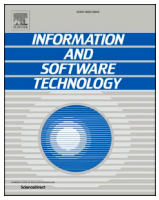
Information and Software Technology 151 (2022) 107020

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof



The essential competencies of software professionals: A unified competence framework

Nana Assyne [a,d,*], Hadi Ghanbari [b,c], Mirja Pulkkinen [a]

[a] University of Jyväskylä, Faculty of Information Technology, P.O. Box 35 (Agora), FI-40014 Jyväskylä, Finland
[b] Aalto University, School of Business, Ekonominaukio 1, 02510, Espoo, Finland
[c] FinEst Centre for Smart Cities, TalTech, Estonia
[d] GIMPA, School of Technology, Accra, Ghana

ARTICLE INFO

Keywords:
Software engineering
Software development
Competence
Competencies
Kano model

ABSTRACT

Context: Developing high-quality software requires skilled software professionals equipped with a set of basic and essential software engineering competencies (SEC). These competencies and the satisfaction levels derived from them change over a project's lifecycle, or as software professionals move from one project to another.
Objective: Previous studies suggest a lack of means enabling SEC stakeholders to identify and assess competencies suitable for different projects. Additionally, previous research has mainly portrayed SEC to be static and overlooked their evolution over time and across projects. We investigate how we could effectively identify and match the competencies of software professionals necessary for different projects.
Method: We follow a mixed-method approach to iteratively develop and evaluate a framework for identifying and managing SEC. In so doing, we use the results of an extensive literature review, focus group discussions with experts from academia and industry, and data collected through interviews with 138 individuals with a supervisory role in the software industry.
Results: Drawing on the Kano model and Competency Framework for Software Engineers, we propose a Unified Competence Gate for Software Professionals (UComGSP), a framework for identifying and managing SEC. The UComGSP consists of 62 hard competencies, 63 soft competencies, and 25 essential SEC competencies. Additionally, we propose three stakeholders' satisfaction levels for SEC assessment: basic, performance, and delighter. Furthermore, based on empirical observation, we report 27 competencies not mentioned in the reviewed literature; 11 of them are considered essential competencies.
Conclusion: Competence development involves different stakeholders, including software professionals, educators, and the software industry. The UComGSP framework enables SEC stakeholders to (i) identify SE competencies, (ii) identify the essential SEC, and (iii) assess the satisfaction levels that can be derived from different competencies. Future research is needed to evaluate the effectiveness of the proposed framework across software development projects.

1. Introduction

Considering the critical role of software in modern societies, there is an increasing demand for highly competent software professionals. This is because software development is a complex human-intensive process, and therefore, the quality of its outcomes is influenced by the skills and competencies of software professionals [1]. In such a complex environment where personal skills and teamwork are essential, having the means to know in advance the potential of the competencies of software professionals may serve as an assurance for the stakeholders. Colomo-Palacios et al. [1] suggest that the development and management of human competencies is one of the key concerns of the software industry. These authors also point out to the lack of established career paths for software professionals, due to no clear role definitions agreed on in the field, and inadequate verification of software professional competencies. Additionally, not having capable people assigned to the roles and problems with team management are two human factors that are found to contribute to failures in software projects [2–4]. In short, the competencies of software professionals are a key to successful software project management, and for this, essential competencies can be

* Corresponding author at: Information Technology, University of Jyväskylä Faculty of Information Technology, FI-40014 Jyväskylä, Jyväskylä, Finland.
E-mail address: nana.m.a.assyne@student.jyu.fi (N. Assyne).

https://doi.org/10.1016/j.infsof.2022.107020
Received 22 November 2021; Received in revised form 23 June 2022; Accepted 20 July 2022
Available online 22 July 2022

discerned [5,6]. However, a covering search of the literature did not present a holistic Software Engineering Competence (SEC) model or a framework to identify competencies, competence satisfaction levels, and the essential competencies to be used in different software projects (cf. Section 2.1). Furthermore, the analysis of the existing SEC models and frameworks (cf. Section 2.1) suggests that they are resource-intensive and complex to use, while they cannot be customized according to the specifics of different software projects. Thus, in most software projects, people are assigned to roles and teams based on the sometimes limited experience of the project manager or team leader [4]. Previous research has looked at competence identification and classification [6–8], competence measurement and assessment [9,10], and curriculum development [11–13]. However, we lack studies looking at SEC from the point of view of staffing software development projects. Thus, we look at SEC from a software engineering management perspective.

Over the past years, SEC research has attempted to identify different sets of essential software engineering competencies (e.g. [1,5,6,14–18]). These previous studies suggest that the essential competencies are same for every software development project. We, however, posit that these competencies change over a project's lifecycle or as software professionals move from one project to another. Following the assertion of previous SEC research, we emphasize the importance of the essential competencies for software development research and practice. However, we highlight the need for novel solutions that make the identification of the essential competencies less complex and according to the needs of a particular software project. Thus, we aim to develop a holistic framework for determining the essential competencies for software development that can be customized according to the characteristics of a particular software project. Such framework better aligns with the recent software development methodologies such as agile [19] and DevOps [20]. To achieve this goal, we set out to answer the following question:

How could we effectively identify and match the competencies of software professionals necessary for different software development projects?

To answer our research question, we first examine the extant literature on SEC. Adopting the Kano model [21] and the Competency Framework for Software Engineers [8] as a basis, we propose a framework called **Unified Competence Gate for Software Professionals** (UComGSP). Next, through several rounds of stakeholder consultations and interview data, we evaluate the applicability and useability of the framework. As such, this study contributes to software engineering research and practice by proposing a framework for managing and developing SEC. It is worth mentioning that the field of education (cf. the ACM community work on defining the educational content and practices [22], is not much touched in the manuscript. This is deliberate because the paper takes the viewpoint of an industry practitioner. Additionally, we define competence satisfaction levels and identify a set of essential competencies for software professionals. Finally, the study shows how the UComGSP framework can be used in practice for determining the competence satisfaction levels and essential competencies for different software development projects.

The rest of the paper is structured as follows: In Section 2 we discuss the research background. In Section 3 the methodology is discussed and in Section 4 the results are presented. In Sections 5 and 6 we discuss the results and conclusions of the study, respectively.

2. Research background

2.1. Concepts and definitions

Competence is defined as "a set of abilities, knowledge, and skills for performing an assigned task in a given context" [23] or "the personal qualities causally related to effective performance in an area of work, where individual competence integrates Knowledge, Skills, and Dispositions in a professional Context" [24]. We define competence in software engineering as *"a complete set of abilities, skills, knowledge, and capabilities needed to actively engage in a software development effectively."* On the other hand, the essential competence of a software professional is defined as knowledge, skills, and attitudes for exceptional performance in a software project [6]. Competence is mainly categorized into soft and hard skills. Soft competencies relate to the personal behavior of an individual in its interaction with others within a setup [25,26]. Hard competencies are defined as skills, knowledge, abilities, and attitude that are teachable, acquired mostly through formal training and studies, that one needs to perform a job or an assignment [27,28].

2.2. Background in the context of software engineering bodies of knowledge

A concern of the software industry is the development of the talents of human resources. This is so 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 [4,8]. As already stated, the development of software does not require complex machinery, rather it requires competent software professionals. However, the software industry faces a significant shortage of skilled software professionals [29]. To identify and train such professionals, studies have proposed various curricula to support the training development of skills, the identification and classification of competencies of SEC, and a means to assess software professionals' competencies [23,30–33].

Various attempts have been made to define the competencies needed by software professionals for software development [34,35]. Their success in doing so, however, is debatable [35]. For example, several works such as [6,30–32, 36–38] have defined, identified, and classified competencies for software engineering. In proposing a Software Engineering Body of Skills (SWEBOS), Sedelmaier & Landes identified and structured competencies of software professionals into three categories. 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 [33]. Also, practitioner guide documents such as SWECOM is for assessing SEC by looking at the skill area and work activity for each skill activities in an increasing level of 5 stages. are [23], The Software Assurance (SwA) Competency Model is for assessing and providing assurance to software professionals. It has five competence levels [22]. The European e-Competence Framework (e-CF) aims at standardizing the ICT professionals' competencies within the European Union. It has 40 reference competencies and five e-CF areas [39]. The essence kernel of OMG looks at the capabilities required to carry a 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 of which teams can assess the competencies [40]. These models or frameworks mentioned above have attempted to consolidate their assessment to five levels, maybe because they take more fine-grained approaches suitable for education and related assessment. It is worth mentioning that the competencies of the software professional have not kept pace with what the industry requires [8,23,41].

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 [31,33,42–44]. Also, it has been established that the software industry faces a shortage of skilled software professionals. Although there are scientific studies (e.g., [11,45,46]) and practitioner documents (e.g. [47,48]) for training software professionals, the gap remains between what the educational institutions produce and what the industries require. We acknowledge the existence of well-structured curricula to support the training of software professionals such as the current CC2020 and the previous ones. However, they are mostly viewed from an academic or scientific perspective. As suggested by studies such as [31,33,42], there exists a gap between competencies that the education institute produce and what the industry needs. SEC

development is not the sole responsibility of one institution. For this reason, in this study, we approach SEC from software business and project management viewpoints. Therefore, the study departs from the frameworks that assess the gradual development of an individual's skills, abilities, and knowledge in their journey to becoming a professional or a more proficient professional. Instead, it fills the gap in developing a framework for managing competencies needed for software projects. In short, a better way of staffing a software development efforts.

2.3. Software engineering competency frameworks and models

To develop an appropriate competence framework, we have analyzed the existing SEC models or frameworks proposed by previous studies (see Table 1). To do that, we synthesized the concepts and results reported by these studies, including stakeholders, software engineering roles, competence categorization, competence satisfaction levels, personnel competence, organizational competence, and valued competence. Stakeholders are a person(s) or party(/ies) with interest in SEC development. According to [24], 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 purpose). Connected to the categorization are the satisfaction levels. In looking at the prerequisites for software engineering education, Thurner et al. [7] argued that the "core problem is not a lack of general intellectual capacity, but rather significant deficiencies in specific base competencies" [20, p. 1069]. Thus, they defined the base competence with a stack level necessary for acquiring technical competencies as three levels, that is, self-competencies, social competencies, and practical and cognitive competencies [7].

The software assurance competency model enables software organizations to assess the capabilities of software assurance of professionals on a scale of one to five [22]. Therefore, we propose a satisfaction level to represent studies that report some form of assessment level for SEC. We also suggest classifying the frameworks and models into two categories: Personnel research, and organizational research, according to their research area of origin. Thus, we define the personnel competence area as the research that focuses on the software professional competencies (i.e., the skills, the abilities, and the attitudes required for developing software products or services). And the organizational competence area is the research that focuses on tools or instruments used for organizing, assessing, measuring, and managing the personnel competencies (e.g., SEC assessment and identification models and frameworks). Last but not least, work such as [6,29,52,23,24], etc., have studied and established that there exist some competencies that are essential for software development. To observe and analyze any phenomenon requires a framework or model of the entire area. Therefore, we identify existing models and frameworks created for SEC to collect the elements for a comprehensive framework development.

The list of models and frameworks in Table 1 is based on an extensive review on previous studies that present a 'framework' or a 'model' for the SEC area [53]. To identify potential studies that have been published after our earlier literature review study [53], we searched Google Scholar , using the search terms such as 'software competency model', 'software competency framework', 'software competence model', and 'software competence framework'. From the listed models or frameworks in Table 1, three models consider some form of stakeholder in their analysis [14,24,49,7,17,25]. Five studies propose models or frameworks that consider different software engineering roles [4,8,14,49,50,7,25–27]. Concerning competence categorization, four models or frameworks provide the means to categorize competencies [4,14,24,49,7,17,25]. However, we did not find any model from the identified studies that could be used to identify the satisfaction levels of SEC. All the models identified in this analysis represent the personnel competence research area. With respect to the organizational competence research area, there is only one model, which is the work of [49]. We also did not find any model that could be used to identify the essential competencies.

Based on the above comparison of models and frameworks of SEC, we conclude that no model or framework considers the assurance of competence satisfaction level to the stakeholders. Additionally, we were not able to find any model or framework for assessing the essential competencies of software professionals. Lastly, from the categories listed as valuation points by this study, we also did not find any previous model or framework that would consider all the elements identified in Table 1. Therefore, to fill this gap, we set out to develop a unified SEC framework that not only considers the key elements of SEC (i.e., competence identification, competence assessment, and the key stakeholder of competence development), but also can be used to determine the essential competencies of software professionals and also to provide competence assurance to the stakeholders. It must be noted that we have specifically focused on identifying different competency models and frameworks relevant for software engineers, rather than studies proposing frameworks for general human resources competencies (e.g., see Bohlouli et al. [54]).

Table 1
Comparison of competence models and framework for software professionals.

The study	Stakeholders	Roles	Competence categorization	Satisfaction levels	Personnel competence	Organizational competence	Essential competence
Competence learning framework (CoLeaF) [24]	Yes	No	No	No	Yes	No	No
A Competency Framework for the Stakeholders of a Software Process Improvement Initiative [49]	Yes	Yes	Yes	No	Yes	Yes	No
Competence model for testing teams [14]	Yes	Yes	Yes	No	Yes	No	No
Formal model for assigning human resources to teams in software projects [4]	No	Yes	Yes	No	Yes	No	No
Best-fitted resources methodology [50]	No	Yes	No	No	Yes	No	No
Competency framework for software engineers [8]	No	Yes	No	No	Yes	No	No
A competency framework for software development organization [17]	No	No	No	No	Yes	No	No
Linking personality traits and interpersonal skills to gamification [51]	No	No	No	No	Yes	No	No

2.4. Kano model

The Kano model is a quality framework for mapping and prioritizing product features to customer needs [21]. 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 [55]. According to [23,24], competence development is not the sole responsibility of software professionals, rather it involves key stakeholders like software companies and the software industry in general. In our view, this is similar to the objective of the Kano model. Additionally, the Kano model provides a means for assessing the quality of the product to be developed. Similarly, in SEC studies measurement and assessment is one of the main areas according to [6,8–12, 23,22,45, 56]. Therefore, we claim that the identification, assessment, and development of SEC is an interrelated function of the developer and the customer, hence our choice of the Kano model, which has recently been gaining grounds in SE studies and it related field with studies like [55, 57]. 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 the IT systems and concluded that the model prioritizes user involvement, i.e., it allows the inclusion of customers' views to the development of a system [55,57–62, 35]. In this study we use 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 as a product or service.

According to Kano et al. [21], a customer's decision-making options on product or service acquisition are based on conscious and subconscious deliberations [21]. There is, therefore, the need to understand these processes of decision-making to help develop products or services. Kano et al. [21] 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, on the other hand, are an expected pre-requisite that customers know, and they are essential influential factors in the customer's decision-making. These critical pre-requisite requirements create high levels of satisfaction when employed appropriately. The last requirement termed 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 [21].

2.5. Competency framework for software engineers

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 [8]. The framework is based on the activities and interactions of professionals during the software development process. The constructs of this framework are the main classification of competence: hard and soft. The hard competency category relates to the technical aspects of software engineering based on the definition of the software engineering roles proposed in SWEBOK [63]. They are project management, requirement analysis, software design, programming, validation and verification tests, configuration management, quality, tests, documentation, and maintenance. 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, we also want to acknowledge that the field of software engineering has been undergoing a paradigm shift, such as agile [19] and DevOps [20] approaches to developing software. Such a shift brings into focus some emerging roles and responsibilities that hitherto were not part of the one being used for the study, 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. The soft part of the categorization is classified into social competencies and personal competencies. Social competencies include interpersonal relations, cooperation and working in a team, and handling and conflicts resolution. Personal competencies, on the other hand, include development in the job, personal development, 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" [8]. It is important to note that CFSE was chosen because, in organizational settings, competencies are linked to roles. CFSE provides software engineering roles in the hard competence category.

2.6. Competence satisfaction levels

The development of UFSCSL and UFHCSL (see Fig.1), which are derived from the CFSE and the Kano model, is reported in detail in previous work [64,65]. To enable the classification based on soft and hard competencies, the CFSE was divided into two separate frameworks. This is because each category, either hard or soft, has a different granularity. From among frameworks such as (e.g., [66–69]) for identifying software engineering competencies, the CFSE framework is the one with more granularity, thus making it easy for an in-depth analysis. In addition, the Kano model is acknowledged and used for research in software engineering, but so far, not for analyzing competencies. However, the Kano model provides a means for prioritizing features on competencies development path based on how they satisfy the "customers", meaning in our case, the software community stakeholders. Thus, this provides a means to chart a new path for competence research. Therefore, these frameworks (UFHCSL and UFSCSL) support determining the satisfaction levels in competencies and, further, determining the essential competencies of software professionals.

To use the UFSCSL, first, the competencies are identified and classified using the variables in [8] within the frameworks. Then each competence identified or classified is subjected to the metrics of the Kano model to determine its satisfaction levels. Thus, given as basic, performance and delighter competencies for the social competencies (interpersonal relation, cooperation and work in a team, and handling and conflicts resolution) and the personal competencies (development in the job, personal development, rights, and limits) were used for the classification the soft competencies. On the UFHCSL, hard competencies are identified and classified using the hard category in [8] framework, followed by competency identification or classification subjected to the metrics of the Kano model (see Table 2) to determine its satisfaction levels. The Categorization metrics are divided into three main parts (satisfaction levels): basic, performance, and delighter competencies. In each part a number of parameters are considered for the roles (project management, requirement analysis, software design, programming, validation & verification, configuration management, test & quality, and documentation). The key in this is the satisfaction levels and the determination of the essential competencies. Therefore, one can avoid using the soft and hard categorization and use any categorization suitable to the project. The development of the two models above was iterative and conducted with expert consultation that will be detailed next.

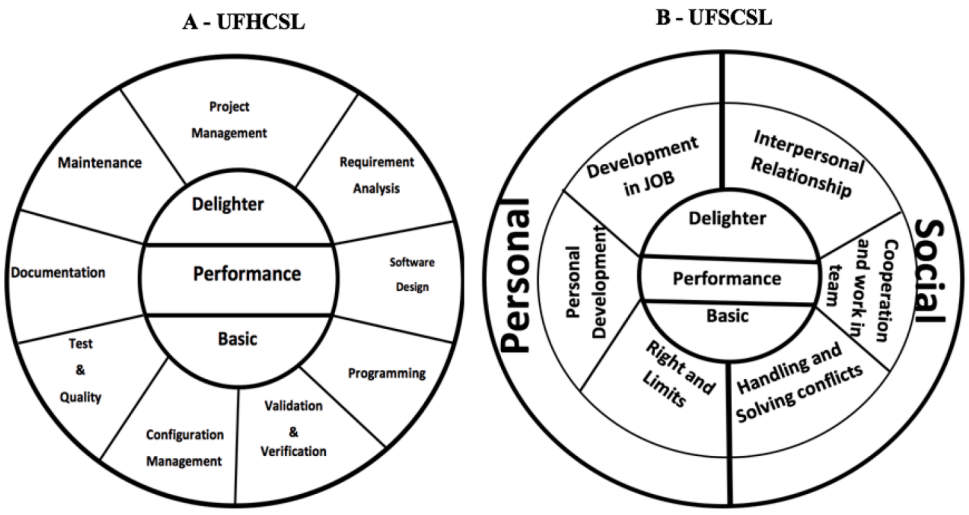


Fig. 1. Competence framework for software professionals [56,57].

Table 2
Categorization metrics for Kano analysis [21,70].

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 variables in nature	Innovation or wow factor They delight customers when delivered They will not dissatisfy customers when missing They are most unspoken of by customer Can be attributes or variables in nature

3. Research approach

Fig. 2 shows the research process phases followed in this study. In each of the three phases, we used different methods and sources for collecting research data.

3.1. Phase 1

In phase 1, we reviewed the SEC literature (cf. Section 2.3) to understand different models and frameworks used for identifying, assessing, and evaluating or managing competencies in software engineering. To that end, several studies were identified and considered including [4,8,71–74, 10,14,24,49,51,66,68,69]. After an analysis, it was apparent that the SEC research area lacks a holistic model or framework that can be used by both practitioners and academics without resorting to a heavy academic exercise. Furthermore, existing models dealt with the stakeholders separately [24]. Therefore, to develop a unified and comprehensive framework, we adopted the Kano model and CFSE as a starting point. Thus, the initial version that had the intention of identifying competencies of software development, in general, was achieved through the literature review and the competence analysis. This version was subsequently carried to phase 2 for further development.

3.2. Phase 2

In phase 2, different versions of the framework were published in peer-reviewed conferences. This was done to obtain feedback for the incremental design of the final framework. The first version (i.e., version 1) was published as a work-in-progress in the proceedings of Euromicro SEAA 2019, Greece. The framework was presented to the conference participants of the work-in-progress track, to collect feedback for the incremental development of the framework. Thus, the conference was used as a forum for collecting data from a group of experts on the subject (i.e., similar to a focus group discussion). Participants in the session were informed of this intended purpose of the presentation, the framework was presented to them, and their comments and suggestions were collected by the first author. The participants suggested two main areas for improvement. First, competence categorization (i.e., soft and hard) is essential for the development of the framework, mainly because both categories have different meanings and usage [32]. Second, the satisfaction levels of SEC must be added as a key new feature to the framework.

In the follow-up versions 2a and 2b, the categorization of the competencies was considered. Thus, two different frameworks were created, for soft and hard competencies respectively. The SEC research has so far treated hard and soft competencies as separate categories. Therefore, creating a framework for each was necessary at this point. Additionally, a minimum satisfaction level was introduced in both versions, 2a and 2b

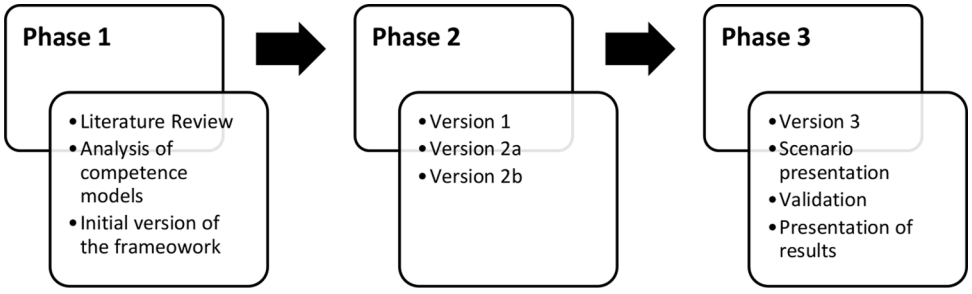


Fig. 2. Framework development process.

(cf. Section 2.6). These two versions of the framework were presented at separate conferences. This was deliberate, to validate the feedback on the initial version (version 1). The hard competence framework (version 2a, (UFHCSL) was presented at the 10th International Conference on Software Business 2019 (more details in [65]). Participants of the conference were at all occasions informed that the paper presentation will be used as a focus group discussion to take feedback for the incremental development of the framework. At the end of the discussion, it was noticed that presenting only the hard competence framework does not argue strongly for a full development stack of software. Also, since the framework is not only focusing on the identification of competencies, but it includes the determination of satisfaction levels, there is a need to complement it with the soft competence framework. Additionally, it was also noticed that role titles differ in different software development processes; therefore, it will be appropriate to use generic role titles for the framework. Furthermore, presenting only the proposed framework without some preliminary results, to justify its validity is not enough for the participants to appreciate its usefulness. Version 2b (UFSCSL) was presented at the 12th International Conference of Software Quality Days 2020. In this conference, we presented the framework for soft competencies (more details in [64]). Additionally, we acted upon one of the feedback items from version 2a, which is to use the proposed framework to analyze competence data and include it as a preliminary result. The focus group discussions focused on version 2b framework, with identified competencies and the satisfaction levels proposed. At the end of the discussions, the following were noted for further development: (i) for the study to be appreciated by the community, both categorizations must be considered, and (ii) the determination must be shown as a blueprint rather than a static identified competence for all projects. It is worth mentioning that the conference setup, which has similarities to a focus group discussion, adds to the reliability of the evaluative feedback: The authors have no direct connections to the attendees, nor can they influence the discussion at a forum moderated by the track chair. Additionally, the conferences were targeted both at the academic and professional communities. The track attendees have interest in the subject area, and diverse research and practice experiences and perspectives. Therefore, their opinion on how to move forward with the development can be considered objective.

In all three stakeholder's consultation was administered before the final framework was developed. Fig. 3. shows participants of the stakeholder consultations. The participants were people who participated in the specific track in which the said papers were presented. That is, for the Euromicro SEAA 2019 Greece (Work-in-progress track), 10th International Conference on Software Business 2019, Finland (Software Business Education track), and International Conferences on Software Qualities 2020, Austria (Industry Challenges and Collaborations track).

3.3. Phase 3

To achieve the goal of this study, which is to develop a holistic framework for the determination of satisfaction levels for SE competencies and the essential competencies for SE, we propose a framework that can be used by both academia and practitioners. In phase 3, we developed the final proposed framework, called **Unified Competence Gate for Software Professionals** (UComGSP), the assumptions of the framework, and a scenario of using the framework. As already stated, the proposed framework must support the key SEC stakeholders in determining the satisfaction levels and essential competencies for software engineering. As suggested earlier, here 'competence' is treated as a 'product' (cf. the Kano model) to be developed, as expected by the 'customers' of that product, meaning the software community.

Fig. 4. shows an overview of the three steps for using the UComGSP. Before entering step 1, the definition of the project is provided by the project manager or management team; this includes details to enable a concise project type selection. In the first step, based on the project definitions, the project management identifies the competencies needed for the project. In the second step of the process, each identified competence is assessed using the metrics in Table 2. Based on the assessment, the competencies are classified to be either 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 metrics 3, it is delighter (see Table 2). In step 3, the prioritized list of competencies generated in step 2 is used to select a candidate with the most suitable profile.

In the next section, we present a scenario to illustrate the use of the proposed framework (UComGSP). Table 3 contains competencies that have been identified and classified using the categorization of hard and soft competencies.

To make a choice using the metrics stated in Table 2, the following must be observed. The categorization of soft competencies (teamwork, knowledge transfer, and cooperation) was used as an example to illustrate how the choice is made. On the hard competencies' categorization (coding competencies across platforms, good coding skills, and basic coding skills) were also used to show how the choice is made. Making a choice as preference will mean that, cooperation which is a behavioral competence is defined as the ability to work with others in a software project which is required in any software project. This competence meets metrics 1 in Table 2. Thus, cooperation will be classified as 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 a software project. Therefore, they are expected pre-requisites that are known, and they are essential influential factors on the software industry decision-making options on competence. These are critical pre-requisite requirements that create high levels of satisfaction when employed appropriately. Therefore, teamwork meets metric 2 in Table 2 and is classified as performance competence. Knowledge transfer competence is a behavioral competence of having an outstanding ability to impart knowledge to others in a software project. 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 metrics 3 and is classified delighter. Basic coding skills are technical skill 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 are technical competence such as commenting well and self-reliance in coding. Such technical competencies are expected pre-requisites 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 software community can use the framework proposed (UComGSP) above to determine the satisfaction levels of competencies for any software project. Table 4a. provides details of the competencies and their satisfaction levels used in the scenario.

Basic competencies (satisfaction level) are pre-requisite competencies that are necessary and are expected by the software industry. Mostly they are taken for granted. The software industry sees these competencies as natural when delivered properly. However, when delivered poorly, the software industry will complain. **Performance competencies (satisfaction level)** 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 "uni-dimensional" competence. **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. It is important to note that the satisfaction levels have been

Fig. 3. Participants characteristics.

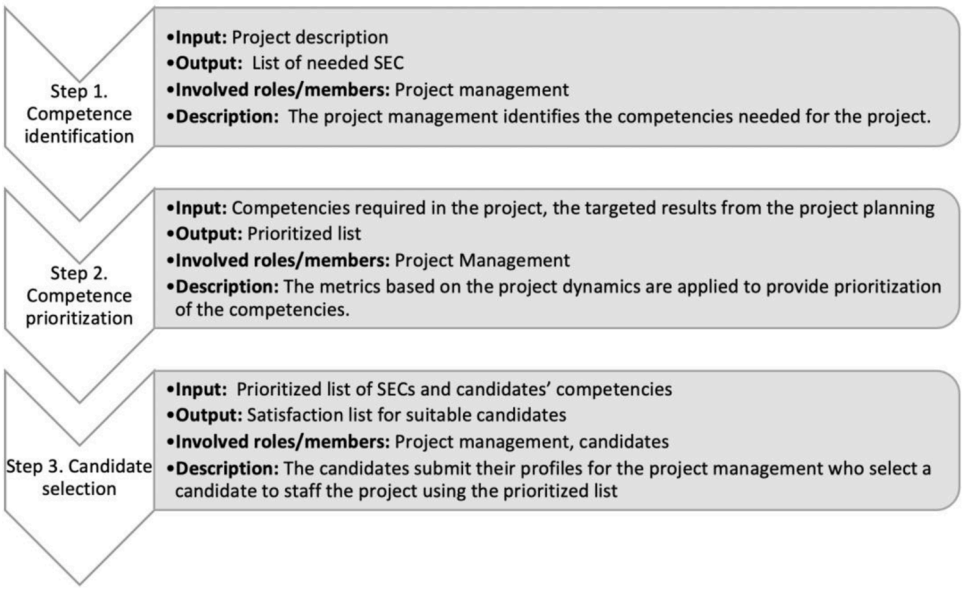


Fig. 4. Process steps for using the competence gate for software professionals (UComGSP).

Table 3
Identified competencies.

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

Table 4a
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

formed by the researchers and not professionals (interview data). However, the models and the levels were presented to professionals (in workshops) to get their feedback for further development of the final model.

In developing the final version, which is presented in this paper, all feedback from the group discussions was taken into consideration. Thus, in this paper, to provide an avenue for different usage of the framework, we present the two separate frameworks, one for soft and one for hard (cf. Section 2.5) competence for determination of competence and satisfaction levels. Additionally, we present a scenario for determining satisfaction levels based on different software projects. Thus, Fig. 5 is the final framework. The A - UComGSP shows the framework without the competencies used to illustrate how to use the framework. The 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

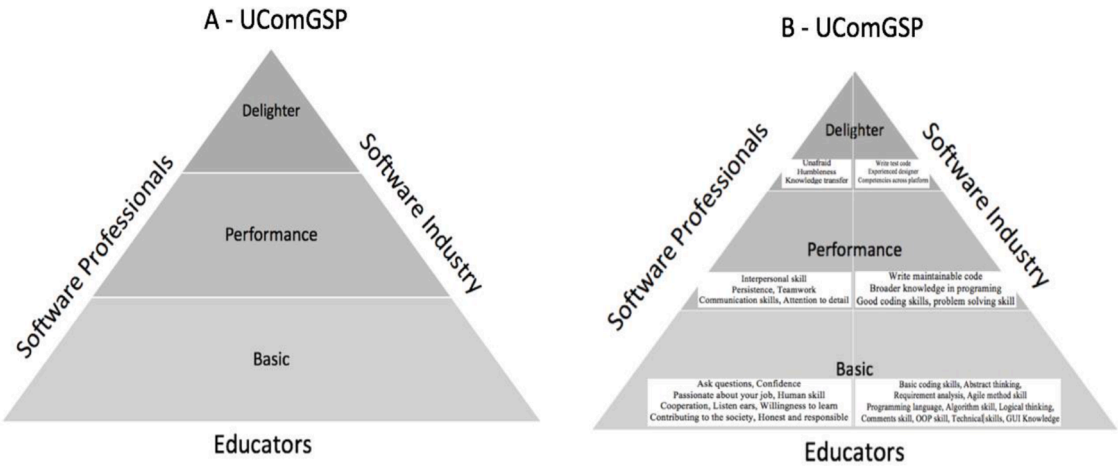


Fig. 5. Unified competence gate for software professionals (UComGSP).

determine which employers they should employ and at the same time use it to validate competencies needed by a particular software project.

3.4. Framework validation

We use a job advertisement for the position of software developer published on monster.com (see Appendix E) to show how the UComGSP can be used for the selection of a suitable candidate depending on the project, and the satisfaction level is expected to be achieved in the project. As pointed out by [75], a competence framework enables companies to efficiently manage the competencies of the employees in terms of recruiting and optimizing their employment. To that end, depending on the objective of the software project, the UComGSP helps in prioritizing the competencies needed to successfully complete a software project. A common description of necessary and desirable competencies in job advertisements include statements like "required skills to succeed in this job" and "valuable skills but not necessary". The question is how project managers determine these different categories and how do they select people for the said positions. In our example, we assume that a project manager uses the UComGSP to guide the job profiling process for one-person software development project or a small software development team. In step 1 **competence identification**, the project manager identifies the competence needed for the job. As can be seen these competencies are identified and are listed. In this scenario, the competencies are listed in Table 4b column 2 and categorized according to the satisfaction levels proposed in the UComGSP. In step 2 **competence prioritization**, we used the gate to prioritize the competencies (cf. Section 3.3). Table 4b column 3,4, and 5 show the prioritization for this job seeking to fill a software developer role.

Step 3 **candidate selection**, based on the resume of the candidates, selection is made using the prioritization guide for step 3. Now, let us use two different scenarios to demonstrate how our framework can help managers to choose a suitable candidate. In the first scenario, the new software developer is expected to complete a one-person software development project (i.e., all the activities in the project are conducted by one person). In this scenario, someone with all the five competencies is an ideal candidate, someone with the first four competencies (i.e., all the basic and performance competencies) is desirable, and someone with the first two competencies is satisfactory for completing this project.

In the second scenario, the new software developer will join a small software development team. Using Table 4b, the project manager can prioritize candidates so that the combination of team members' skills would suit their intended satisfaction level. In small development teams, the members must collectively have the skills and competencies required for delivering the project on time and within budget [76]. However, considering the collaborative nature of such projects communication and teamwork skills are crucial [29]. Therefore, in this scenario, someone with all the five competencies is an ideal candidate, someone with great communication skills and the skills missing in the team is desirable, and someone with skills missing in the team is satisfactory for completing this project.

Even though we used "project manager" for the illustration, as suggested by [23,24], the development of competencies is not the sole responsibility of one role or even only one institution. We believe that other stakeholders can also consider the framework in a similar manner.

Table 4b
Competence satisfaction level framework for validation.

No.	Competencies	Basic	Performance	Delighter
I	Programming languages (JavaScript, Scala, Go, Python, Java or PHP)	x		
ii	Understanding of web development process	x		
iii	Great communication skills		x	
iv	Analytical mindset and problem-solving skills		x	
v	Experience in consulting or digital commence			x

3.5. Data collection and analysis

To validate the framework, an interview data set 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 course at Norwegian University of Science and Technology (135) conducted the interviews with a given interview structure from the course lecturer.[1] The interview data was processed by 4–5 students into a spreadsheet (Appendix A). To analyze the data, we followed the guidelines suggested by [77,78]. Fig. 6 represents an overview of the respondents' characteristics.

As the data set was qualitative data and the authors also aimed to analyze it with 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 [79,80]. It is defined as exploring large amounts of textual information in an orderly way to establish a trend or pattern [81]. "Content analysis is a research technique for making replicable and valid inference from texts (or other

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

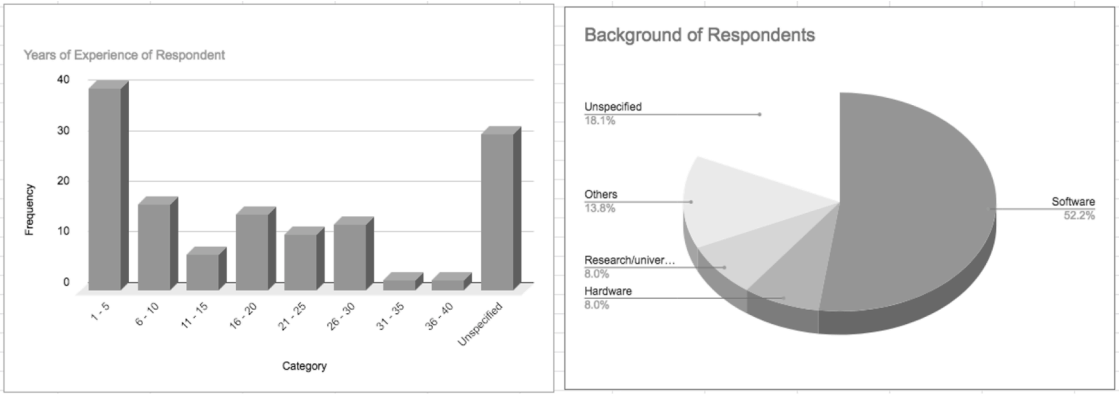


Fig. 6. Respondents characteristics.

meaningful matter) to the contexts of their use" [80]. In this study, it will be used in determining the presence of certain themes or concepts within an interview data of 138 participants into an organized and concise conceptual structure. The expected outcome of content analysis can be quantitative or qualitative.

We examine meanings of the content of the interview data for the identification of relevant concepts based on the research question "what are the competencies expected from persons working as software professionals in your organization". A content analysis must follow a certain procedure [80]. Thus, [80] 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 validating (evidence to justify the analysis). In addition, [82] proposed the following six steps: design, unitizing, sampling, coding, drawing inferences, and validation for conducting a content analysis. [79] also suggest the following: start with a research question, decide on a sampling strategy, define the recording unit, construct categories for analysis, test the coding on samples of the text and assess reliability, and carry out the analysis as what to consider, when using content analysis for empirical study. Both [79] and [80] agree that there is a certain methodological approach for conducting a content analysis study, and that the methodology adds to the rigor. Therefore, adopting the steps recommended by [79] and [82], we followed the stages and steps shown in Fig. 7 for perfroming the content analysis.

As can be seen from Fig. 7, the processing of the interview data and the analysis followed in three stages, familiarizing with the data set, coding and classification, and developing the framework. We explain each of these stages, steps, and the outcome as sub-sections below.

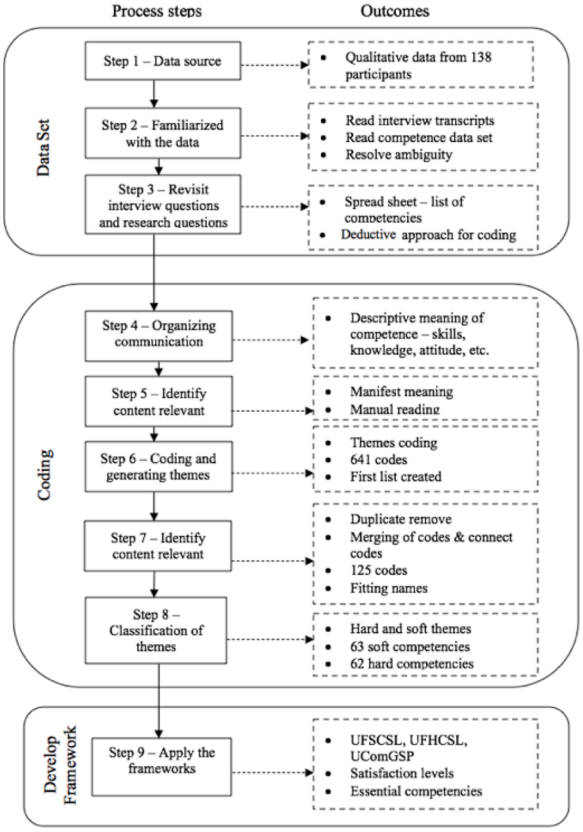


Fig. 7. Content analysis process follows in this study.

3.5.1. Stage one – data set

Step 1 in Fig. 7 shows the selection of dataset used for the content analysis. The dataset consists of interviews with 138 software professionals in a supervisory role including senior software engineers, team leads, managers, and mentors from Norway. The data set includes the transcripts of interviews in English, demographic information about the participants, and the lists of competencies mentioned by them. In Step 2 (Fig. 7), the first author familiarized themselves with the data set. This was done by reading through the original transcripts and the competencies extracted from the transcript. The listed competencies were entered into a spreadsheet to allow for easy coding. Discussions were made with the head of the data collection team to resolve any ambiguity in the data. In Step 3, the first author revisited the interview questions and the listed competencies to compare both data. This was to enable the first author to properly get acquainted with the data with a specific focus on the interview questions and the research question for this study. At the end of stage one, it was concluded that the data source contains texts that can be used for the analysis. Table 5 shows a sample of interviewees and the competencies that they expected from a software professional. The directed or theory-led approach for coding is appropriate for this text because it allows for a top-down or theory-driven approach to coding based on emerging themes from literature. This process was done together by all the authors.

3.5.2. Stage two – coding

As can be seen from Table 5, the data source brings forth competencies mentioned by different participants that are the same, similar, or unique depending on which part of the information communication technology industry the participants are coming from. However, our aim is to identify competencies for software development. This we did by

Table 5
Excerpt of text from the interview data.

Participant	Field	Years of experience	Listed competencies
A	Software	3	Motivation / go-getter-attitude; Ability to work in teams; Work efficiently as an individual; Experience with version control tools; Sharing knowledge/ experience in team; Ask others for help;
B	Hardware	9	Versatile with working individually/ in teams; Creativity; Cooperation; Passionate for what they do; Thrive working with others;
C	Research/ University	30	Pragmatic; Solution-oriented; Cooperating with others in a team; Agile and general interest in programming; Humble to accept other solutions;
D	Others	20	Well written code; Well documented code; Versatile; Curious/passionate about programming/computers; Ask when you don't understand/admit a mistake;
E	Unspecified	3	Specialty in specific developer language; Broad understanding of design; Broad understanding of the business proposition; Broad understanding of user applicability; An open mind for collaboration and problem solving; Work in agile process in multidisciplinary teams;

Table 6
Category names and codes after content analysis.

Proposed fitting name	Different codes identified
Basic Programming Skills	coding skills, be able to code, write understandable code, skill to assess code, able to interpret the code, programming skills, programming languages, programming, basic principles of programming, basic principles of programming, basic programming skills, fundamentals in programming, enthusiasm for programming, knowledge and skills related to programming, good knowledge of the fundamentals
Good Programming Skills	good at coding, writing good code, well-written code, write good and effective code, writes code that is robust and easily maintainable, write code of high quality, good at coding, ability to program well, ability to program well, ability to program well, being a good programmer, good at programming, broad knowledge of programming, programming experience and coding skills, being a proficient coder
Development Process Skills	broad understanding of design, user-oriented design, utilize existing solutions, design methodology/documentation requirements, knowledge of the whole software development process, basic knowledge of systematic working methods, development process, adapt methods and software
Leadership Skills	Leadership, arouse enthusiasm in others, ability to inspire, understand people they work with, estimation of workload, approachable, ability to handle a high workload, understanding of the corporate structure, constantly communicate status
Teamwork	team spirit, communication and working together in a team, master cooperative teams, communication and teamwork
Self-Reliance Skills	eager and independent, independent, work independently, self-reliance, ability to work independently, able to work independently, work independently, independency, work efficiently as an individual

coding and classification of the data set by manually reading the listed competencies in the spreadsheet. Therefore, we made an attempt to find consistent meaning by finding patterns and themes. In Step 4 to Step 7, the coding is performed. Coding and classification are defined as the segmenting and categorizing meaningful analytical units from a text [80].

Step 4 (Fig. 7), Organizing communication is the starting point of the coding process. In this process, we focus our reading of the text by identifying the exact meanings of the phrases or terms provided by the participants specifically, mentioning words as skills, knowledge, or attitude. We focused on the manifest meaning of the obvious contents of the text. This is because the interview questions asked the participants to list the competencies, they expect from software professionals. Thus, the intent was obvious in the text. In Step 5 Identify relevant content, we focus on meaning relevant to our question, therefore, the manifest meaning was used for coding. Therefore, the descriptive meaning of skills, knowledge, attitude, etc., were used to identify the relevant content. This was done by working manually through the text in the spreadsheet. In Step 6, we classified the identified relevant contents to themes. That is, codes with the same meaning were grouped together. This was done by identifying patterns and connections in the codes. Thus, we developed a master list with all identified codes. At this stage, 641 codes were identified (1st order concepts) without paying attention to duplicates.

Therefore, in Step 7, we removed duplicate codes, i.e., coded units with the same meaning. This was done according to the objective of the study, to identify competencies of software professionals. At the end of Step 7, a new master list of 125 codes was generated by assigning fitting or more appropriate names to the generated codes, with reference to previous literature. Table 6 shows some of the codes (different codes identified) and final codes names (proposed fitting name) after the coding process (*2nd order themes*).

In Step 8, we approached the master list generated using the deductive approach. That is, the generated codes were classified using the categories "hard" and "soft" competencies (*aggregate dimensions*). Thus, we identified 63 soft competencies and 62 hard competencies (see Fig. 8). In all, three levels of grouping were applied to our raw data during the analysis. As suggested by Gioia et al. [83], one of the features of rigor in qualitative research is how analysis is approached. As explained above, We illustrate our approach by adopting the concepts from the work of [83]. That is, "1st order concepts", "2nd order themes", and "aggregated dimensions". The 1st order concepts were identified leading to creating the 641 list codes. These 1st order concepts where categorised as 2nd order themes, which were ultimately were aggregated resulting in aggregated dimensions such as hard competencies, soft competencies, satisfaction levels, and the identification of competencies using the roles of software engineering.

3.5.3. Stage three – develop

As stated in Section 3.5 the participants were asked open-ended questions about the competencies needed and expected from software professionals working in their organization. However, the theoretical lens, the UComGSP was used during the analysis of the data. Section 3.5.3 Therefore, we applied our development frameworks (UFSCSL, UFSCSL, and UComGSP) to respond to the main research question of the study as the final analysis (cf. Section 3.3). This results in the development of the soft and hard satisfaction levels of software professionals, the identification of the essential competencies, and the proposed UComGSP for future competence identification template. UComGSP is developed based on Kano as a theoretical base, secondary data from SE literature, and feedback from experts. It entails Kano and CFSE.

Soft Competencies - Social	Interpersonal relation	**1) communicate to the outside world; 2) sociable**
		3) communication skills; 4) adaptability; 5) human skills; 6) interpersonal skills
		7) social skills; 8) contributing to the society
	Cooperation and work in a team	**9) knowledge transfer; 10) see the bigger picture; 11) leadership**
		12) teamwork; 13) team organizer; 14) approachable; 15) open and communicating; 16) learn from others; 17) voice your own opinions
		18) Cooperation; 19) maturity; 20) teach and share knowledge; 21) dedication to work
	Handling and solving conflicts	**22) humbleness; 23) customer awareness; 24) understand customer needs**
		25) meeting skills; 26) contact with clients
		27) Listening ears; 28) compromise; 29) empathy
Soft Competencies - Personal	Development in the job environment	**30) unafraid; 31) creative and brave; 32) think outside the box**
		33) persistence; 34) flexible; 35) versatile; 36) focus; 37) accuracy; 38) analytical skills; 39) logical mindset and keep an overview; 40) creativity
		41) Willingness to learn; 42) curious; 43) passionate about their job; 44) ask questions; 45) confidence; 46) honest and responsible
	Personal development	**47) can apply theories in development; 48) see opportunity in systems; 49) initiative; 50) separate work and being available; 51) self-sufficient**
		52) precise and detail-oriented; 53) self-reliance; 54) independence; 55) understand needs for further development; 56) know the working environments
		57) pragmatic; 58) patience; 59) open to new ideas
	Right and limits	---
		60) attention to detail
		61) Introspection and admit error; 62) admit ignorance; 63) interest in the field
Hard Competencies	Project management	---
		1) team organization and communication skills; 2) project schedule skills
		3) project management skills
	Requirement analysis	**4) new idea promotion; 5) business knowledge**
		6) broader knowledge on systems; 7) problem solving skills; 8) connectivity skills
		9) open mind; 10) abstract thinking; 11) requirement analysis; 12) understanding clients need; 13) communicate ideas
	Software design	**14) experienced designer; 15) knowledge of industry standards; 16) mathematical knowledge in core sciences subject**
		17) broader design skills; 18) technical proficiency; 19) learn new skills; 20) scientific approach
		21) system design; 22) technical skills; 23) agile methods skills; 24) mathematics skills
	Programming	**25) coding competencies across platforms; 26) best practice and standard skills**
		27) good coding skills; 28) broader knowledge in programing; 29) write maintainable code; 30) proficiency in scriptwriting (JS, PHP & Java, CSS); 31) proficiency in algorithms; 32) patterns development skills; 33) problem analyzer; 34) good UX
		35) basic coding skills; 36) programming language; 37) SQL skills; 38) algorithm skills; 39) logical thinking; 40) comments skills; 41) framework development skills; 42) OOP skills; 43) HTML skills; 44) GUI knowledge
	Validation and verification	---

	Configuration management	---
		45) knowledge in specifications and account for future changes; 46) knowledge in distributed systems design
		47) version control; 48) knowledge in architecture; 49) knowledge in networking
	Test &Quality engineering	**50) write test code**
		51) test driven development; 52) quality check; 53) testing and verification
		54) testing skills; 55) benchmarking
	Documentation	---
		56) writing documentation
		57) knowledge in documentation
	Maintenance	**58) electronics and mechanics skills**
		59) server management/deployments; 60) technical evaluation skills
		61) knowledge in network OS and PC OS; 62) IT security

Fig. 8. Soft and hard competencies and their satisfaction levels (Green = Delighter; Grey = performance; Orange = Basic). The essential competencies are shown in **bold**.

4. Results

4.1. Competencies of software professionals

Fig. 8 shows all the competencies identified during the content analysis. These competencies are classified into soft competencies (n = 63) and hard competencies (n = 62). As can be seen in the figure, soft competencies are divided into two main groups of social and personal competencies. The figure also shows the satisfaction levels for each group of competences.

4.2. Satisfaction levels of soft and hard competencies

The results show the individual competencies and their satisfaction levels, meaning basic, performance, and delighters. The soft competence was grouped according to the broader category of social and personal competencies. With regards to hard competence as stated in Section 3.4.3, the analysis was done using UFHCSL. Therefore, we present the result using the roles of software engineering from SWEBOK [63]. The roles as stated in the UFHCSL are project management, requirement analysis, software design, programming, validation and verification tests, configuration management, tests and quality engineering, documentation, and maintenance. We also provided definitions using the classification levels from the Kano model for the competencies. It is worth mentioning that the satisfaction levels are dynamic and might differ based on a role or a project. Fig. 8 shows soft and hard competencies and their satisfaction levels.

As shown in Fig. 8, out of the 63 soft competencies s, 34 were personal competencies and 29 were social competencies. **Personal competencies** are personal attributes for working well in different spheres of life, while social competencies are competencies for organizing cooperation and interpersonal relations in a software development project. In the **social competencies category**, cooperation and work in a team had the highest number of items (13), followed by interpersonal relation and

Table 7
Essential competencies.

Competence category	Essential competencies
Hard competencies	
Project management	-
Requirement analysis	(1) new idea promotion, (2) business knowledge
Software design	(3) experienced designer, (4) knowledge of industry standards, (5) mathematical knowledge in core sciences subject
Programming	(6) coding competencies across platforms, (7) best practice and standard skills
Validation and verification	–
Configuration management	–
Test & Quality engineering	(8) write test code
Documentation	.-
Maintenance	(9) electronics and mechanics skill
Soft competencies	
Interpersonal relation	(1) communicate to the outside world, (2) sociable
Cooperation and work in a team	(3) knowledge transfer, (4) see the bigger picture, (5) leadership skills
Handling and solving conflicts	(6) humbleness, (7) customer awareness, (8) understand customer needs
Development in the job environment	(9) unafraid, (10) creative and brave, (11) think outside the box
Personal development	(12) can apply theories in application development, (13) see opportunity in systems, (14) initiative, (15) separate work and being available, (16) self-sufficient

handling and solving conflicts with eight each. In the **personal competencies category**, development in the job environment has the highest number of competencies with 17, personal development is next with 14 competencies, followed by rights and limits with four competencies. The result also shows that **performance satisfaction level** is highest with 26 competencies. These are what the software community expects and can articulate. They are mostly in the minds of the software community and when they are delivered well, they create more satisfaction. These competencies can be described as a "uni-dimensional" competence in that the satisfaction grows exponentially when executed properly. **Basic satisfaction** is second highest with 21 competencies. Basic competencies are pre-requisite competencies that are necessary and are expected by the software community. Mostly they are taken for granted. The software community sees these competencies as natural when delivered properly. However, when delivered poorly, they will complain. Next is **delighter** with 16 competencies. They are competencies that are unexpected by the software community. Mostly unexpected by the software community but increases the delight and surprise when available however its absence may have no effect on the software community.

Fig. 8 further shows hard competencies and their satisfaction levels using the roles of software engineering. Programming has the most, 20 competencies. It is followed by software design with 11 competencies, requirement engineering with ten competencies and test & quality engineering with six competencies. Configuration management and maintenance have five competencies each, whereas project management has only three, and documentation has two. In our data, no competencies were identified for validation and verification. As stated earlier, this phase of the study focuses on hard competencies and their satisfaction levels. Table 6 shows only data regarding hard competencies. Readers interested in the satisfaction levels of soft competence can read more detail on the soft competencies and their satisfaction levels in the work of [84]. For the satisfaction levels, 28 hard competencies were identified as pre-requisite competencies necessary and are expected by the software community: 25 performance satisfaction levels and nine delight satisfaction levels. Despite categorizing the competencies based on the traditional software engineering roles proposed by SWEBOK, we believe that the UComGSP framework can be used by newer methods such as agile and DevOps. For instance, effective communication, collaboration, and teamwork are three agile SEC proposed by [85]. These soft competencies are also considered in our framework, as shown in Fig. 8. Therefore, it is important to note that the competencies included in our framework may change depending on a specific methodology. However, as stated the framework allows for providing inputs and output based on the dynamics of the projects.

4.3. Essential competencies of software professionals

According to the definition of [21], delighters are attractive or so-called "wow"-factors that are valued in a product. Therefore, we present our delight competencies as the essential competencies for software engineering. Since we intend to show the essential competence of software professionals, we present the essential competencies for both hard and soft competence categories

Table 7 shows the competencies we classified as the essential competencies. This is divided into soft and hard competence categories. A total of 25 essential competencies were identified from our data. This is made up of 9 hard essential competencies and 16 soft essential competencies.

5. Discussion

In this study we aimed at developing a holistic framework for determining the essential competencies for software development that can be customized according to the type of a software project. Upon analyzing the extant literature, we identified a lack of frameworks that take into consideration the key SEC stakeholders (i.e., software industry, software professionals, and educators). Therefore, we propose the UComGSP, a framework that enables different SEC stakeholders to identify software professional competencies, including the essential ones, and to determine their satisfaction levels.

5.1. Contributions

Our study has several contributions to research and practice. First, even though previous research has treated soft and hard competencies separately, our study contributes with a framework that allows for a holistic view. The difference between some of these soft and hard competencies is becoming narrower. An example is the definition that hard competencies can be taught [86]. In fact, these days, soft competencies are also being taught. With the holistic framework resulting from our study, SEC studies can consider the key stakeholders in one look instead of looking at them individually. Additionally, to advance research on SEC, we have analyzed different models and frameworks of SEC to elicit variables (stakeholders, software engineering roles, competence categorization, competence satisfaction levels, personnel competence, organizational competence, and essential (cf. Table 1)) from literature. This analysis provides a pretext for studying SEC or analyzing and comparing different SEC models and frameworks.

Additionally, in this study, we identify 25 essential competencies of software professionals. Sixteen were soft essential competencies and 9 were hard essential competencies. Most of these identified competencies are consistent with the ones reported in previous studies such as [6,38,87,88]. However, we made a total of 11 new observations of essential competencies (e.g., *business knowledge, knowledge of industry standards, coding competencies across platforms, knowledge transfer)*. Under them, seven belong to hard competencies and four to soft competencies. We present the complete list using the categories UFSCSL and UFHCSL in

appendices B and C. It is important to note that we do not mean the new observations are new competencies. Rather, they are not reported in previous SEC literature, to the best of our knowledge. Thus, we call for further empirical research to learn more about these new observations in the SE environment. For instance, software security used to be considered after the development of the software, but this is not the case anymore [89]. For this reason, IT security, which is observed as a new competence for software professionals, needs to be investigated further to understand the implications on software development. We argue the same for all new observations identified in this study. Previous studies suggest that the essential competencies are a core part of SEC studies and that a landmark study on the essential competence was published in 1995 [6]. The SE field has evolved greatly since the 1990′s. In Appendix D, we compare the essential competencies identified in this study with prior studies. Furthermore, according to the findings of this study, agile methodology skills are a prerequisite competence necessary and expected from software professionals. Therefore, we cannot separate agile methodology skills from other SEC, but pay attention to them in any curriculum development for software engineering.

In practice, the UComGSP framework can be used for SEC management. This includes the development, assessment, and customization of SEC, across software projects. The SEC stakeholders can use the framework to identify competencies, determine their satisfaction levels, and identify the essential competencies for software development. Development teams and companies can use this framework to identify competencies that are needed for a particular software project. As discussed before, and also highlighted by [90], staffing people for software development is not straightforward. Using the UComGSP, organizations can assess the competencies needed for a particular type of software or project and fill the software engineering roles accordingly. Similarly, considering that the quality of software products is influenced by software professionals competencies [31], an organization can assess the satisfaction level they may derive from a particular competence. In other words, software developed by employing the most suitable and relevant bundle of SEC more likely excels in quality and is successful in the market.

Another important contribution of this study is categorizing SEC into three satisfaction levels. As such, the UComGSP provides a means to assess the competence level of software professionals (i.e., the satisfaction level of competence) at three levels: basic, performance, and delighter. The UComGSP can be used by companies to determine the basic competencies expected from software professionals before they can be employed. Similarly, there are some competencies that enable an organization to increase its performance in developing software products and services. Lastly, delighter competencies are not expected from software professionals before they can be employed. However, they give an edge to the companies in the competition. The software professionals can also compare their competencies to the framework. They can use it to determine pre-requisite competencies (basic competencies) and competencies that they can develop when they are working in the industry (performance and delighter competencies). For a professional, performance competencies become a benchmark for employment while delighters give them a competitive advantage in the employment market. Therefore, software professionals can use these levels as a career-building ladder and manage their career path, which, according to [31], is one of the problems facing software professionals. The educational institutions providing the foundational competencies and training software professionals can use the framework to identify and consider for their curricula the competencies expected by the software industry.

5.2. Limitations

To discuss the limitations of this study, we use four types of threat to validity suggested by [91] and [92]. 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 [93,94]. Therefore, the concept sometimes becomes misconstrued when studying it. Thus, in this study, to mitigate this threat, we based this study using operational words from the definition of [23–25, 27]. Thus, the transcribed interviews were analyzed using well-established operational words from literature. Additionally, 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 see the consistency of our results.

Causal relations as it relates to the internal validity was not an issue with this study. This is because we did not apply statistical inference in this study for causal relationships. Thus, this study did not suffer this threat.

External validity concerns the extent to which the findings of a study can be generalized [91]. Our study is based on literature review, interview data, and group discussion. The first limitation is related to the scope of the literature review. As mentioned in Section 3.1, we have specifically focused on identifying different models and frameworks related to SEC. Therefore, because of this methodological decision we might have excluded some studies that do not focus on SEC, but still proposing competency frameworks (e.g., on human resources in general) that could be used also in the SEC context. Another limitation of the study is the scope of the data collection was conducted in companies situated in one country (Norway). Nevertheless, most of the companies that the interviewees worked for have global representation and businesses outside Norway. Additionally, it must be noted that the cultural settings of the Scandinavia counties are similar. Notwithstanding this, the limitation invites further studies with data from other countries to test the generalizability of the results. With the development of the 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 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 [91] 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 that was followed has been provided in the study to allow for replication of the study. Again, since the authors of this study were not directly involved in the data collection, to increase the reliability of the finding, 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 process of validating the frameworks was done by the key stakeholders in SEC. Last but not least, different data sources were used to attain the findings.

6. Conclusions

This study aimed to provide a holistic framework enabling SEC stakeholders (i.e., software professionals, educators, and the software industry) to (i) identify SE competencies, (ii) identify the essential SEC, and (iii) assess the satisfaction levels derived from those competencies. A qualitative approach was adopted for the study for the reason to be open to emerging issues. The research literature on SEC, interview data from 138 participants in various positions within the industry in Norway, and expert consultations were used to develop and validate the proposed framework. In total, we identified 63 soft competencies and 62 hard competencies and mapped them to the acknowledged roles in the software engineering practice. Amongst the identified competencies, 25 essential competencies of software professionals were identified. We have also provided a working definition for the essential competencies as ***the skills, knowledge, and attitudes of software professionals***

necessary for excellent performance in a software project. Next, using the Kano model of quality assessment, the competencies were analytically assigned to three satisfaction levels (i.e., basic, performance, and delighters) with the respective definitions. Through a scenario illustration, we demonstrated how SEC stakeholders could evaluate the competence levels for different software projects. However, we suggest further research to understand how competencies within the satisfaction levels can change (e.g., from basic to performance or delighter). Future research is needed to further study and validate the newly observed competencies, from which 11 were classified as essential competencies. Additionally, the developed framework must be validated by allowing the stakeholders to use it in software development projects.

CRediT authorship contribution statement

Nana Assyne: Conceptualization, Methodology, Formal analysis, Data curation, Writing – original draft, Writing – review & editing, Visualization. **Hadi Ghanbari:** Conceptualization, Methodology, Supervision, Visualization, Writing – original draft, Writing – review & editing. **Mirja Pulkkinen:** Methodology, Writing – original draft, Writing – review & editing, Supervision.

Declaration of Competing Interest

None

Data Availability

Data will be made available on request.

Acknowledgments

The authors would like to thank the anonymous interviewees and reviewers and Prof. Pekka Abrahamsson for their insights and contributions to this research. We would like to thank Prof. Abrahamsson and his research assistants at the Norwegian University of Science and Technology for their invaluable help with data collection in Norway. Hadi Ghanbari's contributions to this research have been partly supported by the European Union's Horizon 2020 Research and Innovation Programme, under Grant Agreement No. 856602.

A. Appendices

Appendix A - Interview structure

1. Participant details:

a Name
b Position in the organization
c Name of the organization
d Years of experience

2. Competence question

a What competencies are expected working as a software professional in your company?
b Why are they important?

Appendix B

Identified soft competencies and prior work.

Category	Identified soft competency	Comparison
Social competencies		
Interpersonal relation	Sociable, communication skill, adaptability, human skill, interpersonal skill, social skills	Consistent with prior work
	Communicate to the outside world, contributing to the society	New observations
Cooperation and work in team	See bigger picture, leadership, teamwork, Cooperation, teach and share knowledge, team organizer, approachable, open and communicating, learn from others	Consistent with prior work
	Maturity, knowledge transfer, 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
Personal competencies		
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
Personal development	Separate work and being available, self-sufficient, precise and detail oriented, self-reliance, independence, pragmatic, patience, initiative, open to new ideas.	Consistent with prior work
	Can apply theories in application development, 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

Appendix C

Identified hard competencies and prior studies.

Category	Software engineer competencies	Comparison
Project management	Team organization and communication skill, project schedule skills, project management skill	Consistent with prior work
	Communicate Ideas, business knowledge, new idea promotion	New observations
Requirement analysis	See bigger picture, broader knowledge on systems, problem solving skill, connectivity skill, open mind, abstract thinking, requirement analysis	Consistent with prior work
	Understanding clients need	New observations
Software design	Experienced designer, mathematical knowledge in Core sciences subject, broader design skill, technical proficiency, learn new skills, scientific approach, system design, technical skills, agile skill, mathematics skills	Consistent with prior work
	Knowledge of industry standards	New observations
Programming	Best practice and standard, good coding skill, broader knowledge in programing, write maintainable code, proficiency in script writing (JS, PHP & Java, CSS), proficiency in algorithms, problem analyzer, good UX, basic coding skills, programming language, SQL skill, algorithm skill, logical thinking, comments skill, framework skill, OOP skill, skills in HTML, GUI Knowledge	Consistent with prior work
	Coding competencies across platforms, patterns and best practice	New observations
Configuration management	Distributed systems design, version control, knowledge in architecture, knowledge in Networking	Consistent with prior work
	Specifications and account for future changes	New observations
Test &Quality engineering	Write test code, test driven development, quality check, testing and verification, testing skill	Consistent with prior work
	Benchmarking	New observations
Documentation	Writing documentation., documentation skill	Consistent with prior work
	–	New observations
Maintenance	Electronics and mechanics skill, server management/deployments, technical evaluation skill, knowledge in network OS and PC OS	Consistent with prior work
	IT security	New observations

Appendix D

Essential competencies and prior studies.

Category	Identified soft competency	Comparison
Hard competence	Experienced designer, write test code	Consistent with prior work
	New idea promotion, business knowledge, knowledge of industry standards, mathematical knowledge in core sciences subject, coding competencies across platforms, best practice and standard, electronics and mechanics skill	New observations
Soft competence	Sociable, see bigger picture, leadership skills, customer awareness, understand customer needs, unafraid, creative and brave, think outside the box, initiative, separate work and being available, self-sufficient	Consistent with prior work
	Communicate to the outside world, knowledge transfer, can apply theories in application development, see opportunity in systems	New observations

Appendix E

The link to the advert on monster.com
Screenshots of the Monster.com Adverts

Software Developer - Helsinki
Columbia Road
Helsinki, UUSI

Columbia Road is an independent Futurice subsidiary challenging the old ways of running ecommerce and digital business. We exist to increase our clients' revenue and help them get more customers in the digital era. Our tight-knit team consists of business consultants who understand software and of highly talented business-driven design and technology specialists.

We believe that technology is a tool that we need to master, but the actual business impact of the service is the first thing we want to understand and select proper tooling after that.

We, like Futurice, are a great place to work. Currently, we have over 15 nationalities working in the company and the official language is English. Our company spirit is one of a true start-up, but with a twist of hard-core experience. We believe in giving a lot of responsibility and freedom to our employees. Diversity, equity, and inclusion are at the heart of our essence and everyone at Columbia Road is committed to these values.

Job description

You build businesses with code. You build storefronts, APIs and the integrations needed to glue them together. Your focus is likely on some part of the stack, be it frontend, backend or something like complex integrations but you aren't too afraid to get your hands dirty with the other areas when necessary. You are comfortable working with a variety of programming languages and technology stacks - or at least you are interested to jump in and learn new skills. Columbia Road is not trying to provide the same solution to everyone, and sometimes that requires quite a bit of adapting from our side.

The focus on digital sales still means that we are tackling a large variety of projects. These range from ecommerce platform and data platform projects to site optimisation and mobile development. For a few concrete examples of our tech cases, take a look at the Servaali case of a new microservice business-to-business shop, the full webstore renewal project for Oura, and our work for Matsmart in the optimisation of an existing store.

All of our consultants work directly with client stakeholders to build a good understanding of their business and its challenges. While development is probably where most of your time goes, we are in the business of creating impact for our clients, and that means spending some time to understand how the business functions. We help our clients in translating business opportunities into actionable plans and concrete solutions.
Read more about how we code and design high performing digital sales channels.

We work as diverse teams of digital sales experts, tailored to each project. In some projects, this means working as part of a larger development team, while sometimes the setup needed is a growth team. As we believe that best results arise from employee freedom, all teams have wide latitude to choose the best tools and ways of working for themselves. This also means that your technology stack and specific role may vary between projects. We see this as a great opportunity for professional growth and are looking for colleagues who also value this. The more experience you have, the more you'll be able to take responsibility for your client projects and help other Roadies grow as well.

We pay extra attention to professional growth and learning at Columbia Road. We encourage you to follow your own points of interest, and we do this proactively. Together we'll continuously think about the best career advancement possibilities for you, both during and in between your client projects. Our internal learning opportunities come in many forms: events, courses, certifications, and afterworks, for example.

- Experience in building web services and familiarity with both frontend and backend development. Your focus can be more on the frontend, backend or broader systems but you aren't afraid to jump in at any point. Languages familiar to you could include e.g. JavaScript (and frameworks such as React, Angular, or Vue.js), Scala, Go, Python, Java, or PHP.
- Understanding of the entire web development process (design, development, deployment, operations)
- Great communication skills
- Analytical mindset and problem solving skills
- Past experience in consulting or digital commerce would be very valuable but is not necessary

Recruitment process

First interview: before diving deeper into your skills, we want to know who you are and what motivates you. This is also an opportunity for you to learn more about Columbia Road. We also let you present highlights from your career and tell us about your roles and learnings.

Second interview: you get to meet more Columbia Road people, and the conversation will zoom in to the areas that we've identified as most interesting in the first meeting. We're trying to get a solid understanding of your skills and ambitions. You will understand better how you would potentially fit into our team.

Sometimes it's useful to have a third round. After all, we're making a match between our organisation and you, as a human being. We want to get this right.

When we're done with the face-to-face discussions, we'll contact the people you have chosen as references.

- As a Columbia Road employee, you have the option to buy stocks of the company through the common employee share program
- We offer a wide range of possibilities to shape your own role as you see best fit for your personal development goals and growth
- You'll have the best people to work with and learn from in a truly flat organisation
- A diverse and inclusive workplace
- Competitive salary
- Brick-n-mortar office in downtown Helsinki and flexible remote working
- Lunch benefit
- Phone benefit
- Pick the laptop and phone of your choosing
- Health care & service, including dental and mental health
- Childcare services when your child falls ill
- Insurance for travel and free-time
- Recurring training & learning sessions
- Wellness and personal development (tools, events, etc.) reimbursement
- Weekly breakfasts & healthy daily refreshments
- Sports, culture & massage vouchers
- Company-wide excursion trips
- Option to have housing as a part of total compensation (tax-deductible)
- Compensation for working on open source or social programs in your free time

Take a look at our Culture Code & People to get to know us better.

References

[1] R. Colomo-Palacios, C. Casado-Lumbreras, P. Soto-Acosta, F.J. García-Peñalvo, E. Tovar-Caro, Competence gaps in software personnel: a multi-organizational study, Comput. Human Behav. 29 (2) (2013) 456–461, https://doi.org/10.1016/j.chb.2012.04.021.

[2] R. Nelson, IT project management: infamous failures, classic mistakes, and best practices, MISQE 6 (2) (2007) 67–78.

[3] B.R.N. Charette, Why software fails we waste billions of dollars each year on entirely preventable mistakes market crash : after its new, IEEE Spectr (2005) 1–10, no. September.

[4] M. André, M.G. Baldoquín, S.T. Acuña, Formal model for assigning human resources to teams in software projects, Inf. Softw. Technol. 53 (2011) 259–275, https://doi.org/10.1016/j.infsof.2010.11.011.

[5] C.D. Manawadu, M.G.M. Johar, S.S.N. Perera, Essential technical competencies for software engineers : perspectives from Sri Lankan undergraduates, Int. J. Comput. Appl. 113 (17) (2015) 27–34 [Online]. Available: https://pdfs.semanticscholar.org/c01d/90376b8d36bea073190aee76f77ec883f1f6.pdf.

[6] T. Turley, M. Bieman, Competencies nonexceptional of exceptional and software engineers, J. Syst. Softw. 1995 (28) (1995) 19–38, 28.

[7] V. Thurner, B. Axel, K. Andreas, Identifying base competencies as prerequisites for software engineering education. IEEE Global Engineering Education Conference (EDUCON), 2014, pp. 1069–1076, https://doi.org/10.1109/EDUCON.2014.6826240. April, pp.

[8] J.G. Rivera-Ibarra, J. Rodríguez-Jacobo, M.A. Serrano-Vargas, Competency framework for software engineers, in: 2010 23rd IEEE Conference on Software Engineering Education and Training, 2010, pp. 33–40, https://doi.org/10.1109/CSEET.2010.21.

[9] R. Studt, G. Winterfeldt, J. Mottok, Measuring software engineering competencies, in: IEEE Global Engineering Education Conference, EDUCON, 2015, pp. 908–914, https://doi.org/10.1109/EDUCON.2015.7096081, no. March, pp.

[10] Y. Sedelmaier, D. Landes, A multi-perspective framework for evaluating software engineering education by assessing students ' competencies SECAT – a software engineering competency assessment tool, in: 2014 IEEE Frontiers in Education Conference (FIE) Proceedings, 2014, pp. 1–8, https://doi.org/10.1109/FIE.2014.7044331.

[11] K. Kobata, T. Uesugi, H. Adachi, M. Aoyama, A curriculum development methodology for professional software engineers and its evaluation, in: Proceedings of IEEE International Conference on Teaching, Assessment and Learning for Engineering: Learning for the Future Now, TALE, 2014, pp. 480–487, https://doi.org/10.1109/TALE.2014.7062552, 2015no. December, pp.

[12] P. Hubwieser, et al., Pedagogical content knowledge for computer science in German teacher education curricula, Comput. Educ. (2013) 95–103, https://doi.org/10.1145/2532748.2532753.

[13] CC2020 task force, Comput. Curricula (2020), 2020.

[14] J. Saldaña-Ramos, A. Sanz-Esteban, J. García-Guzmán, A. Amescua, Design of a competence model for testing teams, IET Softw 6 (5) (2012) 405–415, https://doi.org/10.1049/iet-sen.2011.0182.

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

[16] S. Goel, Competency focused engineering education with reference to IT related disciplines: is the Indian system ready for transformation? J. Inf. Technol. Educ. 5 (2006) 27–52, https://doi.org/10.28945/233.

[17] A. Orsoni, B. Colaco, A competency framework for software development organizations, in: 2013 UKSim 15th International Conference on Computer Modelling and Simulation, 2013, pp. 507–511, https://doi.org/10.1109/UKSim.2013.101.

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

[19] P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta, "Agile software development methods: review and analysis," VTT Publ., no. 478, pp. 3–107, 2002.

[20] P. Debois, Devops: a software revolution in the making, J. Inf. Technol. Manag. 24 (8) (2011) 3–39.

[21] N. Kano, N. Seraku, F. Takahashi, S. Tsuji, Kano, Attract. Qual. Must-Be Qual. J. Japanese Soc. Qual. Control 14 (1984) 39–48.

[22] T. Hilburn, M. Ardis, G. Johnson, A.J. Kornecki, and N.R. Mead, "Software assurance competency model," 2013. [Online]. Available: http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=47953.

[23] IEEE, Software engineering competency model (SWECOM), IEEE (2014) [Online]. Available, http://www.dahlan.web.id/files/ebooks/SWECOM.pdf.

[24] S. Frezza, et al., Modelling competencies for computing education beyond 2020 : a research based approach to defining competencies in the computing disciplines, in: 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education, 2018, pp. 148–174.

[25] F. Ahmed, L.F. Capretz, S. Bouktif, P. Campbell, Soft skills and software development: a reflection from software industry, Int. J. Inf. Process. Manag. 4 (3) (2013) 171–191, https://doi.org/10.4156/ijipm.vol4.issue3.17.

[26] M.L. Matteson, L. Anderson, C. Boyden, M.L. Matteson, L. Anderson, Search of meaning, portal Libr. Acad. 16 (1) (2016) 71–88, https://doi.org/10.1353/pla.2016.0009.

[27] D.S. Urs, Soft skills for the engineering students, Synergy 9 (2) (2013) 137–142 [Online]. Available, https://pdfs.semanticscholar.org/fbf4/0a9446331b41d3f11ef3b7035fe33e2b452f.pdf.

[28] M. Omar, N.L.A. Khasasi, S.L.S. Abdullah, L. Hashim, R. Romli, N. Katuk, Defining skill sets requirements for agile scrum team formation, J. Eng. Appl. Sci. 13 (3) (2018) 784–789, https://doi.org/10.36478/jeasci.2018.784.789.

[29] A. Calazans, et al., Software requirements analyst profile : a descriptive study of Brazil and Mexico, in: 2017 IEEE 25th International Requirements Engineering Conference, 2017, pp. 204–212, https://doi.org/10.1109/RE.2017.22.

[30] J. Pérez, C. Vizcarro, J. García, A. Bermúdez, R. Cobos, Development of procedures to assess problem-solving competence in computing engineering, IEEE Trans. Educ. 60 (1) (2017) 22–28, https://doi.org/10.1109/TE.2016.2582736.

[31] R. Colomo-Palacios, C. Casado-Lumbreras, P. Soto-Acosta, F.J. García-Peñalvo, E. Tovar-Caro, Computers in human behavior competence gaps in software personnel : a multi-organizational study, Comput. Human Behav. 29 (2) (2013) 456–461, https://doi.org/10.1016/j.chb.2012.04.021.

[32] A.M. Moreno, M. Sanchez-segura, F. Medina-dominguez, L. Carvajal, Balancing software engineering education and industrial needs, J. Syst. Softw. 85 (7) (2012) 1607–1620, https://doi.org/10.1016/j.jss.2012.01.060.

[33] Y. Sedelmaier, D. Landes, Software engineering body of skills (SWEBOS), in: IEEE Global Engineering Education Conference, EDUCON, 2014, pp. 395–401, https://doi.org/10.1109/EDUCON.2014.6826125, no. April, pp.

[34] W.S. Humphrey, Managing the Software Process, Addson Wesley, 1989.

[35] N.R. Mead, D. Shoemaker, The software assurance competency model: a roadmap to enhance individual professional capability, in: Software Engineering Education Conference, Proceedings, 2013, pp. 119–128, https://doi.org/10.1109/CSEET.2013.6595243.

[36] A. Zendler, D. Klaudt, C. Seitz, Empirical determination of competence areas to computer science education, J. Educ. Comput. Res. 51 (1) (2014) 71–89, https://doi.org/10.2190/EC.51.1.d.

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

[38] R. Colomo-Palacios, E. Tovar-Carlos, Å. Garcia-Crespo, J.M. Gómez-Berbís, The case of software engineers identifying technical competences of IT professionals, Int. J. Hum. Cap. Inf. Technol. Prof. vol. 1 (2010) 31–43, https://doi.org/10.4018/jhcitp.2010091103, no. March, pp.

[39] CEN, "European e-competence framework 3.0 - a common European framework for ICT professionals in all industry sectors," 2014.

[40] Object Management Group, Essence - kernel and language for software engineering methods (Essence), no. Versión 1.2. 2018.

[41] J.M. Pawlowski, P. Holtkamp, H. Kalb, Globalization competences in information systems and e-learning. Lecture Notes in Business Information Processing, 2010.

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

[43] H. Chassidim, D. Almog, S. Mark, Fostering soft skills in project-oriented learning within an agile atmosphere, Eur. J. Eng. Educ. 43 (4) (2018) 638–650, https://doi.org/10.1080/03043797.2017.1401595.

[44] H. Cornide-Reyes, et al., Introducing low-cost sensors into the classroom settings: improving the assessment in agile practices with multimodal learning analytics, Sensors (Switzerland) 19 (15) (2019), https://doi.org/10.3390/s19153291.

[45] M. Ardis, D. Budgen, G.W. Hislop, J. Offutt, M. Sebern, W. Visser, Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering, IEEE Computer Society, 2014, 2014.

[46] J.M. Pawlowski and P. Holtkamp, "Towards an internationalization of the information systems curriculum," in *Multikonferenz Wirtschaftsinformatik 2012 - Tagungsband der MKWI 2012*, 2012, vol. 2011, no. March, pp. 437–449, [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84879862734&partnerID=40&md5=377ec064ba5a4994dcda2fb9148041fa.

[47] A. (Editor) Pyster, "Graduate software engineering 2009 (GSwE2009) curriculum guidelines for graduate degree programs in software engineering," 2009.

[48] IEEE-CS and ACM, "Curriculum guidelines for undergraduate degree programs in software engineering," 2015. [Online]. Available: http://usir.salford.ac.uk/6939/.

[49] I.E. Espinosa-curiel, J. Rodríguez-jacobo, J.A. Fernández-zepeda, A competency framework for the stakeholders of a software process improvement initiative, in: International Conference on Software and Systems Process, 2011, pp. 139–148.

[50] L.D. Otero, G. Centeno, A.J. Ruiz-Torres, C.E. Otero, A systematic approach for resource allocation in software projects, Comput. Ind. Eng. 56 (4) (2009) 1333–1339, https://doi.org/10.1016/j.cie.2008.08.002.

[51] M. Papoutsoglou, G.M. Kapitsaki, N. Mittas, Linking personality traits and interpersonal skills to gamification awards, in: 2018 44th Euromicro Conference on Software Engineering and Advanced Applications, 2018, pp. 214–221, https://doi.org/10.1109/SEAA.2018.00042.

[52] J. Chang, T. Wang, M. Lee, C.-Y. Su, P.-C. Chang, 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, 2016, pp. 396–400, https://doi.org/10.1109/ICALT.2016.78.

[53] N. Assyne, H. Ghanbari, M. Pulkkinen, The state of research on software engineering competencies: A systematic mapping study, J. Syst. Softw. 185 (111183) (2021) 1–18, https://doi.org/10.1016/j.jss.2021.111183.

[54] M. Bohlouli, N. Mittas, G. Kakarontzas, T. Theodosiou, L. Angelis, M. Fathi, Competence assessment as an expert system for human resource management: a mathematical approach, Expert Syst. Appl. 70 (2017) 83–102, https://doi.org/10.1016/j.eswa.2016.10.046.

[55] Y.C. Lee, L.C. Sheu, Y.G. Tsou, Quality function deployment implementation based on Fuzzy Kano model: an application in PLM system, Comput. Ind. Eng. 55 (1) (2008) 48–63, https://doi.org/10.1016/j.cie.2007.11.014.

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

[57] J. Huang, "Application of Kano model and IPA on improvement of service quality of mobile healthcare Jui-Chen Huang," vol. 16, no. 2, 2018.

[58] S. Gangurde, S. Patil, Benchmark product features using the Kano-QFD approach: a case study, Benchmarking Int. J. 25 (2) (2018) 450–470, https://doi.org/10.1108/MRR-09-2015-0216.

[59] L. Lehtola, M. Kauppinen, Suitability of requirements prioritization methods for market-driven software product development, Softw. Process Improv. Pract. 11 (1) (2006) 7–19, https://doi.org/10.1002/spip.249.

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

[61] C. Piaszczyk, Model based systems engineering with department of defense architectural framework, Syst. Eng. 14 (3) (2011) 305–326, https://doi.org/10.1002/sys.

[62] I. Richardson, Software process matrix: a small company SPI model, Softw. Process Improv. Pract. 6 (1992) 157–165, https://doi.org/10.1002/spip.144. Daft2001.

[63] P. Bourque and R.E. Fairley, Guide to the software engineering - body of knowledge (SWEBOK (R)): version 3.0. 2014.
[64] N. Assyne, Soft competencies and satisfaction levels for software engineers: a unified framework, in: Lecture Notes in Business Information Processing, 371, LNBIP, 2020, https://doi.org/10.1007/978-3-030-35510-4_5.
[65] N. Assyne, Hard competencies satisfaction levels for software engineers: a unified framework, in: Lecture Notes in Business Information Processing, 370, LNBIP, 2019, https://doi.org/10.1007/978-3-030-33742-1_27.
[66] A. Orsoni, B. Colaco, A competency framework for software development organizations, in: 2013 UKSim 15th International Conference on Computer Modelling and Simulation, 2013, pp. 507–511, https://doi.org/10.1109/UKSim.2013.101.
[67] S.T. Acuña, N. Juristo, Assigning people to roles in software projects, Softw. - Pract. Exp. 34 (7) (2004) 675–696, https://doi.org/10.1002/spe.586.
[68] B. Linck, et al., Competence model for informatics modelling and system comprehension. 2013 IEEE Global Engineering Education Conference (EDUCON), 2013, pp. 85–93, https://doi.org/10.1109/EduCon.2013.6530090.
[69] D. Tuffley, Optimising virtual team leadership in global software development, IET Softw 6 (March 2011) 176–184, https://doi.org/10.1049/iet-sen.2011.0044, 2012.
[70] Kano, "What is the Kano model? KanoModel.com," 2016. https://kanomodel.com/ (accessed Nov. 19, 2021).
[71] V. Seppanen, A relationship-based view to software engineering competence, in: International Conference on Product Focused Software Process Improvement, 2000, pp. 376–390, https://doi.org/10.1007/b72823.
[72] A.N. Aisha, J. Siswanto, I. Sudirman, Competencies model for entrepreneur development in software industries, in: 2016 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), 2016, pp. 184–188, https://doi.org/10.1109/IEEM.2016.7797861.
[73] C. Schulte, J. Magenheim, K. Muller, and L. Budde, "The design and exploration cycle as research and development framework in computing education," in *2017 IEEE Global Engineering Education Conference (EDUCON)*, 2017, no. April, pp. 867–876, doi: 10.1109/EDUCON.2017.7942950.
[74] C.F. Salviano, et al., Developing a process assessment model for technological and business competencies on software development, in: 2012 Eighth International Conference on the Quality of Information and Communications Technology 15504, 2012, pp. 125–130, https://doi.org/10.1109/QUATIC.2012.27.
[75] E. Moustroufas, I. Stamelos, L. Angelis, Competency profiling for software engineers: literature review and a new model, in: Proceedings of the 19th Panhellenic Conference on Informatics, 2015, pp. 235–240, https://doi.org/10.1145/2801948.2801960.
[76] S.S. Gharebagh, P. Rostami, M. Neshati, T-shaped mining: a novel approach to talent finding for agile software teams, in: Advances in Information Retrieval, 10772, LNCS, 2018, pp. 411–423.
[77] J. Mason, Qualitative Researching, vol. 41, no. 1. 2002.
[78] M.D. Myers, M. Newman, The qualitative interview in IS research: examining the craft, Inf. Organ. 17 (1) (2007) 2–26, https://doi.org/10.1016/j.infoandorg.2006.11.001.
[79] C. Robson, Research Real World, Second Ed., Blackwell Publishing, 2002.
[80] K. Krippendorff, Content Analysis: An Introduction to Its Methodology, SAGA Publications, 2018. Fourth Ed.
[81] M. Vaismoradi, H. Turunen, T. Bondas, Content analysis and thematic analysis : implications for conducting a qualitative descriptive study, Nurs. Heal. Sci. 15 (2013) 398–405, https://doi.org/10.1111/nhs.12048.
[82] K. Krippendorff, Content analysis, Int. Encycl. Commun. 1 (1989) 403–407, https://doi.org/10.1002/9781118541555.wbiepc065.
[83] D.A. Gioia, K.G. Corley, A.L. Hamilton, Seeking qualitative rigor in inductive research: notes on the Gioia methodology, Organ. Res. Methods 16 (1) (2013) 15–31, https://doi.org/10.1177/1094428112452151.
[84] N. Assyne, Soft competencies and satisfaction levels for software engineers : a unified framework, in: International Conference on Software Quality, 2020, pp. 69–83, https://doi.org/10.1007/978-3-030-35510-4.
[85] H. Cornide-Reyes, et al., Key skills to work with agile frameworks in software engineering: Chilean perspectives, IEEE Access 9 (2021) 84724–84738, https://doi.org/10.1109/ACCESS.2021.3087717.
[86] K.S. Harris, G.E. Rogers, Soft skills in the technology education classroom : what do students need, Technol. Teach. 68 (3) (2008) 19–25.
[87] A. Patel, Y. Benslimane, B. Bahli, Z. Yang, 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, 2012, pp. 899–903, https://doi.org/10.1109/IEEM.2012.6837870.
[88] C. Gold, J. Abke, Y. Sedelmaier, A retrospective course survey of graduates to analyse competencies in software engineering, in: 2014 IEEE Global Engineering Education Conference (EDUCON), 2014, pp. 100–106, https://doi.org/10.1109/EDUCON.2014.6826075, no. April, pp.
[89] C.D. Mano, L. Duhadway, A. Striegel, A case for instilling security as a core programming skill, in: Proceedings. Frontiers in Education. 36th Annual Conference, 2006, pp. 13–18, https://doi.org/10.1109/FIE.2006.322347.
[90] A. Barreto, M. de O. Barros, C.M.L. Werner, Staffing a software project: a constraint satisfaction and optimization-based approach, Comput. Oper. Res. 35 (10) (2008) 3073–3089, https://doi.org/10.1016/j.cor.2007.01.010.
[91] C. Wohlin, P. Runeson, M. Höst, M.C. Ohlsson, B. Regnell, and A. Wesslén, Experimentation in software engineering.2012.
[92] P. Runeson, M. Höst, Guidelines for conducting and reporting case study research in software engineering, Empir. Softw. Eng. 14 (2) (2009) 131–164, https://doi.org/10.1007/s10664-008-9102-8.
[93] F.D. Le Deist, J. Winterton, What is competence? Hum. Resour. Dev. Int. 8 (1) (2005) 27–46, https://doi.org/10.1080/1367886042000338227.
[94] B. Wernerfelt, A resource-based view of the firm, Strateg. Manag. J. 5 (2) (1984) 171–180, https://doi.org/10.1002/smj.4250050207.