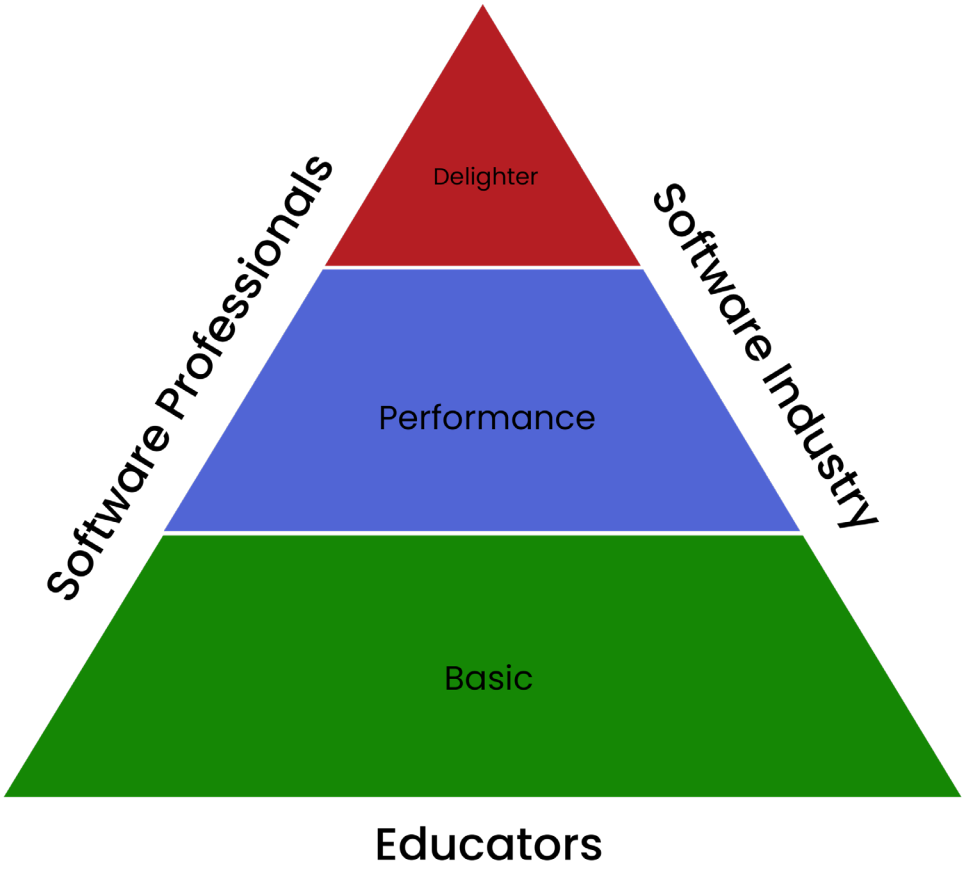
JYU DISSERTATIONS 467

Nana Assyne

Determining the essential competencies of software professionals

A unified framework



UNIVERSITY OF JYVÄSKYLÄ
FACULTY OF INFORMATION TECHNOLOGY

JYU DISSERTATIONS 467

Nana Assyne

Determining the essential competencies of software professionals

A unified framework

Esitetään Jyväskylän yliopiston informaatioteknologian tiedekunnan suostumuksella
julkisesti tarkastettavaksi Agora-rakennuksen auditoriossa Aud 2
marraskuun 30. päivänä 2021 kello 12.

Academic dissertation to be publicly discussed, by permission of
the Faculty of Information Technology of the University of Jyväskylä,
in building Agora, auditorium Aud 2 on November 30, 2021 at 12 o'clock noon.

JYVÄSKYLÄN YLIOPISTO
UNIVERSITY OF JYVÄSKYLÄ

JYVÄSKYLÄ 2021

Editors
Marja-Leena Rantalainen
Faculty of Information Technology, University of Jyväskylä
Timo Hautala
Open Science Centre, University of Jyväskylä

ISBN 978-951-39-8947-7 (PDF)
URN:ISBN:978-951-39-8947-7
ISSN 2489-9003

Permanent link to this publication: http://urn.fi/URN:ISBN:978-951-39-8947-7

DEDICATION

I dedicate this work to my late mother Mrs. Emilia Mary Assyne and my wife Mavis Assyne for their selfless support in my life. You girls will continue to be my motivator in all that I do. To my mother, even though you did not live to read the final work, "the original" (Mrs. Assyne) is in charge. For Mavis, on this long journey, you have been by my side encouraging and pushing me to take the next step. Without your support, it would have been impossible for me to complete this journey. Thank you.

I also dedicate this work to my Dad (Hon. S.K. Assyne), my daughters (Mame Esane and Naa Aso), and the entire Assyne family. It is your advice and support that has brought me this far; I say thank you and remember to continue to pray for me to be a good boy.

ABSTRACT

Assyne, Nana
Determining the Essential Competencies of Software Professionals: A Unified Framework
University of Jyväskylä, 2021, 74 p.
(JYU Dissertations
ISSN 2489-9003; 467)
ISBN 978-951-39-8947-7

The competencies of software professionals have been under the radar of software engineering research and practice for decades. Different models and frameworks, as well as identification and assessment criteria, have been developed to understand and manage software engineering competencies (SEC). Although research on software engineering competencies is not lacking, there appears to be insufficient measures for stakeholders (software professionals, educators, and the software industry) to identify and assess SEC based on different software development projects. Previous studies have portrayed SEC as static in software projects; thus, their evolution is not covered in the literature. To the best of our knowledge, no holistic software engineering competence model or framework has been presented to identify competencies, competence satisfaction levels, and the essential competencies to be used in different software projects.

In light of these observations, we first conducted a mapping study to understand the state of research on SEC, revealing gaps in the knowledge. We then attempted to address some of the gaps by building models and frameworks for managing SEC using findings from the literature and several rounds of stakeholder consultations. Data from interviews with supervisors in software development were used to construct a holistic framework to identify competencies, competence satisfaction levels, and the essential competencies for software projects or software development assignments.

The outcome of this dissertation is an in-depth analysis of SEC and frameworks for managing SEC. We identified 62 hard competencies, 63 soft competencies, and a combination of 25 essential SEC competencies. We propose three stakeholder satisfaction levels for SEC assessment: basic, performance, and delighter. The most significant contribution of our study is the holistic SEC framework for both software engineering research and practice. However, based on empirical observations, we also report 27 competencies not mentioned in the reviewed literature, 11 of which are considered essential competencies for software professionals.

Keywords: software engineering competence, essential competence, competence framework, competence model, Kano model, competence satisfaction levels, systematic mapping study

TIIVISTELMÄ (ABSTRACT IN FINNISH)

Assyne, Nana
Ohjelmistoammattilaisten olennaisten pätevyyksien määrittäminen yhtenäisen kehyksen avulla
University of Jyväskylä, 2021, 74 s.
(JYU Dissertations
ISSN 2489-9003; 467)
ISBN 978-951-39-8947-7

Ohjelmistoammattilaisten osaamista eli kompetensseja (Software Engineering Competence, SEC) on tarkasteltu ohjelmistotuotannon tutkimuksessa ja käytännössä vuosikymmeniä. Niiden ymmärtämiseksi ja hallitsemiseksi on kehitetty malleja ja viitekehyksiä sekä tunnistus- ja arviointikriteereitä. Vaikka alan tutkimus on ollut laajaa, eri sidosryhmillä (ohjelmistoammattilaiset, kouluttajat ja ohjelmistoteollisuus) ei kuitenkaan näytä olevan riittävästi keinoja tunnistaa ja arvioida ohjelmistokehitysprojekteissa tarvittavaa osaamista. Ennen kaikkea tarvittavat kompetenssit on ollut tapana kuvata pysyviksi, joten niiden muutosta ei kirjallisuudessa juurikaan käsitellä. Kokonaisvaltaista ohjelmistokehityksen osaamisen hallintamallia tai viitekehystä ei näytä tutkimuskirjallisuudesta löytyvän osaamisen, eri ohjelmistoprojekteissa tarvittavien olennaisten kompetenssien ja tyytyväisyystasojen tunnistamiseksi.

Väitöstutkimuksessa kartoitimme ensin kompetenssitutkimusta. Tämä toi esiin tutkimusaukkoja, joita täyttääksemme rakensimme asteittain malleja tai kehyksiä osaamisen hallintaan kirjallisuuden sekä sidosryhmien kuulemisen pohjalta. Tästä syntyi lopuksi kokonaisvaltainen ohjelmistokompetenssien kehys. Tutkimusaineistosta eli ohjelmistokehitystyön esimiestehtävissä olevien henkilöiden haastatteluista tunnistimme kokonaisvaltaista viitekehystä käyttäen eri kompetenssit ja ohjelmistoprojektien kannalta olennaiset kompetenssit sekä tyytyväisyystasot.

Tutkimuksen tuloksena saimme syvällisen analyysin kompetensseista sekä niiden hallinnan malleista ja viitekehyksistä. Lisäksi tunnistimme 62 ns. ”kovaa” kompetenssia (hard competencies) ja 63 ns. pehmeää kompetenssia (soft competencies) sekä 25 olennaisen kompetenssin yhdistelmän. Arviointia varten määritimme perus-, suoritus- ja ilahduttavuustason. Kokonaisvaltainen kehys on väitöstutkimuksen keskeisin tulos. Empiiristen havaintojen perusteella raportoimme myös 27 kompetenssia, joita tarkastelemamme kirjallisuus ei sisällä. Niistä 11 katsotaan ohjelmistoalan ammattilaisille välttämättömiksi kompetensseiksi.

Avainsanat: ohjelmistotekniikan osaaminen, olennainen osaaminen, osaamiskehys, osaamismalli, Kano-malli, tyytyväisyystasot, systemaattinen kartoitustutkimus

Author

Nana Assyne
Faculty of Information Technology
University of Jyväskylä
Finland
ORCID 0000-0003-0469-6642

Supervisors

Mirja Pulkkinen
Faculty of Information Technology
University of Jyväskylä
Finland

Hadi Ghanbari
School of Business
Aalto University
Finland

Reviewers

Petri Kettunen
Department of Computer Science
University of Helsinki
Finland

Eleni Berki
Software Engineering Education and Quality Management SEEQMA Ltd
London
UK

Opponent

Markku Oivo
Department of Information Processing Science
University of Oulu
Finland

ACKNOWLEDGEMENTS

Throughout my Ph.D. journey, there have been individuals and groups that have stood by my side in this journey. Indeed, how could I have achieved this if you were not there for me? You all did your best, and I say thank you from the bottom of my heart.

First of all, I would like to thank my supervisors, who through their diligent advice and support have made my journey successful. Special thanks to Dr. Timo Käkölä and the late Dr. Eetu Luoma for their supervisory role in the earlier part of my Ph.D. journey. I am indebted to you, Prof. Pekka Abrahamsson, for your guidance and generosity in providing the main data for this dissertation. My gratitude also goes to the students of the SE course at Norwegian University of Science and Technology. In all, this Ph.D. journey could not have been fruitful without the support of my main supervisors, Dr. Mirja Pulkkinen and Dr. Hadi Ghanbari, for their immense support and diligent guidance to me on this journey. To both of you, you have imparted so much to my life, and I will be indebted to you for the rest of my academic career. All I can say to both of you is Thank You!

I would like to thank Prof. Pasi Tyväinen, Prof. Tuure Tuunanen, Prof. Mikko Siponen, and the entire staff of the Faculty of Information Technology of the University of Jyväskylä. Your support made it happen. I cannot forget to mention Nina Pekkela and Marha-Leena Rantalainen. I must acknowledge the Ghana Institute of Management and Public Administration (GIMPA) for their support on this journey. With the mention of GIMPA, I take the opportunity to thank certain individuals: Prof. Frank Manu, Prof. Gamel Wiredu, Seth Twum, Mansah Preko, Faculty and staff of SOT-GIMPA, and the entire GIMPA community.

In all, there are certain individuals and groups that I cannot forget to thank. They include Offornze family, Emmanuel Owusu-Marfo, my brother from another mother, Justice Kweku Ackaah-Boafo, Victor Steward Sabuka, my boss Dr. Isaac Wiafe, my Jyväskylä school mates Dr. Jonas Kodwo Boateng, and Truth Lumor, Pro. Duku Osei, Prof. George Armah, Dr. Charles Kessey, my mentor Dr. Myriam Menezero, Dr. Richard A. Owusu, Joseph Osei-Amoah, Belayneh Bekele, Ethel Awoonor-Williams, and the entire Awoonor Williams family. I would like to acknowledge the contributions of Stein Rudvin, Geoff Utberg, Dr. Leah Riungu-Kalliosaari, Patrick Eshun, Dr. Perpetual Crentsil, and Adelaide Lönnberg, who read through my work for me. On groups, I cannot forget the support of my "A" and "O" level group, my church members—International Evangelical Church Finland, the Nzema group of Finland, Think Africa, and anyone who contributed to the success of this journey but whose name I could not mention here.

Finally, I would like to thank my girls, Mavis, Naa Aso, Mame Esane, and the entire Assyne family for your selfless support. I say "Meda bɛ ase". To crown it all, I would like to thank Almighty God for his blessings on me.

Espoo 15.11.2021
Nana Assyne

LIST OF INCLUDED ARTICLES

I Assyne, N., Ghanbari, H., & Pulkkinen, M. (2021). The state of research on software engineering competencies: A systematic mapping study. *Journal of Systems and Software* (Revised and resubmitted for review)

II Assyne, N. (2019). Hard competencies satisfaction levels for software engineers: a unified framework. In S. Hyrynsalmi, M. Suoranta, A. Nguyen-Duc, P. Tyrväinen, & P. Abrahamsson (Eds.), *ICSOB 2019: 10th International Conference of Software Business* (pp. 345–350). Springer. Lecture Notes in Business Information Processing, 370.

III Assyne, N. (2020). Soft competencies and satisfaction levels for software engineers: A unified framework. In D. Winkler, S. Biffl, D. Mendez, & J. Bergsmann (Eds.), *Software quality: Quality intelligence in software and systems engineering. Proceedings of the 12th International Conference, SWQD 2020, Vienna, Austria, January 14–17, 2020* (371, pp. 69–83). Springer. Lecture Notes in Business Information Processing. 371.

IV Assyne, N., Ghanbari, H., & Pulkkinen, M. (2021). The essential competencies of software professionals. A unified competence gate framework. *Information and Software Technology* (Submitted for review)

V Assyne, N. (2020). Towards a security competence of software developers: A literature review. In W. Yaokumah, M. Rajarajan, J.-D. Abdulai, I. Wiafe, & F. A. Katsriku (Eds.), *Modern theories and practices for cyber ethics and security compliance* (pp. 73–87). IGI Global.

LIST OF ACRONYMS

ACM	Association for Computing Machinery
AIS	Association of Information Systems
CBK	Common Body of Knowledge
CFSE	Competency Framework for Software Engineers
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronics Engineers
IT	Information Technology
LNBIP	Lecture Notes in Business Information Processing
SE	Software Engineering
SEC	Software Engineering Competence
SQL	Structured Query Language
SWEBOK	Software Engineering Body of Knowledge
SWECOM	Software Engineering Competency Model
SWQD	Software Quality Days Conference
UComGSP	Unified Competence Gate for Software Professionals
UFHCSL	Unified Framework of Hard Competency Satisfaction Levels
UFSCSL	Unified Framework of Soft Competency Satisfaction Levels

FIGURES

TABLES

CONTENTS

1 INTRODUCTION

The way we build software has changed drastically over the past three decades. However, challenges due to the complexity and size of software products and software environments continue to grow. Research has attempted to solve these challenges by studying both practical and academic implications in various areas of the software engineering field (Silveira Neto et al., 2013). For instance, in the area of software engineering competencies (SEC), Lenberg et al. (2015) corroborate this by suggesting that there is no lack of literature on SEC. However, according to Calazans et al. (2017) and Gimenes et al. (2012), the software industry is facing a significant shortage of skilled software professionals. Currently, some 23.9 million developers are employed worldwide, and this is expected to grow to 28.7 million by 2024 (Data, 2019).

To engineer software does not require complex machinery (Casale et al., 2016); rather, it requires the competence of the software professionals, making it the essential asset for software development. Despite this, IEEE (2014) points out that the development of the competencies has not kept pace with what the industry needs. To overcome this concern and fill the apparent gap, both practitioners and academics have been looking at ways to identify and train professionals in software engineering (Moreno et al., 2012). *Competence* is generally defined as "a collection of skills, abilities, and attitudes to solve a problem in a given context" (Holtkamp et al., 2015, p. 137). Software engineering competence (SEC) is defined by IEEE (2014) as *a set of skills, knowledge, and attitudes of software professionals to fulfill a task in software development projects*. This covers the entire development process (IEEE, 2014).

SEC has a rich body of literature (Lenberg et al., 2015) and it is a strategic research area in the software engineering discipline (Colomo-Palacios et al., 2013a). Preliminary literature review suggests that the focus areas of SEC research are: (1) competence identification and classification; (2) competence measurement and assessment; and (3) curriculum development (Acuña et al., 2006; Ardis et al., 2014; Hilburn et al., 2013; Hubwieser et al., 2013; IEEE, 2014; Kobata et al., 2015; Rivera-Ibarra et al., 2010; Sedelmaier & Landes, 2014a; Studt et al., 2015; T. Turley & Bieman, 1995). The above research areas are supported by both

scientific (such as above) and practitioner documents, such as the software assurance competency model.

From the viewpoint of both practitioners and academics, the management of competence of software professionals is necessary for successful software development (Colomo-Palacios et al., 2013a). However, to the best of our knowledge, a holistic model or framework for managing SEC that considers the key stakeholders and competence development is missing. Even more importantly, we were unable to find one that considers the current software development methodology, such as agile (Abrahamsson et al., 2002; Dyba & Dingsoyr, 2008; Kropp et al., 2016) and DevOps (Debois, 2011). In all, the SEC area lacks a holistic model or framework that could be used by different SEC stakeholders, including software professionals, educators, and the software industry, in determining the essential competencies for software development, customized according to the characteristics of a particular software project or software development assignment.

In view of this, this doctoral dissertation aims to investigate and provide an in-depth analysis of the SEC of software professionals for managing software development. The aim is further, to develop models or frameworks for identifying competencies, competence satisfaction levels, and the essential competencies required for software projects or software development assignments. We sought to achieve our aim by using findings from previous literature and by conducting an empirical study using qualitative data from the industry.

The results of this study not only provide a holistic framework for managing the SEC but also provide an in-depth analysis of the SEC research area (the current state of research on SEC). Further, we provide a practical illustration of how the developed frameworks can be used according to the characteristics of a particular software project or software development assignment. The results of this dissertation provide a means to manage SEC through the different viewpoints of stakeholders of competence development. Therefore, this dissertation contributes to both research and practice.

1.1 Background in the context of the software engineering bodies of knowledge

The competence of software professionals has over the years been a focus research area among academics and practitioner studies (Silveira Neto et al., 2013). This has led to the development of standard documents such as SE 2014: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering (IEEE-CS & ACM, 2015), Graduate Software Engineering 2009 (GSwE2009), Curriculum Guidelines for Graduate Degree Programs in Software Engineering (Pyster, 2009), Software Engineering Competency Model (SWECOM) (IEEE, 2014), Software Assurance Competency Model (Hilburn et al., 2013), and E-Competence Framework (CEN, 2014). Both the scientific literature and standard

documents, such as the aforementioned, have provided different ways of observing and organizing the understanding of SEC. Thus, there is a rich body of literature on SEC (Lenberg et al., 2015).

However, theory development is an area that requires more studies to help grow the software engineering field (Johnson et al., 2012; Johnson & Ekstedt, 2015; Päivärinta & Smolander, 2015). Theories help in explaining and predicting the phenomena of the discipline (Johnson & Ekstedt, 2015), particularly those that can be used to organize and observe the understanding of stakeholders involved in the development of SEC (Frezza et al., 2018). In answering the question "Why do we need one more professional competency model?," Mead and Shoemaker (2013) pointed out that "the answer lies in the significant difference between the competencies required to produce working code and those that are needed to produce software free from exploitable weaknesses. That difference is underscored by the presence of the adversary" (Mead & Shoemaker, 2013, p. 119). Hence, to help solve the differences, there is a need for models or frameworks, including holistic ones and those for specific areas of SEC.

A concern of the software industry is the development of the talents of human resources. This is because the quality and innovation of products and services produced by the industry are dependent on the knowledge, abilities, and skills of the software professionals (André et al., 2011; Rivera-Ibarra et al., 2010). As already stated, the development of software does not require complex machinery; rather, it requires the competence of the software professionals. However, the software industry is facing a significant shortage of skilled software professionals (Calazans et al., 2017). To identify and train such professionals to fill the gap, studies have proposed various curricula to support the training and development of skills, the identification and classification of SEC competencies, and measures to assess software professionals' competencies (Colomo-Palacios et al., 2013b; IEEE, 2014; Moreno et al., 2012; Pérez et al., 2017; Sedelmaier & Landes, 2014b).

Various attempts have been made to define the competencies needed by software professionals for software development (Humphrey, 1989; Mead & Shoemaker, 2013). Their success in doing so, however, is debatable (Mead & Shoemaker, 2013). For example, several works (Alavi et al., 2012b; Colomo-Palacios et al., 2010, 2013b; Moreno et al., 2012; Pérez et al., 2017; T. Turley & Bieman, 1995; Zendler et al., 2014) have defined, identified, and classified competencies for software engineering. In proposing a software engineering body of skill (SWEBOS), Sedelmaier and Landes (2014b) identified and structured competencies of software professionals into three categories: (1) comprehension of the complexity of software engineering processes, (2) awareness of problems and understanding of cause-effect relationships, and (3) team competency, including communication skills. There are also practitioner guide documents, such as SWECOM, which assesses SEC by considering skill area and work activity for each skill activity in an increasing level of five stages (IEEE, 2014) and the software assurance (SwA) competency model for assessing and providing assurance to software professionals. SwA has five competence levels (Hilburn et al., 2013).

The people capability maturity model (People CMM) is a workforce practice guide to continuously improve the capability of the organizational workforce. It has five maturity levels (the initial level, the managed level, the defined level, the predictable level, and the optimizing level) (Curtis et al., 2009). The European e-competence framework (e-CF) aims at standardizing ICT professionals' competencies within the European Union. It has 40 reference competencies and 5 e-CF areas (CEN, 2014). The Essence kernel by Object Management Group, Inc (OMG) (Object Management Group, 2018) focuses on providing a common basis for defining the software development practices, which are organized using three areas: alphas, activity spaces, and competencies. Each of these organized areas is further examined using three discrete areas: customer, solution, and endeavor. The competencies subset of the essence kernel assesses the capabilities required to conduct the work of software engineering. The kernel competencies are further subdivided using the three discrete areas into stakeholder representation, analysis, development, testing, leadership, and management as competency areas for competency management. Each competency area has five levels by which teams can assess the competencies (Object Management Group, 2018).

The models or frameworks mentioned above have attempted to consolidate their assessments into five levels, perhaps because they take more fine-grained approaches suitable for education and related assessments. Thus, the competencies of the software professional have not kept pace with what the industry requires (IEEE, 2014). Some studies in the SEC area suggest that there is a gap between the competencies needed by the industry and what the educational institutions produce (Colomo-Palacios et al., 2013b; Radermacher et al., 2014; Sedelmaier & Landes, 2014b). It has been established that the software industry faces a shortage of skilled software professionals. Although there are scientific studies (e.g., Ardis et al., 2014; Kobata et al., 2015; Pawlowski & Holtkamp, 2012) and practitioner documents (e.g. IEEE-CS & ACM, 2015; Pyster, 2009) for training software professionals, the gap remains between what educational institutions produce and what the industry requires.

Since our main audience are the stakeholders involved in staffing development projects, or teams and recruiting SE professionals, this dissertation develops a new framework that departs from the frameworks that assess the gradual development of skills, abilities, and knowledge of an individual in their journey to becoming a professional or a more proficient professional.

1.2 Research objectives

There have been studies that have examined the models or frameworks for organizing and observing SEC. The literature on software engineering competence models or frameworks is not necessarily lacking. For example, Acuña and Juristo (2004), Acuña et al. (2006), Bröker (2014), Rivera-Ibarra et al. (2010), Thurner et al. (2016), and IEEE (2014) have studied and created models or frameworks for organizing and observing the SEC. However, their focus was on the identification,

assessment, and classification of SEC in isolation of the different stakeholders of SEC development. Manawadu et al. (2015) and Turley (1991) indicated the existence of certain competencies of software professionals that are essential for software development. However, a comprehensive study on the essential competencies of software professionals dated back to 1994, which is the work of Turley and Bieman (1994).

Previous research work on SEC has addressed competence assessment levels of software professionals; thus, we know of base competencies (e.g., Thurner et al., 2016), essential technical competencies (e.g., Broadbent et al., 1992; Colomo-palacios et al., 2013; Moreno et al., 2012), and models for identifying and classifying SEC (e.g., Pérez et al., 2017; Rivera-Ibarra et al., 2010). However, the assessment levels of these models did not consider the performance levels associated with the competence. Therefore, there are no measures to determine the satisfaction levels to assure the stakeholders of SEC. Thus, there is a need for an in-depth analysis of SEC that encompasses the identification, assessment, and essential competencies of software professionals. It is clear that understanding the competences of software professionals is essential for software development. (Alavi et al., 2012a; Colomo-Palacios et al., 2013a; Goel, 2006; Manawadu et al., 2015; Orsoni & Colaco, 2013; Robinson et al., 2005; Saldaña-Ramos et al., 2012).

Thus, this dissertation, through an in-depth analysis, attempts to understand the software engineering research area and to develop a framework for managing the competencies of software professionals for software development. The results of this study not only show models and frameworks for identifying and assessing SEC but also extend the use of the models to identifying and assessing the competencies of software professionals and the essential competencies for software development. The study exceptionally pays attention to the stakeholders of the SEC. Furthermore, the results show the practical determination of the assessment levels (the satisfaction levels) according to different software projects or software development assignments.

The creation of good methods and tools has never been sufficient for software development. For this reason, the strategic use of people competencies is inevitable (Casale et al., 2016). To function effectively and productively in this ever-evolving environment, there is a need to develop strategic competencies, especially the employment of human resources with requisite competencies to use the methods and tools (Rivera-Ibarra et al., 2010). As pointed out by Acuña and Juristo (2004), we risk developing tools and methods that are beyond the capabilities of the people, if their competencies are not known and developed. Therefore, we used a comprehensive literature review, expert consultation, and interview data to find solutions to the identified gaps.

A model that can help explain and predict the SEC needed for the development of software must consider the stakeholders involved in the development of the SEC. Such model(s) must also consider the dynamic nature of different unique projects or assignments and the current development methods, such as agile (Abrahamsson et al., 2002; Dyba & Dingsoyr, 2008; Kropp et al., 2016) and DevOps (Debois, 2011). The models must consider the logic and rationale of

using competencies in the development of software and examine the behavior of software professionals in the role of software development.

In this research, using a literature review, expert consultations, and interview data, we constructed a theoretical model for explaining and predicting, organizing, and observing SEC. By building the model, we aim to provide a measure that explains SEC for the use of both research and practice. The main research question (RQ) that guided this effort is as follows:

RQ How do the essential competencies of software professionals evolve over time?

The following sub-questions were addressed in targeting the main RQ:

RQ1 What is the state of research related to software engineering competencies and their evolution?
RQ2 What are the different satisfaction levels of software professional's competencies?
RQ3 What are the different competencies of software engineering roles?
RQ4 What are the essential competencies of software professionals?

We approached this study in stages using theories and empirical evidence, as discussed in the next chapter.

To contextualize our work, we provide definitions of some key terms used in this dissertation. According to Frezza et al. (2018), stakeholders of SEC development may include "educators, students, industry, and other employers of computer graduates, policymakers, professional societies, etc." Thus, for our study, we simplify them to include **software professionals** (i.e., individuals who hold software engineering competencies), **educators** (i.e., institutions that provide software engineering education to software professionals and communities of practice within the software engineering field), and **software industry** (i.e., entities who utilize the competencies held by the software professionals for-profit or for non-profit purposes). The term software industry is sometimes complex to define due to the nature and how we use and develop software. The work of Tyrväinen et al. (2008) makes a distinction between the software industry business as primary software industry and secondary software industry. The primary software industry develops software as its core activity and may include some auxiliary activities. The secondary software industry is hosted by companies focusing on another type of business but developing software as part of the development of their processes, products, or services. Since both of these software organization types employ software professionals, our usage of the term 'software industry' encompasses both the primary and the secondary software industry types. We also use the phrase holistic **framework**. According to previous studies, competence frameworks or models are for identification and assessment. Thus, we define our holistic framework as one that can be used by the key stakeholders of SEC to identify competencies, assess competence satisfaction levels, and identify the essential competencies for software development.

The structure of this dissertation is as follows. Chapter 1 describes the motivation for the study, provides background information in the context of the

software engineering body of knowledge, and states the objectives of the study. Chapter 2 gives an overview of the dissertation, a summary of the five articles of the dissertation, the publication status of the articles, and the contributions of the coauthors of the articles. Chapter 3 presents the scope of the research and the research approach adopted for the individual articles. Chapter 4 presents the theoretical foundation for the dissertation by examining concepts such as competence, software roles in software engineering, competency framework for software engineering, the Kano model, and the UComGSP and how it can be used. Finally, Chapter 5 presents the contributions limitations and future research topics.

2 OVERVIEW OF CHAPTERS

Given the importance of competencies of software professionals to the development of software, Barreto et al.'s (2008) emphasis that software development is human-intensive, and the significant shortage of skilled software professionals in the software industry, we consider it prudent to examine the current state of the literature on SEC. This enabled us to identify and investigate the gaps in the SEC field. In this chapter, we review our studies, Articles I, II, III, IV, and V.

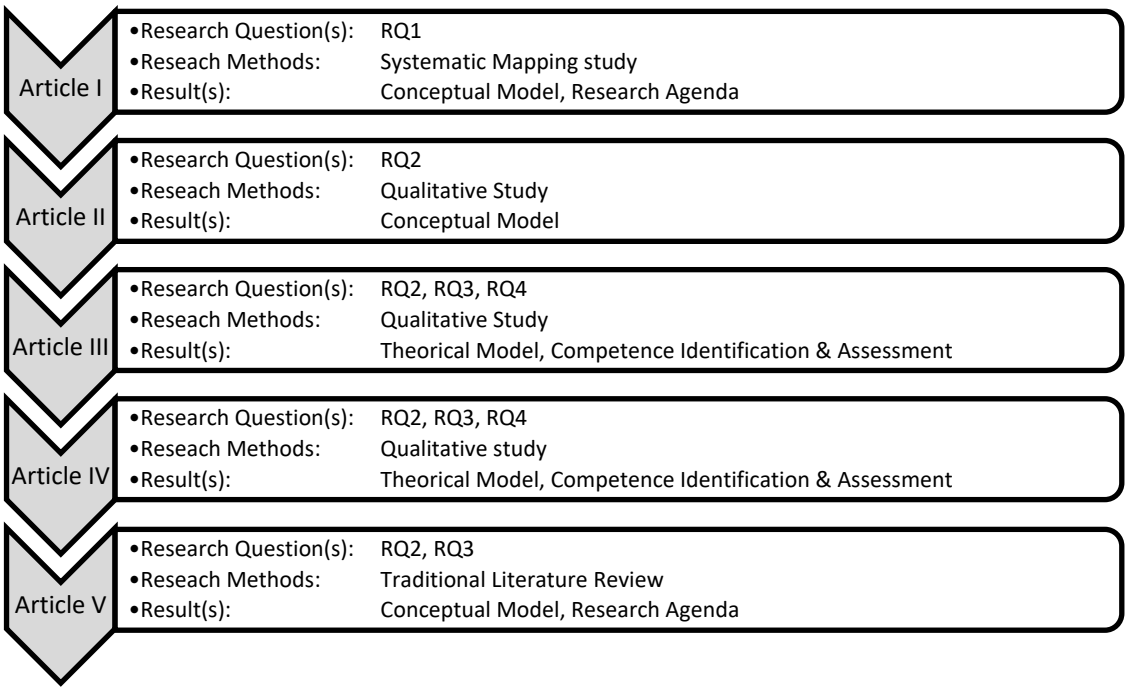


FIGURE 1. Overview of the studies in this dissertation

2.1 Article I – The state of research on software engineering competencies: A systematic mapping study

Research objectives

Over the past decades, significant studies have been conducted on the SEC. Software development is considered a human-intensive field. Colomo-Palacios et al. (2013) stated that the SEC research area is a strategic research area for software engineering. Lenberg et al. (2015) established that there is no lack of literature on SEC. Notwithstanding, the industry is facing a significant shortage of skilled developers. Acuña and Juristo (2004) argued that we risk developing tools and methods that are beyond the capabilities of the people if their competencies are not known and developed (Acuña & Juristo, 2004). Educating and training software professionals to acquire the requisite skills for software development is never an easy task; as Sedelmaier and Landes (2013) pointed out, there are no cookbooks for this task.

However, it is worth mentioning that software professionals are the key drivers for software development. Ignoring their development (competencies) is invariably an oversight of robustness and innovation in software development. Therefore, it is important to know and understand the state of affairs regarding software professionals' competencies. We define SEC as the knowledge, skills, and attitudes of software professionals to fulfill a task in a software development project (IEEE, 2014).

The main objective of this study was to understand the current literature on SEC. To this end, a mapping study was conducted using the guidelines of Petersen et al. (2008) and Petersen et al. (2015) to provide a comprehensive overview of the SEC research area.

Research results

Through an extensive search of previous studies and rigorous inclusion and exclusion processes, we identified 60 relevant primary studies for the review study. By analyzing these primary studies, we provide an overview of the current state of research on SEC, with a particular focus on common SEC research areas, available SEC models and frameworks, and the essential competencies of a software professional.

Our results indicated that despite a rich body of literature, several areas of SEC need further scientific investigation. Future studies are needed to propose better models and frameworks for providing theoretical accounts as well as practical implications on different aspects of SEC, especially assessing the satisfaction levels of SEC stakeholders. More empirical research is also needed to provide a better understanding of how the competencies of software professionals change over time or as they move from one role to another. Lastly, further research is needed to assess and provide a fresh account of the essential competencies of future software professionals, especially concerning modern development

methods and techniques, such as agile methods (Abrahamsson et al., 2002) and DevOps (Debois, 2011).

The results also showed two main research areas (personnel and organizational research areas) and six subcategories. The personnel competence research area focuses on software professional competencies, that is, the skills, abilities, and attitudes required for developing software products or services. These are the catalysts for developing a software product or service, and they include the soft and the hard competence areas. Examples are creative thinking and programming skills. The organizational competence research focuses on tools and instruments, such as assessment and identification models and frameworks, which are used for organizing, assessing, measuring, and managing personnel competencies.

The study identified 14 different models or frameworks that enable stakeholders to understand the underlying logic of the SEC in the context of software development and on which further SEC research is scaffolded. In the area of personnel competence research, three of the models and frameworks are for the generic identification of competencies of software professionals, one is for defining the roles and competencies of software testers specifically, and one is for assessing the competencies of software professionals. The models and frameworks in the organizational research area are for (1) managing competence research and learning, (2) the competence process model is for design, development, and implementation of software, (3) human resource management, (4) competence evolution identification and competence stakeholder identification.

According to Turley and Bieman (1995), essential competencies of software engineering are the skills, knowledge, and attitudes of software professionals necessary for excellent performance in a software project or software development assignment. Forty-nine essential competencies were identified in nine primary studies. The coded items identified in the primary studies were classified, and 11 themes emerged from those essential competencies. The themes were mapped to the top-level themes of Rivera-Ibarra et al.'s (2010) framework for identifying competencies. They are technical knowledge or skills (referred to as the essential hard competencies, and defined as task-oriented competencies), social knowledge or skills (competencies for organizing cooperation and interpersonal relations in a software development project), and personal traits or skills (personal attributes for working well in different spheres of life).

The results also showed changes in SEC research over the past two and a half decades. The first trend that we observed from the primary studies was an increase in the total number of primary studies that have used quantitative research methods since 2011. Regarding the contribution types, it seems that proposing a set of lessons learned continues to be the focus of SEC research, as the number of primary studies that have this type of contribution remains the highest over time. Regarding research areas, however, it seems that since 2012, soft competencies have been receiving some attention from SEC research. Another trend we observed in the essential competence studies is the number of competencies identified per study; that is, fewer essential competencies are being identified in

recent studies. Thus, this highlights a need for a fresh understanding of the essential competencies, which especially considers the current software development trends, such as using agile methods and DevOps.

Summary

In this study, the authors strived to improve SEC research and practice by providing up-to-date information on software development methods to sensitize the key players in the field. The goals of this study were two-fold. The initial step was to obtain a comprehensive overview of the current state of research on SEC, and the next phase involved identifying potential gaps in SEC research to guide future studies. By analyzing these studies, we identified two main SEC research areas: personnel and organizational. We also identified and presented a set of SEC models and frameworks that could be used by SEC research and practice.

Furthermore, we identified a set of essential competencies of software professionals, most of which deal with their social and personality skills and competencies (i.e., soft skills). Based on this observation, we argue that separating soft and hard competencies may soon be a concept of the past, and future research and practice should consider them as two equally critical pillars of software engineering competencies. Our findings show, among other things, that the human-intensive nature of software development requires further attention from both research and practice. Therefore, we argue that the development of the SEC cannot be conducted in isolation but must consider the viewpoints of different SEC stakeholders, including software professionals, educators, and the software industry. Furthermore, future research should seek to identify and provide a better understanding of the essential software engineering competencies that contribute to developing high-quality software products and systems in modern societies.

2.2 Article II - Hard competencies satisfaction levels for software engineers: A unified framework

Research objectives

Previous studies have suggested that software development is a human-intensive field. Software development requires a combination of soft and hard competencies for successful development (Moreno et al., 2012; Sedelmaier & Landes, 2014b). The research area of software engineering competence has become a strategic research area for academicians in software engineering. However, understanding any phenomenon, such as SEC, requires structures such as models and frameworks. Bhattacherjee defined a model or framework as tools required to classify or organize an observation for a general understanding of the phenomenon (Bhattacherjee, 2012). However, previous studies have examined the availability of models or frameworks for managing the SEC, neglecting to consider all the key stakeholders of SEC as part of an overall puzzle. These stakeholders include the educators, the software industry, and the software professionals

(Frezza et al., 2018); not considering them under one framework limits our understanding of the phenomena with SEC. Consequently, this study addressed this apparent gap.

Most previous studies on SEC have split the categorization of competence study into hard or technical and soft or behavioral competence categorization. Thus, the authors' initial step was to utilize this categorization. This study focused on the hard/technical competencies required for software engineering. Hard or technical competence is defined as the technical skills required to perform a given software development task. The authors set out to understand (1) the models and frameworks available for organizing and observing hard competencies and (2) how best to utilize available models to determine the satisfaction level of a competence. Thurner et al. (2014) defined basic competencies as those that are crucial for studying software engineering. Previous SEC studies have indicated competencies that are essential for software development (e.g., Alavi et al., 2012; Colomo-Palacios et al., 2013; Goel, 2006; Manawadu et al. 2015; A. Orsoni & Colaco, 2013; Robinson et al. 2005, Saldaña-Ramos et al. 2012l; Turley & Bieman, 1995) . However, we did not find any study that examined these different types of competencies from the perspective of the different stakeholders of SEC development or, more importantly, for accessing the competence satisfaction levels that will provide assurance to the stakeholders. Against this background, we sought to develop a model that will consider the key stakeholders (software professionals, educators, and software industry) of competence studies as suggested by Frezza et al. (2018) and IEEE (2014), and that will be capable of determining the satisfaction levels of competence while also determining competencies essential to software engineering.

We identified the following model and framework: (1) the Kano model determines the satisfaction of a customer related to product development (Kano et al., 1984) and (2) the competency framework for software engineers (Rivera-Ibarra et al., 2010), which considers the roles of software engineering, focusing on technical competencies, as was the goal of this study.

Research findings

We identified that there was no model or framework for determining the essential competencies of software engineering without resorting to a typical academic exercise. Article I also revealed that the model or framework does not consider the key stakeholders of competence development.

Using existing models from the SEC research area and other research areas, we developed a unified framework of hard competency satisfaction levels for software engineers (UFHCSL) by employing previous literature and focus-group discussion. UFHCSL enables the determination of satisfaction levels of hard competencies and the essential hard competencies for software engineering. The satisfaction levels determined by UFHCSL are basic competency, performance competence, and delighter competence. The results revealed the three-satisfaction level in project management roles, requirement analysis role, software design role, programming role, validation and verification role, configuration management role, test and quality role, documentation role, and maintenance role. The

UFHCSL also allows the determination of the essential hard competencies of software professionals. Lastly, the framework considers the key stakeholders of SEC: software professionals, educators, and the software industry.

Summary

The resulting competence framework known as the UFHCSL can be utilized by software professionals, educators, and the software industry to determine the satisfaction derived from a competence. The model can be employed by practitioners and academics. Thus, this research contributes to the SEC field by developing a framework for determining the satisfaction levels of hard competencies and the essential competencies for software development.

2.3 Article III - Soft competencies and satisfaction levels for software engineers: A unified framework

Research Objectives

The competencies of professionals are the driving force of software development. In human resources studies, they are mainly classified as hard and soft skills. Previous studies have focused on the hard skills of developers (Lenberg et al., 2015). However, Article I found that this is changing and that the focus of SEC research is drifting toward the study of soft skills. Harris and Rogers defined soft skills or competencies as "work ethics, positive attitude, social grace, facility with language, friendliness, integrity and the willingness to learn" (Harris & Rogers, 2008).

Previous studies have focused on identifying these skills without considering the assurance that the competencies may give to the software industry or software professional. In short, the benefits that can be derived from using a soft competence are not known beforehand. Some studies have mentioned the base competencies of software students. This is defined by Thurner et al. (2014) as the prerequisite competencies needed by software engineering students to acquire technical competencies. However, they are not the only satisfaction level or category in SEC. In this study, we argue for the existence of other levels of satisfaction, hence the aim of Article II and Article III.

Furthermore, Article I also found that models and frameworks for understanding SEC lack the viewpoints of all the stakeholders involved in SEC. Thus, this study focused on soft competencies of software engineering by addressing the following: (1) assessing existing models or frameworks for organizing and observing the understanding of SEC, (2) developing a model/framework for determining the satisfaction levels of soft competencies, (3) developing a model/framework for the determination of the essential soft competencies for software engineering, and (4) using the model/framework to validate a dataset to produce competency satisfaction levels and essential competencies for software engineering. To do this, we selected some existing models, and through expert discussions using an iterative approach, we developed the unified

framework of soft competence satisfaction levels for software engineers (UFSCSL). The model was later used to identify the satisfaction levels of SEC. Basic, performance, and delighter competencies satisfaction were identified for software engineering. The paper also identified the essential competencies for software engineering.

Research findings

The results of Article III, which complement Article II, support the assertion that soft and hard competencies are the main driving force of software development. Combining the two means successful and robust software development. The result shows the UFSCSL for determining satisfaction levels for soft competencies for software engineering and the essential soft competencies for software engineering. Thus, this study provided a framework that determines the satisfaction levels and essential soft competencies for software development.

Our results showed three types of competence satisfaction levels in software engineering: basic, performance, and delighter. By using the main actors of competence development, we determined the competencies using the following categories of personal and social competencies: personal category—development on the job, personal development, and rights and limits; social categories—interpersonal relations, cooperation, and teamwork; and handling and solving conflicts. Thus, we provided a second-level granularity of the soft competencies of essential competencies for software engineering.

Summary

In Article III, based on the data collected from expert discussions and supervisors in software development, we developed a framework that can be used to identify satisfaction levels for soft competencies and further identify the essential soft competencies for software engineering. The framework (UFSCSL) is capable of producing outcomes useful for software professionals, educators, and the software industry.

2.4 Article IV – The essential competencies of software professionals: A unified competence gate framework

Research objectives

In Article IV, using an extensive literature review, focus group discussions, and empirical evaluation, we developed a framework called the Competence Gate for Software Professionals (UComGSP). UComGSP can be used to identify and manage SEC. This study, which combined Articles II and III, examined the future of SEC. The framework developed was based on the Kano model (Kano et al., 1984) and the competency framework for software engineers (Rivera-Ibarra et al., 2010). Based on Article I, which argued that separating soft and hard competencies may soon be a concept of the past, and that future research and practice should consider them as two equally critical pillars of software engineering competence

studies, UComGSP can be used to identify and assess SEC based on different software projects or software development assignments. Thus, this study contributes to SEC research and practice.

Even though previous studies have established that essential competencies are important for software development, a model for the determination according to different projects has yet to be found in the literature. André et al. (2011) suggested that, in most software projects or software development assignments, people are assigned to roles and teams based on the experience of the project or team leader. Turley and Bieman (1995) argued for the identification of exceptional competencies to enhance software development. Frezza et al. (2018) and IEEE (2014) pointed out that the development of competence requires different stakeholders. Frezza et al. (2018) listed the following as those involved in competence development (stakeholders of competence development): educators, students, industry and other employers of computing graduates, policymakers, and other professional societies. Article I established that future research would need to identify and provide a better understanding of the essential software engineering competencies contributing to developing high-quality software products and systems in modern societies. Thus, this study aimed to fill these gaps by developing a holistic framework for determining and identifying competencies for both academic and practitioner use, a competence satisfaction level that serves as an assurance to stakeholders, and the essential competencies of software professionals that can vary according to a different software project or software development assignment.

Research findings

We have provided a framework for analyzing competence models or frameworks on SEC. The analysis of the competence models or frameworks should be considered as a steppingstone to developing a holistic model for SEC. In this analysis, we provided some variables for analyzing competence models or frameworks involving stakeholders in SEC and identifying the essential competence for software engineering. Thus, we have provided a framework and a tool for assessing competencies that will support the strategic nature of SEC research.

The results presented individual competencies and their satisfaction levels: basic, performance, and delighter. The roles as stated in the competency framework for software engineers (CFSE) are project management, requirement analysis, software design, programming, validation and verification tests, configuration management, tests and quality engineering, documentation, and maintenance. The results also show the essential competencies of software engineering. A key competence that was highlighted in Article I is the shift in the development of the agile methodology as a competence for software professionals. It is important to note that agile competence was identified as a basic competence. That is, it is a prerequisite competence that is necessary and expected from software professionals. Therefore, we must pay attention to agile methodology as a competence in any curriculum development in software engineering.

The study resulted in the development of a holistic framework for identifying and assessing competencies, revealing 63 soft competencies and 62 hard

competencies mapped to the roles of software engineering, 3 satisfaction levels (basic, performance, and delighter) and their definitions, and 25 identified essential competencies of software professionals. We also report 27 competencies not mentioned in the reviewed literature; 11 of them are considered essential competencies for software professionals. We have also provided a working definition for the essential competence as skills, knowledge, and attitudes of software professionals necessary for excellent performance (a desirable outcome to the project owners) in a software project or software development assignment. Furthermore, the study provided an analysis of the models and frameworks of SEC, which can be used as a starting point for research on SEC.

Summary

This study aimed to provide a holistic framework enabling SEC stakeholders to (1) identify SE competencies, (2) identify the essential SEC, and (3) assess the satisfaction levels derived from those competencies. The study achieved its aim by developing a holistic framework for managing SEC. This holistic framework (UComGSP) can be used by the key stakeholders of SEC for developing SEC.

2.5 Article V - Towards a security competency of software developers': A literature review

Research Objectives

Article IV used the UComGSP to identify SEC for software development. From the identified SEC, some new competencies were observed. As stated in Article IV, those new observations do not mean new competencies. Thus, Article IV called for more investigations to elucidate and expand on those competencies. For this reason, Article V used a traditional literature review to identify the security competencies of software developers and set an agenda for the future direction of research on these competencies.

The ubiquitous nature of computing adds complexity to software development. Software development is human-intensive. However, previous studies suggest that the security competence of software developers has been treated as a subsidiary of security engineer's rather than software engineer's competence, thus limiting our understanding of how to improve software developers' knowledge of software security skills. Security competence of software developers is essential in software development, because security matters must be addressed right from the start of the software development process (Mano et al., 2006). However, the security competence of software developers has not been adequately addressed in previous studies.

In advocating for security engineering environment studies for software developers, Cheng et al. (2008) pointed out that there is an urgent need to create an environment that integrates various tools and provides comprehensive facilities to the designers, developers, users, and maintainers of a software system (Cheng et al., 2008). Yet, the skills needed for such development are not well known or

structured in previous studies. Therefore, there is a need to examine the security skills of the developers; hence, the purpose of Article V. To develop SEC, Article I suggests the need to consider the competencies of developers vis-à-vis the roles and duties of the developer. As a first step, we set up an agenda for assessing the security competencies of software developers.

Research findings

A traditional literature review was chosen as the method for data collection for this study. In the review study, 13 security competencies were identified after the analysis. They were classified into two groups: programming-related competencies and non-programming-related competencies. In the area of programming-related competencies, the following were identified: secure programming or coding skills, secure mobile software development skills, secure socket layer skills, web application security skills, integrated development environment (IDE) security skills, code analysis tool skills, modeling SQL injection skills, handling buffer overflow skills, and security pattern skills. In the area of non-programming-related competencies, the following were identified: software security policy skills, security best practice and standard skills, system security assurance tool skills, and vulnerability assessment tool skills. The study provided a framework for understanding the security competencies of software developers by mapping the identified competencies to the common body of knowledge (CBK) framework of information security professionals' skills. The study also sets out the implications of not having these competencies.

Summary

We identified 13 security competencies of software developers from the literature, using a traditional literature review. The competencies were grouped into two categories: programming-related skills and non-programming-related skills. Nine competencies were programming related and four were non-programming related. To create a framework for help with future studies, we mapped the identified competencies to the CBK framework of information security professionals' skills. Seven of the competencies were mapped to both information communication technology and security criteria, and four to information communication technology. The study set an agenda for the future direction of research on the security competencies of software developers.

2.6 Publication status

Given the importance of the research topic and the fact that the main driving force of software development is software professionals (Casale et al., 2016), we hope our findings will receive consideration from the software development community for practice and research. Thus, we have prepared several scientific papers and submitted them to different SE outlets. As part of this dissertation, five papers were prepared. Two peer-review conference papers and a book

chapter have been published and two journal papers have been submitted or resubmitted for review (see Table 1).

Article I, which is the justification of this study, is a literature review. It has passed the second round of peer review and received a "revise and submit as new" request from the Journal of Systems and Software, a highly regarded peer-review journal published by Elsevier Inc. Articles II and III were published separately in annual peer-review conferences: the 10th International Conference on Software Business (ICSOB) and the 12th International Conference on Software Quality (SWQD). The papers were presented at these conferences and used as a data source for Article IV.

Article IV has been submitted to Information and Software Technology, one of the leading journals in the SE discipline published by the Association for Computing Machinery. Finally, Article V, which is a book chapter, is a study on one of the newly observed competencies from Article IV. The book, Modern Theories and Practices for Cyber Ethics and Security Compliance, is published by IGI Global. Research contributions of the coauthored papers are given in Table 2.

TABLE 1. Publication plan

Article	**Author(s)**	**Title**	**Forum**	**Status**
Article I	Assyne, Ghanbari, & Pulkkinen	The State of Research on Software Engineering Competencies: A Systematic Mapping Study	Journal of Systems and Software	Revised and resubmitted for review
Article II	Assyne	Competencies and Satisfaction Levels for Software Engineers: Unified Framework	10th International Conference, ICSOB 2019 – 370 LNBIP	Published
Article III	Assyne	Soft Competencies and Satisfaction Levels for Software Engineers: Unified Framework	12th International Conference, SWQD 2020 – 371 LNBIP	Published
Article IV	Assyne, Ghanbari, & Pulkkinen	The Essential Competencies of Software Professionals. A Unified Competence Gate Framework	Information and Software Technology	Submitted for review
Article V	Assyne	Towards a Security Competence of Software Developers: A Literature Review	2020 IGI Global	Published

TABLE 2. Contributions of the co-authored papers

Article	Title	Author	Contributions
Article I	The State of Research on Software Engineering Competencies: A Systematic Mapping Study	Nana Assyne	Conceptualization, Methodology, Formal analysis, Data Curation, Writing - Original Draft, Writing- Reviewing and Editing, Visualization
		Hadi Ghanbari	Conceptualization, Methodology, Writing - Original Draft, Writing- Reviewing and Editing, Supervision
		Mirja Pulkkinen	Methodology, Writing - Original Draft, Writing- Reviewing and Editing, Supervision.
Article IV	The Essential Competencies of Software Professionals. A Unified Competence Gate Framework	Nana Assyne	Conceptualization, Methodology, Formal analysis, Data Curation, Writing - Original Draft, Writing- Reviewing and Editing, Visualization
		Hadi Ghanbari	Conceptualization, Methodology, Writing - Original Draft, Writing- Reviewing and Editing, Supervision
		Mirja Pulkkinen	Methodology, Writing - Original Draft, Writing- Reviewing and Editing, Supervision.

3 RESEARCH APPROACH

This doctoral research seeks to understand the SEC area by providing an in-depth analysis of essential competencies for software engineering. By this, we aim to build a framework that is empirically grounded and can be used by both academics and practitioners. Software development is complex and requires dynamism in its processes. As already stated, software development is human-intensive, and software professionals have a direct influence on the quality of the software they develop. A study in such an area requires an approach that enables us to understand human behavior. Wohlin and Aurum (2015) posited that an individual's behavior is influenced by the meanings endowed to an event. Thus, the selected approach must support our understanding of the people and the environment in which the research is being conducted.

3.1 Critical realism

Critical realism as a research paradigm overcomes an odd dualism such as objectivism and subjectivism (Bhasskar, 2008; Vincent & O'Mahoney, 2018). It provides a means to distinguish between what is real and what we know (Vincent & O'Mahoney, 2018). In short, critical realists see the world through objectivism while recognizing the fact that knowledge is subjective (Bhasskar, 2008; Vincent & O'Mahoney, 2018). As suggested by Vincent and O'Mahoney (2018), crucial to understanding the world well, good research requires a bridge between ontology and epistemology. It also provides a means to use theories to analyze and evaluate data that suggest appropriate changes to a problem (Vincent & O'Mahoney, 2018). By contrast, interpretive methods, such as action research and ethnography, are mainly for building theories compared to testing theories (Bhattacherjee, 2012).

Researchers are influenced by their beliefs and assumptions. The most common sets of beliefs and assumptions are positivist, interpretivist, or critical realism. Positivism is the scientific study of the social world. It is applied to separate

facts and values to express the cause of something, or for an explanation of universal laws. Those with this belief are of the view that reality is objectively given and can be measured by its properties (Myers, 1997). Interpretivists assume that the world can be seen or interpreted with an agreed language with meaning (Myers, 1997), and designers deal with the creation and evaluation of technology artifact (Cole et al., 2005). Between these two assumptions is critical realism. With this in mind, my personal belief that guided the conduct of this Ph.D. study is critical realism (Bhasskar, 2008). We show in the methodology section how we used theories in the analysis and evaluation of the data to gain the results, that is, the outcomes of this dissertation.

3.2 Methodology

Methods are processes or techniques used by researchers to empirically validate a phenomenon of interest or under investigation (Kaplan & Maxwell, 1994; Nandhakumar & Jones, 1997; Orlikowski & Baroudi, 1991). Such techniques or processes are used to gather and analyze data (empirical evidence) to discover a new understanding of a phenomenon. The research method is described as "the procedures and techniques used to compile systematic observations and to make sense of those observations in the generation and examination of ideas and theories"(B. Lee & Cassell, 2013, p. 123). There are different types of research methods available to researchers. However, the selection of a method by a researcher is dependent on the phenomenon to be investigated, and more importantly, the belief and the assumption of the researcher (Orlikowski & Baroudi, 1991). Thus, a qualitative research approach (Myers & Newman, 2007; Schultze & Avital, 2011; Venkatesh et al., 2003) is suitable for the studies proposed in this dissertation. That is, our phenomenon is related to humans, thus, a natural context. Therefore, different techniques were used to gather data for the investigation. They include a literature review, stakeholders' consultations, and interviews. Figure 2 shows how the different techniques were combined to validate SEC.

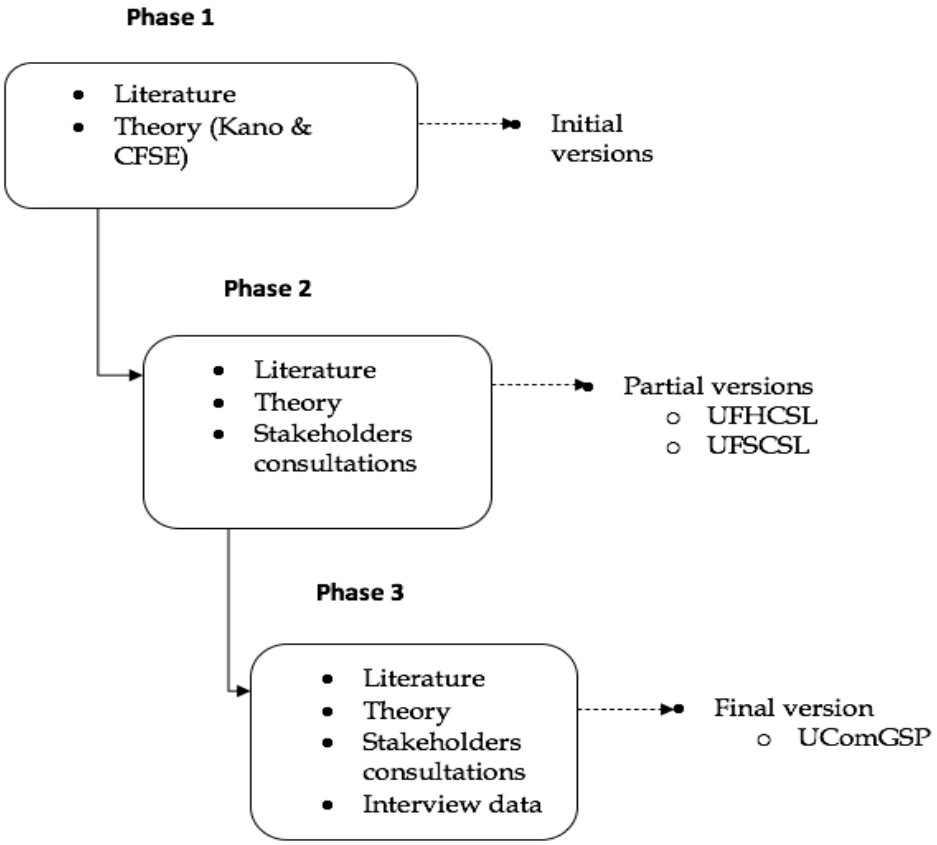


FIGURE 2. Research development approach

3.2.1 Literature review

Following the initial traditional literature of understanding of the research topic area, a systematic literature review was conducted to justify the study and to identify a clear gap for the dissertation. As already stated, literature on SEC is not lacking; therefore, to gain a good understanding of the research area, a systematic mapping study was conducted based on the guidelines of Kitchenham and Charters (2007) and Petersen et al. (2008, 2015). A mapping study provides an overview of a research area by identifying the quantity, type of research, and results within the area (Petersen et al., 2008). It was apparent that there were gaps in the SEC literature, such as the lack of a holistic model or framework for managing SEC and the lack of understanding of the essential competencies that contribute to the development of high-quality software product systems in modern societies. Thus, in phase 1, the study chose to collect data by analyzing a range of previous SEC studies to initiate the filing of the gaps identified (Figure 2).

To ensure a wider coverage of the extant literature, we searched several databases, including IEEE Xplore, ACM Digital Library, Scopus, AIS eLibrary, and Science Direct. After selecting these databases, search strings were developed. In the literature review for Article I, a total of 12,250 potentially relevant papers were retrieved from all five databases. After applying the inclusion/exclusion

plus quality assessment, 60 papers qualified for the review. Data analysis was performed on the extracted primary studies to answer the research questions.

To analyze the extracted data, we employed content analysis (Vaismoradi et al., 2013). According to Vaismoradi et al. (2013), content analysis is well suited for analyzing multifaceted data by labeling relevant items (coding) in the text and interpreting the content. The coding procedure also assists in quantifying the qualitative analysis results, for example, counting the frequency of occurrence, which can indicate the significance of an issue. Content analysis is defined as the use of a "systematic coding and categorizing approach used for exploring large amounts of textural information unobtrusively" (Vaismoradi et al., 2013, p. 400). Hsieh and Shannon (2005) discussed different approaches for conducting content analysis: conventional or open content analysis, directed content analysis, and summative content analysis.

We combined these approaches. Conventional content analysis aims at an open and data-driven approach to describe a phenomenon, with no prior theory or framework guiding the analysis. Directed content analysis is led by a chosen theory or prior research findings, and it aims at completing or refining the existing knowledge with new findings from the analyzed data. Summative content analysis involves counting or comparing words or content. Based on the literature analysis, two theories (the Kano model and CFSE) were identified for the creation of the initial version of the framework. In the next phase of the investigation, we used field data to validate the initial version (Figure 2), which was based only on literature and theories.

3.2.2 Design process

The initial versions were subjected to stakeholders' consultations that involved academics and practitioners. This led to the development of partial versions, such as the UFHCSL and UFSCSL. Thus, in phase 2 (Figure 2), we embarked on a field validation that involved stakeholder consultations. This involves submitting partial versions of the framework to conferences for feedback.

The first partial version of the framework was published as a work-in-progress at the proceedings of Euromicro SEAA 2019, Greece. A presentation Work-in-progress track was held to the conference participants to collect feedback for the incremental development of the framework. Thus, the conference was used as a focus group discussion to collect data for incremental design. At the end of the focus group discussion, two main areas for improvement were suggested. First, competence categorization (i.e., soft and hard) is essential for the development of the framework, mainly because both categories have different meaning and usage (Moreno et al., 2012). Second, the satisfaction levels of SEC must be added as a key feature to the framework.

Based on the suggestions from the group discussions, two separate partial versions (UFHCSL and UFSCSL) were created. The two versions were submitted to two separate conferences, and the frameworks were presented for feedback. The following tracks were held: Software Business Education in the 10th International Conference on Software Business 2019, Finland, and Industry Challenges

and Collaborations in the International Conferences on Software Qualities 2020, Austria. Participants were informed of the purpose of the presentation, that is, to use the venues as focus group discussions.

A summary of the participants in these three stakeholder consultations is presented in Figure 3. Feedback was received and incorporated into the next iteration process of the framework development. According to Peffers et al. (2008), artifact development requires some form of iteration. Thus, we performed iteration accordingly until a meaningful framework was achieved. This led to the partial creation of the UComGSP.

The development of the framework was continued with pre-processed interview data in phase 3. Since the interview data were pre-processed, we adopted an exploratory approach in which the focused research questions did not restrict the data collection effort. In phase 3 (see Figure 2), an interview dataset with 138 participants from various positions within the industry (i.e., software engineers, managers, supervisors, mentors) was used. All the participants were from Norway. Students of SE courses at Norwegian University of Science and Technology (135) conducted the interviews with a given interview structure from the course lecturer[1]. The interview data were processed by 4–5 students into a spreadsheet. To analyze the data based on the research objectives, we used a (Mason, 2002; Myers & Newman, 2007) qualitative research guide. Figure 4 represents the distribution of respondents' characteristics.

FIGURE 3. Participants in stakeholder consultations

[1] The course was lectured by professor Pekka Abrahamsson, who generously made the data available for further study.

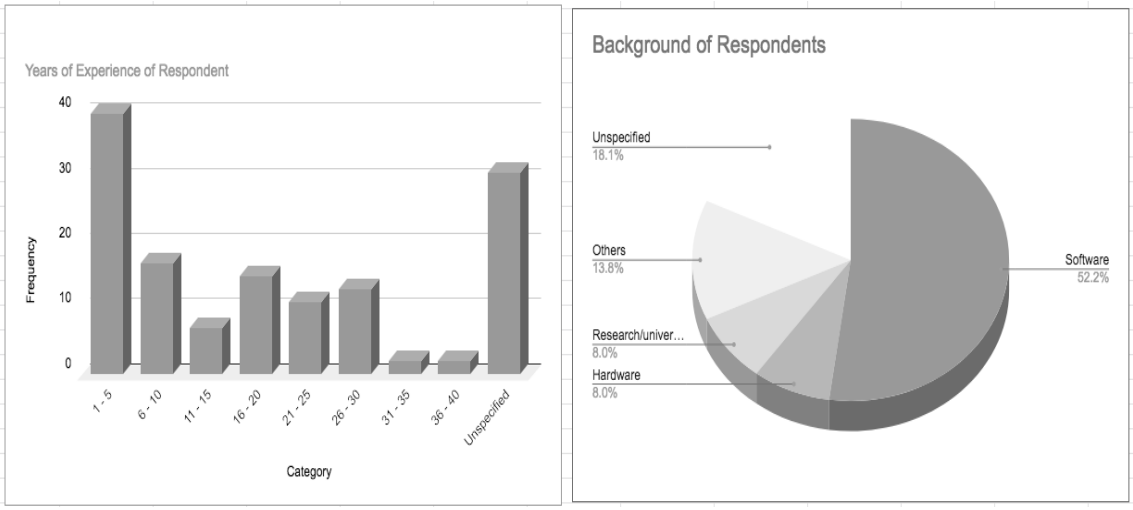


FIGURE 4. Respondents characteristics based on interview data

As the dataset was qualitative (interview data), we also aimed to analyze it using a qualitative approach; we made use of content analysis. Content analysis can be used to characterize responses in an open-ended survey, focus group, and interview transcripts (Krippendorff, 2018; Robson, 2002). It is defined as exploring large amounts of textual information in an orderly way to establish a trend or pattern (Vaismoradi et al., 2013). "Content analysis is a research technique for making replicable and valid inference from texts (or other meaningful matter) to the contexts of their use" (Krippendorff, 2018). In this dissertation, with particular reference to the interview data, content analysis was used to determine the presence of certain themes or concepts within the data of 138 participants into an organized and concise conceptual structure.

The expected outcome of content analysis can be quantitative or qualitative. We examined the meanings of the content of the interview data to identify relevant concepts based on the interview question: What are the competencies expected from persons working as software professionals in your organization? A content analysis must follow a certain procedure (Krippendorff, 2018). Thus, Krippendorff (2018) developed a framework for content analysis with the following components: texts (a body of the text of which the analytical effort will begin), research questions (a question of which the text will help to answer), context (context of the analyst's choice of making sense of the text), analytical constructs (units of words for the operationalization of the research question), inference (meaning extracted to answer the research question), and validity (evidence to justify the analysis).

Conducting these steps led to developing a holistic framework for managing SEC and determining the essential competencies that contribute to the development of high-quality software products and systems in modern societies. Thus, the final result of this study included a holistic framework for determining the essential competencies for software development, which can be customized according to the type of software project or software development assignment.

Therefore, the final framework, UComGSP (Figure 2, phase 3), was achieved using literature, theories, stakeholder consultations, and pre-processed interview data.

4 THEORETICAL FOUNDATION

A competence model is used for defining and assessing competencies. It is generally defined as a framework for defining the competencies requirement of a job. It can be used to identify, measure, assess, or evaluate the skills, attitudes, and knowledge needed to perform a task. A competence model "is a catalogue in which both general and technical competencies needed to perform a professional role are defined, including the level required for each one" (Saldaña-Ramos et al., 2012, p. 405). Satisfaction level is a term used in this study to mean competence development stages or levels. Therefore, it determines the performance level of competence in a software project or software development assignment by addressing the question: What value or performance level will competence "x" add to the development of a software project or software development assignment?

4.1 Competence versus competency, soft and hard competence, and essential competencies

The word competency or competence sometimes has varied and ambiguous meanings, depending mainly on the content or cultural contexts (Le Deist & Winterton, 2005). In this dissertation, we avoid these conceptual ambiguities and the debates by adopting the descriptive narratives used by Sedelmaier and Landes (2014):

> Competency denotes a comprehensive capability to act appropriately in complex situations. The capability to act includes technical knowledge, also called factual knowledge. The capability to cope with complex and new situations also presupposes additional skills, which are often subdivided into social, personal and methodological competence. (Sedelmaier & Landes, 2014, p. 395)

To this end, we define competence in software engineering as "a complete set of abilities, skills knowledge and capabilities needed to engage in a software development activity effectively." As such, competencies are associated with both

individuals and enterprises. From the individual perspective, they are related to education and human resource management (Le Deist & Winterton, 2005), while from the enterprise perspective, they are concerned with the resource-based view of the firm (Wernerfelt, 1984). Competencies can be divided into two main types: hard and soft competencies (Harzallah & Vernada, 2002; Havelka et al., 2009; Rainsbury et al., 2002; Tahvanainen & Luoma, 2018).

In summarizing other definitions of soft skills, Harris and Rogers (2008) stated that they are "work ethics, positive attitude, social grace, facility with language, friendliness, integrity, and the willingness to learn" and complement hard or technical skill (Harris & Rogers, 2008, p. 19). For the authors, most soft skills do not require formal training. Until recently, these skills were typically self-taught and self-developed. They are usually not industry specific. Further, a soft skill mostly requires emotional intelligence (Andrews & Higson, 2008; Trivellas & Reklitis, 2014). "Soft skills are the personal individuality that has a major impact on the behavior of a person, while having interaction with others in a working setup" (Ahmed e al., 2013, p. 172). They include communication, flexibility, leadership, motivation, patience, persuasion, problem-solving abilities, teamwork, time management, and work ethics.

By contrast, hard skills are needed to perform a job or assignment (Urs, 2013). These skills are teachable and acquired mainly through formal training and studies. Often the trainer is required to be smart or must possess a good IQ to acquire the required skill. However, with this sort of skill, the rules remain the same regardless of the industry (Andrews & Higson, 2008; Harzallah et al., 2002; Trivellas & Reklitis, 2014). They include language, typing speed, degree or certificate, and machine operation.

We use the term essential competence to denote the "essential competencies," "most relevant competencies," and "most needed competencies" (Calazans et al., 2017; Chang et al., 2016; Engelbrecht et al., 2018; Gimenes et al., 2012; Magenheim et al., 2010; Sedelmaier & Landes, 2012; Suhartono & Sudirwan, 2016; T. Turley & Bieman, 1995). As defined by Turley and Bieman (1995), essential competencies of software engineering are the skills, knowledge, and attitudes of software professionals necessary for excellent performance in a software project or software development assignment.

4.2 Software roles, associated positions, and tasks

In organizational settings, competencies are linked to roles. Thus, we examined the roles and tasks or responsibilities associated with software engineering. Depending on the organization, different names may be associated with certain roles (e.g., software developer, software designer, software engineer). This makes the study of the competencies of these roles difficult. To avoid such confusion, in this study we use the term *software professional* to address individuals who employ the necessary skills to design, construct, test, and maintain computer

software (Kalliamvakou et al., 2019; Kobata et al., 2013; León-sigg et al., 2018). Thus, we provide the context in which this competence study can be viewed.

Defining roles and responsibilities has not been easy in software engineering as with any other field. In most cases, different names are given to the same role. However, West (2004) suggested that the name(s) must be contextualized to the organization culture or the activities to be performed. For this reason, we make use of the roles defined in the software engineering body of knowledge (SWEBOK) (Bourque & Fairley, 2014) and the software engineering competency model SWECOM (IEEE, 2014). SWECOM, inspired by SWEBOK, provides various knowledge and skillsets for software engineering. These skill sets are grouped into five areas:

1. cognitive skills,
2. behavioral attributes and skills,
3. requisite knowledge,
4. related disciplines,
5. technical skills.

SWECOM further divides technical skills into life cycle skill areas and cross-cutting skill areas. While cross-cutting skill areas are skill sets needed throughout the life cycle of developing software, life cycle skill areas are those needed to perform a task in different phases of software development. Thus, since this study aimed to identify competencies and tasks associated with them and relate them to the roles of software development, we used the life cycle skill (comprising skills and activities) and mapped it to the software engineering roles in SWEBOK. Some of the different roles, their associated positions, and tasks or responsibilities for a typical software development project are requirement analyst, designer, programmer, test and quality engineer, and configuration and maintenance engineer. Figure 5 shows these roles and how they can overlap.

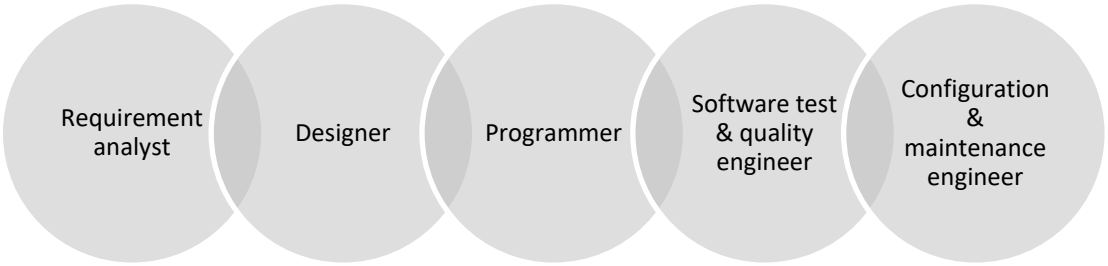


FIGURE 5. Roles of software professionals in the software development project

The different positions assigned to software professionals depend on the size of the organization or type of software project to be undertaken (Saiedian & Dale, 2000). Again, it is important to note that these roles can be more or less to what is in Figure 5. We acknowledge that these roles can be performed by a dedicated person or be shared and performed by all members of a team. The latter especially applies to agile software development methodologies, which rely on small and self-organized teams. In adopting these roles in Table 3, we also want to acknowledge that the field of software engineering has been undergoing a paradigm shift, such as agile (Abrahamsson et al., 2002) and DevOps (Debois, 2011) approaches to developing software. Such a shift brings into focus some emerging roles and responsibilities that hitherto were not part of the one presented in Table 3, for example, "product owner" and "scrum master". Notwithstanding this paradigm shift of approach, our study shows that even if agile methods are broadly adopted in practice and are also present in software engineering research, their specificity has not yet received much attention in SEC research; therefore, we relied on the traditional role definitions as the basis for our competence study. Therefore, to put the study into the context of competencies of software engineering, we use requirement analyst, designer, programmer, test and quality engineer, and configuration and maintenance engineer to illustrate different roles, positions, and responsibilities in a typical software development project.

Software professionals with certain competencies discover the requirements for developing software. As shown in Figure 5, a software professional with the role of requirement analyst discovers the software requirements. Table 3 shows some of the typical positions, roles, and responsibilities associated with the requirement discovery of a software professional in a software project. After the elicitation of the requirements, the software professionals take the role(s) of designing the architecture of the software based on the requirements. In Figure 5, we use the designer to show the stage of software professionals' role(s) in a software project. Typical positions, roles, and responsibilities associated with software professionals in designing the architecture of the software are presented in Table 3. The next role to be performed by software professionals is the construction of the software. In Figure 5, we use the programmer to represent the construction role(s). The programmer converts the design specifications into functional software. In the next stage, the software professional in his or her capacity as a test and quality engineer (Figure 5) oversees the testing and quality issues of the software. After performing the test and being satisfied with the quality of the software, the software professional moves to the role of configuration and maintenance engineer. In that role, the software professional configures and oversees the maintenance of the software. It is important to note that the activities to be performed by the software professionals can or may overlap during the software development stage. In Table 3, we have provided a framework depicting the typical positions, roles, and responsibilities associated with software professionals.

TABLE 3. Roles, associated positions and tasks of software professionals

Roles	Description	Associated positions	Tasks
Requirement analyst	Requirement analyst role represents software professional(s) with the responsibility of eliciting the functional and non-functional attributes of a software system to meet the goals of a customer.	System engineer, requirement technical leader, requirement engineer, requirement analyst, requirement lead manager, requirement technician, etc.	Identify stakeholders, perform analysis on the requirement, use appropriate means to describe the requirements, construct, and analyze prototypes, etc.
Designer	Designer role represents software professional(s) responsible for designing the architecture of software based on the requirement.	Software designer, lead designer, technical designer, design technician, etc.	Design technics for software design, manage software design activities, specify a common interface, and the use software design review, etc.
Programmer	Programmer is the software professional(s) responsible for constructing the software. It involves converting the design specifications into functional software.	Senior software engineer, lead developer, software technical leader, software technician, etc.	Select the environment for developing the software, monitor the software development process, create code and implement the design, and document and comment the codes, etc.
Software test & quality engineer	Software test and quality engineer represent software professional(s) responsible for overseeing the testing and quality issues of software.	Senior test/quality engineer, lead software test/quality engineer, software test/quality engineer, test/quality engineer, test technician, etc.	Identify stakeholders and tools for testing the software, develop a test plan for testing the software, and collect and report data resulting from testing/demonstration, etc.
Configuration & maintenance engineer	Configuration & maintenance engineer is the software professional(s) responsible for maintaining and sustaining the software during its lifecycle.	Senior maintenance/configuration engineer, lead maintenance/configuration engineer, maintenance/configuration engineer, maintenance technician, etc.	Develop transition and identify stakeholders for transition, maintain software configuration, perform problem identification and correction, and monitor and analyze software maintenance activities, etc.

To quote West (2004), people are of the view that "I am the universe and the universe is me, what I do matters and what other people do does not matter until it messes with my universe" (West, 2004, p. 49). Such an assumption hinders people from perceiving how different roles in an organization need to work hand-in-hand to accomplish an organizational goal. Thus, identifying the roles and their responsibilities provide a means to know which skills are needed to accomplish the organizational goal (IEEE, 2014). Furthermore, it enables people to understand how their roles overlap especially when it is being performed by different people. It also provides a means to switch from one role to another internally or externally. (Chrissis et al., 2011; West, 2004). We make use of West's (2004) definition of roles, which states that a role is "a brief summary description of a person's function in a relationship to a particular aspect of a business." In this illustration using the requirement engineer as a position for software professionals, we provide the following brief summary: requirement elicitation, requirement analysis, and requirement verification and validation are the different roles associated with the requirement engineer.

Knowing what to do also serves to help people evaluate the skills they have. Therefore, there is a need to describe the responsibilities of each role for software professionals. In the above illustration of the software professionals, using requirement engineers, we have shown in Table 3 that there can be different positions and that the positions may show hierarchy. Regarding the roles, we also showed that there can be different roles to be performed by different people or the same person depending on the organization.

4.3 Competency framework for software engineers (CFSE)

The competency framework for software engineers (CFSE) is a framework that facilitates and guides the development of software professional competencies as well as identifies the training needs for developing those competencies. The framework, which is built on the previous classification of the competence subject, is categorized into hard and soft. The design is based on the activities and interactions of the engineers during the software development process.

The hard competency category, which is about technical competencies, focuses on the roles of software engineering and the use of technology. These roles are based on the definition of SWEBOK roles in software engineering. They are project management, requirement analysis, software design, programming, validation and verification tests, configuration management, quality, tests, documentation, and maintenance. The soft competency category is divided into social competencies and personal competencies. Social competence is further classified into interpersonal relations, cooperation and work in a team, and handling and solving conflicts. The personal competencies area is also further classified into development in the job, personal development, and rights and limits. The authors of CFSE defined it as "a set of knowledge, abilities and key behaviors, with special emphasis on the soft skills" (Rivera-Ibarra et al., 2010).

TABLE 4. Competency Framework for Software Engineers (CFSE)

General Competence Classification	First Level Classification	Second Level Classification
Hard	Technical knowledge	Project management
		Requirement analysis
		Software design
		Programming
		Validation and verification tests
		Configuration management
		Quality
		Tests
		Documentation
		Maintenance
	Use of technology	Evaluation and selection of tools to support influenced areas
		Adaption and use of tools to support influenced areas
Soft	Social competencies	Interpersonal relations
		Cooperation and teamwork
		Handling and solving conflicts
	Personal competencies	Development in the job environment
		Personal development
		Rights and limits

With the main aim of extrapolating the essential competencies of software professionals in general, we see the framework as fitting, since it considered both soft and hard competencies which are the generally accepted categorization in competency literature. The framework also considers granularity, which is essential for fitting the work to the community. Lastly, CFSE was chosen because, in organizational settings, competencies are linked to roles. CFSE describes the roles of software engineering in the hard competence category. CFSE and the constructs in the framework are shown in Table 4. For details, readers can refer to the original study by Rivera-Ibarra et al. (2010). For our study, we adapted the technical knowledge (hard competencies), social, and personal competencies (soft competencies), which are connected to SWEBOK and SWECOM.

4.4 Kano model

The Kano model is a quality framework for mapping and prioritizing product features to customer needs. The model was initially introduced in the manufacturing industry, but more recently, it has been applied in the software development industry. The model takes into consideration the views of both the customer and developer in the development of a product instead of a passive approach of only developers (Y. C. Lee et al., 2008). As such, the model assists software development teams in determining the basic, performance, and delighter

categories of features of a product or service. Previous studies have deployed the Kano model for the development of IT systems and concluded that the model prioritizes user involvement, that is, it allows the inclusion of customers' views in the development of a system (Gangurde & Patil, 2018; Huang, 2018; Y. C. Lee et al., 2008; Lehtola & Kauppinen, 2006; Liu, 2000; Piaszczyk, 2011; Richardson, 2001). In this study, we used the Kano model to consider the views of SEC stakeholders about the set of competencies that they value the most. In this scenario, the "customer" is the software industry (i.e., entity using the competencies), and the set of competencies that they value the most is considered a product or service.

According to (Kano et al., 1984), a customer's decision-making options on product or service acquisition are based on conscious and subconscious deliberations (Kano et al., 1984). There is, therefore, the need to understand these processes of decision-making to help develop products or services. Kano et al. (1984) categorized these processes into three requirement levels: basic, performance, and delighter. Basic requirements relate to the customer's expectations about a product or service. These requirements are classified as basic since their presence is not dynamic enough to change the options and the opinion a customer has about the product. However, their absence may result in complaints from the customer. Performance requirements, by contrast, are the expected prerequisites that customers know, and they are essential influential factors in the customer's decision-making. These critical prerequisites create high levels of satisfaction when employed appropriately. Delighters are those requirements that do not engender any complaints from the customers when absent; however, they surprise the customer pleasantly when present. Delighters are sometimes referred to as attractive or "wow" factors (Kano et al., 1984). The variables or metrics originally used by Kano for classifying or describing these three requirements are shown in Table 5. Each competence from our collected data is associated with the following metrics for classification and categorization.

TABLE 5. Categorization metrics for Kano analysis (reproduced from (Kano, 2016))

Metric 1 (Basic)	**Metric 2 (Performance)**	**Metric 3 (Delighter)**
• Must-Be's • Threshold attributes, price of entry • Taken for granted and expected by customers • Can be attracted or variable in nature	• One dimensional • Result in satisfaction when fulfilled • Consciously evaluated when looking at alternative • Often "more the better" requirements • Can be attributes or variable in nature	• Innovation or wow factor • They delight customers when delivered • They will not dissatisfy when missing • They are most unspoken of by customers. • Can be attributes or variable in nature

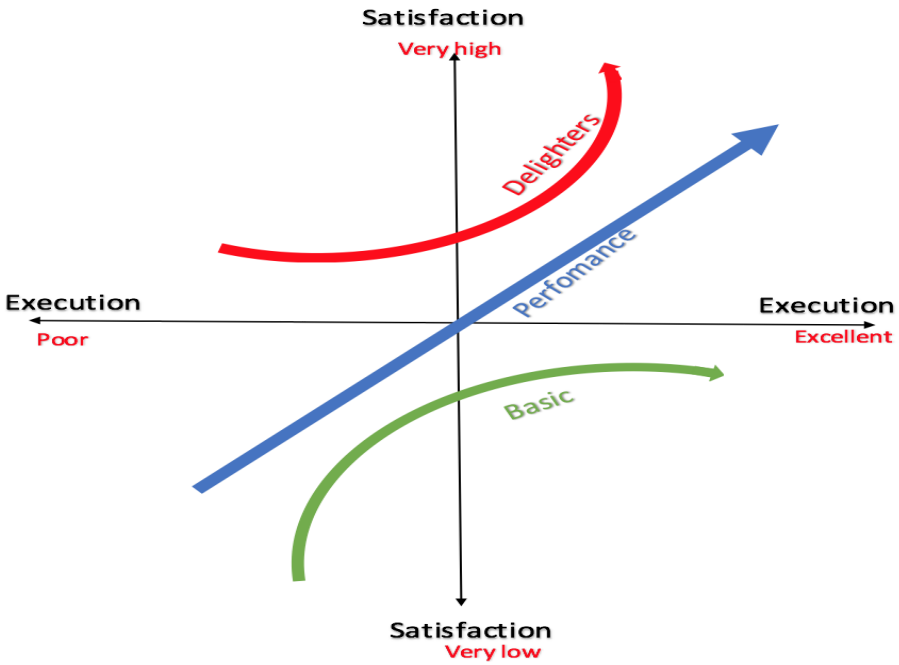


FIGURE 6. Graph used in the Kano model (Kano, 2016)

Kano originally illustrated these requirements using a graph, see Figure 6. It is important to note that the graphical representation of these competencies is beyond the scope of this research. However, we reproduced the original graph to further emphasize our arguments for understanding purposes.

We used the metrics in Table 5 to classify the competencies derived from the data. It is important to state that even though the Kano model has been used in software engineering literature specifically on products and services, this work is the first to use it on competencies, a concept that is directly related to humans. Thus, this work charts a new direction in competence studies.

4.5 Framework construction and its applications

The aim of this dissertation is to develop a framework that is useful for academics and practitioners. We will now illustrate how to use the proposed framework for identifying software professionals' competencies, determining the satisfaction levels of SEC, and identifying the essential competencies of software professionals in different software projects. Figure 7 shows the process steps for using the UComGSP, which is the final framework as a result of this study.

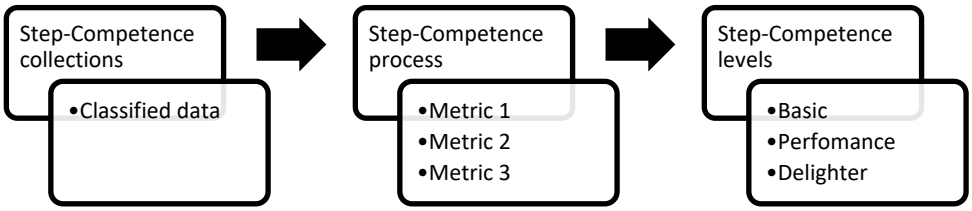


FIGURE 7. Process steps for using the unified competence gate for software professionals (UComGSP)

TABLE 6. Identified competencies

Competence Category	SE Identified competencies
Soft	Teamwork, knowledge transfer, cooperation
Hard	Coding competencies across platforms, basic coding skills, good coding skills

UComGSP consists of three steps. In the first step, particular attention is not paid to the categorization of the competencies, but it is rather assumed that this is done before using the competence gate. That is, competencies are identified and classified according to the choice of classification, for example, hard or soft competence. Details of such classifications can be seen in Assyne (2020, 2019). In the second step of the process, each identified competence is accessed using the metrics in Table 5. Based on the assessment, in the third step, the competencies are classified as basic, performance, or delighter. If the value of a competence satisfies Metric 1, it is classified as basic; if it satisfies Metric 2, it is performance; and if it satisfies Metric 3, it is delighter. The basis satisfaction levels following the Kano model are presented in Table 5.

Next, we present a scenario to illustrate the use of UComGSP. Table 6 contains competencies that have been identified and classified using the categorization of hard and soft competencies. To make a choice using the metrics stated (Table 5), the following must be observed. In the competence categorization of soft competencies, teamwork, knowledge transfer, and cooperation are used as examples to illustrate how the choice is made. On the hard competencies' categorization, coding competencies across platforms, good coding skills, and basic coding skills are also used to show how the choice is made. Making a choice as preference will indicate that cooperation, a behavioral competence, is defined as the ability to work with others on a software project or software development assignment, which is required in any software development. This competence meets Metric 1 in Table 5. Thus, cooperation will be classified as a basic competence.

Teamwork is a competence that allows software professionals to work together for the effective development of a software product or service. Teamwork leads to higher performance in software development. Therefore, they are expected prerequisites that are known, and they are essential influential factors in the software industry decision-making options on competence. These are critical

prerequisites that create high levels of satisfaction when employed appropriately. Therefore, teamwork meets Metric 2 in Table 5 and is classified as performance competence.

Knowledge transfer competence is the behavioral competence of having an outstanding ability to impart knowledge to others in a software development. This type of competence does not engender any complaints from the software industry when absent from the software professional. However, it surprises the software industry when present. Thus, a knowledge transfer meets Metric 3 and is classified as delighter. Basic coding skills are technical skills, such as the ability to read and understand code. This competence is required from any software professional. Such competencies are taken for granted; however, their absence will be noticed by the software industry. Therefore, they meet Metric 1 and are classified as basic competence. Good coding skills include technical competence, such as commenting well and self-reliance in coding. Such technical competencies are expected prerequisites that are known, and they are essential influential factors on the software industry decision-making options on competence. Therefore, it is a performance competence. Coding competencies across platforms is a technical competence of being able to code on multiple platforms. Thus, it is a technical delighter competence.

Using the scenario illustrated above, the stakeholders can use the framework proposed (UComGSP) to determine the satisfaction levels of competencies for any software project or software development assignment. *Basic competencies* are prerequisite competencies that are necessary and are expected by the software industry. Mostly, they are taken for granted. The software industry considers these competencies as natural when delivered properly. However, when delivered poorly, the software industry complains. *Performance competencies* are competencies that the software industry expects and can articulate. They are mostly in the minds of the software industry actors, and when they are delivered well, they create more satisfaction. These competencies can be described as "unidimensional" competence. *Delighter competencies* are competencies unexpected by the software industry. Mostly unexpected by the software industry but increases the delight and surprise when available; however, its absence may have no effect on the software industry. Table 7 provides details of the competencies and their satisfaction levels used in the scenario.

TABLE 7. Competence satisfaction level framework

Competency category	**Competency**	**Satisfaction levels**
Soft	Cooperation	Basic
	Teamwork	Performance
	Knowledge transfer,	Delighter
Hard	Basic coding skills	Basic
	Good coding skills	Performance
	Coding competencies across platforms	Delighter

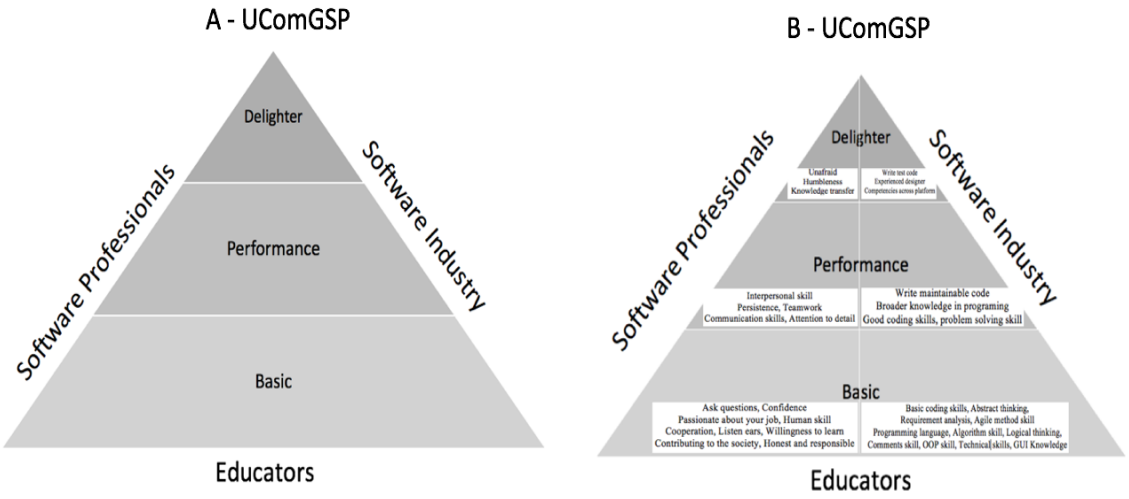


FIGURE 8. Unified Competence Gate for Software Professionals (UComGSP)

In developing the final version, which is presented in this study, all feedback from the group discussions was taken into consideration. Thus, in this study, to provide an avenue for different usages of the framework, we present two separate frameworks, one for soft and one for hard competence for determining competence and satisfaction levels. We also present a scenario for determining satisfaction levels based on different software projects or software development assignments. Thus, the final framework is as follows (see Figure 8):

- A–UComGSP shows the framework without the competencies used to illustrate how to use the framework.
- B–UComGSP shows the framework with specific competencies stacked to show the levels.

The proposed framework can be used by software professionals, educators, and the software industry. Software professionals can use the framework to determine with which competencies they are employable. Educators can also use the framework to determine which competencies they need to teach to the software professionals for them to be employable. Finally, the software industry can also use the framework to determine which employers they should employ and at the same time use it to validate competencies needed by a particular software project or software development assignment. A key consideration of the use of this framework is that it can be adjusted to different software projects or software development assignments, paving a way for its use with new methodologies used to develop software such as agile and DevOps.

5 CONTRIBUTIONS, LIMITATIONS, AND FUTURE RESEARCH TOPICS

In this chapter, we present the contribution of this dissertation, which is twofold: that is, from a theoretical and a practical perspective. Limitations of this study and future topics are also discussed in this chapter.

The results add to SEC studies by providing a new dimension in both research and practice in the SEC. The dissertation aimed to investigate and provide an in-depth analysis of the SEC of software professionals for software development. It provides insight into the management of SEC as well as how competencies can vary according to different projects. The dissertation explored means of identifying SEC, determining satisfaction levels of SEC, and the identification of essential competencies of software professionals. Thus, the results show how to organize and observe the understanding (Bhattacherjee, 2012; Mead & Shoemaker, 2013) of the SEC. Section 5.1 shows the research questions, summary of results, and contributions.

5.1 Summary of results and contributions

RQ1: What is the state of research related to software engineering competencies and their evolution?

The literature on SEC is not lacking; SEC is considered to be the backbone for the successful development of software products or services. The previous review studies have only examined specific areas in SEC, such as the role of personality in software engineering. However, personnel and organizational research areas were identified as the main research areas. We also identified SEC models and frameworks for research and practice, as well as a set of essential competencies of software professionals. The SEC overview and the research gaps identified in this dissertation will help to provide a better understanding of future research on SEC, particularly on how to deal with the essential competencies of software

professionals that contribute to the development of high-quality software products and systems in modern societies.

RQ2: What are the different satisfaction levels of software professional's competence?

Previous literature categorized competence into soft and hard competencies. Other studies have established that there are some competencies of software professionals that are necessary for excellent performance in software development. A competence model was developed to enable the understanding of the underlying logic of the SEC in terms of satisfaction. The model was developed in an iterative process. This model enables the determination of the satisfaction level by considering the stakeholders of the SEC. The model enables the determination of competencies according to the dynamics of the projects. Three satisfaction levels were identified: basic, performance, and delighter. Thus, it provides a means to understand and observe competencies during software development.

RQ3: What are the different competencies of software engineering roles?

In organizational settings, competencies are linked to roles. Previous studies on software engineering have linked roles with tasks or responsibilities. Software professionals' competencies can also be associated with the tasks or responsibilities in a software project.

Thus, this study identified the competencies for software development and classified them into behavioral and technical competencies with a further granularity as follows:

- Behavioral competencies:
  - Interpersonal relationship
  - Cooperation and working in team
  - Handling and solving conflicts
  - Development in the job environment
  - Personal development
  - Rights and limits
- Technical competencies using the software engineering roles:
  - Project management
  - Requirement analysis
  - Software design
  - Programming
  - Validation and verification
  - Configuration management
  - Test and quality

We used the literature to set an agenda for the identification of competencies associated with the software developer's security competencies. We identified programming-related security competencies and non-programming-related security competencies. We also developed a framework for identifying the software developer's security competence.

RQ4: What are the essential competencies of software professionals?

Essential competence is defined as the skills, knowledge, and attitudes of software professionals necessary for excellent performance in software development. We identified essential competencies of software professionals and mapped them to the roles of software engineering. The identification of essential competencies was not dynamic according to different projects. This study identified the essential competencies for behavioral competencies and technical competencies. The essential competencies were mapped to the roles of software engineering. Newly observed essential competencies were brought up in the study.

5.2 Contributions to the body of knowledge

Through an in-depth analysis, this dissertation determines the satisfaction levels of the competencies of software professionals. Of particular importance is the determination and identification of the essential competencies of software professionals. Using the three key stakeholders of SEC, we identified gaps in SEC research. This dissertation has addressed several of these gaps, which serve as theoretical contributions to advance knowledge. In the next sections, we provide more detail on the theoretical contributions.

5.2.1 Conceptualization of Software Engineering Competencies of Software Professionals

Developing software successfully requires competent software professionals (Casale et al., 2016), as they directly influence the quality of the software development process. The mapping study was conducted to provide an overview of the current state of research on SEC. The review revealed that there is a rich body of literature published in this area (Lenberg et al., 2015); however, there is a need to study the evolution of software engineering competencies over time as well as to identify essential competencies necessary for developing the next generation of software products. The mapping study also revealed that identification, assessment, and development of SEC are interrelated functions, which must be considered as a whole and from the perspective of the stakeholders (software professionals, educators, and software industry). However, the mapping study did not identify any study that proposed a comprehensive model or framework that could be used for such a holistic approach. More importantly, we did not find any model or framework for assessing the benefit of SEC for these stakeholders, for instance, by determining satisfaction levels derived for possessing or using a competence. As mentioned by Mead and Shoemaker (2013), the SEC models and frameworks are beneficial in organizing and observing the understanding of the SEC. According to Frezza et al. (2018) and IEEE (2014), competence development is not the sole responsibility of software professionals. It also requires the participation of other stakeholders, including software companies and the software

industry in general, as well as educators who provide education and training to software professionals. Therefore, providing a model or framework that considers the key main stakeholders for organizing and observing the SEC will enhance software development in practice and research.

In view of this, the first contribution of the dissertation was to develop a holistic model for organizing and observing the SEC, with a special focus on identifying, assessing, and developing SEC, secondary to the use of the model to observe and organize SEC. Developing software successfully requires competent software professionals (Casale et al., 2016), as they directly influence the quality of the software development process. Thus, to answer the question of the efficacy of the SEC model and framework for managing the SEC (Mead & Shoemaker, 2013), the studies resulting in Articles II, III, and IV were conducted. The models and frameworks contribute theoretically and practically to the advancement of knowledge. Typically, the holistic model developed using an iterative process in this dissertation addresses to Frezza et al. (2018) and IEEE (2014) argument that competence development is not the sole responsibility of software professionals. Rather, it requires the participation of other stakeholders, such as software companies and the software industry in general, as well as the educators who provide education and training to software professionals. Thus, this study developed a framework that can be used for the identification, assessment, and development of the SEC.

The second contribution is the determination of competence satisfaction levels that serve as an assurance to the stakeholders of competence development. Barreto et al. (2008) pointed out that staffing people for software development is not a straightforward decision-making process. Therefore, having a model that helps in determining the satisfaction levels that serve as an assurance for the stakeholder involved in competence development is helpful. Additionally, certain studies have provided different forms of competencies necessary for software development. These include base competence defined by Thurner et al. (2014) and essential competence defined by Calazans et al. (2017), Chang et al. (2016), and Turley and Bieman (1995). Also, the software assurance competency model enables software organizations to access the capabilities of software assurance professionals. It has 1-5 levels (Hilburn et al., 2013). However, specific performance levels have never been defined. Thus, this study defined three stages of competencies for software development. Basic competencies (satisfaction level) are prerequisite competencies that are necessary and are expected by the software industry. They are mostly taken for granted. The software industry perceives these competencies as natural when delivered properly. However, when delivered poorly, the software industry complains. Performance competencies (satisfaction level) are competencies that the software industry expects and can articulate. They are mostly in the minds of the software industry, and when they are delivered well, they create more satisfaction. These competencies can be described as "unidimensional" competence, in that satisfaction grows exponentially when executed properly. Delighter competencies (satisfaction level) are competencies unexpected by the software industry. Mostly unexpected by the

software industry but increases the delight and surprise when available however its absence may have no effect on the software industry. We state that certain factors, such as innovation, training, etc., can cause a competence to change from one level (state) to the other. This can be either to improve or reduce performance. Therefore, we describe these levels (basic, performance, and delighter) as stages in the competence growth path (competence evolution). This is because competence in the state of performance can be affected by training and cause it to change to basic or delighter and vice versa.

Last is the contribution of the identification of competencies of software professionals for software development. This is made up of the identification of competencies of software professionals and the essential competencies of software professionals. André et al. (2011), Charette (2005), and Nelson (2007) establish that not having the right people assigned to the roles as well as problems with team management are two human factors that contribute to the failure of software projects or software development. assignments. In short, the right competencies of software professionals are key to successful software development, especially essential competencies (Manawadu et al., 2015). Thus, this dissertation has identified the competencies of software professionals for software development. More importantly, the thesis also identified the essential competencies of software professionals. From our Article I, we found that the major research on the essential competencies of software professionals is over two decades old. Thus, this thesis served as a review of the essential competencies study on SEC. The thesis compared the identified essential competencies with those in the literature and listed the new observations of essential competencies of software professionals. Finally, in organizational settings, competencies are linked to roles. André et al. (2011), Charette (2005), and Nelson (2007) also established that not having the right people assigned to the roles as well as problems with team management are two human factors that contribute to the failure of software projects or software development assignments. Therefore, this thesis answers this by assigning the hard competencies identified to the roles of software engineering. Furthermore, the satisfaction levels of the hard competencies were also classified using the roles.

5.2.2 Contextualization of SEC of software professionals

We further analyzed our contributions using the theoretical contribution framework of Corley and Gioia (2011). In their framework, they pointed out that theoretical contribution is not only when a variable is added or subtracted from a theory, but it must also explain the additions and subtractions of the variables. The authors provided two main dimensions with their subdimensions:

1. Originality
    a. revelatory
    b. incremental
2. Utility
    a. practical usefulness
    b. scientific usefulness.

TABLE 8. Analysis of the theoretical contribution of the dissertation

Originality	**Utility**
Incremental - The combination of the constructs in both the CFSE and Kano models allows for the creation of a holistic framework for managing the SEC	**Scientific** - The model developed can be applied in other fields of study
Revelatory - Kano model use in SEC studies - Soft and hard competencies can be seen as equal pillars of software engineering	**Practice** - Organization can use the model to determine the competencies for project - The identified competencies can be used as yardsticks for software development

Incremental as a sub-dimension of originality asks whether there is an addition or subtraction of a variable and how such addition and subtraction affect the new phenomena. The subdimension revelatory is the contribution toward the advancement of knowledge. The dimension of utility entails how the theory is practically implemented, that is, the use of the concept in an organization or society. Practical utility involves using the concept in an organization for an outcome. By contrast, scientific utility is being able to use the theory in other fields to increase knowledge advancement (Corley & Gioia, 2011). The theoretical contribution of this dissertation using the framework of Corley and Gioia (2011) is presented in Table 8.

In terms of revelatory of originality contribution, the Kano model has only been applied to products and services. Our study is the first to apply the Kano model to the competencies of software professionals (human resources), which charts a new path in software engineering. Through the use of the combined models and frameworks, we were able to advance knowledge on SEC. Our studies propose that the soft and hard competence can be seen as equal pillars of software engineering. The originality dimension of theoretical contribution is further subdivided into incremental and revelatory (Corley & Gioia, 2011). During the theorization process, the incremental contribution was used to enable organizing and observing the SEC. In this process, we combine the constructs of the Kano model with the CFSE (Rivera-Ibarra et al., 2010). This was deliberately done to enable understanding of the SEC from the traditional classification of the SEC since each model or framework had a different perspective. However, the combination of the variables enables us to develop a holistic framework for identifying, assessing, and developing the SEC. This hitherto was handled separately by different models or frameworks. Thus, the competence framework for managing hard (Article II) and soft (Article III) competencies was developed. The determination of competencies on the traditional levels enables us to understand the trend in both had and soft competencies. Thus, addition and subtraction provide a means of understanding the phenomena and advance knowledge in SEC research. In this dissertation, using the separate models for soft and hard, we were

able understand the different satisfaction levels (assessment), essential competencies (identification), and their contribution to any software development. Additionally, we were able to determine competence using the roles of software engineering. This is because in organizational settings, competencies are linked to roles. Our findings therefore provide a means to link the role, associated position, and responsibilities.

With regard to scientific utility, our frameworks can be applied to other fields where the use of human resources is applicable. Thus, the frameworks can be used to advance knowledge in the domain of human resources in general. Additionally, the process used to develop the UComGSP illustrates how to develop a framework or model using an iterative process, which is key for developing an artifact. On a practical level, an organization can use the framework to determine the competencies for different software projects or software development assignments. Furthermore, the identified competencies can be used as a yardstick for software development. It is worth mentioning that there are other works, such as the people capability maturity model (People CMM) (Curtis et al., 2009), the Essence Kernel by Object Management Group, Inc. (OMG) (Object Management Group, 2018), and the European e-competence framework (e-CF) (CEN, 2014), that can be used to assess the competence levels of software professionals. However, we provide another framework, one that considers the main audience of SEC—the stakeholders involved in staffing development projects, teams, or recruiting SE professionals. We also recognize the existence of software process improvement (SPI) (McFeeley, 1996), capability maturity model (CMM) (Software Engineering Institute, 2010), and related, multiple standards and standardization organizations, such as IEEE and ISO/IEC that provide standards to software organizations to guide and improve productivity and quality, reduce costs and time to market, and increase customer satisfaction.

5.3 Limitations and future research

This dissertation acknowledges some limitations. These limitations range from generalizability of the results, data collection, data analysis, and limitations on the concepts. To discuss the limitations of this study, we use four types of threats to validity suggested by Wohlin et al. (2012) and Runeson and Höst (2009). Construct validity concerns the use of the right operational measures to study the main research phenomenon, in our case SEC. SEC is associated with individuals and enterprises (Le Deist & Winterton, 2005; Wernerfelt, 1984). Therefore, the concept sometimes could have become misconstrued during its study.

This dissertation made use of literature data and interview data. Regarding construct validity to the literature data, there is a concern about using the right measure for studying SEC. To avoid potential misconceptions, the definitions and keywords were carefully selected from previous SEC studies and with the contribution of all the research team. Further, to identify relevant literature, the identified keywords were used to develop and test a search string. Even though

we had used known papers from some selected literature review in the SEC area to validate the use of the term competence in the string statement as against terms such as skills, knowledge, attitude, etc., we also want to acknowledge that we may have missed some papers during the selection process. Nevertheless, a thoughtful process was considered before the use of that term competence. For the interview data, to mitigate this threat, we based this study using operational words from the definition of (Ahmed et al., 2013; Frezza et al., 2018; IEEE, 2014; Urs, 2013). Thus, the transcribed interviews were analyzed using well-established operational words from the literature. This study used the identified competencies to propose the essential competencies of software professionals. It is important to note that previously studied models and frameworks were used to attain the results. Furthermore, the results were compared with previous studies on the subject matter to assess the consistency of our results.

Causal relations as they relate to internal validity were not an issue with this study. This is because we did not apply statistical inference in this study for causal relationships. External validity concerns the extent to which the findings of a study can be generalized (Wohlin et al., 2012). Our study is based on a literature review, interview data, and group discussion. One of the limitations of the study is the scope of the data collection, which was confined to companies situated in one country (Norway). Nevertheless, most of the companies that the interviewees worked for have global representation and businesses outside Norway. However, this limitation invites further studies with data from other countries to test the generalizability of the results. With the development of competence satisfaction levels, we call for further studies to understand how specific competencies evolve within different projects. That is, how the competencies of an individual evolve and how the competency within a software organization (e.g., a software or IT company, or IT organization within a user organization) develops its SE competency as part of their HR management.

Finally, reliability, as defined by Wohlin et al. (2012), is the extent to which the data collection and analysis processes are influenced by the researchers involved in the study. To this end, a well-structured process for collecting and analyzing the data was developed and followed. This process allowed for replication of the study. Further, since the authors of this study were not directly involved in the data collection, to increase the reliability of the findings, the authors consulted the leader of the data collection team during the analysis stage to resolve ambiguity in the dataset. On the development of the frameworks, we purposefully selected different conferences that are attended by both practitioners and academics. Thus, the frameworks were validated by key stakeholders in SEC. We also used different data sources to attain the findings.

Regarding future research, we suggest that studies validate the observed competencies, especially those identified as essential competencies of software professionals. We further call for more studies to validate the proposed framework. We suggest further research to understand how competencies within the satisfaction levels can change (e.g., from basic to performance or delighter) and its implications for software development.

6 CONCLUSION

This doctoral dissertation aimed to investigate and provide an in-depth analysis of the SEC of software professionals for managing software development. The goal was to enable SEC stakeholders (i.e., software professionals, educators, and the software industry) (1) identify SE competencies, (2) identify the essential software engineering competence, and (3) assess the satisfaction levels derived from those competencies. Hence, the studies in this dissertation used the literature on SEC, interview data from 138 participants in various positions within the industry in Norway, and expert consultations.

The dissertation resulted in the development of frameworks including the unified framework of hard competency satisfaction levels, the unified framework of soft competency satisfaction levels, the unified competence gate for software professionals, and the software professional security competencies framework. The frameworks can be used to manage software engineering competencies. More importantly, as pointed out throughout the different studies in this dissertation, the development of software engineering competencies cannot be done in isolation. Thus, the unified competence gate for software professionals is an excellent framework that considers the key stakeholders of software development. Using the frameworks developed, the study also identified some competencies and essential competencies for software development. The study also discovered some new competencies that were previously not in the extant literature. Of important to this study is the use of a unified competence gate for software professionals, which can be used to evaluate the competence levels for different software projects or software development assignments. The dissertation also calls on researchers to examine some specific competencies and their implications for the industry. This dissertation demonstrated the security competencies of software professionals to exemplify the use of the frameworks.

In the software business area, competitive advantage is usually short-lived; therefore, ingenuity and competencies of software professionals are needed to continue to create groundbreaking innovations (Sambamurthy et al., 2003). For this reason, the use of the unified competence gate for software professionals, which allows for the assessment of the needed competencies according to

different software projects or software development assignments, will be an answer to the creation of innovations that can affect a firm's performance, as pointed out by Sambamurthy et al. (2003). This will create avenues for software businesses to flourish. Further, having the right competencies for a software project or software development assignment will also indicate being able to deliver the project on time, which will also affect the cost of delivery of software. Delivering software projects or software development assignments on time will also lead to customer satisfaction, which, in effect, influences software businesses. The studies in this dissertation can have an impact on software businesses when applied in software development projects.

An area of consideration for this dissertation was the data for the empirical evaluation. Although there was already processed interview data to start the research, the style of the research required other sources of data as well. Focus group discussion was selected to complement the interview data. However, having a venue that could bring the key stakeholders of SEC (software professionals, educators, and the software industry) under one roof was a concern. Conferences were selected as the appropriate venue. This required presenting the studies' findings at the conferences, which required having the paper accepted through the conferences' peer-review system since they were academic conferences. This process was time-consuming and significantly extended the time of the study; however, it provided rich data to complement the interview data. Given that research on software engineering competence is not lacking, conducting a systematic review of such extensively studied concepts entailed dealing with many articles. This was another time-consuming process that delayed the reporting of the results. Overall, we have provided an outcome that can be beneficial to software practitioners and researchers.

YHTEENVETO (SUMMARY IN FINNISH)

Tämän päivän yhteiskunta on vahvasti riippuvainen erilaisten ohjelmistojen toiminnasta, siksi pätevien ohjelmistoalan ammattilaisten kysyntä on kasvamassa. Ohjelmistokehitys on monimutkainen prosessi, jossa hyödynnetään intensiivisesti inhimillistä pääomaa, toisin sanoen monitahoista osaamista. Sen vuoksi tulosten laatuun vaikuttavat ohjelmistoalan ammattilaisten taidot ja pätevyys (Colomo-Palacios ym., 2013). Monimutkaisessa ympäristössä, jossa sekä henkilökohtaiset taidot että tiimityö ovat välttämättömiä, eri sidosryhmille on tärkeää varmistaa etukäteen ohjelmistoalan ammattilaisten osaamispotentiaali. Colomo-Palacios et al. (2013) näkevät, että inhimillisen osaamisen kehittäminen ja hallinta on yksi ohjelmistoteollisuuden keskeisistä huolenaiheista. Kirjoittajat korostavat myös ohjelmistoalan ammattilaisten vakiintuneiden urapolkujen puutetta, koska alalla ei ole sovittu roolimäärityksistä ja pätevyyden todentamisesta järjestelmällisesti. Lisäksi puute osaavista ihmisistä ja ongelmat tiimin johtamisessa ovat kaksi ihmisten kykyihin liittyvää tekijää, joiden on havaittu vaikuttavan ohjelmistoprojektien epäonnistumiseen (André et al., 2011; Charette, 2005; Nelson, 2007).

Tiivistäen ohjelmistoalan ammattilaisten kompetenssit ovat avain onnistuneeseen ohjelmistoprojektien hallintaan. Tätä varten voidaan erottaa välttämättömät kompetenssit (Manawadu et al., 2015; Turley & Bieman, 1995). Kattava kirjallisuushaku ei kuitenkaan löytänyt kokonaisvaltaista ohjelmistotuotannon (Software Engineering Competence, SEC) osaamismallia tai -viitekehystä, jonka avulla voitaisiin tunnistaa osaaminen, osaamisen tyydyttävyystasot ja eri ohjelmistoprojekteissa tarvittavat välttämättömät kompetenssit. Lisäksi olemassa olevien mallien ja kehysten analyysi osoittaa, että ne ovat resurssi-intensiivisiä ja monimutkaisia käyttää eikä niitä voida räätälöidä eri ohjelmistoprojektien erityispiirteiden mukaan. Näin ollen useimmissa ohjelmistoprojekteissa ihmiset jaetaan rooleihin ja ryhmiin projektipäällikön tai tiiminvetäjän joskus rajallisen kokemuksen perusteella (André et al., 2011).

Ohjelmistoliiketoiminnassa kilpailuetu on pääosin lyhytaikaista, joten ohjelmistoammattilaisilta vaaditaan kekseliäisyyttä ja osaamista jatkaa uraauurtavien innovaatioiden luomista (Sambamurthy ym., 2003). Tästä syystä yhtenäinen osaamiskartoitus ohjelmistoalan ammattilaisille on vastaus yrityksen suorituskykyyn vaikuttavien innovaatioiden luomiseen, kuten Sambamurthy et. al. (2003) ehdottavat. Se mahdollistaisi tarvittavien kompetenssien arvioinnin eri ohjelmistoprojektien mukaan, mikä vahvistaisi ohjelmistoyritysten liiketoimintaa. Lisäksi ohjelmistoprojektien oikeanlainen osaaminen tukee myös kykyä toimittaa projekti ajoissa, mikä puolestaan vaikuttaa ohjelmistojen kustannuksiin. Kun ohjelmistoprojektit toimitetaan ajallaan, seuraa asiakastyytyväisyyttä, mistä edelleen syntyy myönteisiä vaikutuksia ohjelmistoliiketoimintaan. Tämän väitöstutkimuksen tulokset on suunnattu näiden myönteisten kehityskulkujen vahvistamiseen ohjelmistoliiketoiminnassa, ohjelmistokehitysprojekteihin sovellettaessa.

Väitöskirjassa vastasimme seuraaviin tutkimuskysymyksiin:

Tutkimuskysymys 1. Mikä on ohjelmistotekniikan osaamiseen ja sen kehitykseen liittyvän tutkimuksen tilanne?
Tutkimuskysymys 2. Mitkä ovat ohjelmistoalan ammattilaisten tyytyväisyystasot?
Tutkimuskysymys 3. Mitkä ovat ohjelmistosuunnitteluroolien erilaiset kompetenssit?
Tutkimuskysymys 4. Mitkä ovat ohjelmistoalan ammattilaisten olennaiset pätevyydet?

Tuloksena kehitimme kokonaisvaltaisen viitekehyksen, jonka avulla ohjelmistoalan sidosryhmät (ohjelmistoammattilaiset, kouluttajat ja ohjelmistoteollisuus) voivat (1) tunnistaa alan kompetenssit, (2) tunnistaa välttämättömät ohjelmistotekniset kompetenssit ja (3) arvioida näistä kyvyistä johdettuja tyytyväisyystasoja. Tutkimukseen valittiin laadullinen lähestymistapa, koska se on avoin esiin nouseville ongelmille. Ehdotetun viitekehyksen kehittämiseen ja validointiin käytettiin ohjelmistotekniikan osaamista koskevaa tutkimuskirjallisuutta, haastattelutietoja 138 alan eri tehtävissä Norjassa työskenteleviltä henkilöiltä sekä asiantuntijakonsultaatioita.

Kaikkiaan tunnistimme 63 niin sanottua pehmeää kompetenssia (soft competence) ja 62 teknis-tiedollista kompetenssia (hard competence), jotka kartoitimme tunnustettuihin rooleihin käytännön ohjelmistokehitystyössä. Kompetenssien joukosta tunnistimme ohjelmistoammattilaisen 25 keskeistä kompetenssialuetta. Olemme myös laatineet toimivan määritelmän näille välttämättömille kompetensseille eli ohjelmistoalan ammattilaisten taidoille, tiedoille ja asenteille, jotka ovat välttämättömiä erinomaisen suorituskyvyn saavuttamiseksi ohjelmistoprojekteissa. Seuraavaksi kompetenssit jaettiin Kano-laadunarviointimallia (Kano model) käyttäen analyyttisesti kolmelle tyytyväisyystasolle (perus-, suoritus- ja ilahduttavuustaso) vastaavien määritelmien mukaan. Esitimme skenaariokuvauksen avulla, kuinka sidosryhmät voivat arvioida ohjelmistoprojektissa tarvittavia eri osaamistasoja. Suosittelemme kuitenkin lisätutkimuksia ymmärtääksemme, kuinka tyytyväisyystasojen kompetenssit voivat muuttua (esim. perustasosta suoritus- tai ilahduttavuustasoon). Lisätutkimusta tarvitaan havaittujen uusien kompetenssien tutkimiseksi ja validoimiseksi. Niistä 11 luokiteltiin oleellisiksi kompetensseiksi.

REFERENCES

Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J. (2002). Agile Software Development Methods: Review and Analysis. *VTT Publications*, *478*, 3–107.

Acuña, S. T., & Juristo, N. (2004). Assigning people to roles in software projects. *Software - Practice and Experience*, *34*(7), 675–696. https://doi.org/10.1002/spe.586

Acuña, S. T., Juristo, N., & Moreno, A. M. (2006). Emphasizing human capabilities in software development. *IEEE Software*, *23*(2), 94–101. https://doi.org/10.1109/MS.2006.47

Ahmed, F., Capretz, L. F., Bouktif, S., & Campbell, P. (2013). Soft Skills and Software Development: A Reflection from Software Industry. *International Journal of Information Processing and Management*, *4*(3), 171–191. https://doi.org/10.4156/ijipm.vol4.issue3.17

Alavi, S. B., Moteabbed, S., & Arasti, M. R. (2012a). A qualitative investigation of career orientations of a sample of Iranian software engineers. *Scientia Iranica*, *19*(3), 662–673. https://doi.org/10.1016/j.scient.2011.08.033

Alavi, S. B., Moteabbed, S., & Arasti, M. R. (2012b). Sharif University of Technology A qualitative investigation of career orientations of a sample of Iranian software engineers. *Scientia Iranica*, *19*(3), 662–673. https://doi.org/10.1016/j.scient.2011.08.033

André, M., Baldoquín, M. G., & Acuña, S. T. (2011). Formal model for assigning human resources to teams in software projects. *Information and Software Technology*, *53*, 259–275. https://doi.org/10.1016/j.infsof.2010.11.011

Andrews, J., & Higson, H. (2008). Graduate Employability, 'Soft Skills' Versus 'Hard' Business Knowledge: A European Study. *Higher Education in Europe*, *33*(4), 411–422. https://doi.org/10.1080/03797720802522627

Ardis, M., Budgen, D., Hislop, G. W., Offutt, J., Sebern, M., & Visser, W. (2014). *SE 2014: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering*. IEEE Computer Society.

Assyne, N. (2019). Hard competencies satisfaction levels for software engineers: A unified framework. *Lecture Notes in Business Information Processing*, *370 LNBIP*. https://doi.org/10.1007/978-3-030-33742-1_27

Assyne, N. (2020). Soft Competencies and Satisfaction Levels for Software Engineers: A Unified Framework. *Lecture Notes in Business Information Processing*, *371 LNBIP*. https://doi.org/10.1007/978-3-030-35510-4_5

Barreto, A., Barros, M. de O., & Werner, C. M. L. (2008). Staffing a software project: A constraint satisfaction and optimization-based approach. *Computers and Operations Research*, *35*(10), 3073–3089. https://doi.org/10.1016/j.cor.2007.01.010

Bhasskar, R. (2008). A Realistic Theory of Science. In *Taylor & Francis* (Vol. 24, Issue 5). Routledge. https://doi.org/10.2307/2215817

Bhattacherjee, A. (2012). Social Science Research: principles, methods, and practices. In *Textbooks collection* (Vol. 9). https://doi.org/10.1186/1478-4505-9-2

Bourque, P., & Fairley, R. E. (2014). Guide to the Software Engineering - Body of Knowledge (SWEBOK (R)): Version 3.0. In *IEEE Computer Society Press*. https://doi.org/10.1234/12345678

Broadbent, M., Lloyd, P., Hansell, A., & Dampney, C. N. G. (1992). Roles , Responsibilities and Requirements for Managing Information Systems in the 1990s. *International Journal of Information Managemet*, *72*(12), 21–38.

Bröker, K. (2014). Identification and measurement of computer science competencies in the area of software development, software engineering and programming. *Proceedings of the Tenth Annual Conference on International Computing Education Research - ICER '14*, 141–142. https://doi.org/10.1145/2632320.2632322

Budgen, D., & Brereton, P. (2006). Performing systematic literature reviews in software engineering. *Proceedings of the 28th International Conference on Software Engineering*, *45*(4ve), 1051–1052.

Calazans, A., Paldês, R., Masson, E., Rezende, K., Braosi, E., Perera, N., & Brito, I. S. (2017). Software Requirements Analyst Profile : A Descriptive Study of Brazil and Mexico. *2017 IEEE 25th International Requirements Engineering Conference*, 204–212. https://doi.org/10.1109/RE.2017.22

Casale, G., Chesta, C., Deussen, P., Di Nitto, E., Gouvas, P., Koussouris, S., Stankovski, V., Symeonidis, A., Vlassiou, V., Zafeiropoulos, A., & Zhao, Z. (2016). Current and Future Challenges of Software Engineering for Services and Applications. *Procedia Computer Science*, *97*, 34–42. https://doi.org/10.1016/j.procs.2016.08.278

CEN. (2014). *European e-Competence Framework 3.0 - A common European Framework for ICT Professionals in all industry sectors*.

Chang, J., Wang, T., Lee, M., Su, C.-Y., & Chang, P.-C. (2016). Impacts of Using Creative Thinking Skills and Open Data on Programming Design in a Computer-supported Collaborative Learning Environment. *2016 IEEE 16th International Conference on Advanced Learning Technologies*, 396–400. https://doi.org/10.1109/ICALT.2016.78

Charette, B. R. N. (2005). Why Software Fails We waste billions of dollars each year on entirely preventable mistakes Market Crash : After its new. *IEEE Spectrum*, *September*, 1–10.

Cheng, J., Goto, Y., Morimoto, S., & Horie, D. (2008). A Security Engineering Environment Based on ISO / IEC Standards : Providing Standard , Formal , and Consistent Supports for Design , Development , Operation , and Maintenance of Secure Information Systems. *2008 International Conference on Information Security and Assurance*, 350–354. https://doi.org/10.1109/ISA.2008.106

Chrissis, M. B., Konrad, M., & Shrum, S. (2011). *CMMI for Development, Guidlines for process integration and product improvement* (Third Edit).

Cole, R., Purao, S., Rossi, M., & Sein, M. (2005). Being proactive: where action research meets design research. *ICIS 2005 Proceedings*, 27.

Colomo-Palacios, R., Casado-Lumbreras, C., Soto-Acosta, P., García-Peñalvo, F. J., & Tovar-Caro, E. (2013a). Competence Gaps in Software Personnel: A

Multi-Organizational Study. *Computers in Human Behavior*, *29*(2), 456–461. https://doi.org/10.1016/j.chb.2012.04.021

Colomo-Palacios, R., Casado-Lumbreras, C., Soto-Acosta, P., García-Peñalvo, F. J., & Tovar-Caro, E. (2013b). Computers in Human Behavior Competence gaps in software personnel : A multi-organizational study. *Computers in Human Behavior*, *29*(2), 456–461. https://doi.org/10.1016/j.chb.2012.04.021

Colomo-Palacios, R., Tovar-Carlos, E., Garcia-Crespo, Å., & Gómez-Berbís, J. M. (2010). The Case of Software Engineers Identifying Technical Competences of IT Professionals : *International Journal of Human Capital and Information Technology Professionals*, *1*(March), 31–43. https://doi.org/10.4018/jhcitp.2010091103

Corley, K. G., & Gioia, D. A. (2011). Building theory about theory building: what constitutes a theoretical contribution? *Academy of Manaagement Review*, *36*(1), 12–32. https://doi.org/10.5465/AMR.2011.55662499

Curtis, B., Hefley, B., & Miller, S. A. (2009). People Capability Maturity Model (P-CMM) Version 2 . 0 , Second Edition. In *Business Process Management Journal* (Issue July). https://doi.org/Report CMU/SRI-2001-MM-001

Data, E. (2019). Global Developer Population and Demographic Study 2019. In *Evans Data Corporation, Tech. Rep.* (Vol. 1).

Debois, P. (2011). Devops: A software revolution in the making. *Journal of Information Technology Management*, *24*(8), 3–39.

Dyba, T., & Dingsoyr, T. (2008). Empirical Studies of Agile Software Development : A Systematic Review. *Information and Software Technology*, *50*, 833–859. https://doi.org/10.1016/j.infsof.2008.01.006

Engelbrecht, L., Landes, D., & Sedelmaier, Y. (2018). A Didactical Concept for Supporting Reflection in Software Engineering Education. *2018 IEEE Global Engineering Education Conference (EDUCON)*, 547–554. https://doi.org/10.1109/EDUCON.2018.8363278

Frezza, S., Daniels, M., Pears, A., Cajander, A., Viggo, K., Kapoor, A., Mcdermott, R., Peters, A.-K., Sabin, M., & Wallace, C. (2018). Modelling Competencies for Computing Education beyond 2020 : A Research Based Approach to Defining Competencies in the Computing Disciplines. *23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*, 148–174.

Gangurde, S., & Patil, S. (2018). Benchmark product features using the Kano-QFD approach: a case study. *Benchmarking: An International Journal*, *25*(2), 450–470. https://doi.org/http://dx.doi.org/10.1108/MRR-09-2015-0216

Gimenes, I. M. S., Barroca, L., & Barbosa, E. F. (2012). The Future of Human Resources Qualifications in Software Engineering – Meeting Demands from Industry and Benefiting from Educational and Technological Advances. *2012 26th Brazilian Symposium on Software Engineering*, 181–185. https://doi.org/10.1109/SBES.2012.19

Goel, S. (2006). Competency Focused Engineering Education with Reference to IT Related Disciplines: Is the Indian System Ready for Transformation?

Journal of Information Technology Education, *5*, 27–52. https://doi.org/10.28945/233

Harris, K. S., & Rogers, G. E. (2008). Soft Skills in the Technology Education Classroom : What Do Students Need. *Technology Teacher*, *68*(3), 19–25.

Harzallah, M., Giuseppe, B., & Vemadat, F. (2002). A Formal Model for Assessing Individual Competence in Enterprises. *IEEE International Conference on Systems, Man and Cybernetics*, 6. https://doi.org/10.1109/ICSMC.2002.1173300

Harzallah, M., & Vernada, F. (2002). IT-Based Competency Modeling and Management : from Theory to Practice in Enterprise Engineering and Operations. *Computer in Industry*, *48*, 157–179.

Havelka, D., & Mermout, Jeffrey, W. (2009). Toward a Theory of Iformation Technology professional Competence. *Jounal of Computer Information Systems*, *Winter 2009*.

Hilburn, T., Ardis, M., Johnson, G., Kornecki, A. J., & Mead, N. R. (2013). *Software Assurance Competency Model* (Issue March). http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=47953

Holtkamp, P., Jokinen, J. P. P., & Pawlowski, J. M. (2015). Soft Competency Requirements in Requirements Engineering, Software Design, Implementation, and Testing. *Journal of Systems and Software*, *101*, 136–146. https://doi.org/10.1016/j.jss.2014.12.010

Hsieh, H.-F., & Shannon, S. E. (2005). Three Approaches to Qualitative Content Analysis. *Qualitative Health Research*, *15*(9), 1277–1288. https://doi.org/10.1177/1049732305276687

Huang, J. (2018). *Application of Kano model and IPA on improvement of service quality of mobile healthcare Jui-Chen Huang*. *16*(2).

Hubwieser, P., Berges, M., Magenheim, J., Schaper, N., Bröker, K., Margaritis, M., Schubert, S., & Ohrndorf, L. (2013). Pedagogical content knowledge for computer science in German teacher education curricula. *Computing Education*, 95–103. https://doi.org/10.1145/2532748.2532753

Humphrey, W. S. (1989). *Managing the Software Process*. Addson Wesley.

IEEE-CS, & ACM. (2015). Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering. In *Computing Curricula Series* (Issue February). http://usir.salford.ac.uk/6939/

IEEE. (2014). *Software Engineering Competency Model (SWECOM)*. IEEE. http://www.dahlan.web.id/files/ebooks/SWECOM.pdf

Johnson, P., & Ekstedt, M. (2015). The Tarpit - A general theory of software engineering. *Information and Software Technology*, *70*, 181–203. https://doi.org/10.1016/j.infsof.2015.06.001

Johnson, P., Ekstedt, M., & Jacobson, I. (2012). Where's the theory for software engineering? *IEEE Software*, *29*(5), 94–95. https://doi.org/10.1109/MS.2012.127

Kalliamvakou, E., Member, S., Bird, C., Zimmermann, T., Begel, A., Deline, R., & German, D. M. (2019). What Makes a Great Manager of Software Engineers ? *IEEE Transactions on Software Engineering*, *45*(1), 87–106.

Kano. (2016). *What is the Kano Model? KanoModel.com*. https://kanomodel.com/, retrieved 19.04.2018

Kano, N., Seraku, N., Takahashi, F., & Tsuji, S. (1984). Kano. *Attractive Quality and Must-Be Quality. The Journal of the Japanese Society for Quality Control*, *14*, 39–48.

Kaplan, B., & Maxwell, J. A. (1994). Qualitative Research Methods for Evaluating Computer Information Systems. In J. G. Anderson, C. E. Aydin, & S. J. Jay (Eds.), *Evaluating Health Care Information Systems: Methods and Applications* (pp. 45–68). SAGE. https://doi.org/10.1007/0-387-30329-4

Kobata, K., Uesugi, T., Adachi, H., & Aoyama, M. (2015). A curriculum development methodology for professional software engineers and its evaluation. *Proceedings of IEEE International Conference on Teaching, Assessment and Learning for Engineering: Learning for the Future Now, TALE 2014*, *December*, 480–487. https://doi.org/10.1109/TALE.2014.7062552

Kobata, K., Uesugi, T., Adachi, H., & Aoyama, M. (2013). Software Engineering Education Program for Software Professionals of High Competency at. *2013 20th Asia-Pacific Software Engineering Conference (APSEC)*, *2*, 117–122. https://doi.org/10.1109/APSEC.2013.125

Krippendorff, K. (2018). *Content Analysis: An Introduction to Its Methodology* (Fourth Edi). SAGA Publications.

Kropp, M., Meier, A., & Perellano, G. (2016). Experience Report of Teaching Agile Collaboration and Values Agile Software Development in Large Student Teams. *2016 IEEE 29th International Conference on Software Engineering Education and Training (CSEET)*, 76–80. https://doi.org/10.1109/CSEET.2016.30

Le Deist, F. D., & Winterton, J. (2005). What is competence? *Human Resource Development International*, *8*(1), 27–46. https://doi.org/10.1080/1367886042000338227

Lee, B., & Cassell, C. (2013). Research methods and research practice: History, themes and topics. *International Journal of Management Reviews*, *15*(2), 123–131. https://doi.org/10.1111/ijmr.12012

Lee, Y. C., Sheu, L. C., & Tsou, Y. G. (2008). Quality function deployment implementation based on Fuzzy Kano model: An application in PLM system. *Computers and Industrial Engineering*, *55*(1), 48–63. https://doi.org/10.1016/j.cie.2007.11.014

Lehtola, L., & Kauppinen, M. (2006). Suitability of requirements prioritization methods for market-driven software product development. *Software Process Improvement and Practice*, *11*(1), 7–19. https://doi.org/10.1002/spip.249

Lenberg, P., Feldt, R., & Wallgren, L. G. (2015). Behavioral Software Engineering: A Definition and Systematic Literature Review. *Journal of Systems and Software*, *107*, 15–37. https://doi.org/10.1016/j.jss.2015.04.084

León-sigg, M. De, Pérez-valenzuela, B. J., Vázquez-reyes, S., & Cisneros, J. L. V. (2018). Adaptation of the Initial Software Development Method for a

Single Developer. *6th International Conference in Software Engineering Research and Innovation*. https://doi.org/10.1109/CONISOFT.2018.00013

Liu, X. F. (2000). Software quality function deployment. *Potentials, IEEE*, *19*(5), 14–16. https://doi.org/10.1109/45.890072

Magenheim, J., Nelles, W., Rhode, T., Schaper, N., & Schubert, S. (2010). Competencies for Informatics Systems and Modeling Results of Qualitative Content Analysis of Expert Interviews. *IEEE EDUCON 2010 Conference*, 513–521. https://doi.org/10.1109/EDUCON.2010.5492535

Manawadu, C. D., Johar, M. G. M., & Perera, S. S. N. (2015). Essential Technical Competencies for Software Engineers : Perspectives from Sri Lankan Undergraduates. *International Journal of Computer Applications*, *113*(17), 27–34. https://pdfs.semanticscholar.org/c01d/90376b8d36bea073190aee76f77ec883f1f6.pdf

Mano, C. D., Duhadway, L., & Striegel, A. (2006). A Case for Instilling Security as a Core Programming Skill. *Proceedings. Frontiers in Education. 36th Annual Conference*, 13–18. https://doi.org/10.1109/FIE.2006.322347

Mason, J. (2002). Qualitative Researching. In *Qualitative Research Journal* (Vol. 41, Issue 1). https://doi.org/10.1159/000105503

McFeeley, B. (1996). IDEAL: User's Guide for Software Process Improvement. In *Cmu/Sei-96-Hb-001*.

Mead, N. R., & Shoemaker, D. (2013). The Software Assurance Competency Model: A Roadmap to Enhance Individual Professional Capability. *Software Engineering Education Conference, Proceedings*, 119–128. https://doi.org/10.1109/CSEET.2013.6595243

Moreno, A. M., Sanchez-segura, M., Medina-dominguez, F., & Carvajal, L. (2012). Balancing Software Engineering Education and Industrial Needs. *The Journal of Systems & Software*, *85*(7), 1607–1620. https://doi.org/10.1016/j.jss.2012.01.060

Myers, M. D. (1997). Qualitative research in information systems. *Management Information Systems Quarterly*, *21*(June), 1–18. https://doi.org/10.2307/249422

Myers, M. D., & Newman, M. (2007). The qualitative interview in IS research: Examining the craft. *Information and Organization*, *17*(1), 2–26. https://doi.org/10.1016/j.infoandorg.2006.11.001

Nandhakumar, J., & Jones, M. (1997). Too close for comfort? Distance and engagement in interpretive information systems research. *Information Systems Journal*, *7*(2), 109–131. https://doi.org/10.1046/j.1365-2575.1997.00013.x

Nelson, R. (2007). IT Project Management: Infamous Failures, Classic Mistakes, and Best Practices. *MISQE*, *6*(2), 67–78.

Object Management Group. (2018). Essence - Kernel and Language for Software Engineering Methods (Essence). In *OGM* (Issue Versión 1.2). https://www.omg.org/spec/Essence/1.2

Orlikowski, W. J., & Baroudi, J. J. (1991). Studying information technology in organizations: Research approaches and assumptions. *Information Systems Research*, *2*(1), 1–28. https://doi.org/10.1287/isre.2.1.1

Orsoni, A., & Colaco, B. (2013). A Competency Framework for Software Development Organizations. *2013 UKSim 15th International Conference on Computer Modelling and Simulation*, 507–511. https://doi.org/10.1109/UKSim.2013.101

Päivärinta, T., & Smolander, K. (2015). Theorizing about software development practices. *Science of Computer Programming*, *101*, 124–135. https://doi.org/10.1016/j.scico.2014.11.012

Pawlowski, J. M., & Holtkamp, P. (2012). Towards an internationalization of the information systems curriculum. *Multikonferenz Wirtschaftsinformatik 2012 - Tagungsband Der MKWI 2012*, *2011*(March), 437–449. https://www.scopus.com/inward/record.uri?eid=2-s2.0-84879862734&partnerID=40&md5=377ec064ba5a4994dcda2fb9148041fa

Pawlowski, J. M., Holtkamp, P., & Kalb, H. (2010). Globalization Competences in Information Systems and E-Learning. In *Lecture Notes in Business Information Processing*.

Peffers, K., Tuunanen, T., Rothenberger, M., & Chatterjee, S. (2008). A Design Science Research Methodology for Information Systems Research. *Journal of Management Information Systems*, *24*(3), 45–77. https://doi.org/10.2753/MIS0742-1222240302

Pérez, J., Vizcarro, C., García, J., Bermúdez, A., & Cobos, R. (2017). Development of Procedures to Assess Problem-Solving Competence in Computing Engineering. *IEEE TRANSACTIONS ON EDUCATION*, *60*(1), 22–28. https://doi.org/10.1109/TE.2016.2582736

Petersen, K., Feldt, R., Mujtaba, S., & Mattsson, M. (2008). Systematic Mapping Studies in Software Engineering. *12Th International Conference on Evaluation and Assessment in Software Engineering*, *17*, 10. https://doi.org/10.1142/S0218194007003112

Petersen, K., Vakkalanka, S., & Kuzniarz, L. (2015). Guidelines For Conducting Systematic Mapping Studies in Software Engineering : An Update. *Information and Software Technology*, *64*, 1–18. https://doi.org/10.1016/j.infsof.2015.03.007

Piaszczyk, C. (2011). Model Based Systems Engineering with Department of Defense Architectural Framework. *Systems Engineering*, *14*(3), 305–326. https://doi.org/10.1002/sys

Pyster, A. (Editor). (2009). *Graduate Software Engineering 2009 (GSwE2009) Curriculum Guidelines for Graduate Degree Programs in Software Engineering: Vol. Integrated*.

Radermacher, A., Walia, G., & Knudson, D. (2014). Investigating the skill gap between graduating students and industry expectations. *Companion Proceedings of the 36th International Conference on Software Engineering - ICSE Companion 2014*, 291–300. https://doi.org/10.1145/2591062.2591159

Rainsbury, E., Hodges, D., Burchell, N., & Lay, M. (2002). *Ranking Workplace Competencies: Student and Graduate Perceptions*. 8–18.

Richardson, I. (2001). Software Process Matrix: A Small Company SPI Model. *Software Process: Improvement and Practice*, *6*(Daft 1992), 157–165. https://doi.org/10.1002/spip.144

Rivera-Ibarra, J. G., Rodríguez-Jacobo, J., & Serrano-Vargas, M. A. (2010). Competency Framework for Software Engineers. *2010 23rd IEEE Conference on Software Engineering Education and Training*, 33–40. https://doi.org/10.1109/CSEET.2010.21

Robinson, M. A., Sparrow, P. R., Clegg, C., & Birdi, K. (2005). Design engineering competencies: Future requirements and predicted changes in the forthcoming decade. *Design Studies*, *26*(2), 123–153. https://doi.org/10.1016/j.destud.2004.09.004

Robson, C. (2002). *Research Real World* (Second Edi). Blackwell Publishing.

Runeson, P., & Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, *14*(2), 131–164. https://doi.org/10.1007/s10664-008-9102-8

Saiedian, H., & Dale, R. (2000). Requirements Engineering : Making the Connection Between the Software Developer and Customer. *Information and Software Technology*, *42*, 419–428.

Saldaña-Ramos, J., Sanz-Esteban, A., García-Guzmán, J., & Amescua, A. (2012). Design of a Competence Model for Testing Teams. *IET Software*, *6*(5), 405–415. https://doi.org/10.1049/iet-sen.2011.0182

Sambamurthy, V., Bharadwaj, A., & Grover, V. (2003). Shaping Agility Through Digital Options: Reconceptualizing The Role Of Information Technology In Contemporary Firm. *MIS Quartely*, *27*(2), 237–263.

Schultze, U., & Avital, M. (2011). Designing interviews to generate rich data for information systems research. *Information and Organization*, *21*(1), 1–16. https://doi.org/10.1016/j.infoandorg.2010.11.001

Sedelmaier, Y., & Landes, D. (2013). A Research Agenda for Identifying and Developing Required Competencies in Software Engineering. *International Journal of Engineering Pedagogy*, *3*(2), 0–4.

Sedelmaier, Y., & Landes, D. (2014a). A Multi-Perspective Framework for Evaluating Software Engineering Education by Assessing Students ' Competencies SECAT – A Software Engineering Competency Assessment Tool. *IEEE Frontiers in Education Conference (FIE)*.

Sedelmaier, Y., & Landes, D. (2012). A Research Agenda for Identifying and Developing Required Competencies in Software Engineering. *2012 15th International Conference on Interactive Collaborative Learning (ICL)*, *01*, 1–5. https://doi.org/10.1109/ICL.2012.6402195

Sedelmaier, Y., & Landes, D. (2014b). Software Engineering Body of Skills (SWEBOS). *IEEE Global Engineering Education Conference, EDUCON*, *April*, 395–401. https://doi.org/10.1109/EDUCON.2014.6826125

Silveira Neto, P. A. D. M., Gomes, J. S., De Almeida, E. S., Leite, J. C., Batista, T. V., & Leite, L. (2013). 25 years of software engineering in Brazil: Beyond an

insider's view. *Journal of Systems and Software*, *86*(4), 872–889. https://doi.org/10.1016/j.jss.2012.10.041

Software Engineering Institute. (2010). CMMI for Development, Version 1.3. In *Software Engineering Process Management Program* (Issue November).

Studt, R., Winterfeldt, G., & Mottok, J. (2015). Measuring Software Engineering Competencies. *IEEE Global Engineering Education Conference, EDUCON*, *March*, 908–914. https://doi.org/10.1109/EDUCON.2015.7096081

Suhartono, J., & Sudirwan, J. (2016). Academic Competence of Computer Science Graduate Degree from the Employer ' s Perspective. *2016 International Conference on Information Management and Technology (ICIMTech)*, *November*, 176–181. https://doi.org/10.1109/ICIMTech.2016.7930325

Tahvanainen, S., & Luoma, E. (2018). Examining the Competencies of the Chief Digital Officer. *Twenty-Fourth Americas Conference on Information Systems*, *2009*, 1–10.

Thurner, V., Axel, B., & Andreas, K. (2014). Identifying Base Competencies as Prerequisites for Software Engineering Education. *IEEE Global Engineering Education Conference (EDUCON)*, *April*, 1069–1076. https://doi.org/10.1109/EDUCON.2014.6826240

Thurner, V., Schlierkamp, K., Bottcher, A., & Zehetmeier, D. (2016). Integrated Development of Technical and Base Competencies: Fostering Reflection Skills in Software Engineers to be. *IEEE Global Engineering Education Conference, EDUCON*, *April*, 340–348. https://doi.org/10.1109/EDUCON.2016.7474576

Trivellas, P., & Reklitis, P. (2014). Leadership Competencies Profiles and Managerial Effectiveness in Greece. *Procedia Economics and Finance*, *9*(Ebeec 2013), 380–390. https://doi.org/10.1016/S2212-5671(14)00039-2

Turley, R. T. (1991). *Essential Competencies of Exceptional Professional Software Engineers* (Issue July). Colorado State University.

Turley, R. T., & Bieman, J. M. (1994). Identifying essential competencies of software engineers. *Proceedings of the 22nd Annual ACM Computer Science Conference on Scaling up : Meeting the Challenge of Complexity in Real-World Computing Applications - CSC '94*, 271–278. https://doi.org/10.1145/197530.197637

Turley, T., & Bieman, M. (1995). Competencies Nonexceptional of Exceptional and Software Engineers. *J. Systems Software*, *1995:28*(28), 19–38.

Tyrväinen, P., Warsta, J., & Seppänen, V. (2008). Evolution of secondary software businesses: Understanding industry dynamics. In *IFIP International Federation for Information Processing: Moving Towards Cooperative IT Transfer and Knowledge Diffusion* (Vol. 287, pp. 381–401). Springer. https://doi.org/10.1007/978-0-387-87503-3_22

Urs, D. S. (2013). Soft Skills for the Engineering Students. *Synergy*, *9*(2), 137–142. https://pdfs.semanticscholar.org/fbf4/0a9446331b41d3f11ef3b7035fe33e2b452f.pdf

Vaismoradi, M., Turunen, H., & Bondas, T. (2013). Content Analysis and Thematic Analysis : Implications for Conducting a Qualitative Descriptive Study. *Nursing and Health Science*, *15*, 398–405. https://doi.org/10.1111/nhs.12048

Venkatesh, V., Morris, M. G., Davis, G. B., Davis, F. D., Cole, R., Purao, S., Rossi, M., Sein, M. K., Dyer, W. G., Wilkins, a. L., Easton, G., Eisenhardt, K. M., Academy, T., Review, M., Freeze, R., Raschke, R., Germonprez, M., Hovorka, D., Gal, U., … Walsham, G. (2003). Management Information Systems Research Center, University of Minnesota. *MIS Quarterly*, *27*(3), 533–556. https://doi.org/10.2753/MIS0742-1222240302

Vincent, S., & O'Mahoney, J. (2018). Critical Realism and Qualitative Research: An Introductory Overview. In *The SAGE Handbook of Qualitative Business and Management Research Methods: History and Traditions* (pp. 201–216). https://doi.org/10.4135/9781526430212.n13

Wernerfelt, B. (1984). A Resource-Based View of the Firm. *Strategic Management Journal*, *5*(2), 171–180. https://doi.org/10.1002/smj.4250050207

West, M. (2004). *Real Process Improvement Using the CMMI* (Taylor & F). AUERBACH PUBLICATIONS.

Wohlin, C., & Aurum, A. (2015). Towards a decision-making structure for selecting a research design in empirical software engineering. *Empirical Software Engineering*, *20*(6), 1427–1455. https://doi.org/10.1007/s10664-014-9319-7

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., & Wesslén, A. (2012). Experimentation in Software Engineering. In *Springer Science & Business Media.* https://doi.org/10.1037/12345-011

Zendler, A., Klaudt, D., & Seitz, C. (2014). Empirical Determination of Competence Areas to Computer Science Education. *Journal of Educational Computing Research*, *51*(1), 71–89. https://doi.org/10.2190/EC.51.1.d

ORIGINAL PAPERS

I

THE STATE OF RESEARCH ON SOFTWARE ENGINEERING COMPETENCIES: A SYSTEMATIC MAPPING STUDY

by

Nana Assyne, Hadi Ghanabari & Mirja Pulkkinen

Journal of Systems and Software, under revision.

Request a copy from the author.

II

HARD COMPETENCIES SATISFACTION LEVELS FOR SOFTWARE ENGINEERS: A UNIFIED FRAMEWORK

by

Nana Assyne, 2019

Lecture Notes in Business Information Processing vol 370, LNBIP

https://doi.org/10.1007/978-3-030-33742-1_27

Hard Competencies Satisfaction Levels for Software Engineers: A Unified framework

Nana Assyne

Faculty of Information Technology, University of Jyväskylä, Finland
nana.m.a.assyne@student.jyu.fi

Abstract. Software engineer's/developer's competency has long been established as a key pillar for the development of software. Nevertheless, the satisfaction levels derived from using a competency needs more investigation. The aim of this paper is to propose a framework for identifying hard competencies and their satisfaction levels. The paper contributes to the software engineering competency research by highlighting the satisfaction levels of hard competence for the benefit of the educators (academia), software engineers and users of software competence (practitioner).

Keywords: Hard competency, Technical competency, Software engineers' competencies, Competence satisfaction levels.

1 Introduction

Software are the principal driving force for hardware that currently run our daily lives. As such software development is propelled by the competency of the software developers. Competency is said to be the combination of abilities, knowledge, and skills for performing an assigned task. Competency then includes both soft and hard competencies [1]: a hard skill is or are the skill(s) one needs to be able to perform a job or assignment. Hard skills are teachable and acquired mostly through formal training and studies, and are sometimes referred to as technical skill. Often for example a trainee is required to be smart or must possess a good IQ to acquire the required skill. Thus, hard/technical skills are pre-requisite skills required by software engineers/developer in software development process.

Whereas both practical and empirical knowledge on technical competencies of software developers is not lacking, competency study has become an important and fundamental strategic area for academic research. Colomo-palacios et al. identify the competency levels relevant to software engineering of professional profiles [2]. Turley and Bieman in an attempt to identify non-exceptional and exceptional competencies of software engineers, also provided the technical competencies of software engineers [3]. Yet – there is paucity of studies that examines the satisfaction levels derived for possessing or using a competence.

Though the works of [4]; [2] and [5] establish the essence of hard or technical competence to software development, if we do not know the satisfaction level derived

as assurance for the possessor or the user, beneficiary cannot know which competency will be demanded or be needed. Our initial study looked at [2] work, which examined relevant levels of profile of software engineers and professional. Also the work of [6] assesses base competencies necessary for software engineering students. We do agree with the said work and argue further that it gives credence to the software engineering competency. However, we are of the view that additional satisfaction levels of the competency will provide assurance for both possessor and users in the software engineering community. Thus, there is a need to provide strategic frameworks for the various satisfaction levels of hard or technical competencies of software developers. This paper forms part of broader research on software developer's competency study.

The goal of this paper is to use existing models to create classification levels for the benefit of the users and possessors of software engineering competency. We therefore set our research question as: how do we determine the benefit or satisfaction of a competency of technical or hard competencies for software developers, thus, the research question for this paper is:

> *What are the different satisfaction levels derived from using a software technical or hard competency?*

Research on software competency is not necessarily lacking in software engineering studies [7], however, in this study the Kano model, which is the main framework for this study is being used for the first time on competencies as against it original use on products. To structure this study to fit into previous studies for practical use, we also made use of Competency Framework for Software Engineers (CFSE) [8]. The framework has two main areas, that is soft and hard competency. Since this paper focuses on hard competency, we make use of that as part of the framework. This paper, is structured as follow: section 2 discusses the theoretical foundations, section 3, methodology and the proposed framework, section 4, conclusions and future work.

2 Theoretical foundations

2.1 Kano model

The Kano model provides a quality function-deployment framework that aids products or service developers to take into consideration the customer's voice and preferences in the development phase instead of a passive approach of only developers [9–15] employed the Kano model for ICT system development and established that the model highlights user involvement. The model assists in determining basic, performance and delighters of a product or service.

In this paper, we conceptualize the customer as the software community (organization using the competencies) and the product or service as the needed competency. According to Kano et al. [16], customer's decision-making options on product or service acquisition, are founded on conscious and subconscious deliberations. For effective product and or service development there is the need to understand these deliberative conscious and subconscious processes of decision-

making. Kano et al.'s categorization of these processes into three-requirement levels (basic, performance and delighters) is relevant. For instance, basic requirements emanate from customer's expectations about a product or service, since their presence are immutable to influence customer options and opinion about the product. However, their absence may result in complaints from the customer. By extension, performance requirements, are expected pre-requisites knowledge factor vital in influencing customer decision-making options. These are critical pre-requisite requirements when appropriately adopted yields high levels of satisfaction. Meanwhile, at the delighter level, product and service developers are required to include surprise elements often referred to as 'wow' factors to entice, attract and influence customer choice options and preferences [16].

2.2 Competence framework for software engineers

Competency Framework for Software Engineers (CFSE) is a framework proposed by [17]. It identifies the training needs of software community and also serves as a guide for competency identification. The framework is divided into two main categories with sub-categories under main categories. The main categories are hard and soft competency. The soft competency category has socials and personals. The hard competency category has subcategories similar to roles for software development identified in SWEBOK. These includes project management, requirement analysis, software design, programming, validation and verification tests, configuration management, quality, tests, documentation and maintenance.

Our study, forms part of a broader software engineering competency study, which aims at creating classification maps for the satisfaction levels of software engineers' competencies. Specifically, in this paper, we focus on hard competency. Since CFSE serve the purpose of identifying hard soft competencies, we make use of the hard category side. This framework provides a granularity which align closes with the roles of software engineering. Thus, we make use of hard category aspect and the kano model to create our desired framework for the study. The result will be a unified framework to identify and classify the satisfaction levels of hard competencies for the use of the software engineering community.

3 Methodology and proposed framework

According to [18] framework as design science artifact requires some iteration in the validation of the process in developing. Justification for the need of the artifact has been presented through using literature, but it also requires stakeholder input, Thus, we present the proposed model for validation in this conference.

3.1 Propose Model: A Unified framework of Hard competency satisfaction levels for software engineers (UFHCSL)

This framework originates in the Kano model and CFSE. The Kano model as quality function-deployment model has been used for research work in software engineering.

Our study is the first to apply the Kano model on human resources as a means to determine the competency satisfaction levels of software engineers. CFSE is a framework for identifying competencies of software engineers, and there are more compatible frameworks available, such as [19–22] which provide a means to identify competencies of software developers. However, in line with our objectives, the CFSE frame work provides required granularity and align with the roles of software engineering, we think the work of Rivera-ibarra et al. (2010) is suitable for our objectives.

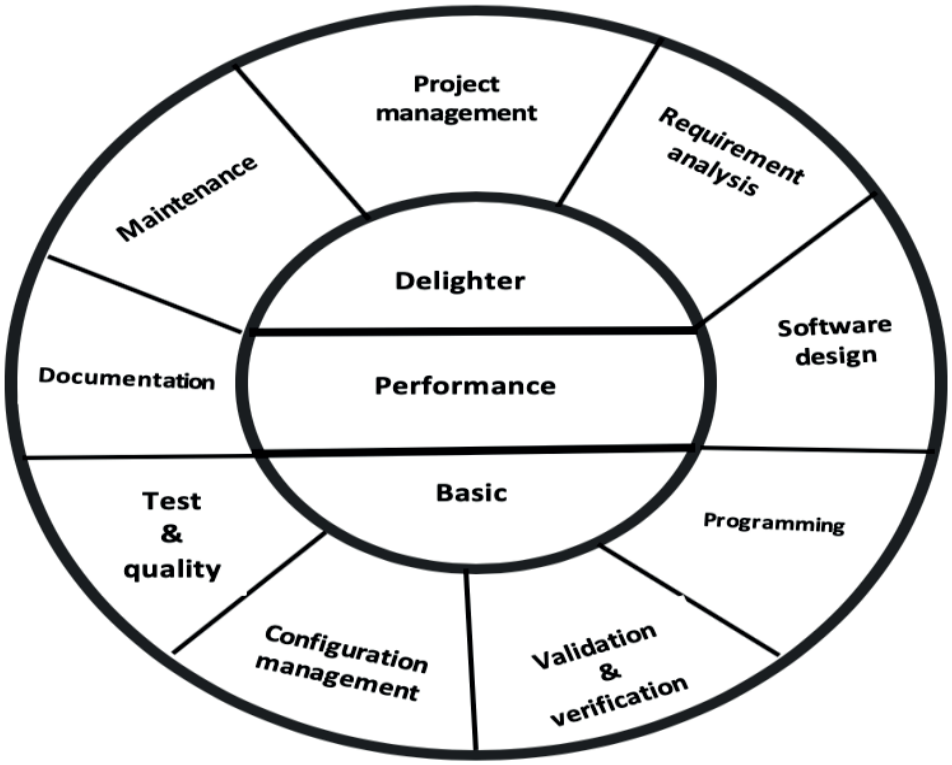


Figure 1. Unified framework of hard competency satisfaction levels for software engineers

To use the presented framework (UFHCSL), hard competencies are identified and classified using the hard category in [17] framework, followed by competency identification or classification subjected to the metrics of Kano model (we provide the metrics as table 1) to determines its satisfaction levels. The Categorization metrics is divided into three main parts (satisfaction levels): 1) basic, 2) performance and 3) delighter competencies. In each part a number of parameters are considered e.g. socials (interpersonal relations, cooperation and work in a team, and handling and conflicts resolution) and personals (development in the job, personal development, rights and limits).

4 Conclusion and future work

The proposed framework UFHCSL uses the kano model and the CFSE framework to create framework that can be use to identify hard competencies of software developers, their satisfaction levels and the most valued competencies of the developers. This framework add to the work of [23]. Thus, we have provided a framework that can be beneficial to educators, competency users, and possessors of hard competencies. The future work will be to use empirical data to evaluated the framework.

References

1. Sedelmaier, Y., & Landes, D. (2014). Software Engineering Body of Skills (SWEBOS). In *2014 IEEE Global Engineering Education Conference (EDUCON)* (pp. 395–401)
2. Colomo-palacios, R., Carlos, U., & Madrid, I. I. I. De. (2010). The Case of Software Engineers Identifying Technical Competences of IT Professionals : *International Journal of Human Capital and Information Technology Professionals*, *1*(March), 31–43
3. Turley, T., & Bieman, M. (1995). Competencies Nonexceptional of Exceptional and Software Engineers. *J. Systems Software*, *1995:28*(28), 19–38.
4. Patel, A., Benslimane, Y., Bahli, B., & Yang, Z. (2012). Addressing IT Security in Practice : Key Responsibilities , Competencies and Implications on Related Bodies of Knowledge. In *2012 IEEE International Conference on Industrial Engineering and Engineering Management* (pp. 899–903)
5. Manawadu, C. D., Johar, M. G. M., & Perera, S. S. N. (2015). Essential Technical Competencies for Software Engineers : Perspectives from Sri Lankan Undergraduates. *International Journal of Computer applications*, *113*(17), 27–34
6. Thurner, V., Axel, B., & Andreas, K. (2014). Identifying Base Competencies as Prerequisites for Software Engineering Education. In *IEEE Global Engineering Education Conference (EDUCON)* (pp. 1069–1076)
7. Lenberg, P., Feldt, R., & Wallgren, L. G. (2015). Behavioral software engineering: A definition and systematic literature review. *Journal of Systems and Software*, *107*, 15–37
8. Holtkamp, P., Jokinen, J. P. P., & Pawlowski, J. M. (2015). Soft competency requirements in requirements engineering, software design, implementation, and testing. *Journal of Systems and Software*, *101*, 136–146
9. Lee, Y. C., Sheu, L. C., & Tsou, Y. G. (2008). Quality function deployment implementation based on Fuzzy Kano model: An application in PLM system. *Computers and Industrial Engineering*, *55*(1)
10. Gangurde, S., & Patil, S. (2018). Benchmark product features using the Kano-QFD approach: a case study. *Benchmarking: An International Journal*, *25*(2), 450–470
11. Huang, J. (2018). Application of Kano model and IPA on improvement of

service quality of mobile healthcare Jui-Chen Huang, *16*(2).

12. Lehtola, L., & Kauppinen, M. (2006). Suitability of requirements prioritization methods for market-driven software product development. *Software Process Improvement and Practice*, *11*(1), 7–19
13. Liu, X. F. (2000). Software quality function deployment. *Potentials, IEEE*, *19*(5), 14–16
14. Piaszczyk, C. (2011). Model Based Systems Engineering with Department of Defense Architectural Framework. *Systems Engineering*, *14*(3), 305–326
15. Richardson, I. (2001). Software Process Matrix: A Small Company SPI Model. *Software Process: Improvement and Practice*, *6*(Daft 1992), 157–165
16. Kano, N., Seraku, N., Takahashi, F., & Tsuji, S. (1984). Kano. *Attractive Quality and Must-Be Quality. The Journal of the Japanese Society for Quality Control*, *14,* 39–48.
17. Rivera-ibarra, J. G., Rodríguez-jacobo, J., Fernández-zepeda, J. A., & Serrano-vargas, M. A. (2010). Competency Framework for Software Engineers and. In *2010 23rd IEEE Conference on Software Engineering Education and Training* (pp. 33–40)
18. Peffers, K., Tuunanen, T., Rothenberger, M., & Chatterjee, S. (2008). A Design Science Research Methodology for Information Systems Research. *J. Manage. Inf. Syst.*, *24*(3), 45–77
19. Linck, B., Ohrndorf, L., Kiel, T. D. L., Magenheim, J., & Neugebauer, J. (2013). Competence model for informatics modelling and system comprehension. In *2013 IEEE Global Engineering Education Conference (EDUCON)* (pp. 85–93)
20. Tuffley, D. (2012). Optimising virtual team leadership in Global Software Development. *IET Software*, *6*(March 2011), 176–184
21. André, M., Baldoquín, M. G., & Acuña, S. T. (2011). Formal model for assigning human resources to teams in software projects. *Information and Software Technology*, *53*, 259–275
22. Schulte, C., Magenheim, J., Kathrin, M., & Budde, L. (2017). The Design and Exploration Cycle as Research and Development Framework in Computing Education. In *2017 IEEE Global Engineering Education Conference (EDUCON)* (pp. 867–876)
23. Rivera-Ibarra, J. G., Rodríguez-Jacobo, J., & Serrano-Vargas, M. A. (2010). Competency Framework for Software Engineers. In *2010 23rd IEEE Conference on Software Engineering Education and Training* (pp. 33–40) 1

III

SOFT COMPETENCIES SATISFACTION LEVELS FOR SOFTWARE ENGINEERS: A UNIFIED FRAMEWORK

by

Nana Assyne, 2020

Lecture Notes in Business Information Processing vol 371, LNBIP

https://doi.org/10.1007/978-3-030-35510-4_5

Soft Competencies and Satisfaction Levels for Software Engineers: A Unified framework

Nana Assyne

Faculty of Information Technology
University of Jyväskylä, Jyväskylä, Finland
nana.m.a.assyne@student.jyu.fi

Abstract. The importance of software engineers' competency has long been established as a key pillar for the development of robust software in order to achieve quality software. Software engineering competency research is not necessarily lacking. Nevertheless, the satisfaction derived from using software competency needs more investigation. The aim of this study is to identify soft competencies from empirical data and create satisfaction levels for software engineers' soft competencies. The result shows 63 soft competencies with three different satisfaction levels consisting of basic, performance and delighters. The paper contributes to the SEC research by highlighting the satisfaction levels of soft competency for the benefit of the educators (academia), software engineers (possessor) and users of software competency (practitioner).

Keywords: Soft competency, Software engineers' competencies, Competency satisfaction levels, Essential competencies.

1 Introduction

The competencies of software engineers have long been recognized as essential for the development of efficient and robust software [1]. According to IEEE software engineering competency is defined as the knowledge, skills and attitudes of software developers to fulfill a task in a software development project [2]. This includes both soft and hard competencies [3]. Lenberg et al. pointed out that research work on software engineering competency (SEC) is not necessarily lacking. Yet, most of the earlier research on SEC focused on technical or hard competencies as against soft or behavioral competencies [4]. Harris & Rogers, define soft skills or competencies as "work ethics, positive attitude, social grace, facility with language, friendliness, integrity and the willingness to learn" [5, p.19]. Thus, the identification and use of soft competencies help in the development of complex software, because the software development involves a combination of soft and hard competencies [3, 6].

Works of authors such as Broadbent et al., Moreno et al., and Colomo-palacios et al. have established that soft competency is essential for development of software [6–8]. More importantly, recent literature suggests an increase in the number of software soft competencies studies with emphasis on identification of soft competencies [4].

Holtkamp et al. argued that, soft competencies are crucial for the development of global software engineering [9]. Nonetheless, the satisfaction levels of these competencies have not been adequately explored. Accordingly, this paper reports the identification and satisfaction levels of soft SEC as part of a bigger research on SEC.

The knowledge or identification of competencies is one phase of competency equation. The other phase is the benefit derived in the using such competency. The second phases have not received much attention in SEC research. Thurner et al., argue for minimum or base competency as a basic requirement for students of software engineering [10]. We support the base competency requirement and advocate for further investigations to determine the various levels of satisfaction of competencies. We therefore argue for satisfaction levels of soft competencies for software engineers, and state our research questions as:

> *RQ1: What are the different satisfaction levels derived from using a software soft competency?*
> *RQ2: Which of these soft competencies are perceived as most valuable for Software engineering?*

Knowledge of soft competencies and their satisfaction levels serve as insurance for users (people or organizations who use the competencies possessed by the developers to produce a product or a service), educator (people who train the developers to acquire the competencies), and the engineers' (people who receive training and therefore possesses some competencies). Therefore, we have adopt the Kano model [11] and Competency Framework for Software Engineers (CFSE) [12] as a lens to develop satisfaction rankings that can be employed by (i) the possessor of the competencies, (ii) users of the competencies and (iii) by the trainer of competencies possessors. The rest of the paper is presented as follows: section 2 looks at the background and related works and discuss the research models; section 3 discusses the methodology; section 4 presents the results; and section 5 and 6 looks at the discussions and conclusions respectively.

2 Theoretical foundation and Related Works

2.1 Soft competency

According to Harris & Rogers, soft skill is or are skills that mostly do not require formal training [5]. Until recently, these skills were mostly self-taught and self-developed. They are mostly not industry specific. In addition, they mostly require emotional intelligence [13][14] E.g. communication flexibility, leadership, motivation, patience, persuasion, problem-solving abilities, teamwork, time management, work ethics.

Soft competency connotes skills that complement technical skills; therefore, it cannot be overlooked in the development of software engineering. [They complement technical skills and thus cannot be overlooked in software engineering]. They are considered to be essential for global software projects [9, p.136]. (Broadbent et al.

established that the biggest skill gaps for software engineers were business strategies and marketing of their services [7]. This was emphasized by Moreno et al. [6]. Other studies have argued that more attention must be given to social and inter-personal competencies [15] and emotional intelligence [16].

In proposing a body of skills (SWEBOS) for software engineering, Sedelmaier and Landes identified and structured soft competencies of software engineers into three categories [3]. These include (i) comprehension of the complexity of software engineering processes, (ii) awareness of problems and understanding of cause-effect relationships, and (iii) team competency including communication skills. Although, this provided useful information that facilitates software development practices, it fails to provide relevant information regarding the satisfaction levels derived for possessing or using a competency. Evidently, there is a gap in existing literature. Perhaps, this is because researchers in the area of behavioral of software engineering have been focusing on few concepts [4] and ignore other relevant issues such as the assurance for using or possessing a particular competency

To address this, this study seeks to identify and also create a satisfaction level of the competencies from perspective of users, educators, and engineers. This will complement research on SEC in general and soft competency research specifically. To enable us to achieve our research objectives, we make use of CFSE and Kano model. The next sub-sections discuss CFSE and the Kano model.

2.2 Kano model

The Kano model is a quality function-deployment framework that helps developers of product or service to include customer's voice in the development phase. It has been applied mostly in the development of products. This is because it takes into consideration the views of both the customer and developer in the development of a product instead of a passive approach of only developers [17]. [17–23] used the Kano model for the development of ICT system and concluded that the model prioritizes user involvement. It assists in determining basic, performance and delighters of a product or service.

In our scenario, the customer is the software community (organization using the competencies) and the product or service is the competency. According to Kano et al., customer's decision-making options on product or service acquisition, are based on conscious and subconscious deliberations [11]. There is therefore the need to understand these deliberative conscious and subconscious processes of decision-making to help develop products or services. Kano et al., categorized these processes into three-requirement levels (basic, performance and delighters). Basic requirements relate to customer's expectations about a product or service. These requirements are classified as basic since their presence are not dynamic enough to change the options and opinion a customer has about the product. However, their absence may result in complaints from the customer. Performance requirements, on the other hand, are expected pre-requisites that customers know and they are essential influential factors on the customer's decision-making options on products or services. These are critical pre-requisite requirements that create high levels of satisfaction when employed

appropriately and otherwise if not used. The last requirement termed delighters are those requirements that do not engender any complaints from the customers when absent however surprises the customer when present. Delighters are sometimes referred to as attractive or “wow” factors [11].

2.3 Competency framework for software engineers

Competency Framework for Software Engineers (CFSE) is a framework that facilitates, identifies the training needs, and guides the design of software engineers’ competencies. The design is based on the activities and interactions of engineers during the software development process. The constructs of this framework are under the main classification of competency (Hard and Soft). Hard competency category relates to the technical aspects of software engineering. These aspects are based on the definition of the SWEBOK roles in software engineering. They are project management, requirement analysis, software design, programming, validation and verification tests, configuration management, quality, tests, documentation and maintenance. The soft part of the categorization is classified into social and personal. Social aspects include interpersonal relations, cooperation and work in a team, and handling and conflicts resolution. Personals on the other hand includes development in the job, personal development, rights and limits. It can broadly be considered as “a set of knowledge, abilities and key behaviors, with special emphasis on the soft skills” [12].

The objective is to create a classificatory system that identifies and explains satisfaction levels of software engineers' competencies. Therefore, we consider the framework suitable. This is because it considers both soft and hard competency and this is the bigger objective we intend to achieve. Furthermore, the framework considers granularity, which is essential for fitting the work to the community. In line with the objective of this study, we focus on the soft competency aspect of the framework and merge it with Kano model. This resulted in a unified framework for identifying and classifying the satisfaction levels of soft competencies. For detailed analysis of the individual meanings of critical variables of CFSE, readers can refer to the original paper of [12]. The detail of the proposed framework for this paper is explained in the next section.

3 A Unified framework of Soft competency satisfaction levels for software engineers (UFSCSL)

As mentioned earlier, the framework is derived from the CFSE and Kano model. From the CFSE we made use of the soft competency category since our aim is to identify and classify only the soft competency. From among frameworks such as [24–27] for identifying software engineering competencies CFSE framework is the one that has more granularity, thus making it easy for in-depth analysis. In addition, the Kano model has been used for research work in software engineering, but not for analyzing competencies. Thus, this provides a means to chart a new path for competency research. The soft part of the CFSE framework is first categorized into socials and personals and

each have lower granularity as shown in figure 1. The variables of the Kano model (basic, performance, and delighters) were included (see section 2.4), to provide the satisfaction levels for the competencies. See figure 1 for the "soft satisfaction levels of software engineers" framework.

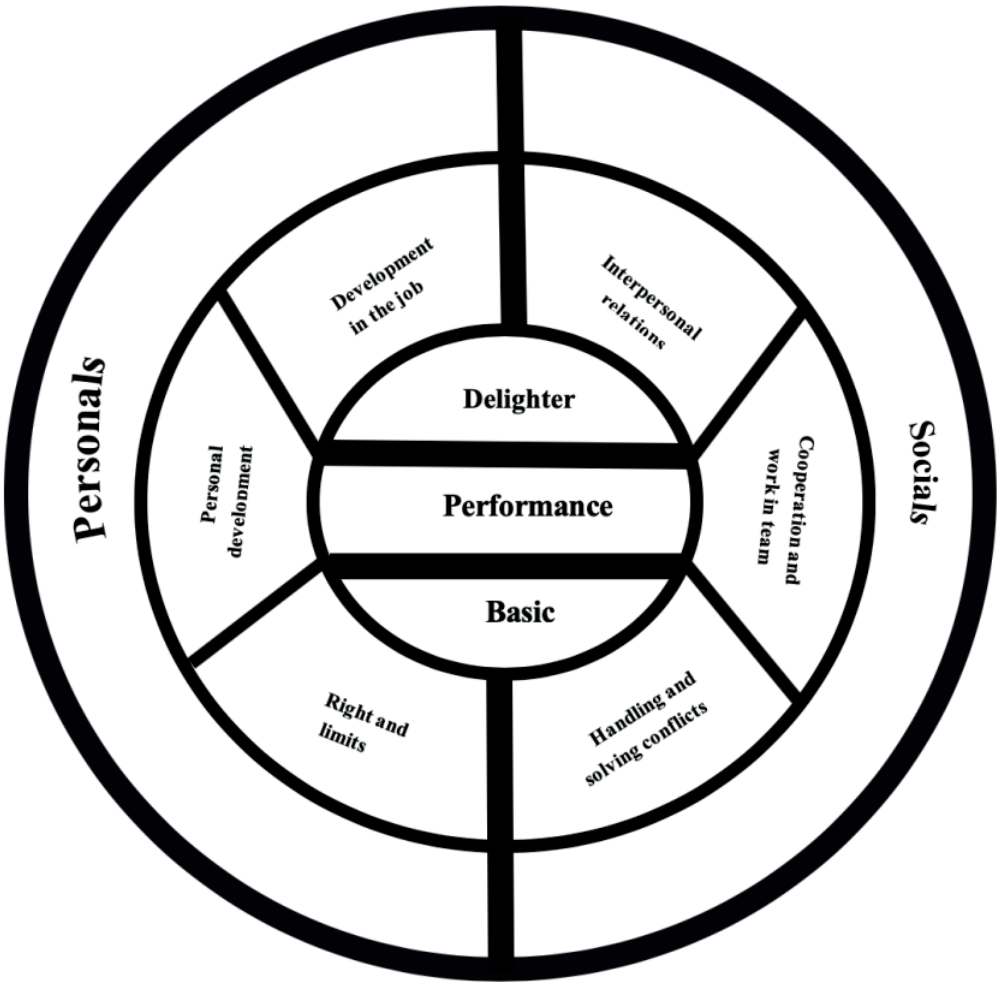


Fig. 1. Unified framework of soft competency satisfaction levels for software engineers (UFSCSL)

To use the UFSCSL, first, the competencies are identified and classifying using the variables in [12]within the frameworks. Then each competency identified or classified is subjected to the metrics of Kano model to determines its satisfaction levels. Thus, given as basic, performance and delighter competencies for socials (interpersonal relations, cooperation and work in a team, and handling and conflicts resolution) and personals (development in the job, personal development, rights and limits).

4 Methodology

4.1 Data Collection

An exploratory qualitative study was adopted. Specifically, [28, 29] qualitative research guide was employed to extrapolate the required data. We agree with the philosophy that an individual's behavior is influenced by the meanings attached to events [30]. Thus, one hundred and thirty-eight (138) participants were drawn from workers in various positions within the industry: practitioner/software engineers/managers/supervisors/mentor. All participants were from software industries based in Norway. A semi-structured interview was used for data collection. Interviews were face-to-face and focused on expected skills of a software developer. Each interview session lasted for about 1 hours. The interview was conducted with the support of assistants. Table 1 represents the distribution of respondents' characteristics.

Table 1. Respondents Characteristics

Years of Experience	**Category**	**Freq**
	1 - 5	40
	6 – 10	17
	11 - 15	7
	16 - 20	15
	21 - 25	11
	26 - 30	13
	31 - 35	2
	36 - 40	2
	unspecified	31
Background	Software	72
	Hardware	11
	Research /university	11
	Others	19
	Unspecified	25

4.2 Data analysis

A thematic analysis offers an accessible means for organizing and describing a dataset under specific themes. Currently, there is no widely agreed way of going about how to use the method [31]. The soft competency satisfaction framework was therefore adopted to guide the analysis.

Both inductive analysis and deductive analysis were used. The coding of the data was done without any pre-defined framework. This enable the themes to emerge from the data. The framework (UFSCSL) was then applied to further code the theme that emerged from the data. Two categories were used on the bases of the epistemology of this research. That is, we were aware of the competencies that have been identified and exist in literature, but our epistemology was that within those identified there will be different satisfaction levels. Hence, we employed both categories in this paper. We

outline the following steps below based on the outcome of our analysis and guided by the steps of [31].

Step 1 Familiarization of the data

The interview was conducted with the help of assistants, with the aim of capturing large groups of respondents. Each interviewer transcribed his or her own interview. The author of this paper acquainted himself by reading through the transcribed scripts. During this stage, notes were taken in cases where there were difficulties in understanding aspects of the data. Further discussions were made with the head of data collection to resolve any ambiguity in the data.

Step 2 Generating initial codes

Initial codes were generated from the data by extracting keywords. This was done without recourse to initial pre-defined coding framework. The total number of competencies that were identified from the transcribed data were six hundred forty-one.

Step 3 Searching for themes

After the initial code, all initial codes were grouped into themes, this facilitated the identification of themes. These themes were generated without resort to pre-defined coding framework. Three hundred sixty soft competencies were identified at this stage.

Step 4 Reviewing themes

The themes were compared with existing themes. That is, a pre-defining coding framework was also used. In this case, the Rivera-Ibarra et al. [12] CFSE framework was used.

Step 5 Defining and naming

Next defined themes and meanings were assigned. These names and meanings were reviewed with literature before the competencies were validated using the variables in the Kano model. This stage resulted in 22 basics, 26 performance and 16 delighter competencies.

Phase 6 Producing the report

The emerged themes that resulted from comparing data themes and themes from the framework were used to produce the results discussed in the next section.

5 Results

5.1 RQ1: What are the different satisfaction levels derived from using a software soft competency?

We present the result in Table 2 using the framework (UFSCSL) developed for this paper. The results show the individual competencies and their satisfaction levels, that is: basic, performance, and delighters. They were grouped according to the broader

theme of soft competency: social and personal. We also provided definitions using the classification levels from the Kano model for the competencies.

Table 2. soft competencies and their satisfaction Levels

Socials		
	Satisfaction levels	***Software engineer competencies***
Interpersonal relations	Delight	(i) communicate to outside world, and (ii) sociable
	Performance	(i) communication skill, (ii) adaptability, (iii) human skill, and (iv) interpersonal skill.
	Basic	(i) social skills and (ii) contributing to the society.
Cooperation and work in team	Delight	(i) Excellent teacher, (ii) see bigger picture, and (iii) leadership.
	Performance	(i) team work, (ii) team organizer, (iii) approachable, (iv) open and communicating, (v) learn from others, and (vi) voice your own opinions.
	Basic	(i) Cooperation, (ii) maturity, (iii) teach and share knowledge, and (iv) dedication to work.
Handling and solving conflicts	Delight	(i) humbleness, (ii) customer awareness, and (iii) understand customer needs.
	Performance	(i) meeting skills, and (ii) contact with clients.
	Basic	(i) Listen ears, (ii) compromise, and (iii) empathy.
Personals		
Development in the job environment	Delight	(i) unafraid, (ii) creative and brave, and (iii) think outside the box.
	Performance	(i) persistence, (ii) flexible, (iii) versatile, (iv) focus, (v) accuracy, (vi) analytical skills, (vii) logical mindset and keep and overview, and (viii) creativity.
	Basic	(i) Willingness to learn, (ii) curious, (iii) passionate about your job, (iv) ask questions, (v) confidence, (vi) honest and responsible.
Personal development	Delight	(i) can apply theories in application, (ii) see opportunity in systems, (iii) initiative, (iv) separate work and being available, and (v) self-sufficient.
	Performance	(i) precise and detail oriented, (ii) self-reliance, (ii) independence (iv) understand needs for further development, and (v) know the working environments.
	Basic	(ii) pragmatic, (iii) patience, and (iii) open to new ideas.
Right and limits	Delight	-
	Performance	(i) attention to detail
	Basic	(i) Introspection and admit error, (ii) admit ignorance, and (iii) interest in the field.

Basic

From the interview data and the analysis, basic competencies are pre-requisite competencies that are necessary and are expected by the users of the competency. Mostly they are taken for granted. Users see these competencies as natural when delivered properly. However, when delivered poorly, users will complain.

Performance

From the interview data and the analysis performance competencies are what users expect and can articulate. They are mostly in the minds of the users and when they are delivered well, they create more satisfaction. These competencies can be described as "uni-dimensional" competency, in that the satisfaction grows exponentially when executed properly.

Delighters

From the interview data and the analysis, the delighter competencies are unexpected by the user. Mostly unexpected by the user but increases the delight and surprise when available however its absence may have no effect on user.

5.2 RQ2: Which of these soft competencies are perceived as most valuable for Software engineering?

As mentioned earlier, delighters are attractive or wow factors that valuable for the development of a product [11]. Therefore, we present our delighter competencies as the most valuable or essential competencies for software engineering. The table 3 shows the competency based on Rivera-Ibarra et al. CFSE framework [12]. The table shows the competency category and the identified essential soft competency for software engineers that are useful for software development.

Table 3. Most valuable competencies.

Competency area	**Competency name**
Socials	
Interpersonal relations	Communicate to outside world
	Sociable
Cooperation and work in team	Excellent teacher
	See bigger picture
	Leadership
Personals	
Handling and solving conflicts	Humbleness
	Customer awareness
	Understand customer needs
Development in the job environment	Unafraid
	Creative and brave

	Think outside the box
Personal development	Can applied theories in application
	See opportunity in systems
	Initiative
	Separate work and being available
	Self-sufficient

6 Discussions

Following our analysis, we aimed to provide a satisfaction level for the competencies identified from our primary data. A total of 63 competencies emerge from our data. Out of that 29 was for social competencies and 34 was for personals competencies with three satisfaction levels.

Table 4. Total number of soft competencies based on the satisfaction levels classification

Competency area	***Satisfaction levels***	***Total Number***	
Socials			
Interpersonal relations	Delight	2	8
	Performance	4	
	Basic	2	
Cooperation and work in team	Delight	3	13
	Performance	6	
	Basic	4	
Handling and solving conflicts	Delight	3	8
	Performance	2	
	Basic	3	
Personals			
Development in the job environment	Delight	3	17
	Performance	8	
	Basic	6	
Personal development	Delight	5	13
	Performance	5	
	Basic	3	
Right and limits	Delight	0	4
	Performance	1	
	Basic	3	
Total			

Table 4 shows the number of competencies and number of satisfactions of the competency area. Under socials competency area cooperation and work in team had 13 competencies, interpersonal relations and handling and solving conflicts had 8 competencies each. The cooperation and work in team competency reflect the team competency category of Sedelmaier and Landes [3]. Under personals competency area,

development in the job environment had 17 competencies, followed by personal development with 13 and right and limits with 4 competencies.

Table 5. Identified soft competencies and prior work.

Category	Identified soft competency	Comparison
Socials		
Interpersonal relations	Sociable, communication skill, adaptability, human skill, interpersonal skill, social skills	Consistent with prior work
	Communicate to outside world, contributing to the society,	New observations
Cooperation and work in team	See bigger picture, leadership, team work, Cooperation, teach and share knowledge, team organizer, approachable, open and communicating, learn from others	Consistent with prior work
	Maturity, Excellent teacher, voice your own opinions. dedication to work	New observations
Handling and solving conflicts	Customer awareness, understand customer needs, meeting skills, contact with clients, empathy	Consistent with prior work
	Humbleness, compromise	New observations
Personals		
Development in the job environment	Unafraid, creative and brave, think outside the box, persistence, flexible, versatile, analytical skills, creativity, Willingness to learn, curious, ask questions, confidence, focus, accuracy, logical mindset and keep and overview, honest and responsible	Consistent with prior work
	Passionate about your job	New observations
	Separate work and being available, self-sufficient,	Consistent with prior work

Personal development	precise and detail oriented, self-reliance, independence, pragmatic, patience, initiative, open to new ideas.	
	Can apply theories in application, see opportunity in systems, understand needs for further development, know the working environments	New observations
Right and limits	Attention to detail,	Consistent with prior work
	Introspection and admit error, admit ignorance, interest in the field	New observations

Some of the identified competencies in the categories are consistent with exiting literature such as the work of [3, 15, 16, 32–34]. Table 5 highlights the new observations and comparison to prior work. With regard to satisfaction levels, competencies were identified in all the categories except rights and limits delighters. A total of 16 essential competencies using the Kano model was identified. These competencies are consistent with literatures such as [35, 36]. Furthermore we have been able to create a satisfaction level, that adds to the works of [10, 37] that made argument for based competencies.

On the essential soft competencies for software engineers, we have been able use model analysis to extrapolate the essential competencies that are in agreements with the work of [35, 36, 38]. Thus, providing a new way of identifying essential competencies.

The novelty in this work are: (i) observations of new soft competency from empirical data that are highlighted in table 4 and (ii) the Unified framework of soft competency satisfaction levels for software engineers (UFSCSL). The UFSCSL has the ability to identify soft competencies of software engineers and also provide a satisfaction levels of the competency. Thus, serving as insurance model for users, possessors and the educators. In short, the major stakeholders of software engineering competency development are considered in this framework.

The study has both practical and research implications. From the perspective of the users of competencies, they can use the classification to determine which competencies will be valuable for employment. On the part of the possessor, they can use the classification levels to evaluate what they possess. Furthermore, educators can use the classification levels to adjust their training. Additionally, the framework which was proposed (UFSCSL) can be used for constant evaluation on old competencies and also on new ones.

7 Conclusion

The study has analyzed, identified and created a classification that can be used by the software community. This was done by synthesized existing relevant literature. The empirical work was based on Kano et al. [11] and Rivera-Ibarra et al. [12] CFSE framework. The study resulted in the identification of competencies, classification levels and essential competencies of software engineers. The study charts a new path of identifying essential or valued competencies of software engineers by using Kano model that has on been applied on products and services. Further studies should be done to understand how competencies within the satisfaction level can change.

The scope of the data collection was limited to companies situated in Norway; it may therefore limit the ability to generalize the findings universally. Nevertheless, most of the companies that the interviewees worked for has global representation and dealings outside Norway. With the development of competency satisfaction levels, we call for further studies to understand how specific competencies evolves within the satisfaction level.

Acknowledgement

The author would like to acknowledge Prof. Pekka Abrahamsson for his support in providing the dataset for this research and Dr. Hadi Ghanbari for his guidance.

References

1. Weinberg, G. M. (1971). *The Psychology of Computer Programming* (Silver Ann.). New York, New York 10027 Usa: Dorset House Publishing.
2. IEEE. (2014). *Software Engineering Competency Model (SWECOM)*. IEEE. Retrieved from http://www.dahlan.web.id/files/ebooks/SWECOM.pdf
3. Sedelmaier, Y., & Landes, D. (2014). Software Engineering Body of Skills (SWEBOS). In *2014 IEEE Global Engineering Education Conference (EDUCON)* (pp. 395–401). IEEE. doi:10.1109/EDUCON.2014.6826125
4. Lenberg, P., Feldt, R., & Wallgren, L. G. (2015). Behavioral software engineering: A definition and systematic literature review. *Journal of Systems and Software*, *107*, 15–37. doi:10.1016/j.jss.2015.04.084
5. Harris, K. S., & Rogers, G. E. (2008). Soft Skills in the Technology Education Classroom : What Do Students Need. *Technology Teacher*, *68*(3), 19–25.
6. Moreno, A. M., Sanchez-segura, M., Medina-dominguez, F., & Carvajal, L. (2012). The Journal of Systems and Software Balancing software engineering education and industrial needs. *The Journal of Systems & Software*, *85*(7), 1607–1620. doi:10.1016/j.jss.2012.01.060
7. Broadbent, M., Dampney, C. N. G., Lloyd, P., & Hansell, A. (1992). Roles , Responsibilities and Requirements for Managing Information Systems in the 1990s. *International Journal of Information Managemet*, *72*, 21–38.
8. Colomo-palacios, R., Casado-lumbreras, C., Soto-acosta, P., García-peñalvo, F. J., & Tovar-caro, E. (2013). Computers in Human Behavior Competence gaps in software personnel : A multi-organizational study. *Computers in Human Behavior*, *29*(2), 456–461. doi:10.1016/j.chb.2012.04.021
9. Holtkamp, P., Jokinen, J. P. P., & Pawlowski, J. M. (2015). Soft competency requirements in requirements engineering, software design, implementation, and testing. *Journal of Systems and Software*, *101*, 136–146. doi:10.1016/j.jss.2014.12.010
10. Thurner, V., Schlierkamp, K., Bottcher, A., & Zehetmeier, D. (2016). Integrated development of technical and base competencies: Fostering reflection skills in software engineers to be. In *IEEE Global Engineering Education Conference, EDUCON* (pp. 340–348). Abu Dhabi, UAE: IEEE. doi:10.1109/EDUCON.2016.7474576
11. Kano, N., Seraku, N., Takahashi, F., & Tsuji, S. (1984). Kano. *Attractive Quality and Must-Be Quality. The Journal of the Japanese Society for Quality Control*, *14,* 39–48.
12. Rivera-Ibarra, J. G., Rodríguez-Jacobo, J., & Serrano-Vargas, M. A. (2010). Competency Framework for Software Engineers. In *2010 23rd IEEE Conference on Software Engineering Education and Training* (pp. 33–40). doi:10.1109/CSEET.2010.21
13. Andrews, J., & Higson, H. (2008). Graduate Employability, 'Soft Skills' Versus 'Hard' Business Knowledge: A European Study. *Higher Education in Europe*, *33*(4), 411–422. doi:10.1080/03797720802522627
14. Trivellas, P., & Reklitis, P. (2014). Leadership Competencies Profiles and Managerial Effectiveness in Greece. *Procedia Economics and Finance*, *9*(Ebeec 2013), 380–390. doi:10.1016/S2212-5671(14)00039-2
15. Licorish, S. A., & Macdonell, S. G. (2013). Differences in Jazz Project Leaders '

Competencies and Behaviors : A Preliminary Empirical Investigation. In *2013 6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)* (pp. 1–8). IEEE. doi:10.1109/CHASE.2013.6614725

16. Noorman, M., Akmal, M., Osman, F., & Ibrahim, Z. (2015). Malaysian Computer Professional : Assessment of Emotional Intelligence and Organizational Commitment. In *Procedia - Social and Behavioral Sciences* (Vol. 172, pp. 238–245). Elsevier B.V. doi:10.1016/j.sbspro.2015.01.360
17. Lee, Y. C., Sheu, L. C., & Tsou, Y. G. (2008). Quality function deployment implementation based on Fuzzy Kano model: An application in PLM system. *Computers and Industrial Engineering*, *55*(1), 48–63. doi:10.1016/j.cie.2007.11.014
18. Gangurde, S., & Patil, S. (2018). Benchmark product features using the Kano-QFD approach: a case study. *Benchmarking: An International Journal*, *25*(2), 450–470. doi:http://dx.doi.org/10.1108/MRR-09-2015-0216
19. Huang, J. (2018). Application of Kano model and IPA on improvement of service quality of mobile healthcare Jui-Chen Huang, *16*(2).
20. Lehtola, L., & Kauppinen, M. (2006). Suitability of requirements prioritization methods for market-driven software product development. *Software Process Improvement and Practice*, *11*(1), 7–19. doi:10.1002/spip.249
21. Liu, X. F. (2000). Software quality function deployment. *Potentials, IEEE*, *19*(5), 14–16. doi:10.1109/45.890072
22. Piaszczyk, C. (2011). Model Based Systems Engineering with Department of Defense Architectural Framework. *Systems Engineering*, *14*(3), 305–326. doi:10.1002/sys
23. Richardson, I. (2001). Software Process Matrix: A Small Company SPI Model. *Software Process: Improvement and Practice*, *6*(Daft 1992), 157–165. doi:10.1002/spip.144
24. Orsoni, A., & Colaco, B. (2013). A Competency Framework for Software Development Organizations. In *2013 UKSim 15th International Conference on Computer Modelling and Simulation* (pp. 507–511). IEEE. doi:10.1109/UKSim.2013.101
25. Acuña, S. T., & Juristo, N. (2004). Assigning people to roles in software projects. *Software - Practice and Experience*, *34*(7), 675–696. doi:10.1002/spe.586
26. Linck, B., Ohrndorf, L., Kiel, T. D. L., Magenheim, J., & Neugebauer, J. (2013). Competence model for informatics modelling and system comprehension. In *2013 IEEE Global Engineering Education Conference (EDUCON)* (pp. 85–93). IEEE. doi:10.1109/EduCon.2013.6530090
27. Tuffley, D. (2012). Optimising virtual team leadership in Global Software Development. *IET Software*, *6*(March 2011), 176–184. doi:10.1049/iet-sen.2011.0044
28. Mason, J. (2002). *Qualitative Researching*. *Qualitative Research Journal* (Vol. 41). doi:10.1159/000105503
29. Myers, M. D., & Newman, M. (2007). The qualitative interview in IS research: Examining the craft. *Information and Organization*, *17*(1), 2–26. doi:10.1016/j.infoandorg.2006.11.001
30. Wohlin, C., & Aurum, A. (2015). Towards a decision-making structure for selecting a research design in empirical software engineering. *Empirical Software Engineering*, *20*(6), 1427–1455. doi:10.1007/s10664-014-9319-7
31. Braun, V., & Clarke, V. (2006). Full-Text. *Qualitative Research in Psychology*, *3*(2), 77–101. doi:10.1191/1478088706qp063oa

32. Kropp, M., Meier, A., & Perellano, G. (2016). Experience Report of Teaching Agile Collaboration and Values Agile Software Development in Large Student Teams. In *2016 IEEE 29th International Conference on Software Engineering Education and Training (CSEET)* (pp. 76–80). IEEE. doi:10.1109/CSEET.2016.30
33. Robal, T., Ojastu, D., Kalja, A., & Jaakkola, H. (2015). Managing Software Engineering Competences with Domain Ontology for Customer and Team Profiling and Training. In *2015 Portland International Conference on Management of Engineering and Technology (PICMET)* (pp. 1369–1376). Portland International Conference on Management of. doi:10.1109/PICMET.2015.7273171
34. Samuelsen, T., Colomo-palacios, R., & Kristiansen, M. (2016). Learning software project management in teams with diverse backgrounds. In *Fourth International Conference on Technological Ecosystems for Enhancing Multiculturality – TEEM 16*.
35. Turley, T., & Bieman, M. (1995). Competencies Nonexceptional of Exceptional and Software Engineers. *J. Systems Software*, *1995:28*(28), 19–38.
36. Chang, J., Wang, T., & Lee, M. (2016). Impacts of Using Creative Thinking Skills and Open Data on Programming Design in a Computer-supported Collaborative Learning Environment. In *2016 IEEE 16th International Conference on Advanced Learning Technologies* (pp. 396–400). doi:10.1109/ICALT.2016.78
37. Thurner, V., Axel, B., & Andreas, K. (2014). Identifying Base Competencies as Prerequisites for Software Engineering Education. In *IEEE Global Engineering Education Conference (EDUCON)* (pp. 1069–1076). doi:10.1109/EDUCON.2014.6826240
38. Suhartono, J., Sudirwan, J., & Background, A. (2016). Academic Competence of Computer Science Graduate Degree from the Employer ' s Perspective. In *2016 International Conference on Information Management and Technology (ICIMTech)* (pp. 176–181). IEEE. doi:10.1109/ICIMTech.2016.7930325

IV

THE ESSENTIAL COMPETENCIES OF SOFTWARE PROESSIONALS: A UNIFIED COMPETENCIES GATE FRAMEWORK

by

Nana Assyne, Hadi Ghanabari & Mirja Pulkkinen

Information and Software Technology, under revision

Request a copy from the author.

V

TOWARDS A SECURITY COMPETENCIES OF SOFTWARE DEVELOPERS

by

Nana Assyne, 2020

W. Yaokumah, M. Rajarajan, J.-D. Abdulai, I. Wiafe, & F. A. Katsriku (Eds.),
Modern Theories and Practices for Cyber Ethics and Security Compliance
(pp. 73-87)

Chapter 5
Towards a Security Competence of Software Developers: A Literature Review

Nana Assyne
https://orcid.org/0000-0003-0469-6642
University of Jyväskylä, Finland

ABSTRACT

Software growth has been explosive as people depend heavily on software on daily basis. Software development is a human-intensive effort, and developers' competence in software security is essential for secure software development. In addition, ubiquitous computing provides an added complexity to software security. Studies have treated security competences of software developers as a subsidiary of security engineers' competence instead of software engineers' competence, limiting the full knowledge of the security competences of software developers. This presents a crucial challenge for developers, educators, and users to maintain developers' competences in security. As a first step in pushing for the developers' security competence studies, this chapter utilises a literature review to identify the security competences of software developers. Thirteen security competences of software developers were identified and mapped to the common body of knowledge for information security professional framework. Lastly, the implications for, with, and without the competences are analysed and presented.

INTRODUCTION

The current explosive growth being observed in the software industry requires high-level corresponding software security. This is because "software vulnerabilities or flaws are often key entrance door for attackers" (Sametinger, 2013). They include buffer overflows, SQL injection, cross-site scripting, stack overflow, inconsistent error handling, and so on (McGraw, 2004). Previously, software security used to be an afterthought, but recently it is being addressed actively from the planning stage of software development. Additionally, in today's software development process, software testing includes security testing instead of only functional testing (Mano, Duhadway, & Striegel, 2006), thus making the security

DOI: 10.4018/978-1-7998-3149-5.ch005

competences of the developers more eminent in software development. Coupled with the fact that research work on software developers' competence is not lacking (Lenberg, Feldt, & Wallgren, 2015), the security competences of software developers should be well recorded in literature. But on the contrary, that is not the case. However, when they are recorded, they are recorded as a subsidiary of security engineers' competence instead of software engineers' competence, thus making it counterproductive to develop and maintain the security competences of software developers to the benefit of the possessors (developers), those who train the possessors of the competences (educators), and users of the competences (industry).

McGraw (2004) defines software security as "the idea of engineering software so that it continues to function correctly under malicious attack". And, Hazeyama & Shimizu (2012), goes further with the definition by stating that "software security deals with security during the whole software development process". On the other hand, software engineering competence is defined by the Institute of Electrical and Electronics Engineers (IEEE) as knowledge, skills, and attitudes of software developers to fulfil a given task in a software development project (IEEE, 2014). Thus, the author of this chapter defines security competence of software developers as those specific security competences required by a developer to deal with security during the whole software development process. An example is an SQL injection skills and security pattern skills.

As mentioned above, one cannot afford to leave software security as an afterthought; developers must strive to improve software security issues from the planning stage to the maintenance stage. The works of Cheng et al. (2008), Hilburn and Mead (2013), and Riehle and Nürnberg (2015) are studies that investigated methods to handle software security using the lifecycle of software development. It is also well established that vulnerabilities and flaws are the doors attackers exploit. Works such as Kaur and Kaur (2016), McGraw (2004), Park et al. (2010), and Wegerer and Tjoa (2016) confirm this assertion in literature. In addition, assailants of software systems are persons or entities, who are active and keep on improving their skills in attacking software systems to satisfy their desire (Cheng et al., 2008). However, the security competences of the developers of the software are not well established in literature.

Whilst introducing security engineering environment studies for software developers, Cheng et al. (2008) point out that there is urgent need to create an environment that integrates various tools and provides comprehensive facilities to the designers, developers, users, and maintainers of a software system (Cheng et al., 2008). The development and maintenance of such an environment requires knowledge of security competences of the developers to prepare and develop them to withstand the intrinsic difficulty of assailants of a software system (Cheng et al., 2008). This implies that security know-how of the developer is very crucial. Hazeyama and Shimizu (2012) and Hilburn and Mead (2013) reiterate the need for awareness to be channelled towards developers' skills regarding security. However, previous studies provide less concise and coordinated information on security competences of developers.

Summarily, these competences are scattered in several different studies. Thus, the following questions arise: *what are the security competences of software developers? How can they be improved?* As part of broader research on software developers' competences, we set our research question as *what are the security competences of a software developer that are available in literature?* The remainder of this work includes: Section 2 presents previous studies and background. Section 3 looks at the methodology used in this study. Section 4 looks at the results. Section 5 and 6 presents the discussion and conclusion.

PREVIOUS STUDIES AND BACKGROUND

In this section of the study, three literature review studies on software developers' competences are identified. These literature reviews are Cruz et al. (2015); Moustroufas et al. (2015) and Vishnubhotla et al. (2018). Two of the studies utilized systematic literature review methods and the last study employed a traditional literature review method. Cruz et al. (2015) and Vishnubhotla et al. (2018) that used systematic literature review, focused on specific areas of software developers' competence. Cruz et al. (2015) investigated the personality of software engineers and their roles in software development. Vishnubhotla et al. (2018) also presented the capability and competence measurement of software engineers, including team working in agile software development. Moustroufas et al. (2015) utilized a traditional literature review to evaluate the adequacy of software engineer competences and created a software competence profiling model for recruiting software engineers. Moustroufas et al. (2015) investigated and reviewed software developers' competence in general contrary to the first two that focused on specific areas. The software security competence of developers did not appear in any of the three studies, thus the need for this paper.

It is also worth mentioning that there are several efforts being made to improve security matters in the development of software. They include the development processes and the methods to reduce vulnerabilities and flaws in software. Hazeyama & Shimizu (2012) proposed a software security learning process using the traditional software development cycle. Cheng et al. (2008) reiterated for security engineering environment for software development since security requires continuous support. Thus, they make use of the lifecycle of software engineering for their solution which is based on International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC) standards. The work of Verdon (2006) and McGraw (2004) examined the security policies and best practices that are essential for software developers.

The Open Web Application Security Project (OWASP) that is OWASP top 10 -2017 that focused on software developers and designers stated that "insecure software is undermining our financial, healthcare, defense, energy, and other critical infrastructure." The increasing complexity and the connectedness of software, is making it more difficult in attaining an increase in application security. Additionally, we face the rapid process of developing software which increases our common security risks. This makes it impossible to accept simple security problems as listed in the OWASP top 10 – 2017. The top five on the list are (i) Injection, (ii) Broken Authentication, (iii) Sensitive Data Exposure, (iv) XML External Entities (XXE), and (v) Broken Access Control. The rest of the OWASP top 10 – 2017 are (vi) Security Misconfiguration, (vii) Cross-Site Scripting (XSS), (viii) -Insecure Deserialization, (ix) Using Components with Known Vulnerabilities, and (x) Insufficient Logging & Monitoring (OWASP, 2017). Such security problems require corresponding skills to handle them. Given this, software developers' need to develop their security competences. For them to be able to develop and maintain such competences, it requires that such competences are identified and placed in the appropriate domain. Thus, the need for this study.

A survey to identify the guidance available on the web to help software developers' to fix security matters was conducted by Acar et al. (2017). They concluded that not all the information on the web is readily made for fixing security issues (Acar et al., 2017). Therefore, it may require security competences of the developers' to adjust the available code to meet the security demand. Hilburn & Mead (2013), developed a software security assurance model by providing capabilities. The capability of the assurance model was addressed by utilizing the knowledge areas. The main knowledge areas of assurance model that were identified were: assurance across lifecycles, risk management, assurance assessment, assur-

ance management, system security assurance, system functionality assurance and system operational assurance (Hilburn & Mead, 2013). Even though, this work focused on assurance in software security, it also provided some capabilities or knowledge areas that are useful for this paper. Work such as Meng et al. (2018); Miller and Heymann (2018) and Qian et al. (2018) provide some information on the security competences of software developers. Therefore, we employ these studies stated above and other existing studies to set the agenda for identifying the security competences of software developers and highlight the importance of software developers' security competences for further studies. Thus, this study seeks to employ traditional literature reviews to identify the security competences of software developers as the first step in broader research.

In presenting Common Body of Knowledge (CBK) for Information security professionals, Theoharidou & Gritzalis (2007) made a case for technical and behavioural skills for information security professionals. The framework was achieved using 135 academic intuitions from Africa, Asia, Australia, Europe, and South and North America to provide a skill set for information security professionals. The framework can be utilized in identifying and assessing the skills of information security professionals. The framework has three major areas: information communications technology skills area, security skills area and behavioural skills area. This study aimed at identifying the security competences of software developers from literature using traditional literature review and maps the result to the Common Body of Knowledge for information security professional skills framework (CBK). As a result, the CBK framework will be employed as a theoretical lens for this study.

METHODOLOGY

Primarily a literature review will be mainly employed in this study. Fink defines a research literature review as "a systematic, explicit and reproducible method for identifying, evaluating and synthesizing the existing body of completed and recorded work produced by researchers, scholars, and practitioners" (Fink, 2010, p. 3). In this section, an attempt is also made to distinguish between a traditional literature review and a systematic literature review. Systematic literature review is defined by Kitchenham and Charters as "a form of secondary study that uses a well-defined methodology to identify, analyse and interpret all available evidence related to a specific research question in a way that is unbiased and (to a degree) repeatable" (Kitchenham & Charters, 2007, p. vi, pp. 8). A traditional literature review is used to demonstrate a gap or a problem in an area one seeks to research without an explicit method for reviewing the literature (Moustroufas et al., 2015). Since this is the first step towards broader research, a traditional literature review will be utilized.

Given this, the IEEE database was used as the database to find studies that investigated software security. The identified competences were grouped into two areas: programming related competences and non-programming related competences. The detail of the classification is explained in the result section. The identified competences were then mapped to technical and behavioural skills of information security professionals' skill set framework. With regard to data collection, data was collected from the IEEE database. The search strings that were utilized for the search were: software engineers/developers' skills, competence, and security knowledge. This was done without any strict protocol. Only peer-review papers were employed for the study. The names of the competences were extracted, descriptions of the competences were recorded into an excel sheet for the next stage of the research. On data analysis, competences with the same meaning were group together. Different implications of the competences

were analysed and recorded against the individual competences identified. Using conventional content analysis guideline of Hsieh & Shannon (2005), competences were classified into two areas. They are programming related competences and non-programming related competences. Lastly, the identified competences were mapped to the information security professional skills set framework.

RESULTS

The identified competences were categorized into two. They are programming related competences and non-programming related competences. Programming related competences are those that involve coding. Non-programming related competences are those that do not directly deal with coding. The competences were mapped to the common body of knowledge information security professional skills framework. Table 1 depicts the competence area, the competence name, the citation of the papers that the competences were extracted from and the CBK of information security professional's framework.

Table 1 shows the competences identified, their classifications, the literature from which the competence is extracted from and their relationship to CBK of information security professionals' framework. In all 13 competences were identified, nine competences were programming related and 4 competences were non-programming related. Seven of the competence maps to both information communication technology and security criterial and 6 maps to information communication technology. The next section provides the definition/descriptions of the competences and implications.

PROGRAMMING RELATED COMPETENCES

Secure Programming/Coding Skills

Description

The art of adopting a secure practice in the development of software. This includes the skill of being able to guide against vulnerabilities and flaws in software development. The majority of vulnerabilities and flaws in software appear when developers ignore secure practices in programming. More details of secure programming/coding competences can be found in the works of Mano et al. (2006); Miller & Heymann (2018) and Zainuddin & Normaziah (2011).

Implication

Without the adoption of secure coding, developers may create software with flaws and vulnerabilities. As pointed out by Sametinger (2013), vulnerabilities and flaws are the key entrants for attackers. Improving secure coding or programming will reduce security flaws. Secure coding must be part of a software development curriculum. There is a need to include fundamental security principles programming courses. Organizations must continue to introduce fresh courses on secure coding. In today's software development, secure coding must be started from the planning stage of the development to the end of the software development lifecycle. This implies that developers' competence in secure coding is essential.

Table 1. Security competences of software developers

Competence area	Competence name	Reference	CBK of information security professionals framework (Theoharidou & Gritzalis, 2007)
Programming related skills	Secure programming or coding skills	(Acar et al., 2017; Mano et al., 2006; Miller & Heymann, 2018; Qian, Lo, et al., 2018; Zainuddin & Normaziah, 2011)	Information communications technology/ security
	Secure mobile software development skills	(Meng et al., 2018; Qian, Parizi, & Lo, 2018)	Information communications technology/ security
	Secure socket layer/transport layer security (SSL/TLS) skills	(Verdon, 2006)	Information communications technology/ security
	Web Application security development skills	(Qian, Lo, et al., 2018)	Information communications technology/ security
	Integrated development environment (IDE) security skill	(Meng et al., 2018)	Information communications technology
	Code Analysis tools skills	(Meng et al., 2018)	Information communications technology
	Modelling SQL injection skills	(Kaur & Kaur, 2016; Wegerer & Tjoa, 2016)	Information communications technology/ security
	Handling buffer overflow skills	(Park et al., 2010)	Information communications technology/ security
	Security patterns skills	(Hazeyama & Shimizu, 2012)	Information communications technology/ security
Non-Programming related skills	Software security policy skills	(Verdon, 2006)	Information communications technology
	Software security best practice and standard skills	(McGraw, 2004)(Hazeyama & Shimizu, 2012)(Cheng et al., 2008)	Information communications technology
	System Security assurance skills	(Hilburn & Mead, 2013)(Miller & Heymann, 2018)	Information communications technology
	Vulnerability assessment tool skills	(Miller & Heymann, 2018)	Information communications technology

As suggested by Mano et al. (2006), secured programming must be taught in the early part of a software program. It must also be recognized as important skill for software developers.

Secure Mobile Software Development Skills

Description

Mobile devices may have software applications that we utilize frequently or perhaps even daily. The process of developing apps for these devices differ from the main devices. Furthermore, the database and the storage for these devices also differ. Thus, requiring different programming and security competences for the development of mobile apps. More about secure mobile software development skills can be found in the works of Meng et al. (2018); Qian, Lo, et al. (2018); Qian, Parizi, et al. (2018).

Implication

Most of the developers of these apps lack the necessary skill for developing mobile apps, thereby creating vulnerabilities for attackers to exploit those devices. The common nature (maybe you could be more specific here?) of the devices makes them more vulnerable. Thus, delays in providing bug fixings for new versions of applications can provide a door for attackers. Un-updated operating systems (OS) on mobile devices can allow attackers to exploit the vulnerabilities on the OS to attack the software application. Developers must pay attention to secure mobile development skills since techniques used for developing mobiles are different from that of normal devices. Fundamentally the increased usage of mobile technology is putting pressure on mobile developers. Both the trainers and users of the security competence of developers must adopt modern techniques to upgrade the developers to withstand the modern attackers.

Secure Socket Layer Skills

Description

Communication – data transmission between devices - is important in the applications function. This requires developers' skills in standard cryptographic protocol and technology for communicating on the internet. More importantly the use of transport layer security (TLS). Developers need to have skills in socket programming to enable them to develop this type of communication. More details of secure socket layer skills can be found in the work of Verdon (2006)

Implications

Most attackers take advantage of eavesdropping on transmission and launch their attack. This happens when strong encryptions are not used. Developers are to have skills in SSL or TLS encryptions technology. This is because most devices use the internet as a means to transmit data. Without such skills will mean that most attackers can eavesdrop on the communication and launch attacks. Developers should understand and have skills in symmetric encryption.

Web Application Security Skills

Description

Skills to protect devices or applications against web attacks such as cross-site scripting, SQL injection, denial-of-service, etc. Most attackers use vulnerabilities of web applications to attack. It is important to know that web application security directly relates to websites, web applications and web services such as APIs. Again, one needs to distinguish between network security and web application security. Therefore, the competences may defer. More details of secure socket layer skills can be found in the works Anand & Ryoo (2017); Uskov (2013) and Uskov & Avenue (2013).

Implication

In today's world, most of our business is done using the internet. Thus, not having the skills of developing software that can reduce web vulnerability will mean that most businesses could face catastrophes in their dealings. There is the need to have developers who understand using up-to-date skills in proper authentication methods, encryptions and development of patching for discovered vulnerabilities.

Integrated Development Environment (IDE) Security Skills

Description

Most developers of software make use of IDE for the development of software. They are software applications that provide the environments for software development. Thus, they are attitude, skills, and knowledge for using IDE securities in developing software. More details of IDE security skills can be found in the work of Meng et al. (2018).

Implication

Such environments sometimes if not well protected, can leave vulnerabilities in the software being developed and can be exploited by attackers. Having the skills related to the security of the use of the said IDE provides the developer with an environment free of vulnerabilities and flaws. Security updates are important and other security in the transmission of data. Developers must understand such security environments and use them appropriately to avoid leaving vulnerabilities that can be taken advantage of attackers.

Code Analysis Tools Skills

Description

Code analysis tools are used during coding to aid in analysing the code of the developer. Such tools help in identifying bugs and guide the developer to fix them before deploying the applications. They are attitude, skills, and knowledge for performing code analytics in software development. More details of code analysis tools skills can be found in the work of Meng et al. (2018)

Implication

If developers do not have the skill of using code analysis tools it may mean that time to identify bugs during coding may be long. It can result in leaving bugs to be exploited by attackers. It is also important to note that most of these bugs are difficult to be identified by the human eye. Examples of such tools are PMD java and SonarQube.

Modelling SQL Injection Skills

Description

It is a code injection technique that attackers take advantage of data-driven applications using SQL statements. It mostly happens when user inputs are not well-typed. They are attitude, skills, and knowledge for developing software free of SQL injection. More detail of SQL injection skills can be found in the works of Kaur & Kaur (2016) and Wegerer & Tjoa (2016).

Implication

It allows attackers to use malicious SQL statements to attack. This can be used on websites and databases. This is done by using spoof identity to temper with existing data. Such attacks are known as vector. Without skills in SQL injection handling in web applications and applications using databases, it will give attackers the chance to attack just systems since such vulnerability is commonly committed by developers.

Handling Buffer Overflow Skills

Description

It happens when a program writing to the buffer, which is a memory area set aside to hold data overflow. Mostly, when malformed inputs are used. they are attitude, skills, and knowledge needed to avoid buffer overflows. More details of handling buffer overflow skills can be found in the work of Park et al. (2010).

Implication

This happens when programmers or developers assume that all inputs may be smaller, but this may not always be the case. In case there is an overflow, the system may write beyond the allocated size causing erratic in execution leading to access error or crashing of the system. There is the need to write code that has built-in protections in the programming codes. The possession of such skills may reduce buffer overflows in memory, since not all input size can be predicted well by the developer.

Security Patterns Skills

Description

Security patterns are applied during software development by developers to achieve security goals. Such security patterns are pre-defined to guide developers. Having such skills will enable developers to know what security pattern can be used to achieve a particular security goal. That is the protected system patterns for confidentiality and integrity of information and error detection/correction pattern for deducing errors for corrections. More detail of security patterns skills can be found in the work of Hazeyama & Shimizu (2012).

Implication

Without such patterns, developers are to start from scratch to develop such protections. Understanding or having such skills, they can also develop security patterns to meet a specific goal that is not available.

NON-PROGRAMMING RELATED SKILLS

Software Security Policy Skills

Description

A software security policy defines the specific rules of security that software to be developed must have. That means that developers must frequently reference to make sure that the software obeys such policy. Understanding software security policy as a skill will enable the developer to develop software that will meet the security policy of the organization, the state and the world in general. Thus, they are attitude, skills, and knowledge needed to develop software to meet software security policies of the organization, the state, and the international community. More details of software security policy skills can be found in the work of Verdon (2006).

Implication

If developers do not have the skill to understand security policies and cannot develop software to meet what the organization, the state, and the international community have set as their policy for software security, consumers may not trust those software products. Furthermore, software security policies are standards, established to help reduce security threats. This means that, without them, developers may develop software according to their skills. This can lead to a lower security standard for the software they develop.

Security Best Practice and Standard Skills

Description

Best practice and standard are what has been used, tested and agreed as the best way of handling security in software development. Security best practices and standards can guide developers in secure software development. Thus, they are the attitude, skills, and knowledge needed to develop software security best practices and standards. More details of software security policy skills can be found in the works Cheng et al. (2008); Hazeyama & Shimizu (2012) and McGraw (2004).

Implication

If developers do not have such skills, it will mean they may not follow the best way of developing secure software. Mostly, security best practices and standards serve as a guide, but also provide a means to develop to meet certain accepted way that leads to trust.

This will mean that software developed by such developers with security best practices and standards skills will develop secured software, thereby, reducing the vulnerabilities that an attacker can exploit.

System Security Assurance Tools Skills

Description

These are tools that help developers of software from protecting the data and resources controlled by the software. They are the first line in for defending the attackers and also assessing the software security. Thus, they are the attitudes, skills, and knowledge needed to use system security assurance tools when developing software. More details of system security assurance tools skills can be found in the works of Hilburn & Mead (2013) and Miller & Heymann (2018).

Implication

Mostly, the human resources of the developer alone may not be enough for handling the development of software. Therefore, tools are needed to support the development of secured software. System security assurance tools support developers in such a situation. Not having the skill of using such tools will require more human hand in the development process. Alternatively, they will develop software that does not provide the required assurance for the people.

Vulnerability Assessment Tool Skills

Description

Tools are needed to identify the threats and risks that may be in software during development. In using such tools developers will need some special skills. Thus, they are attitude, skills, and knowledge needed by developers to use vulnerability assessment tools during software development. More detail of vulnerability assessment tool skills can be found in the work of (Miller & Heymann, 2018)

Implication

Without such tools, the human factor is to be used for such identification of vulnerability and threats thus, making such skills important for developers. It is important to note that most of such vulnerabilities are difficult to be identified by the human eye, thus if developers have no skills in using these tools, it may mean suck vulnerabilities and threats may be left in the software for attackers to exploit.

DISCUSSIONS

As stated in the related works, there were three review papers on software developers' competences. Two made use of a systematic review and one used a traditional review. None of these reviews mentioned the security competences of software developers. Nevertheless, there are some similarities. The work of Moustroufas et al. (2015) also used a traditional review, which was the same method used by this paper.

The difference between this paper and Moustroufas et al. (2015) is that they looked at software developers competence in general, whereas this paper looked at is security competence of the developers which is a specific area in software developers' competence. On the other hand, the other two reviews also looked at specific areas of developers' competence similar to this paper but used a systematic literature review as a method. This paper agrees with these authors that competences of software developers are essential for software development and effort must be made to maintain them especially in academia.

In proposing a security engineering environment for software developers, Cheng et al. (2008) claimed that the tools and the developers must integrate for a secure engineering environment. We support their assertion, but their work falls short of the implication of not having such an environment. To add to their work, this paper has provided the security competences of the developers which are essential for the security engineering environment they proposed. Furthermore, this paper has responded to the call by Hazeyama and Shimizu (2012) and Hilburn and Mead (2013), that there is the need to pay attention to security competences of the developers'. This paper has provides some of the competences, therefore agreeing with Hazeyama and Shimizu (2012) and Hilburn and Mead (2013) that the security competences of the developers are an essential parts of software developers' competences. For that reason, we support their call for more research on security competences of software developers'.

Researchers such as Cheng et al. (2008); Hilburn & Mead (2013) and Riehle & Nürnberg (2015) have called for security competence development through the lifecycle of developers. We concede, we could not do that, but we have identified some security competences of the developer that can be used as a starting stage for security competences of the developers' studies. Acar et al. (2017)stated that not all web security resources can be used fully to solve security problems by developers. Therefore, with the identification of the security competences of software developers, industry players can add to such work (web resources) by using the competences they have. Thus, this chapter supports the work of Hilburn & Mead (2013) that, knowing those security competences of software developers will help the users, possessors, and educators. Meng et al. (2018); Miller and Heymann (2018) and Qian et al. (2018) provided individual security competences of software developers, though this paper could not provide a full list, the paper has provided the basis for more work to be done. Theoharidou & Gritzalis (2007) work identified the technical and behavioural competences of information security professionals. This assertion has been established in the literature. We did not identify any behavioural security competences of software developers. Nevertheless, we hold the belief that there are behavioural security competences of developers and that empirical work must be conducted to identify them.

CONCLUSION

This chapter proposes a security competence for software developers. It uses a literature review to identify and classify security competence of software developers. Thirteen security competences of software developers were identified. They were classified as programming related competence and non-programming related competence. The author agrees that the methodology used has some limitations. Nevertheless, the competence identified and the linkage provided between the security competence of software developers and the information security professional framework will serve as a base for the development of the security competence of software developers. Furthermore, this chapter also makes a

call for empirical research to identify the security competence of software developers. By that, the author calls for a systematic literature review on the security competence of software developers. Again, there is the need also to identify those security competences using the lifecycle of the software development process.

REFERENCES

Acar, Y., Stransky, C., Wermke, D., Weir, C., Mazurek, M. L., & Fahl, S. (2017). *Developers Need Support, Too: A Survey of Security Advice for Software Developers. In 2017 IEEE Cybersecurity Development IEEE Secure Development Conference Developers* (pp. 22–26)., doi:10.1109/SecDev.2017.17

Anand, P., & Ryoo, J. (2017). Security Patterns As Architectural Solution - Mitigating Cross-Site Scripting Attacks in Web Applications. In *2017 International Conference on Software Security and Assurance (ICSSA)* (pp. 25–31). IEEE. 10.1109/ICSSA.2017.30

Cheng, J., Goto, Y., Morimoto, S., & Horie, D. (2008). A Security Engineering Environment Based on ISO / IEC Standards: Providing Standard, Formal, and Consistent Supports for Design, Development, Operation, and Maintenance of Secure Information Systems. In *2008 International Conference on Information Security and Assurance* (pp. 350–354). 10.1109/ISA.2008.106

Cruz, S., Fabio, Q. B., & Fernando, L. (2015). Forty years of research on personality in software engineering: A mapping study. *Computers in Human Behavior*, *46*, 94–113. doi:10.1016/j.chb.2014.12.008

Fink, A. (2010). Conducting Research Literature Reviews: From the Internet to Paper (3rd ed.). SAGE.

Hazeyama, A., & Shimizu, H. (2012). Development of a Software Security Learning Environment. In *2012 13th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing* (pp. 518–523). IEEE. 10.1109/SNPD.2012.65

Hilburn, T. B., & Mead, N. R. (2013). Building Security In. *IEEE Security and Privacy*, *11*(October), 89–92. doi:10.1109/MSP.2013.109

Hsieh, H.-F., & Shannon, S. E. (2005). Three Approaches to Qualitative Content Analysis. *Qualitative Health Research*, *15*(9), 1277–1288. doi:10.1177/1049732305276687 PMID:16204405

IEEE. (2014). *Software Engineering Competency Model (SWECOM)*. IEEE. Retrieved from http://www.dahlan.web.id/files/ebooks/SWECOM.pdf

Kaur, N., & Kaur, P. (2016). Modeling a SQL Injection Attack. In *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)* (pp. 77–82). Bharati Vidyapeeth.

Kitchenham, B., & Charters, S. (2007). Guidelines for performing Systematic Literature reviews in Software Engineering Version 2.3. Engineering (Vol. 45). doi:10.1145/1134285.1134500

Lenberg, P., Feldt, R., & Wallgren, L. G. (2015). Behavioral software engineering: A definition and systematic literature review. *Journal of Systems and Software*, *107*, 15–37. doi:10.1016/j.jss.2015.04.084

Mano, C. D., Duhadway, L., & Striegel, A. (2006). A Case for Instilling Security as a Core Programming Skill. In *Proceedings. Frontiers in Education. 36th Annual Conference* (pp. 13–18). IEEE. 10.1109/FIE.2006.322347

McGraw, G. (2004). *Software Security*. IEEE Security & Privacy. doi:10.1109/MSECP.2004.1281254

Meng, X., Qian, K., Lo, D., & Wu, F. (2018). Secure Mobile Software Development with Vulnerability Detectors in Static Code Analysis. *2018 International Symposium on Networks, Computers and Communications (ISNCC)*, 1–4. 10.1109/ISNCC.2018.8531071

Miller, B. P., & Heymann, E. (2018). *Tutorial: Secure Coding Practices, Automated Assessment Tools and the SWAMP. In 2018 IEEE Cybersecurity Development (SecDev)* (pp. 124–125). IEEE; doi:10.1109/SecDev.2018.00025

Moustroufas, E., Stamelos, I., & Angelis, L. (2015). Competency profiling for software engineers: Literature review and a new model. In *Proceedings of the 19th Panhellenic Conference on Informatics* (pp. 235–240). Athens, Greece: ACM. 10.1145/2801948.2801960

OWASP. (2017). *OWASP Top 10 - 2017 The Ten Most Critical Web Application Security Risks*. OWASP.

Park, C. S., Lee, J. H., Seo, S. C., & Kim, B. K. (2010). Assuring software security against buffer overflow attacks in embedded software development life cycle. In *2010 The 12th International Conference on Advanced Communication Technology (ICACT)* (Vol. 1, pp. 787–790). IEEE.

Qian, K., Lo, D., Parizi, R., & Wu, F. (2018). Authentic Learning Secure Software Development (SSD) in Computing Education. *2018 IEEE Frontiers in Education Conference (FIE)*, 1–9.

Qian, K., Parizi, R. M., & Lo, D. (2018). OWASP Risk Analysis Driven Security Requirements Specification for Secure Android Mobile Software Development. In *2018 IEEE Conference on Dependable and Secure Computing (DSC)* (pp. 1–2). IEEE. 10.1109/DESEC.2018.8625114

Riehle, D., & Nürnberg, F.-A.-U. E. (2015). How Open Source Is Changing the Software Developer's Career. *Computer Practice*, *48*(5), 51–57. doi:10.1109/MC.2015.132

Sametinger, J. (2013). Software Security. In *2013 20th IEEE International Conference and Workshops on Engineering of Computer Based Systems (ECBS)* (p. 216). IEEE. 10.1109/ECBS.2013.24

Theoharidou, M., & Gritzalis, D. (2007). Common Body of Knowledge for Information Security. *IEEE Security & Privacy*, 64–67.

Uskov, A. V. (2013). Software and Web Application Security: State-of-the-Art courseware and Learning Paradigm. In *IEEE Global Engineering Education Conference (EDUCON)* (Vol. 0, pp. 608–611). 10.1109/EduCon.2013.6530168

Uskov, A. V., & Avenue, W. B. (2013). Hands-On Teaching of Software and Web Applications Security. 2013 3rd Interdisciplinary Engineering Design Education Conference, 71–78. 10.1109/IEDEC.2013.6526763

Verdon, D. (2006). *Security Policies and the Software Developer*. IEEE Security & Privacy. doi:10.1109/MSP.2006.103

Vishnubhotla, S. D., Mendes, E., & Lundberg, L. (2018). An Insight into the Capabilities of Professionals and Teams in Agile Software Development A Systematic Literature Review. In *ICSCA 2018* (pp. 10–19). Kuantan, Malaysia: ACM. doi:10.1145/3185089.3185096

Wegerer, M., & Tjoa, S. (2016). Defeating the Database Adversary Using Deception - A MySQL Database Honeypot. In *2016 International Conference on Software Security and Assurance (ICSSA)* (pp. 6–10). IEEE. 10.1109/ICSSA.2016.8

Zainuddin, H. N., & Normaziah, A. A. (2011). Secure Coding in Software Development. In *2011 Malaysian Conference in Software Engineering* (pp. 458–464). IEEE. 10.1109/MySEC.2011.6140716

KEY TERMS AND DEFINITIONS

Competence: A set of knowledge, skills, and attitudes for performing a task.

Non-Programming-Related Competences: Software security skills that do not directly deal with coding. For example, software security policy skills and system security assurance tools skills.

Programming Related Competences: Software security skills needed for coding. For example, secure programming/coding skills and secure mobile software development skills.

Security Competence of Developers: A set of specific security competencies required by a developer to deal with security during the whole software development process; For example, SQL injection skills, and security pattern skills.

Software Developer: Individuals who employ software development skills to design, construct, test, and maintain computer software.

Software Engineering Competence: A set of knowledge, skills, and attitudes of software developers to fulfill a given task in a software development project.

Software Security: An art of providing protection to software against hackers and attackers during the life cycle of the software.

Traditional Literature Review: A method used to demonstrate a gap or a problem in an area one seeks to research without an explicit method for reviewing the literature.