrows. Descriptions are light grey, and the symbol is a speech bubble. Decision points are sky blue diamonds. Swimmer lines divide the sections between the "diamonds" and these sections are called phases.

TABLE 10 Action planning phase outputs

Phase	**Output**
-1-0	Product mission statement
0-1	Requirements document draft
1-2	Software specification including the technical design and test plans
2-3	Beta security report
3-4	MVP security report
4-5	Release of a first version of the software
(5-	Update, revision, new release or a new feature)

The phases were examined one at a time and every phase is explained. The first is a pre-study extending from -1 to 0. The second phase was requirements definition from 0 to 1. Third phase was specification from 1-2 and fourth was product and process design from 2 to 3. Fifth was industrialization and market preparation from 3 to 4 and then the launch 4 to 5 and production finalized the phases, occurring after phase 5.

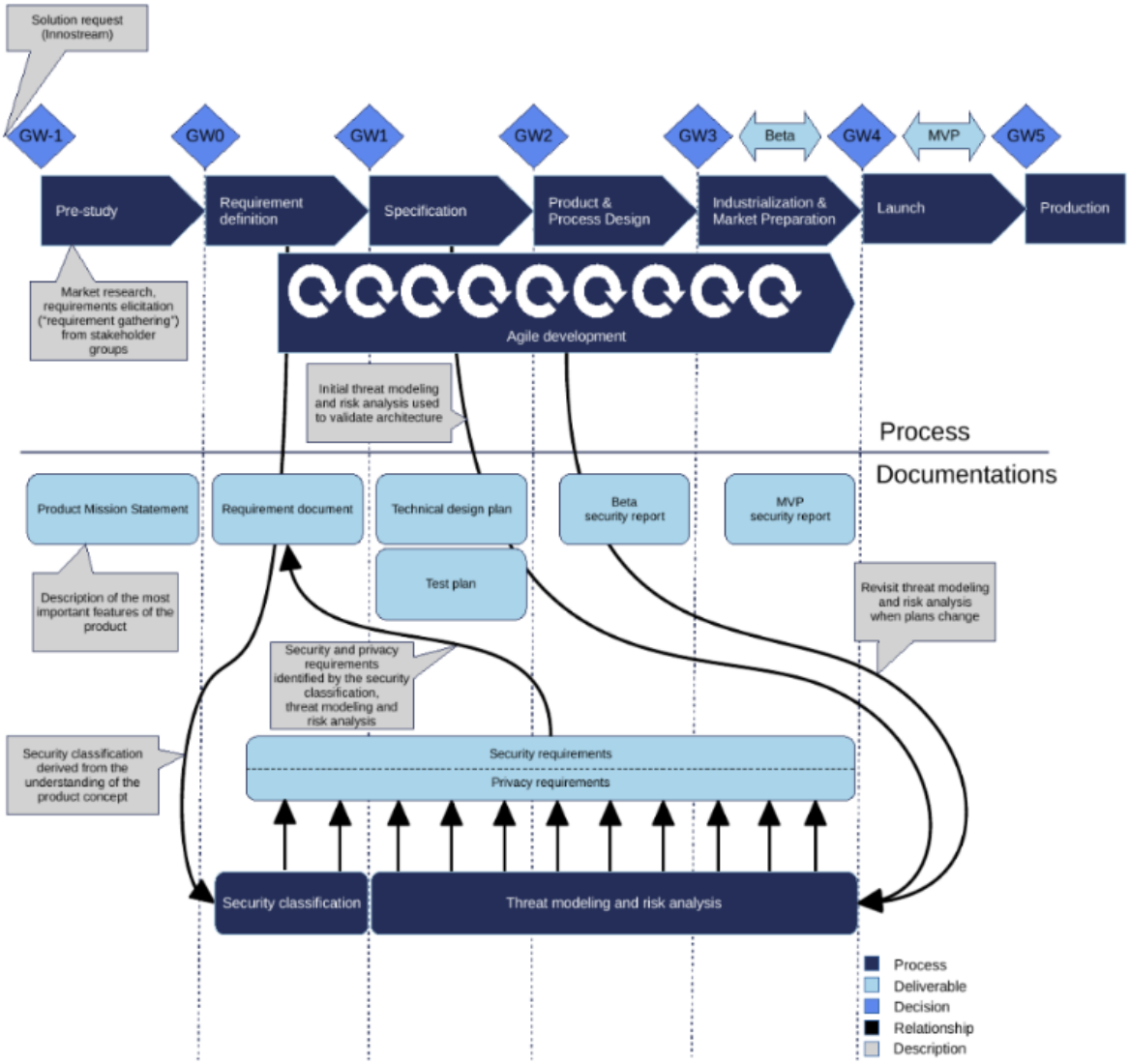

FIGURE 27 Model for RE with implemented information security

The focus of this thesis was between phases 0-2, in these phases the requirements engineering's role is highlighted. Other phases from -1-0 and 2-5- were described in more general terms.

Pre-study (-1-0)

Phase input is an idea from the InnoStream. The pre-study examines if the idea has business potential and this is scrutinized with a market study. Stakeholder analysis provides the correct stakeholders for a specific project. The stakeholder groups are identified, analyzed and their needs are mapped. In this phase the initial requirements are also elicited from the groups. The market study delivers an understanding about the initial requirements, shape of the concept and combining these as the output, a product mission statement.

Requirement definition (0-1)

This phase receives the product mission statement as an input and initialized agile development which continues all the way through the process to industrialization and market preparation phase (4). The requirements definition phase produces the requirement definitions and answers the question; "what kind of software should be produced?". To answer this question the security goals of the software should be defined, and its security level must have a classification. To produce this decision, the company should define what are the classes and the criteria for a security classification. The criteria should be specified, and its usage must be implemented to the organization. This will provide guidance to the employees and aid in the decision of the correct class.

A chosen security classification defines the scope for the security and privacy requirements. This scope includes these initial high-level requirements that can be documented and specified later during the process.

The initial requirements document includes the high-level requirements that have been received through the security classification. In the document all requirements must be prioritized, the class must be labelled, someone must be accountable for them and their status must be clearly marked. The documentation should be stored into a centralized database. A typical output from this phase is a requirements document draft.

Specification (1-2)

Phase input is the requirements document draft. In this phase the product mission statement and the requirements document draft form the input for the technical design including architectural design and test plan of the software.

After the software architecture is drafted the initial threat modelling and risk analysis can be produced. The meaning of this process phase is to identify the most critical assets regarding the developed software. The identification process results in threat recognition, after this the risk related to the threat should be evaluated. Based on the risk evaluation the risks can be prioritized and categorized to identify the most critical threat that must mitigated by the software development team. These countermeasures are used as security controls and these countermeasures can also be perceived as requirements.

This initial threat modelling and risk analysis produces more accurate security and privacy requirements. The re-evaluation of the threats and risks continues from this phase (1-2) through the process until phase 3-4. The requirements documentation is updated after every re-evaluation.

The most essential security and privacy requirements are divided into priority classes, the meaningful ones are chosen and then transferred to a requirements document. The development team combines their knowledge and chooses the most meaningful ones and these choices must be explained and documented. The documented requirements are forwarded to the development team as epics and initial beta and minimum viable product (MVP) content are decided. The typical output from this phase is the software specification including the technical design and test plans.

Product and process design (2-3)

Phase input is the software specification combined with security and privacy requirements and used to produce product and process design. This phase answers to the question; "how should the requirements be implemented?".

The software development team received the plans as an input and initiates a refinement of the plans into individual development tasks. These refinements and tasks are implemented in iterations during the phase. When beta content has been implemented and tested the typical output for this phase; a beta security report (a general comprehension about the security's state at the time) is produced and the beta version of the software is released. The security report includes the work that still needs to be done and risks related to this missing work.

Industrialization and market preparation (3-4)

Phase inputs are the beta version of the software and the security report. The development team continues working on the project tasks and the issues identified during beta testing. The beta software can be evaluated, and its programming adjusted accordingly. During this phase minimum viable product testing is done to ensure that the software responds to the minimum viable product's criteria. When MVP content has been implemented and tested the typical output for this phase; an MVP security report is produced, and the MVP software is released. The aim of MVP security report is to describe the software's security requirements and the decisions made during the project.

Launch (4-5)

Phase inputs are the MVP version of the software and the MVP security report, during this phase the software is made available to the market. The typical output from this phase is the release of a first version of the software, which is maintained and further developed after launch according to market needs.

Production and further development (5-)

The software is reviewed if further development is required and the documentation is updated accordingly the update and change revisions are represented in FIGURE 28. The phase input is a feature request that has not already been accounted for in the previous software release. It must be stored into InnoStream where all the requests are commonly deposited. The request must be evaluated and analyzed according to its business potential, what are the costs of its production and the impact of the change. When impact changes original plans all relevant documentation needs to be updated otherwise development of the feature may begin. In case of a minor change the process jumps from -1 to phase 2 and if the change is to security and privacy requirements a more thorough analysis must be performed similarly to the new software de-

velopment. Based on the changed version and change's security report the release decision is made directly on phase 4.

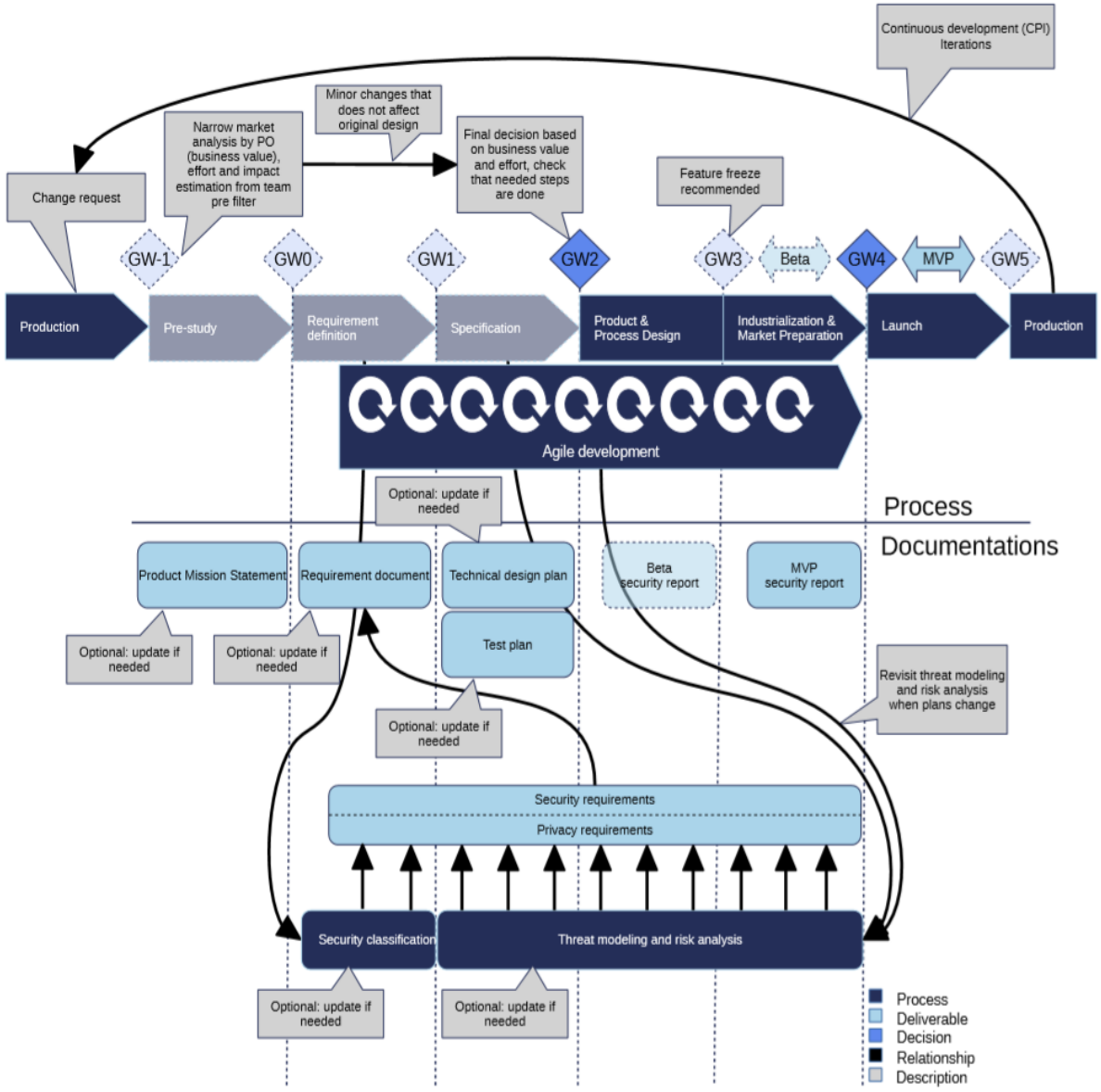

FIGURE 28 Change iterations

This model is a combination of the best methods and practices of the software development field. The foundation of the process is linear and is founded onto Gateway- process model. A linear model enables better management of the process and follows the corporate strategy. The linear model is also used to ensure that the process results are as even in quality as possible while maintaining a good security level.

The operative software development happens between phases 0-4 and the aim is to keep it as agile as possible which is why software development will be fundamentally agile with the commissioner. Programming and work in the process utilize agile development practices which has been done to lighten the too linear aspects of the development process. The central idea of it arises from the process of threat modelling and risk analysis. The purpose of that process is

to investigate all the factors affecting software security and consider their impact if the threats related to those factors will be realized.

The threat modelling and risk analysis process should be implemented by the commissioner. As a recommendation, this process should cover at least the following steps: 1) security classification, 2) asset identification, 3) threat identification 4) risk assessment, 5) countermeasure formulation and 6) security requirements creation. Security classification step includes both security categories and their selection criteria. This step is used to lighten security goals of the software development, meaning confidentiality, integrity and availability evaluation related to the software. Commissioner should develop security classification classes and the criteria for their selection.

Asset identification step is used to recognize all the critical assets of the software, that the attacker could exploit. After the asset identification threat modelling (synonym to threat identification) should be implemented. This modelling should go through all the assets, examine and name all the threats related to them, through the CIA perspective and execute their risk assessment. Risk assessment aids in threat prioritizing and according to these, the countermeasures can be formulated.

Countermeasures of the threats represent the idea of the needed security requirements. Therefore, the last step of the threat modelling and risk analysis is security requirements creation, done according to the formulated countermeasures. Thus, it can be stated that the agile development is threat and risk based, which is still academically understudied area.

Davison et al. (2012, p. 767) write that the research methodology choice of an action research mandates that action planning stage must explain and justify the solution and how exactly is the identified problem solved. During the diagnosis stage and its first part the interviews revealed several problems in the requirements engineering process. The most essential problem being that a structured model was not used. This model solves this problem by providing a structured model for requirements engineering (FIGURE 27).

The second part of the diagnosed problem was the uncertainty towards the most suitable practices of information security and how to implement into requirements engineering. This problem was solved during the comparative study where the most suitable practices were identified. These practices formed the elements of the resulting model. The paramount problem of how to implement information security into requirements engineering process was solved simultaneously by combining the results of diagnosis parts.

5 DISCUSSIONS

Methodological choice of an action research led to two main goals: one goal was academical and the second goal was to provide concrete benefits to the commissioner. The first one required that this thesis would fill the framework of an academic research and the second one to solve an existing organizational problem. Thus, literature review provided an understanding about software development context and document analysis about the business context.

The empirical material was gathered through semi-structured interview, which together with comparative study, formed the expeditionary framework for the current situation diagnosis. The interview material was analyzed with categorizing and coding. Analysis was accomplished by comparing interview material with the observations of theoretical framework. The comparative study, in turn, was completed through five iterations. Every iteration, to the fourth one, was used to exclude models, after which the main comparison was accomplished in the fifth iteration. The fifth iteration benefitted from a recently published research paper, containing a similar set of models but with a different incidence angle. This research paper was written by Higuera et al. and published in 2019.

All four methods were used to accomplish the diagnosis of current situation. They both revealed the problems related to requirements engineering process in commissioner's business context as well as afforded the suitable practices for implementing information security into it. All the results were merged in the action planning stage, where it engendered the answer to the main research question: "What is the best model for secure software development for the commissioner, to implement information security requirements into the requirements engineering process, in order to produce more secure software?".

The action planning stage resulted in the TRD-SGW- model. This model is not only theoretical, but a combination of its stakeholder's needs. The principal idea of the researcher's has been to find a solution, which will be easily adopted by its users. At the same time, it would increase their shared understanding of secure software development and its fundamentals. It is warmly recommended to involve stakeholders to implementation and testing of this model as well as

its continuous development. This model already has multiple earlier versions, which all have been results of regular stakeholder reviews.

The resulting model and its applicability to its purpose can be authenticated only after completing the evaluation and reflecting stages. However, the model can be evaluated according to a criterion created for models of secure software development. Lynn Ann Futcher (2007, p. 43) combined such a criterion, which contains seven points (TABLE 11, adapted from (Futcher, 2007, p. 43)). It is used to measure how satisfyingly the developed model considers the most critical standards and practices on the field. This criterion was generated under the supervision of professor Rossouw von Solms, who is also one of the central influences of this work. Even though technology has evolved, and the threat environment has changed, the secure development's fundamental idea remains the same: to secure software's critical assets from those threat and risks directed at them. For this reason, the criterion is still relevant.

The TRD-SGW- model considers five points out of seven. It integrates information security into SDLC, is threat and risk driven, ensures security requirements elicitation, suggests security controls according the risk assessment results and considers change. The remaining two points should be considered through, before the evaluation of the presented model.

TABLE 11 The criteria for secure software development

No.	Description	Original source	Fulfilled
1.	Ensure that developers are trained in how to develop secure software	NIST SP 800-14, Microsoft TechNet Report	
2.	IS must be integrated into the SDLC. It is essential that security be a well-thought-out process from system inception and design through implementation and deployment, covering all the stages	Jones and Rastogi, ISO/IEC 17799, BS 7799, NIST SP 800-14, RUP, Microsoft TechNet Report	x
3.	Some form of risk analysis, risk assessment and threat modelling must be performed during the initial phase of the SDLC	Howard and LeBlanc, 2003; BS 7799, ISO/IEC 17799, NIST SP 800-14, ISO/IEC TR 133353	x
4.	Security requirements must be identified early in the SDLC	ISO/IEC 17799, BS 7799, NIST SP 800-14, RUP	x
5.	Relevant security services must be assigned	ISO 7498-2, X.800, X.805	
6.	Design appropriate security controls and mechanisms into application systems to meet the security requirements. These IS controls and mechanisms should be selected because of some risk- based approach	ISO 7498-2, ISO/IEC 17799, BS 7799, NIST SP 800-14	x
7.	Ensure that any system changes do not compromise the security of the application	ISO/IEC 17799, BS 7799, RUP	x

As a typical characteristic to an action research, it considers complex organizational situations, involving many actors, subproblems, and subprocesses. All these variables represented a particularly acute problem for this action research. The need was not only to identify and describe the organizational situation, but also to co-operate with different stakeholders to improve the situation. On the other hand, this same characteristic made this research information rich, offering abundant empirical backdrop. Other weakness of an action research is that conversations and social interactions can never be repeated with comparable

results. Thus, this kind of research setting can be challenging to repeat which negatively influences reliability of the results.

The reliability weakness related to conversations and social interactions is shared between action research and semi-structured interview. However, the semi-structured interview as a research method was a perfect match for the goals of this research because the goal was not to form generalizable but specific information for a specific purpose.

The foundation for the comparative study was formed during the literature review, by listing secure software development models befitting to the business frame. That frame was refined from the set of commissioner's goals during the initial phases of this project. Reliability of this research may have been affected by this approach, where the accessibility of the information played a vital role. If the model candidate was for example relatively new or covered only needs of a specific business field, it was excluded already during this first iteration. This means, that some of the potential solutions may have been ignored already in this "data gathering phase". On the other hand, the accomplished comparison included altogether five iterations, where each one of them refined the information and re-examined it from a new perspective. This factor increases the reliability of the end-results and offers the research itself extra value, even as a singular part.

Both essential parts of the current situation diagnosis resulted in decisive research results but also important experience. The semi-structured interview showed how much silent information and great ideas there are hiding among the stakeholders. After this experience, it is not exaggerated to underline the importance of gathering stakeholder's needs as well as feedback and ideas regularly. Comparative study, in turn, highlighted the understanding about the threat and risk-based ideology of all security related actions. Every defensive action should be built on the understanding of the objective and its significance for its assessor. Only through this understanding the potential threats and risks related to asset may be mitigated efficiently. Therefore, security is never off-the-self solution.

Lastly, the organization is as strong as its weakest link. By this statement we want to remind that even if the threats and risks of an asset are identified and mitigated, the significance of a security training and culture cannot be undervalued. Like already mentioned; "You can't calculate the probability that a system is secure based on the risks it handles, if it's certain that insecure humans will form a part of it.".

6 CONCLUSIONS

The aim of this thesis was to produce a model for the commissioner to implement information security to the company's requirements engineering process used in software development. The material was collected as a part of two-stage action research, where the first stage was current situation diagnosis and the second action planning. The research included four research methods: literature review, document analysis, semi-structured interviews, and comparative study.

Software products perform an increasingly critical role in information society. Like Barabas et al. (2019, p. 1) wrote software products perform everyday tasks and ensure that the most critical applications operate uninterrupted. This means that security has become one of the most essential aspects of reliable software product development. Secure software is forged during requirements engineering process, which elicits stakeholder needs to solve customer's problems. In this process, the role of information security requirements is emphasized.

The aim of this thesis was to produce a model for the commissioner to implement information security into the company's requirements engineering process. The represented model, solving this problem, is a TRD-SGW- model, which will create a foundation for secure software development and will later be implemented to the commissioner's use. The model is a threat and risk driven, focusing on requirements engineering perspective and characteristically to requirements engineering, inner and outer stakeholder needs are highlighted. However, the implementation of this model will not alone ensure secure software development, but it will help the organization to organize and perform security driven development activities throughout the SDLC.

The research was founded on action research characteristics and included two stages: diagnosis of the current situation and action planning, respectively. The first stage provided an answer to the question; "What are the current problems of requirements engineering process for the commissioner and what practices in the field of secure software development, would best solve them?". The answer was provided through two parts; interviews about the process problems and a comparison between practices used in the field.

One of the weak points of an action research and an interview is that conversations and social interactions could never be repeated with comparable results. Thus, these kinds of research settings can be challenging to repeat which negatively influences reliability of the results. However, the chosen research methodology was a suitable match for the goals of this research because the goal was not to form generalizable information but specific information for a specific purpose. The interviews revealed several problems with the whole requirements engineering process, the main being that there is no structured model utilized.

Researcher interpretations and inexperience on software development might have affected the end results and the material that was chosen for comparative study. However, researchers acted in co-operation with the commissioner and employee experts. Therefore, their input acted as a kind of vetting process for this thesis's choices. The comparative study was an only methodological option to fulfil the need to find the most suitable practices. It resulted in the practices needed for the creation of a final model. These practices were inspected through four phases of SDLC and compared against each other, after which the most suitable elements needed for the new requirements engineering model were refined.

These elements, together with existing CSSDP formed the TRD-SGW - model. The model is generic in nature for this one company, which enables its usage widely in their context. However, the aim has been to find a balance between a too universal model and a too meticulous one. TRD-SGW will be used project specifically, which is why it endeavors to be straightforward and user-friendly.

The corporation level requires a linear model usage from the commissioner. However, it leaves a freedom to the developers to choose their preferred work practices. The reality is that agile practices are used and preferred in most software development companies and projects, thus TRD-SGW- model includes an agile practice element. Meaning that the model can be seen as a hybrid of a linear foundation and agile practices, with a focus on information security.

The TRD-SGW has a novelty value because it merges agile development practices with the idea of a requirements engineering- process. Bernsmed et al. (2019, p. 2) concur that the combination of threat and risk modelling and agile principles is still an understudied area. Therefore, this work has a novelty value to the existing research. It also brought added value to the commissioner by fulfilling the goals set for it. The model enables a project-specific security requirement definitions and concrete security measures to distinct stages of the SDP. Security measures are based on a risk assessment, which is done to all security requirements and can be traced through the requirements engineering process.

The TRD-SGW- model can be easily implemented to the commissioner's software development and its Gateway- process model because it is built on its existing foundation and structure. New elements of the model surfaced from the comparative study, where the most widely used practices were the focus

point. This was also one of the goals set for the model's creation. The final and the most paramount goal was to ensure that the company could develop software and services with high-quality and security. This will be accomplished by utilizing this model and at the same time it will support security throughout product life cycle.

The methodological choice affected to the research process. It resulted to hardships and miscommunication during the action research. The goals and desires of the parties were wide apart. The need for the organization to receive a concrete and tangible solution and the required framework that had to be reached for this thesis were not always compatible. The aim of an action research is to develop a novel approach to an issue or solve a problem with ties to a practical activity (Davison et al., 2012, p. 763). The value of this thesis is eventually defined by its usability as a model for software development and its ability to further develop the current practices in the company.

Further investigation could be addressed to polishing the model with the stakeholder groups. This action requires an owner for the model, eliciting the improvement ideas and critic regularly. This kind of research could for example be directed at one of the model elements and moved through the process one element at a time. This would improve the quality of the elements and adhere to the ideology of continuous improvements. This model is company specific, which restricts its usage only to this context. Therefore, the other investigation development idea is to examine how usable this model would be with a wider audience. It might be beneficial to study this phenomenon among other companies working on the same business section and conduct the process among their stakeholder groups.

REFERENCES

Abdelaziz, A. A., El-Tahir, Y., & Osman, R. (2015). Adaptive Software Development for developing safety critical software. *2015 International Conference on Computing, Control, Networking, Electronics and Embedded Systems Engineering (ICCNEEE)*, 41–46. https://doi.org/10.1109/ICCNEEE.2015.7381425

Abraham, A., Happe, A., Hudic, A., Krenn, S., Notario-McDonnel, N., Striecks, C., & Thiemer, F. (2016). *System Security Requirements, Risk and Threat Analysis*. 97.

Abran, A., Kotonya, G., Moore, J. W., & Sawyer, P. (2001). *Guide to the software engineering body of knowledge: Trial version : a project of the software engineering coordinating committee*. IEEE Computer Society.

Ackoff, R. (1988). *From Data to Wisdom*.

Ahmad, A., Horne, C. A., & Maynard, S. B. (2016). *A Theory on Information Security*. 13.

Aitken, A., & Ilango, V. (2013). A Comparative Analysis of Traditional Software Engineering and Agile Software Development. *2013 46th Hawaii International Conference on System Sciences*, 4751–4760. https://doi.org/10.1109/HICSS.2013.31

Ajayi, O. B., Onashoga, S. A., & Sodiya, A. S. (2006). *Towards Building Secure Software Systems*. *3*, 12.

Alexander, D., Finch, A., Sutton, D., Taylor, A., & Taylor, A. (2013). *Information Security Management Principles*. BCS Learning & Development Limited. http://ebookcentral.proquest.com/lib/jyvaskyla-ebooks/detail.action?docID=1213992

Alexander, I., & Beus-Dukic, L. (2009). *Discovering Requirements: How to Specify Products and Services*. A John Wiley and Sons Ltd.

Alqudah, M., & Razali, R. (2016). A Review of Scaling Agile Methods in Large Software Development. *International Journal on Advanced Science, Engineering and Information Technology*, *6*(6), 828. https://doi.org/10.18517/ijaseit.6.6.1374

Alshayeb, M., Mahmood, S., Mohammed, N. M., & Niazi, M. (2017). Exploring software security approaches in software development lifecycle: A systematic mapping study. *Computer Standards & Interfaces*, *50*, 107–115. https://doi.org/10.1016/j.csi.2016.10.001

Anderson, R. (2001). Why information security is hard—An economic perspective. *Seventeenth Annual Computer Security Applications Conference*, 358–365. https://doi.org/10.1109/ACSAC.2001.991552

Andress, J. (2011). *The basics of information security: Understanding the fundamentals of InfoSec in theory and practice*. Syngress.

Anuradha, J., & Pawar, M. V. (2015). Network Security and Types of Attacks in Network. *Procedia Computer Science*, *48*, 503–506. https://doi.org/10.1016/j.procs.2015.04.126

Apvrille, A., & Pourzandi, M. (2005). Secure software development by example. *IEEE Security Privacy*, *3*(4), 10–17. https://doi.org/10.1109/MSP.2005.103

Babar, M. A., Liming Zhu, Ming Huo, & Verner, J. (2004). Software quality and agile methods. *Proceedings of the 28th Annual International Computer Software and Applications Conference, 2004. COMPSAC 2004.*, 520–525. https://doi.org/10.1109/CMPSAC.2004.1342889

Bahill, A. T., & Henderson, S. J. (2005). Requirements development, verification, and validation exhibited in famous failures. *Systems Engineering*, *8*(1), 1–14. https://doi.org/10.1002/sys.20017

Balaji, S., & Murugaiyan, M. S. (2012). WATERFALLVs V-MODEL Vs AGILE: A COMPARATIVE STUDY ON SDLC. *International Journal of Information Technology and Business Management*, *2*(1), 6.

Barabas, M., Blazek, P., Borcik, F., Fujdiak, R., Misurec, J., Mlynek, P., & Mrnustik, P. (2019). Managing the Secure Software Development. *2019 10th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, 1–4. https://doi.org/10.1109/NTMS.2019.8763845

Baskerville, R. (1993). Information systems security design methods: Implications for information systems development. *ACM Computing Surveys*, *25*(4), 375–414. https://doi.org/10.1145/162124.162127

Baskerville, R., Kuivalainen, T., & Siponen, M. (2005). Integrating Security into Agile Development Methods. *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*, 1–7. https://doi.org/10.1109/HICSS.2005.329

Baskerville, R., & Myers, M. D. (2004). Special Issue on Action Research in Information Systems: Making IS Research Relevant to Practice: Foreword. *MIS Quarterly*, *28*(3), 329–335. JSTOR. https://doi.org/10.2307/25148642

Baskerville, R., & Wood-Harper, A. T. (1998). Diversity in information systems action research methods. *European Journal of Information Systems*, *7*(2), 90–107. https://doi.org/10.1057/palgrave.ejis.3000298

Bassil, Y. (2012). A Simulation Model for the Waterfall Software Development Life Cycle. *International Journal of Engineering*, *2*(5), 7.

Beatty, J., & Wiegers, K. (2013). *Software Requirements*. Microsoft Press. https://www.dawsonera.com/readonline/9780735679641

Beckers, K., Bruegge, B., Klepper, S., Lachberger, P., & Moyon, F. (2018). Towards Continuous Security Compliance in Agile Software Development at Scale. *2018 IEEE/ACM 4th International Workshop on Rapid Continuous Software Engineering (RCoSE)*, 31–34.

Beg, R., Khan, M. H., & Parveen, N. (2014). *Software Security Issues: Requirement Perspectives*. *5*(7), 5.

Bernsmed, K., Cruzes, D. S., Jaatun, M. G., & Tøndel, I. A. (2019). *Exploring Security in Software Architecture and Design:* (M. Felderer & R. Scandariato, Eds.). IGI Global. https://doi.org/10.4018/978-1-5225-6313-6

Bhatia, P. K., & Kumar, G. (2014). Comparative Analysis of Software Engineering Models from Traditional to Modern Methodologies. *2014 Fourth International Conference on Advanced Computing & Communication Technologies*, 189–196. https://doi.org/10.1109/ACCT.2014.73

Bizzell, A., Clinton, B. D., Prentice, R. A., & Stone, D. N. (2017). *Wiley CPAexcel Exam Review April 2017 Study Guide: Business Environment and Concepts*. John Wiley & Sons.

Boehm, B. (2006). A view of 20th and 21st century software engineering. *Proceedings of the 28th International Conference on Software Engineering*, 12–29. https://doi.org/10.1145/1134285.1134288

Bowen, G. A. (2009). Document Analysis as a Qualitative Research Method. *Qualitative Research Journal*, *9*(2), 27–40. https://doi.org/10.3316/QRJ0902027

Broughton, T., Neailey, K., & Phillips, R. (1999). A comparative study of six stage-gate approaches to product development. *Integrated Manufacturing Systems*, *10*(5), 289–297. https://doi.org/10.1108/09576069910371106

Butler, C. W., & Vijayasarathy, L. R. (2016). Choice of Software Development Methodologies. *IEEE SOFTWARE*, 9.

Buyens, K., De Win, B., Grégoire, J., Joosen, W., & Scandariato, R. (2009). On the secure software development process: CLASP, SDL and Touchpoints compared. *Information and Software Technology*, *51*(7), 1152–1171. https://doi.org/10.1016/j.infsof.2008.01.010

Byrne, M., Keary, E., & Lawton, A. (2012). *How to Conduct a Literature Review*. *38*(9), 8.

Calder, A., & Watkins, S. G. (2010). *Information Security Risk Management for ISO27001/ISO27002*. http://web.a.ebscohost.com.ezproxy.jyu.fi/ehost/ebookviewer/ebook/bmxlYmtfXzM5MTA5Nl9fQU41?sid=d921a1d5-3c01-42a2-9c7e-a269b28ad50a@sdc-v-sessmgr01&vid=0&format=EB&rid=1

Campbell, T. (2016). *Practical Information Security Management: A Complete Guide to Planning and Implementation*. Apress.

CESSDA Training Working Group. (2017). *CESSDA Data Management Expert Guide*.

Cho, J. (2008). ISSUES AND CHALLENGES OF AGILE SOFTWARE DEVELOPMENT WITH SCRUM. *Issues in Information Systems*, *IX*(2), 8.

CNSS, C. on N. S. (2010). *CNSS Instruction No. 4009*.

Collins English Dictionary. (2019). *Motivation definition and meaning | Collins English Dictionary*. https://www.collinsdictionary.com/dictionary/english/motivation

Cooper, R. G. (1990). Stage-gate systems: A new tool for managing new products. *Business Horizons*, *33*(3), 44–54. https://doi.org/10.1016/0007-6813(90)90040-I

Dale, R., & Saiedian, H. (2000). Requirements engineering: Making the connection between the software developer and customer. *Information and Software Technology*, *42*(6), 419–428. https://doi.org/10.1016/S0950-5849(99)00101-9

Dardick, G. S. (2010). Cyber Forensics Assurance [PDF]. 8th Australian Digital Forensics Conference, Edith Cowan University, November 30th 2010. https://doi.org/10.4225/75/57B2926C40CDA

Davison, R., Kock, N., & Martinsons, M. G. (2004). Principles of canonical action research. *Information Systems Journal*, *14*(1), 65–86. https://doi.org/10.1111/j.1365-2575.2004.00162.x

Davison, R., Martinsons, M. G., & Ou, C. X. J. (2012). The Roles of Theory in Canonical Action Research. *MIS Quarterly*, *36*(3), 763. https://doi.org/10.2307/41703480

Dick, J., Hull, E., & Jackson, K. (2005). *Requirements engineering* (2. ed). Springer.

Dick, J., Hull, E., & Jackson, K. (2011). *Requirements Engineering | Elizabeth Hull | Springer* (3.rd). Springer-Verlag London Limited 2011. https://www.springer.com/gp/book/9781447158189

Dorfman, M., & Thayer, R. (2000). *Software requirements engineering*. John Wiley & Sons. https://ieeexplore.ieee.org/servlet/opac?bknumber=5989265

Doyle, S. (2000). *Understanding Information Technology*. Nelson Thornes.

Easterbrook, S., & Nuseibeh, B. (2000). Requirements engineering: A roadmap. *Proceedings of the Conference on The Future of Software Engineering - ICSE '00*, 35–46. https://doi.org/10.1145/336512.336523

Eberlein, A., Maurer, F., & Paetsch, F. (2003). Requirements engineering and agile software development. *WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003.*, 308–313. https://doi.org/10.1109/ENABL.2003.1231428

Ellis, T. J., & Levy, Y. (2006). A Systems Approach to Conduct an Effective Literature Review in Support of Information Systems Research. *Informing Science: The International Journal of an Emerging Transdiscipline*, *9*, 181–212. https://doi.org/10.28945/479

Finnish security committee. (2018). Vocabulary of Cyber Security. In *Turvallisuuskomitea.fi*. https://turvallisuuskomitea.fi/wp-content/uploads/2018/06/Kyberturvallisuuden-sanasto.pdf

Flechais, I., & Sasse, M. A. (2005). Usable Security: Why Do We Need It? How Do We Get It? In L. F. Cranor & S. Garfinkel (Eds.), *Security and Usability: Designing secure systems that people can use* (pp. 13–30). O'Reilly. http://shop.oreilly.com/product/9780596008277.do

Futcher, L. A. (2007). *A Model for Integrating Information Security into the Software Development Life Cycle*. 198.

Futcher, L. A., & von Solms, R. (2007). SecSDM: A Model for Integrating Security into the Software Development Life Cycle. *ResearchGate*. Fifth World Conference on Information Security Education, United States Military Academy, West Point, New York, USA. https://doi.org/10.1007/978-0-387-73269-5_6

Gedam, M. N., & Meshram, B. B. (2019). *Vulnerabilities & Attacks in SRS for Object-Oriented Software Development*. 6.

Ghilic-Micu, B., Mircea, M., & Stoica, M. (2013). Software Development: Agile vs. Traditional. *Informatica Economica*, *17*(4/2013), 64–76. https://doi.org/10.12948/issn14531305/17.4.2013.06

Goertzel, K. M., & Jarzombek, J. (2006). Security in the Software Life Cycle. *The Journal of Defense Software Engineering*, 6.

Gupta, A. (2014). *International Journal of Advance Research in Computer Science and Management Studies*. *2*(12), 6.

Hadavi, M. A., Hamishagi, V. S., & Sangchi, H. M. (2008). Security Requirements Engineering; State of the Art and Research Challenges. *Hong Kong*, 7.

Haley, C. B., Laney, R., Moffett, J. D., & Nuseibeh, B. (2008). Security Requirements Engineering: A Framework for Representation and Analysis. *IEEE Transactions on Software Engineering*, *34*(1), 133–153. https://doi.org/10.1109/TSE.2007.70754

Higuera, B., Mohino, de, & Montalvo, J. A. (2019). The Application of a New Secure Software Development Life Cycle (S-SDLC) with Agile Methodologies. *Electronics*, *8*, 1218. https://doi.org/10.3390/electronics8111218

Hossain, D. S. A., & Moniruzzaman, A. B. M. (2013). *Comparative Study on Agile software development methodologies*. 25.

Howard, M. (2004). Building more secure software with improved development processes. *IEEE Security Privacy*, *2*(6), 63–65. https://doi.org/10.1109/MSP.2004.95

ISO/IEC 25010:2011. (ISO 2011). Retrieved January 24, 2020, from https://www.iso.org/standard/35733.html

ISO/IEC/IEEE 29148-2011. (ISO 2011b). Retrieved January 24, 2020, from https://standards.ieee.org/standard/29148-2011.html

ISO/IEC/IEEE 29148-2018. (ISO 2018). Retrieved February 16, 2020, from https://standards.ieee.org/standard/29148-2018.html

Jain, N. K., & Patel, U. A. (2013). New Idea In Waterfall Model For Real Time Software Development. *International Journal of Engineering Research*, *2*(4), 6.

Joint Pub. (1998). *Joint Doctrine for Information Operations*. Joint Pub 3-13 - Joint Doctrine for Information Operations. http://www.c4i.org/jp3_13.pdf

Joint Task Force Transformation Initiative. (2013). *Security and Privacy Controls for Federal Information Systems and Organizations* (NIST SP 800-53r4; p. NIST SP 800-53r4). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-53r4

Kamkarhaghighi, M., Moghaddasi, H., & Sajjadi, S. (2016). Reasons in Support of Data Security and Data Security Management as Two Independent Concepts: A New Model. *The Open Medical Informatics Journal*, *10*, 4–10. https://doi.org/10.2174/1874431101610010004

Kassab, M., Laplante, P., & Neill, C. (2014). State of Practice in Requirements Engineering: Contemporary Data. *Innovations in Systems and Software* Engineering: A NASA Journal. https://doi.org/10.1007/s11334-014-0232-4

Koskinen, A. (2020). DEVSECOPS: BUILDING SECURITY INTO THE CORE OF DEVOPS [Master thesis, University of Jyväskylä]. https://jyx.jyu.fi/bitstream/handle/123456789/67345/URN%3aNBN%3afi%3ajyu-202001171290.pdf?sequence=1&isAllowed=y

Kotonya, G., & Sommerville, I. (1998). *Requirement Engineering Processes and Techniques*. John Wiley & Sons Ltd.

Lamsweerde, A. van. (2009). *Requirements Engineering From System Goals to UML Models to Software Specifications*. John Wiley & Sons Ltd.

Laplante, P. A. (2017). *Requirements Engineering for Software and Systems* (3rd ed.). CRC Press Taylor & Francis Group.

Lauesen, S. (2002). *Software Requirements: Styles and Techniques*. Pearson Education.

Maciaszek, L. (2007). *Requirements Analysis and System Design*. Pearson Education.

Madden, A. D. (2000). A definition of information. *Aslib Proceedings*, *52*(9), 343–349. https://doi.org/10.1108/EUM0000000007027

Mattord, H. J., & Whitman, M. E. (2009). *Principles of Information Security*. Cengage Learning EMEA.

Mattord, H. J., & Whitman, M. E. (2011). *Principles of Information Security*. Cengage Learning.

Mattord, H. J., & Whitman, M. E. (2013). *Management of Information Security*. Cengage Learning.

Mattord, H. J., & Whitman, M. E. (2017). *Principles of Information Security*. Cengage Learning.

McGraw, G. (2006). *Software Security: Building Security in*. Addison-Wesley Professional.

McGraw, G., Miguels, S., & West, J. (2017). *BSIMM8*. https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=3&cad=rja&uact=8&ved=2ahUKEwjX9-a1zKPoAhVM4KYKHZrcDZMQFjACegQIARAB&url=https%3A%2F%2Fwww.bsimm.com%2Fcontent%2Fdam%2Fbsimm%2Freports%2Fbsimm9.pdf&usg=AOvVaw34fLYhtYTPzlVj2f3Gtdb5

McNiff, J. (2017). *Action research: All you need to know* (1st edition). SAGE Publications.

Mead, N. R., Padmanabhan, D., Raveendran, A., & Viswanathan, V. (2008). *Incorporating Security Quality Requirements Engineering (SQUARE) into Standard Life-Cycle Models:* Defense Technical Information Center. https://doi.org/10.21236/ADA482345

Merkow, M. S., & Raghavan, L. (2010). *Secure and Resilient Software Development*. Auerbach Publications. http://ebookcentral.proquest.com/lib/helsinki-ebooks/detail.action?docID=1446770

Merriam-Webster Dictionary. (2020). Definition of ASSURANCE. In *Merriam-Webster Dictionary*. https://www.merriam-webster.com/dictionary/assurance

Mitchell, S. M., & Seaman, C. B. (2009). A comparison of software cost, duration, and quality for waterfall vs. iterative and incremental development: A systematic review. *2009 3rd International Symposium on Empirical Software Engineering and Measurement*, 511–515. https://doi.org/10.1109/ESEM.2009.5314228

Nielsen, K. (2007). The Qualitative Research Interview and Issues of Knowledge. *Nordic Psychology*, *59*(3), 210–222. https://doi.org/10.1027/1901-2276.59.3.210

NIST. (2020). *Information Assurance (IA) – Glossary | CSRC* [Glossary]. Computer Security Resource Center. https://csrc.nist.gov/glossary/term/information-assurance

Pandey, S. (2011). MODERN NETWORK SECURITY: ISSUES AND CHALLENGES. *International Journal of Engineering Science and Technology*, *3*(5), 7.

Parker, D. B. (1998). *Fighting Computer Crime: A New Framework for Protecting Information*. John Wiley & Sons.

Parnas, D. L. (2000). Requirements documentation: Why a formal basis is essential. *Proceedings Fourth International Conference on Requirements*

Engineering. ICRE 2000. (Cat. No.98TB100219), 81–82. https://doi.org/10.1109/ICRE.2000.855594

Peltier, T. R. (2013). *Information Security Fundamentals, Second Edition*. CRC Press.

Pickvance, C. (2005). *The four varieties of comparative analysis: The case of environmental regulation*. 1–20.

Pressman, R. S. (2005). *Software Engineering: A Practioner's Approach* (6th ed.). McGraw-Hill.

Qadir, S., & Quadri, S. M. K. (2016). Information Availability: An Insight into the Most Important Attribute of Information Security. *Journal of Information Security*, *07*(03), 185–194. https://doi.org/10.4236/jis.2016.73014

Raggad, B. G. (2010). *Information Security Management: Concepts and Practice*. CRC Press.

Rice, R. S., & Tompkins, F. G. (1986). Integrating security activities into the software development life cycle and the software quality assurance process. *Computers & Security*, *5*(3), 218–242. https://doi.org/10.1016/0167-4048(86)90014-3

Ruparelia, N. B. (2010). Software development lifecycle models.html. *ACM SIGSOFT Software Engineering Notes*, *35*(3), 8–13. https://doi.org/10.1145/1764810.1764814

Shoemaker, D., & Sigler, K. (2014). *Cybersecurity: Engineering a Secure Information Technology Organization*. Cengage Learning.

Siponen, M. (2006). Secure-system design methods: Evolution and future directions. *IT Professional*, *8*, 40–44. https://doi.org/10.1109/MITP.2006.73

Siponen, M., & Willison, R. (2009). Information security management standards: Problems and solutions. *Information & Management*, *46*(5), 267–270. https://doi.org/10.1016/j.im.2008.12.007

Stringer, E. T. (2013). *Action Research*. SAGE Publications.

Thamhain, H. J. (2000). *Accelerating product developments via phase-gate processes*. https://www.pmi.org/learning/library/phase-gate-processes-promising-complex-547

Tilly, C. (1984). *Big Structures, Large Processes, Huge Comparisons*. Russell Sage Foundation.

van Niekerk, J., & von Solms, R. (2013). From information security to cyber security. *Computers & Security*, *38*, 97–102. https://doi.org/10.1016/j.cose.2013.04.004

Vuori, M. (2011). *Agile Development of Safety-Critical Software*. *14*, 114.

Watson, R. T., & Webster, J. (2002). Analyzing the past to prepare for the future: Writing a literature review—ProQuest. *MIS Quarterly*, *26*(2), R13.

Zahran, S. (1998). *Software process improvement: Practical guidelines for business success*. Addison-Wesley.

Zave, P. (1984). The operational versus the conventional approach to software development | Communications of the ACM. *Communications of the ACM ACMPUB27*, *27*(2), 104–118. https://doi.org/10.1145/69610.357982

Zave, P. (1997). Classification of research efforts in requirements engineering. *ACM Computing Surveys*, *29*(4), 315–321. https://doi.org/10.1145/267580.267581

ANNEX 1 INTERVIEW TEMPLATE

Mandatory = **M** or Optional question = **O**
Freetext = **FT** or Multiple Choice = **MC**

1. Could you provide your personal information? **(M/FT)**
    a. Title and department
    b. A short job description of your role in the software development organization?
    c. What are the main responsibilities in your role?

2. How do you see your role in the requirements engineering process? (provide a possible role such as: subject matter experts, software systems engineer, architects and so on) **(M/FT)**

3. Which phase/phases of the company specific software development process (CSSDP) are you involved with? (the FIGURE 29 is shown to the interviewees) **(M/MC)**

4. In what phase of the CSSDP do you think your role is the most important? **(O/MC)**
    a. What do you think about the CSSDP? **(O/MC)**

5. Is the company using a product mission statement or any document that would provide that information? (description of the most important features of a product) **(M/FT)**
    a. Who approves such a document? **(O/FT)**

6. Why are requirements collected? **(M/FT)**

7. When are requirements collected? (a timeframe and/or a phase in the CSSDP) **(M/FT)**

8. How are requirements collected? (provide a method) **(M/FT)**

9. How are the "raw" requirements analyzed? **(M/FT)**
    a. Do you do analyzing? **(O/FT)**
    b. What are the methods used? **(O/FT)**

10. How are requirements documented? **(M/FT)**
    a. Do you use a standardized model? **(O/FT)**

11. How do you utilize the requirements? **(M/FT)**

   a. What is their effect on the software development? **(O/FT)**
   b. How do the requirements affect the software development? **(O/FT)**

12. How is it supervised that the requirements get implemented? **(M/FT)**
   a. What are the methods? **(O/FT)**

13. What are the different kinds of requirements you see the most? (such as user, system, design) **(M/FT)**

14. From what stakeholder groups are requirements collected from? (such as user, client, legal and so on) **(M/FT)**

15. What are the models for requirements presentation? (picture, text, other) **(M/FT)**

16. How is it confirmed that the market/user is understood correctly (from the business point of view)? **(M/FT)**
   a. How do you make sure? **(O/FT)**

17. How the system context is understood? **(M/FT)**
   a. System boundaries? **(O/FT)**
   b. Who are involved with system usage? **(O/FT)**
   c. How does the system converse with other systems? **(O/FT)**
   d. What is the (business and usage) environment like? **(O/FT)**

18. Who takes responsibility if the requirements engineering process fails? (from the point-of-view of your role) **(M/FT)**
   a. Product doesn't respond to the need the customer presented, who "takes the fall"? **(O/FT)**

ANNEX 2 THE COMPANY SPECIFIC SOFTWARE DEVELOPMENT PROCESS (CSSDP)

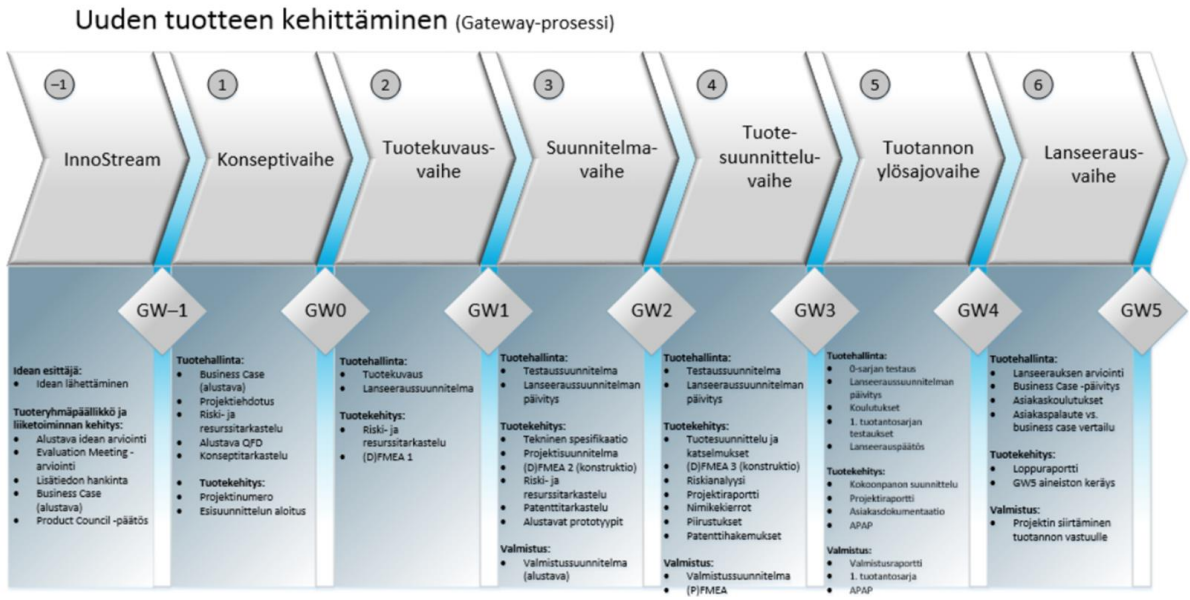

FIGURE 29 The CSSDP- model used in product development

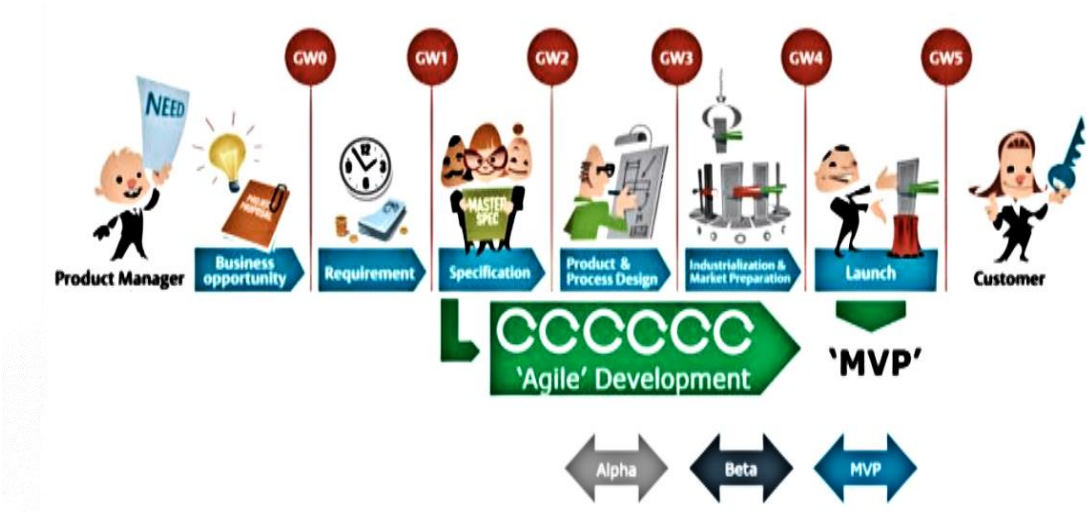

FIGURE 30 The high level CSSDP- model

ANNEX 3 SEMI-STRUCTURED INTERVIEW RESULT TABLES

TABLE 12 Total amount of interviewees per stakeholder group

Groups / interviews	Business Development	Software Development	Product Management	Legal	Sales	Service Center
22	6	5	4	1	5	1

TABLE 13 Usage of a product mission statement (PMS) -document

Answers grouped according the theme	**Number of mentions**
I cannot answer / I do not know; I am not familiar with this kind of a document	7
Yes, I believe that we have such a document. I suppose so	4
We do not have it, but we do need it	2
User Story Mapping works as product mission statement	1
I think, we do not have it, but I suppose that we are creating one	1
Power Point works as product mission statement	1
Service description works as product mission statement	1
We do not have it in its usual form, but we are using version releases as product mission statement	1
We do not have it in its usual form, but we have common product descriptions	1
No reply	1
TOTAL	**20**

TABLE 14 Acceptor of product mission statement (PMS) -document

Answers grouped according the theme	**Number of mentions**
I do not know	2
Product group manager	1
Person named to the task	1
Product owner	1
Concept owner, who will ask steering groups approval for it	1
Steering group of the project	1
Product council	1
TOTAL	**8**

TABLE 15 Reason for requirements elicitation

Answers grouped according the theme	**Number of mentions**
Collected to produce the best product, software or service to fulfil customer needs	5
Collected to improve mutual understanding about things related to the product	3
Collected to gain understanding about the customer	3
Collected to improve business	2
Collected to produce added value to the customers	2
Collected to avoid making over-quality	1
Collected to fulfil the customer needs better via the product	1
Collected to produce products that have a customer orien-tated approach	1
Collected to develop company's market position	1
Collected to put them into InnoStream	1
Collected to understand the user	1
Collected to understand what we are developing	1
Collected to improve business competitiveness	1
Collected to be able to program and test the product	1
Collected to avoid doing waste	1
TOTAL	**25**

TABLE 16 Requirements elicitation time

Answers grouped according the theme	**Number of mentions**
Elicitation is a continuous process	9
I do not know	2
Elicited in the beginning of the process	2
Elicitation should be a continuous process	2
Elicited when we are going to develop something new	1
Elicited and reviewed multiple times	1
Elicitation depends on process and process model used	1
Hardware POW, requirements are elicited before gateway 1	1
Elicited when necessary, but it is not done systematically	1
TOTAL	**20**

TABLE 17 Initial requirements elicitation tools and methods

Answers grouped according the theme	Number of mentions
Via discussions	6
Via observations in the field	4
Via market researches	4
Via interviews	3
Via inquiries	3
Via competitor analyses	3
Via sales events	3
We have no process for requirements elicitation	3
Salesmen forward the information coming from customers to development	2
By discussing with the development team and using experience	2
Via regular customer meetings	2
Via sparring sales (export)	2
By using user story mapping	2
Via email	2
By using VOC -technique (Voice of the customer)	2
By observing distributer's operations	1
By making notes when visiting a customer	1
By collecting these requirements to a data pool when we have one	1
Big operatives on the field are collecting lists about competitive tendering and sharing this information with our company	1
I have not participated in requirements elicitation process	1
By collecting customer feedback (phone calls or problem tickets) and by making my own perceptions about development objectives and by registering them into ticket to commissioner's group ticketing service	1
By visiting customers	1
By getting the information from our distributers, after they have met our customers and heard about their needs	1
TOTAL	**51**

TABLE 18 Requirement elicitation tools and methods

Answers grouped according the theme	Number of mentions
Via discussions	17
Other tools and methods	12
Via observations	5
Via market researches	4
Via competitor analysis	4
Via interviews	3
Via inquiries	3
We have no requirements elicitation process	3
TOTAL	**51**

TABLE 19 Requirements documentation forms and databases

Answers grouped according the theme	**Number of mentions**
As a PowerPoint	6
JIRA	6
Confluence	5
To InnoStream (where they are in excel form)	3
As word -documents	3
Via voice of customer (VOC) -documents	3
We have no structured way/form to save requirements	3
User case and user story mapping	3
CRM	3
To folders in company's network disk	2
I do not document them, there is no place for that	2
As a Value Proposition Canvas (VPC)	2
Miro	2
Wiki	2
Excel	2
Email	2
As CORE -tickets	1
As animated videos	1
As MRD -documents	1
As a Comparative chart	1
By using a customer project form	1
As TST -tickets	1
As documents describing customer value	1
As a Business Proposition Canvas (BPC)	1
SharePoint	1
Teams	1
TOTAL	**59**

TABLE 20 Methods for requirement analysis

Answers grouped according the theme	Number of mentions
What the biggest customers are saying that they need or want. So, according to the potential cash flow	4
Analysis is made in workshops with the development team by prioritizing elicited requirements	4
By discussions with the development team	3
By asking questions repeatedly, until the customer can provide accurate definitions	3
By considering if requirements are executable	3
Via InnoStream	2
By going through the requirements with the customer	2
I discuss with the customer to analyze the requirements	2
Requirements are analyzed in JIRA, where they will get points according to working hours needed to fulfil them	2
Via Comparative Chart	1
Prioritization between ideas that our customers have mostly asked for. All those ideas will be taken into cost calculation process and the best options put forward	1
Via Affinity Wall	1
I believe that we make an analysis from requirements	1
I call to software development and ask if it is possible to produce the product according to these requirements	1
By analyzing the business case	1
Interviewee does not provide an answer to the question	1
In the mechanical side, we are using tool called QFD, for the analysis	1
Via flipchart	1
With a discussion	1
By testing if the requirement can be fulfilled	1
Product owner makes requirements prioritization and elimination	1
Via time-estimation analysis	1
Via user story mapping	1
TOTAL	**38**

TABLE 21 Utilization of requirements

Answers grouped according the theme	Number of mentions
Building future business portfolio and business potential	3
Interviewee gives an answer, but it does not answer to the presented question	2
It is hard to say, which requirements really are utilized, because we do not have visibility to InnoStream	2
Requirements are used in sales cases	2
Requirements are used to make a technical product specification	2
Requirements are used to carry out sprints	2
Requirements are utilized for coding support and testing	2
Strategic decision making in business management level	1
Only those requirements are utilized, which are most highlighted and asked for	1
Requirements are used for prioritization; what things we should do, or which are profitable	1
Requirements are used for specification creation for minimum viable product (MVP) and after that to improve the product	1
In software development requirements are used for new releases and further development	1
Requirements are used through the whole product lifecycle to improve the product	1
Requirements help development team to figure out, how to build the wanted product	1
From hardware POW requirements affect to the product that is sold and through that our customer satisfaction and business	1
Requirements are used for resource planning	1
No reply	1
Programming tasks and documentation are made based on the requirements	1
Requirements are used in requirements execution	1
Requirements are used both in programming and building automated testing	1
Shared technologies has a business which is too far from our customers, which is why we feel that they are making solutions, which do not meet with our customer's needs. We feel that they do not really understand our customers and therefore cannot make solutions that can fulfil their needs. We would like to do this job by ourselves.	1
A Development team starts to program the software based on requirements, which have been elicited and documented (a hardware POW)	1
TOTAL	**30**

TABLE 22 Roles and tools of requirement implementation supervising

Answers grouped according the theme	**Number of mentions**
Implementation is supervised with involving customer to the process	4
Supervised with usability testing	4
Supervised with automatized testing	4
No reply	3
Supervised with Alfa and Beta testing	3
Supervised with Pen-testing	1
We are lacking a systematic process and it should be built	1
Supervised with demos	1
Supervised via testing software in-house	1
Supervised by the product manager, who ensures that requirements have been met	1
Not supervised. It is not scheduled nor included as someone's responsibility	1
Supervised with a testing plan and a product specification	1
Supervised by the project team. They follow the implementation weekly in team meetings (a hardware POW)	1
Supervised by the product owner, who also tests the software	1
TOTAL	**29**

TABLE 23 Occurrence of different requirement types

Answers grouped according the theme	**Number of mentions**
User requirements	7
Functional requirements	7
Business requirements (time, money, resources)	5
Information security requirements	3
GDPR	3
System requirements	3
Technical requirements	3
Administrator requirements	2
Customer (distributor) requirements	2
Company (concern) requirements	2
Law requirements	2
System design requirements	2
Requirements set by standards	2
No reply	1
Architectural requirements	1
(Data) Communication requirements	1
Requirements related to protection practices	1
Non-functional requirements	1
Competitor functionalities that we should be able to answer	1
KATAKRI	1
Data privacy requirements	1
TES (collective labor agreement)	1
Integrations and requirements needed to produce them	1
Definitions given by public authorities of a country related to the data	1
Requirements related to cloud	1
Usability requirements	1
Stakeholders are defining the state of security that the company will try to reach – security level requirements	1
Requirements given by the product owner	1
Requirements set by company lead through the road map given to software development	1
TOTAL	**62**

TABLE 24 Stakeholder groups involved in requirements elicitation

Answers grouped according the theme	**Number of mentions**
Customer (distributor)	10
Customer (end-customer)	9
Law and authorities	6
Third party	5
Sales (in-house stakeholder)	4
User	4
Shared Technologies	4
EMEA level of the concern	4
Product group managers (in-house stakeholder)	3
Competitors	3
Customer (no level specified)	3
In-house customers	2
HID	2
Architect offices	2
No answer	1
Operations -department (in-house stakeholder)	1
Marketing (in-house stakeholder)	1
Export (in-house stakeholder)	1
Law (in-house stakeholder)	1
Software development team in another location (in-house stakeholder)	1
Product management (in-house stakeholder)	1
Human resources admin (in-house stakeholder)	1
IT (in-house stakeholder)	1
Software development team (in-house stakeholder)	1
In-house stakeholders	1
Big customers with biggest business potential	1
Customer's customer	1
Software suppliers	1
Cloud service provider	1
Society	1
Concern	1
Technical support (in-house stakeholder)	1
Sales offices	1
Software sustenance (in-house stakeholder)	1
Maintenance (in-house stakeholder)	1
Architect (in-house stakeholder)	1
TOTAL	**82**

TABLE 25 Models of requirements presentation

Answers grouped according the theme	Number of answers
Plain text	11
Picture	6
Verbal	5
PowerPoint	2
No accurate model	2
Videos	2
Email	2
Drawings (hardware POW)	1
Report	1
Service Blueprint	1
Logical chain of events	1
Value Stream Mapping (VSM)	1
Visualized from customer needs	1
Word and excel	1
JIRA -ticket	1
User story	1
TOTAL	**39**

TABLE 26 Methods for market and customer understanding

Answers grouped according the theme	Number of mentions
By customer understanding	3
No reply	2
By visiting customers	2
By making advance clearance about the customer	1
Via experience from customers and their business	1
By conversations with different teams about the customer needs	1
By discussing with product manager and segment owner	1
By using local people	1
Via sales	1
Via retailers	1
I have not participated to this phase of the process	1
By using open source intelligence	1
By observing	1
Via protos	1
Currently we are not making sure that we really understand the market or the customer	1
By knowing the competitor	1
Via value stream mapping (VSM)	1
Via business model canvas (BMC)	1
By sending someone to visit the customer, who really knows the business of the customer and customer itself (business environment is familiar)	1
By selecting a focus group and focusing on it	1
By exploiting open source intelligence and confirming the information with a party, who has the competence	1
By checking the end-product with the customer agilely	1
TOTAL	**25**

TABLE 27 Methods for system context understanding

Answers grouped according the theme	Number of mentions
This is not part of my job description	3
I do not know, how is it done	2
Via customer visits	2
By integrations	2
There are as many ways to make it as makers	1
Segment owners, sales and product owners are trying to make as accurate hypotheses as possible	1
By interviewing people	1
By investigating business processes	1
By communicating with inhouse stakeholders, such as sales	1
By communicating with customers	1
Direct customer feedback	1
By understanding customers, and their partners as well as ours and by understanding the business environment, where the product is supposed to be sold	1
By understanding customer organization's roles - especially those, which affect purchasing decisions	1
By experience	1
This information comes from our business management	1
By market researches	1
By communicating with inhouse stakeholders, such as support (operations)	1
By communicating with stakeholders, such as maintenance (hardware POW)	1
By performing a technical investigation	1
TOTAL	**24**

TABLE 28 Responsibility of requirement engineering process

Answers grouped according the theme	Number of mentions
Collective responsibility among the steering group of the project	3
No reply	2
Development team	2
The owner of the information (owner of the requirement)	2
Product group manager	2
It is unclear, who is the responsible person	1
I do not know	1
Concept owner	1
Collective responsibility	1
Product development organization as a whole	1
Sales	1
The one who knows the problem best (professional)	1
A party, who has been in contact with the customer	1
The (project/product) owner organization	1
Shared Technologies or the development team if the failure is with a technical solution	1
If the product development has not understood what they are about to develop, the responsibility lies with the project manager	1
If the customer needs are not understood, I have no idea, who is responsible	1
Project steering group	1
With SaaS -solutions, Operations-unit is responsible	1
TOTAL	**23**

ANNEX 4 FIRST ITERATION OF COMPARATIVE STUDY

Abbreviation	Full name	Year	Enforce security based on risk assessment model	A Maturity Model	Security model for RE	SSDLC process	Other
AEGIS	Appropriate and Effective Guidance for Information Security	1999				√	
AOD	Aspect-Oriented design	1999				√	
Apvrille & Pourzandi	-	2005				√	
BLP	Bell-LaPadula model	1973				√	
BSIMM	Building Security in Maturity Model	2008		√			
CbyC	Correctness by Construction	2002				√	
CLASP/by OWASP	Comprehensive Lightweight Application Security Process	2005			√		
Cleanroom	-	1985				√	
CMMI-DEV	Capability Maturity Model Integration for Development	2010		√			
CORAS	Risk Assessment of Security Critical Systems	2001	√				
CRAMM	Central Computer and Telecommunications Agency (CCTA) Risk Analysis and Management Method	1986	√				
EBIOS	Expression des Besoins et Identification des Objectifs de Sécurité/Expression of needs and identification of security objectives	1995	√				
GBRAM	Goal Based Requirement Analysis Methods	1996			√		
Hadawi	Set of Secure Development Activities	2007				√	
ISDF	Integrated Security Development Framework	2017				√	
i*Tropos	i = ("threat" and "security constraint"), Trust, ownership, and permission delegation meta-model	1999				√	
KAOS	Knowledge acquisition in automated specification method/Keep all objectives satisfied	2007			√		
MS SDL	Microsoft software development	2002				√	
Microsoft SDL-Agile	Microsoft software development agile	2009				√	
OCTAVE	Operationally Critical Threat, Asset, and Vulnerability Evaluation	1999	√				
OpenSAMM/by OWASP	Open Software Assurance Maturity Model	2009		√			
Protection Poker	-	2010					√
S2D-ProM	Secure Software Development Process Model	2007				√	
SaFe	Scaled Agile framework	2011				√	
SAFECode	Software Assurance Forum for Excellence in Code	2011					√
SAMM/by OWASP	Software Assurance Maturity Model	2009		√			
SCR	Software Cost Reduction	2008					√
SDLC	Software Development Life Cycle	-				√	
SecSDM	Secure Software Development Model	2007				√	
Securosis SSDL	-	2009				√	
SQUARE	Security Quality Requirements Engineering	2005			√		
SREF	Security Requirements Engineering Framework	2009			√		
SREP	Security Requirements Engineering Process	2006			√		
SSAI	Software Security Assessment Instrument	2003				√	
S-SCRUM	-	2014				√	
SSDLC	Secure Software Development Life Cycle	-				√	
SSDM	Secure Software Development Model	1975				√	
SSE-CMM	The System Security Engineering Capability Maturity Model	1999		√			
Touchpoints/by McGraw	-	2007				√	
Tropos	-	1999					√
TSP-Secure	Team Software Process for Secure Software Development	2002				√	
41			4	5	6	22	4

ANNEX 5 ALL THE COMPARATIVE STUDY ITEARATION OUTPUTS

Iteration	Purpose	Result
1.	Discover and list initial SSDL- models found through literature review	A list of 41 SSDL- models + CSSDP-model
2.	Select the most suitable models to third iteration with the commissioner	A list of 6 SSDL- models
3.	Evaluate models to selection criteria	None of the models fulfils the criteria by itself. Therefore, a combination model is needed. Exclusion of two models
4.	Comparison between models	SQUARE value exceeds its one deficiency, all others (3) fulfil the criteria
5.	Comparison of the model content between Higuera et al. and this thesis	Content of the table 8, including the most suitable practices for the commissioner

ANNEX 6 ALL THE COMPARATIVE STUDY ITEARATION OUTPUTS

Elements	Literature	Document analysis	Interview	Comparative study
Product mission statement (PMS)	Recommends a document form called PMS for shared understanding about the aim of software and its critical features		A unified form was requested by the sales; for customer requests & questions from features & benefit	Product features should be defined
Security classification	Define security categories, their criteria from those security goal for each software product can be specified. RE requires that requirements are classed			Security goals should be defined for each software product
Requirements document	Traceability, management, change management, legal foundation for software	Requirement book (hardware POV)	The requested that it would be developed, unified and standardized	Used for requirements management (ex. elicitation)
Technical design plan		Practice of the CSSDP		Definition of secure design through threat&risk modelling
Test plan		Practice of the CSSDP		
Threat modelling & risk analysis (TMRA) -Security requirements -Privacy requirements	TMRA enables security requirement creation and requires a PMS document as an input	Current practices that CSSDP are built for product devel. & require updating	There are deficiencies on requirement inclusion and comprehension (ex. privacy)	TMRA enables security requirement creation and requires a product definition as an input
Beta security report				Security report should be implemented to present decision-making processes & rationales about implemented requirements
MVP security report				Security report should be implemented to present decision-making processes & rationales about implemented requirements
Requirement changes	Provide unique identifiers for req. managing their evolution, tracing and changes			