# Understanding the Knowledge Gaps of Software Engineers: An Empirical Analysis Based on SWEBOK

VAHID GAROUSI, Queen's University Belfast, Northern Ireland, UK
GORKEM GIRAY, Independent Researcher, Izmir, Turkey
ERAY TUZUN, Bilkent University, Ankara, Turkey

*Context:* Knowledge level and productivity of the software engineering (SE) workforce are the subject of regular discussions among practitioners, educators, and researchers. There have been many efforts to measure and improve the knowledge gap between SE education and industrial needs.

*Objective:* Although the existing efforts for aligning SE education and industrial needs have provided valuable insights, there is a need for analyzing the SE topics in a more "fine-grained" manner; i.e., knowing that SE university graduates should know more about requirements engineering is important, but it is more valuable to know the exact topics of requirements engineering that are most important in the industry.

*Method:* We achieve the above objective by assessing the knowledge gaps of software engineers by designing and executing an opinion survey on levels of knowledge learned in universities versus skills needed in industry. We designed the survey by using the SE knowledge areas (KAs) from the latest version of the Software Engineering Body of Knowledge (SWEBOK v3), which classifies the SE knowledge into 12 KAs, which are themselves broken down into 67 subareas (sub-KAs) in total. Our analysis is based on (opinion) data gathered from 129 practitioners, who are mostly based in Turkey.

*Results:* Based on our findings, we recommend that educators should include more materials on software maintenance, software configuration management, and testing in their SE curriculum. Based on the literature as well as the current trends in industry, we provide actionable suggestions to improve SE curriculum to decrease the knowledge gap.

CCS Concepts: • **Social and professional topics** → **Model curricula**; **Software engineering education**; **Employment issues**;

Additional Key Words and Phrases: Software engineering education, education research, skill gaps, knowledge gaps, opinion survey, empirical study

**ACM Reference format:**
Vahid Garousi, Gorkem Giray, and Eray Tuzun. 2019. Understanding the Knowledge Gaps of Software Engineers: An Empirical Analysis Based on SWEBOK. *ACM Trans. Comput. Educ.* 20, 1, Article 3 (November 2019), 33 pages.
https://doi.org/10.1145/3360497

Authors' addresses: V. Garousi, School of Electronics, Electrical Engineering and Computer Science (EEECS), Queen's University Belfast, Computer Science Building, 18 Malone Rd, Belfast, BT9 5BN, Northern Ireland - UK; email: v.garousi@qub.ac.uk; G. Giray, Narli M. Coskun S. No: 4/3 Narlidere, Izmir - Turkey; email: gorkemgiray@gmail.com; E. Tuzun, Department of Computer Engineering, Bilkent University, Engineering Building, EA-501, Bilkent, Ankara - Turkey; email: eraytuzun@cs.bilkent.edu.tr.
Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2019 Association for Computing Machinery.
1946-6226/2019/11-ART3 $15.00
https://doi.org/10.1145/3360497



RIGHTSLINK

## 1 INTRODUCTION

How good are our software engineering (SE) university programs? Questions such as this are sparked in the minds of many university educators (academics) who are educating SE university students and managers who are hiring SE university graduates.

Knowledge level and productivity of the SE workforce are the subject of regular discussions among practitioners, educators, and researchers, e.g., [1–3]. Both university educators and practitioners are striving to ensure that new hires and fresh SE graduates meet the needs of the industry and possess the "right" skills and knowledge to smoothly integrate into their new teams and start contributing to their projects after being hired. However, SE is a field in which talent differences are disproportionate [4, 5]. There are academic papers from the 1980s that have reported about the impact of developer productivity and skill set on software projects [4], and the community has coined notions such as "10x" [5], which is a concept used to describe the "best" software engineers who could be 10 times as productive as the low-skilled engineers. Companies such as IBM have developed sophisticated approaches and tools to quantitatively measure their developers' productivity [6].

Furthermore, to improve the skill level and productivity of the SE workforce, companies usually have rigorous and formal training programs for their employees and for the external workforce [7]. Large companies such as IBM have systematically studied and shown that effective training has a direct impact on project success [8]. On the other hand, there are studies such as [9, 10] that have reported that many practitioners see the lack of proper skill sets in the new hires and thus many companies put major resource investments into the training of new hires. This raises questions on the quality and skill set of graduates and denotes a mismatch between what graduates have learned in their university education and what industry needs.

This article intends to add to the existing body of knowledge on the issue of knowledge gaps or skill deficiencies in engineers just entering the workforce by identifying the topics that are the most important in practice. We achieve the above goal by conducting assessment and improvement of SE undergraduate university programs based on data gathered from practitioners on what they had learned in university versus what they use in industry. We believe that educators need to instill in SE students both the theoretical foundations of SE/Computer Science (CS) and the latest industrial needs and trends (such as cloud computing and microservice architectures). Based on combined 50+ years of experience in industry and academia and experience in offering both university-level SE education and corporate training in several countries, the authors present a hybrid approach to identify the knowledge gaps of software engineers and to assess and improve SE courses and programs.

The remainder of this article is structured as follows. A review of background and the related work is presented in Section 2. We describe the study's goal, research questions, and research methodology in Section 3. Section 4 presents the results. We discuss a summary of the findings and provide the implication of findings for educators, new graduates, and managers in Section 5. Finally, in Section 6, we draw conclusions and suggest areas for further research.

## 2 BACKGROUND AND RELATED WORK

SE education and training are active areas among practitioners, educators, and researchers. A very large number of university programs are offered across the globe with a focus on CS, SE, and other areas of computing [11]. In parallel to formal university programs across the globe, many firms are specialized in offering training and certifications in various subareas of SE.

According to Evans Data Corp. [12], the total worldwide population of formally educated software engineers will exceed 25 million by 2020. Furthermore, according to a 2012 report [13], only 41% of practicing software engineers have CS- or SE-related degrees. These statistics project



RIGHTSLINK

the number of software engineers (both formally trained and non-formally trained) to be about 61 million by the year 2020. Furthermore, companies spend a huge amount of resources on "proper" education and training of this massive number of professionals to keep their knowledge up to date. Thus, it is clear that proper education and training of software engineers in an effective and efficient manner are very important.

In terms of systematic approaches on what topics are be covered in training and how to train software engineers, a large number of experience reports and approaches are discussed annually in the area's flagship venue, the IEEE Conference on Software Engineering Education and Training (CSEE&T), which was held for its 31st year in 2019 (the first event was held back in 1987). Some studies related to SE education also appear in other similar venues such as the Software Engineering Education and Training (SEET) track of the ICSE conference and the ACM Technical Symposium on Computer Science Education (SIGCSE).

In addition to these venues, some researchers and academics collected and analyzed data from the software development industry to shed light on the opportunities for improving SE education. In a recent systematic literature review (SLR) [14], the authors reviewed the body of knowledge in this area by synthesizing the findings of 33 studies in this area. By doing a meta-analysis of all those studies and using data from 12 countries and over 4,000 data points, the study provided insights that could be useful for educators and hiring managers to adapt their education and hiring efforts to best prepare the SE workforce. For brevity, we do not provide all results of the SLR [14] in this article, but only present a subset of those 33 papers and a brief summary of their results in Table 1.

The works reported in [15] and [16] can be attributed as the pioneer studies that aim to identify the knowledge gap of software engineers conducted by Lethbridge. While [17] shares the same research objective with Lethbridge's studies, it criticizes the analysis method used in [15] and [16]. We provide more information on this in Section 3.5. The study [21] conducted a survey in South Africa using a taxonomy similar to [15] and [16]. [18] focuses on comparing gaps in technical and soft skills by interviewing 22 participants. [19] and [20] identified the areas in which the students struggle when beginning their careers. In both studies, the sample size was 23 and the method for data collection was face-to-face interviews. In [22], the authors mined job ads to extract the skills needed and interviewed 26 professionals to find out knowledge deficiencies. [23] collected data from 283 participants to identify knowledge gaps. They used a taxonomy of topics based on the authors' earlier studies. The taxonomy used seems problematic since the granularity levels of the topics are different. For instance, among the least important skills are "hard sciences," "physics," "assembly," "soft sciences," "spiral development model," "extreme programming," "code generation tools," "calculus," "legal aspects," and "waterfall development model." In addition, they do not differentiate CS and computer engineering (CE) graduates from SE graduates.

In this study, the contributions differ from the related work in the following ways:

- We have selected the latest version of the Software Engineering Body of Knowledge (SWEBOK) version 3 published in 2014 [24] as the basis to conduct a detailed granular analysis of all the SE knowledge areas (KAs) and subareas (sub-KAs). To the best of our knowledge, this is the first time that SWEBOK-2014 (version 3) has been used for identifying knowledge gaps of software engineers. This is the main novelty of our article compared to all the previous works.
- Another unique characteristic of this study is the use of sub-KAs to identify knowledge gaps. This enabled us to identify the gaps at a more fine-grained level and provide better guidance for curriculum improvement.
- We analyzed the knowledge gap (addressed by RQ 3) using the analysis methods used in [15], [16], and [17] and used paired-samples t-test to provide better statistical evidence.



RIGHTSLINK

Table 1. A Selected Subset of Articles Addressing Knowledge Gaps of SEs

| Ref | Year | Research Questions and/or Objective(s) | Body of Knowledge/Taxonomy Used | Number of Participants/Data Points |
|---|---|---|---|---|
| [15] Lethbridge | 1998 | To identify the participants' learning level on a topic during their formal education, current level of knowledge on a topic, and usefulness of a topic in their career | Existing university courses | 168 |
| [16] Lethbridge | 2000 | To identify the participants' learning level on a topic during their formal education, current level of knowledge on a topic, and usefulness of a topic in their career | Existing university courses<br>SWEBOK v1 1999 | 186 |
| [17] Kitchenham et al. | 2005 | To identify the participants' learning level on a topic during their formal education, current level of knowledge on a topic, and usefulness of a topic in their career | Existing university courses<br>SWEBOK v1 1999<br>SEEK 2003 | 30 |
| [18] Colomo-Palacios et al. | 2013 | 1. Do software engineers present more deficiency in technical competences than generic competences?<br>2. Do discrepancies in competences differ along the SE career?<br>3. Are self-evaluations systematically above 360-degree evaluations? | SWEBOK v2 2004 for technical competencies | 22 |
| [19] Radermacher et al. | 2014 | 1. In what areas do recently graduated students most frequently struggle when beginning their careers in industry?<br>2. Are there certain skills, abilities, or knowledge (i.e., knowledge deficiencies) that recently graduated students might lack that can prevent them from gaining employment?<br>3. Are there issues that managers and hiring personnel attempt to identify during interviews with recent graduates that are still common after recently graduated students begin their jobs? | After extracting knowledge deficiencies identified in the existing literature, a taxonomy was created to classify deficiencies into multiple categories | 23 |
| [20] Radermacher et al. | 2015 | To gain a better understanding of knowledge deficiencies of CS graduates | Areas of knowledge deficiencies identified during interviews | 23 |
| [21] Liebenberg et al. | 2016 | 1. What are the topics professional software developers learned in their formal education?<br>2. What topics are important to professional software developers in their actual workplace?<br>3. Is there a gap between software development education and the workplace from the perspective of the software industry? | Similar to [16] | 214 |
| [22] Stevens and Norman | 2016 | 1. What does the IT industry recruit for?<br>2. How does the IT industry define "work ready"?<br>3. How does the IT industry identify a new entrant as "productive"? | List of soft skills was compiled from analyzing job ads | 543 job ads<br>26 interviewees |
| [23] Exter et al. | 2018 | 1. What importance do computing professionals place on specific technical and nontechnical topics and skills?<br>2. What gaps are there between recent graduates' rankings of the importance of each topic or skill and their perception of the degree to which that topic or skill was covered in their undergraduate experiences?<br>3. What are professionals' recommendations for improving education to better prepare students for future employment? | Based on the earlier studies of the authors | 283 |



RIGHTSLINK

- We compared the software engineers who had formally received a SE degree with the software engineers from other computing or noncomputing disciplines.

## 3 RESEARCH GOAL AND RESEARCH APPROACH

We first present the study's goal and research questions. We then provide an overview of our research approach. This section then ends by presenting the design and execution of the opinion survey that we conducted to gather data for our empirical study.

### 3.1 Goal and Research Questions

Formulated using the Goal, Question, Metric (GQM) approach [25], the goal of this study is to characterize the knowledge gap between SE education and the industrial needs. Based on the overall goal, we raised the following research questions (RQs):

- RQ 1: Which KAs of SE are perceived as the most important, at work, by practitioners? The first RQ aims to explore the importance of SE topics in the workplace and is further broken into three sub-RQs as discussed next:
  - RQ 1.1: Which KAs of SE are perceived as the most important by practitioners educated in Turkey?
  - RQ 1.2: How does the perception of importance of KAs differ between experienced practitioners and recent graduates?
  - RQ 1.3: How do the importance of SE topics reported by participants in the survey compare with importance reported by other studies in the literature (as synthesized by the SLR in [14])?
- RQ 2: How does the knowledge obtained in Turkish universities compare to the perceived importance of SE KAs (what software engineers use) at work?
- RQ 3: What SE knowledge gaps are perceived the most by practitioners (the gap between reported importance of SE topics on the job and reported levels of learning in university education)? The third RQ aims to explore the knowledge gaps, i.e., areas in which software engineers have needs for more learning and training and is stated as three sub-RQs as below:
  - RQ 3.1: What are the knowledge gaps of software engineers educated in Turkey?
  - RQ 3.2: Is there a difference between the knowledge gaps of less and more experienced practitioners?
  - RQ 3.3: How do the knowledge-gap data that we have collected compare with the knowledge-gap data reported by other studies in the literature (as synthesized by the SLR in [14])?
- RQ 4: How do the data for graduates with SE degrees compare with the data for graduates with non-SE degrees?

### 3.2 An Overview of Our Research Approach

The work that we report in this article is part of a larger research program that we have developed over the past few years, as shown in Figure 1, to analyze and improve SE courses and programs.

We, as educators and practitioners, have felt the need for assessment and improvement of SE curriculum, according to our active discussions with recent graduates who work in industry and also with managers who have hired our university graduates and are able to comment on their skill sets. This "hybrid" approach analyzes SE programs based on two important inputs: (1) SE body-of-knowledge frameworks (the two well-known ones being the ACM SE 2014 [26] and SWEBOK [24]) and (2) opinions about graduates' skills and industrial needs. We conducted and reported two studies in the former direction (1), based on SWEBOK in two recent papers [27, 28]. In one study



RIGHTSLINK

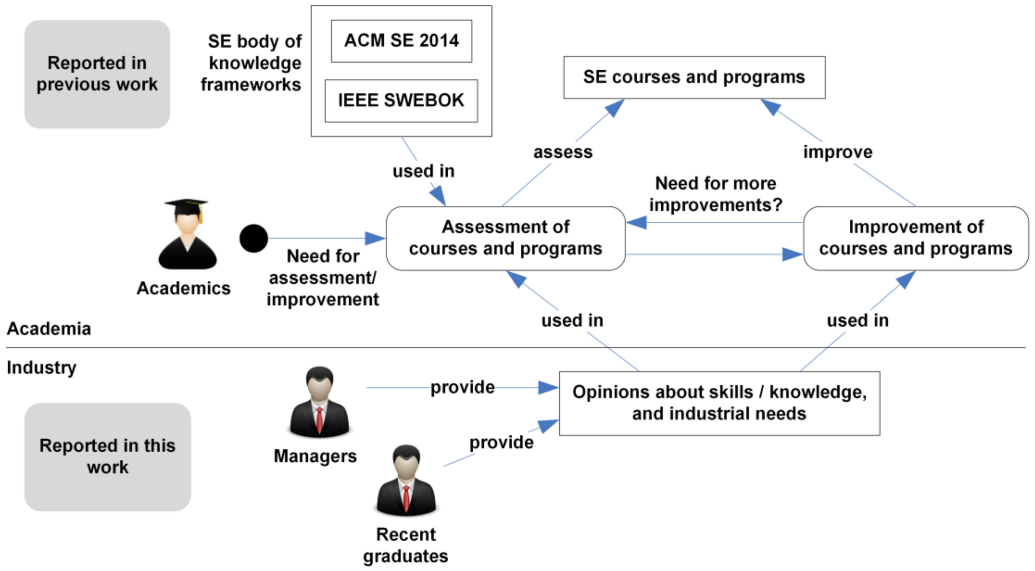



Fig. 1. Our approach to assess and improve SE courses and programs.

Table 2. Structure of the Questionnaire Used for the Survey

| Part | RQs Addressed | Number of Questions |
|---|---|---|
| Part 1: Demographics (profile) of the respondents<br>• Four questions about educational background: highest academic degree (BSc, MSc, etc.), BSc-degree graduation year, last-degree graduation year, degree areas (SE, CS, etc.)<br>• One question about city/country<br>• One question about current position<br>• One question about application domain of the company of employment | Last-graduation year will be used in RQ 1.2 and RQ 3.2<br>"Degree areas" will be used in RQ 4 | 7 |
| Part 2: Importance of SE topics in workplace, and level of SE knowledge obtained in university education | RQ 1–RQ 4 | 1 (large question) |
| Part 3: Comments | None | 1 |

[27], we analyzed and improved the SE undergraduate curriculum of a given SE degree program. One of the actionable recommendations for the SE programs in that study was to include more course materials about SE economics, e.g., cost-benefit and ROI analysis of various SE activities. In one of our earlier studies [28], we assessed all of the 13 SE programs offered in Turkey with respect to SWEBOK and provided improvement recommendations to educators and department heads in those programs.

The work that we report in this article corresponds to the bottom portion of the approach shown in Figure 1, in which we survey the opinions of practitioners.

### 3.3 Survey Design

To survey the opinions of practitioners, we designed an online survey, containing nine questions. We show in Table 2 the structure of the questionnaire used for the survey. The questionnaire



RIGHTSLINK

Table 3. Five-Point Likert Scales Used to Assess University-Level Learning and Importance of SE Topics at Work

| The Likert Scale to Assess University-Level Learning | The Likert Scale to Assess Importance (Usefulness) of Each SE Topic at Work |
|---|---|
| (0)-Learned nothing at all | (0)-Completely useless |
| (1)-Learned the basics | (1)-Occasionally useful |
| (2)-Moderate working knowledge | (2)-Moderately useful |
| (3)-Learned a lot | (3)-Very useful |
| (4)-Learned in depth (became expert) | (4)-Critical to my work |

started with a set of demographic questions, including degree(s) obtained, year of graduation, roles undertaken, and so forth. The survey then asked the participants how much they learned in their university education about each of the 12 KAs and 67 sub-KAs in SWEBOK and also about the level of importance of each topic at work. A summary of the 12 KAs and 67 sub-KAs of the SWEBOK is shown in Figure 2.

For the extent of university-level learning on each topic, we provided a 5-point Likert scale to participants, as shown in Table 3. For the importance (usefulness) of each topic at work, we also used a 5-point Likert scale, as shown in Table 3. Figure 3 shows a screenshot of the online survey, in which eight of the sub-KAs under the software requirements KA are shown. For transparency and replicability of our study, the survey questions (as presented to the respondents) and the raw data are available online in [29] and [30], respectively.

To ensure that participants had a necessary understanding of the scope of SWEBOK KAs and sub-KAs, we provided a summary of SWEBOK (as shown in Figure 4). The participants were able to access this summary text by clicking a link provided in question 9 ("click here" link shown in Figure 3).

After we prepared an initial version of the survey, as the pilot study, we asked five software engineers to fill out our survey. The objective of this pilot study was to check the clarity of the questions. Based on the feedback we received from the pilot study, we performed improvements in our survey.

### 3.4 Survey Execution

We shared the survey link via our social media accounts (Twitter and LinkedIn) to invite software professionals to fill out the survey. To increase the response rate, we also sent personal emails including the survey link to our industry network. This type of sampling is named convenience sampling, which is the dominant survey and experimental approach in SE [31]. The main drawback of convenience sampling is that it may result in a homogenous sampling frame and this sample may have limited generalizability to the broader population [32]. Therefore, we employed snowball sampling [33] to obtain a more heterogeneous sample. In the scope of snowball sampling, the participants in our industry network recruited other participants. We think that our industry network and snowball sampling enabled us to have a sample of participants graduated from various universities.

We targeted software engineers who participate in the development, operation, and maintenance of software professionally. We stated this objective at the beginning of our survey. In addition, we emphasized this objective in our email communications and social media posts. Since the authors have a network of software professionals both in industry and in academia, the respondents are assumed to be eligible to fill out our survey. The same is valid for the respondents who



RIGHTSLINK

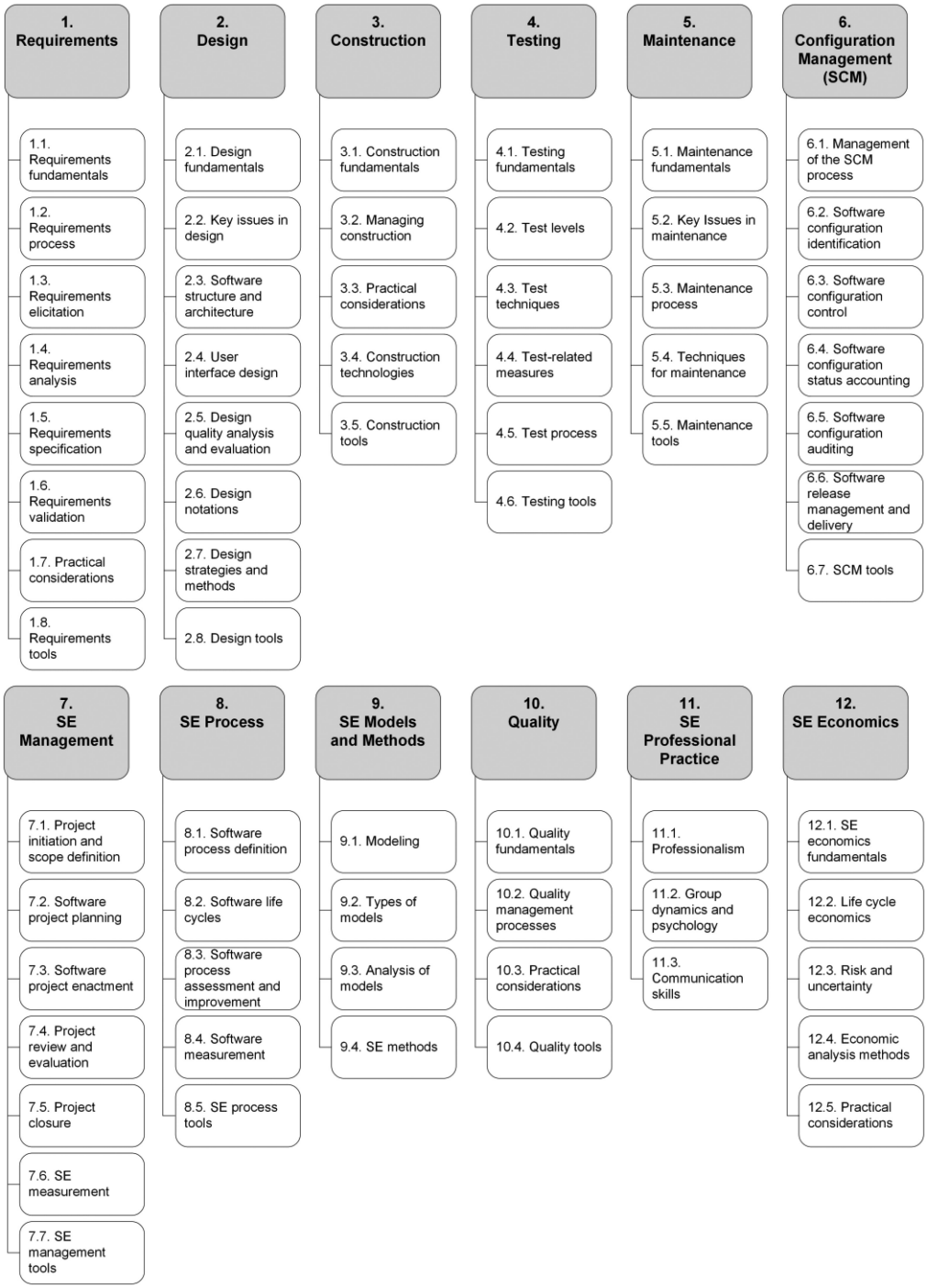



Fig. 2. Twelve KAs of SWEBOK and their sub-KAs (adopted from [24]).



RIGHTSLINK

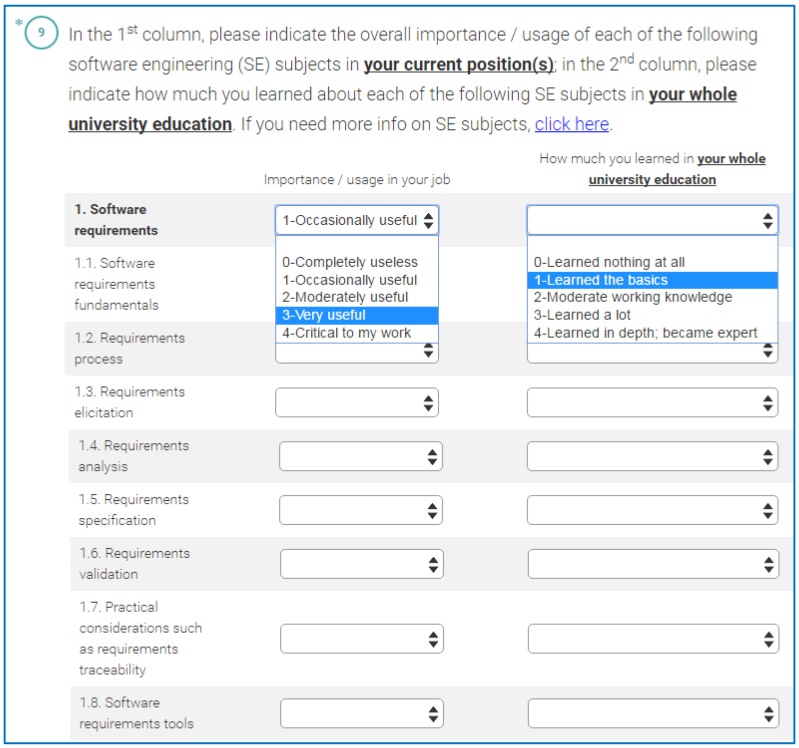



Fig. 3. Screenshot from a part of the online survey.

were reached via the network of the authors. Last but not least, the survey covers the SE topics in detail, so a person who does not have any experience in SE cannot fill out the survey properly. We assume that if a noneligible respondent has accessed our survey, he or she should have cancelled filling out the survey after he or she saw the content.

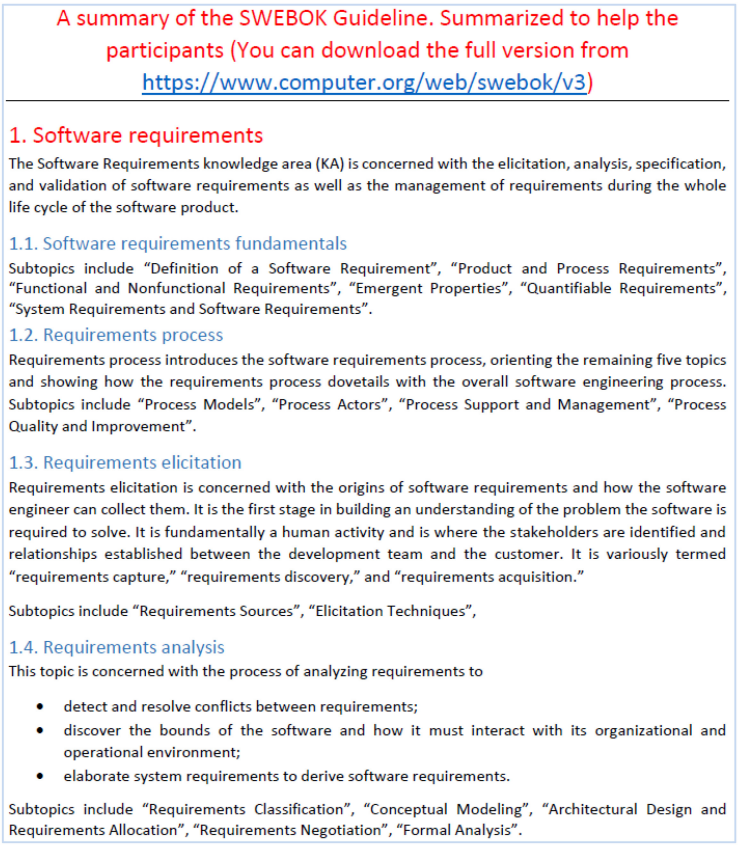
We used an online survey hosted on www.surveymonkey.com to receive responses from the participants at their convenience and hence increase the number of data points. The survey was open for responses between November 2016 and February 2017. We received a total of 132 (raw) data points, before preprocessing. When we reviewed that dataset, we found that the total number of valid responses was 129.

### 3.5 Data Analysis Method

As discussed above, we gathered ordinal data for the importance of SE topics and level of SE knowledge obtained in university education, using the Likert scale. These ordinal data can be treated as either nominal or interval data [34, 35]. An example of nominal data can be the degree obtained by students (such as BSc, MSc, PhD). Some examples of interval include experience of an engineer in years or lines of code written in a specific period. When ordinal data are treated as interval data, mean and standard deviation are appropriate. This way of treatment assumes that the difference between "(1)-Occasionally useful" and "(2)-Moderately useful" is equal to the difference between "(2)-Moderately useful" and "(3)-Very useful." On the other hand, when ordinal data are treated as nominal data [36], the difference between "(1)-Occasionally useful" and "(2)-Moderately useful" is not assumed to be equal to the difference between "(2)-Moderately useful" and "(3)-Very



RIGHTSLINK



A summary of the SWEBOK Guideline. Summarized to help the participants (You can download the full version from https://www.computer.org/web/swebok/v3)

1. Software requirements

The Software Requirements knowledge area (KA) is concerned with the elicitation, analysis, specification, and validation of software requirements as well as the management of requirements during the whole life cycle of the software product.

1.1. Software requirements fundamentals

Subtopics include "Definition of a Software Requirement", "Product and Process Requirements", "Functional and Nonfunctional Requirements", "Emergent Properties", "Quantifiable Requirements", "System Requirements and Software Requirements".

1.2. Requirements process

Requirements process introduces the software requirements process, orienting the remaining five topics and showing how the requirements process dovetails with the overall software engineering process. Subtopics include "Process Models", "Process Actors", "Process Support and Management", "Process Quality and Improvement".

1.3. Requirements elicitation

Requirements elicitation is concerned with the origins of software requirements and how the software engineer can collect them. It is the first stage in building an understanding of the problem the software is required to solve. It is fundamentally a human activity and is where the stakeholders are identified and relationships established between the development team and the customer. It is variously termed "requirements capture," "requirements discovery," and "requirements acquisition."

Subtopics include "Requirements Sources", "Elicitation Techniques",

1.4. Requirements analysis

This topic is concerned with the process of analyzing requirements to

- detect and resolve conflicts between requirements;
- discover the bounds of the software and how it must interact with its organizational and operational environment;
- elaborate system requirements to derive software requirements.

Subtopics include "Requirements Classification", "Conceptual Modeling", "Architectural Design and Requirements Allocation", "Requirements Negotiation", "Formal Analysis".

Fig. 4. Screenshot from a part of SWEBOK summary that was provided to the participants.

useful." In this case, it becomes possible to rank these values but not use the mean and standard deviation of them.

There is an ongoing debate on treating ordinal data as interval data [35, 36]. Some researchers claim that this is permissible with certain caveats [34], e.g., assuming that the ordinal data are normally distributed. [37] and [38] both indicate that under equinormality, power for both the parametric and nonparametric is close, and may actually be greater for the nonparametric under violations of the normality assumption. [35] states that parametric statistics can be used with Likert data, with small sample sizes, with unequal variances, and with nonnormal distributions.

Lethbridge [15, 16] measured the importance and degree of learning using a Likert scale and analyzed the results (ordinal data) using techniques applicable for interval data. Lethbridge calculated the mean and standard deviation of ordinal data and formulated a knowledge gap as the difference between the importance of a topic and the degree of knowledge in that topic.

On the other hand, another pioneering study for exploring knowledge gaps of software engineers, Kitchenham et al. [17], refuses Lethbridge's treatment of ordinal data. They preferred to use only proportions, which are also present in Lethbridge's study, to assess importance and learning level of topics in SE. For instance, the importance of a topic is calculated with the formula: Importance = number of subjects scoring three (3) or more/number of subjects. A 6-point scale, from 1 to 6, was used to measure importance and learning level. The knowledge gap was defined as the difference between importance and learning level, both of which are expressed in percentages.



RIGHTSLINK

Table 4. Academic Degrees of Participants (n = 129)

| | | # of Participants | % of Participants |
|---|---|---|---|
| Degree | BSc | 83 | 64.3 |
| | MSc | 39 | 30.2 |
| | PhD | 7 | 5.4 |
| SE degree | Yes | 27 | 20.9 |
| | No | 102 | 79.1 |

In this study, we used the approaches presented by Lethbridge [15, 16] and Kitchenham et al. [17] as well as paired-samples t-test. We measured knowledge gap by accepting Lethbridge's assumption for treating ordinal data as interval data and enriched it by applying paired-samples t-test. Paired-samples t-test is suitable for measuring the difference between the participants' perception of importance and learning level of each KA. In our case, we formed the following hypotheses to assess the knowledge gap for each KA and answer RQ 3.1:

- $H_0$: The mean of importance for the participants equals the mean of learning level of the participants (there is no statistically significant difference between importance and learning level, and hence there is no knowledge gap).
- $H_1$: The mean of importance for the participants does not equal the mean of learning level of the participants (there is a statistically significant difference between importance and learning level, and hence there is a knowledge gap).

After we calculated the t value, we then compared it to the critical t value with a degree of freedom (df) of 128 (df = n – 1; hence 129 – 1 = 128 in our case) from the t distribution table for a chosen confidence level (0.05 in our case). If the calculated t value is greater than the critical t value, then we reject the hypothesis $H_0$ (and conclude that the means are significantly different). By doing so, we used Lethbridge's analysis method and extended it using paired-samples t-test to test the significance of differences.

We adapted the formula of Kitchenham et al. [17] to our Likert scales (see Table 3) as below and used it to answer RQ 3.1:

$$\text{Importance/Learning Level} = \frac{\text{number of subjects scoring more than zero (0)}}{\text{total number of subjects}}.$$

## 4 RESULTS

We first present the demographics of our dataset. We then answer each of the RQs defined in Section 3.1.

### 4.1 Demographics

Table 4 provides a summary of the demographic data. As their highest degrees, 83 of the respondents reported they had BSc degrees, while 39 had MSc degrees and 7 had PhD degrees; 20.9% of the participants (n = 27) held an SE degree from a university. The rest held degrees mostly from CS and CE departments. A minority had degrees from noncomputing disciplines, including electrical and electronics engineering, mathematics, and statistics.

As displayed in Figure 5, the work experience of the respondents indicates that 51.2% of them had 5 or fewer years of experience; 44.2% had work experience of 6 to 20 years.

As Figure 6 displays, the large majority of the participants were undertaking more than one role concurrently: 87 of them were working as software developers/programmers, 24 as project



RIGHTSLINK

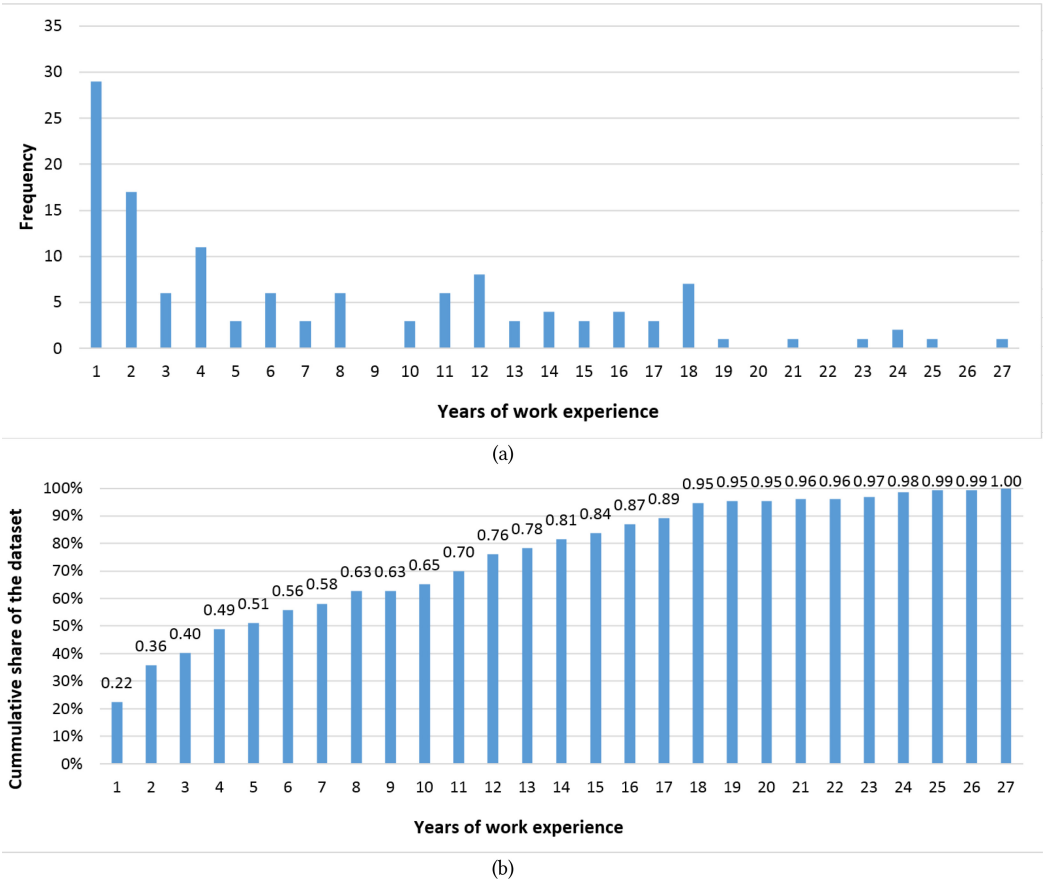



Fig. 5. Histogram of work experience of the participants: (a) Frequency; (b) cumulative share.

managers, 15 as software architects, 13 as business analysts, 10 as consultants, and 7 as software testers. The other respondents reported being in other roles (e.g., higher-level management, requirements engineers, etc.).

In terms of work locations of respondents, 118 were based in different cities of Turkey and 11 reported their location to be in the United States. Although we do not have the data to verify, we think that that subset of respondents is (1) of Turkish origin and have done their studies in Turkey and then have moved to the United States or (2) of non-Turkish origin working in the United States.

### 4.2 RQ 1: Importance of SE Topics in the Workplace

For RQ 1.1, we present the importance data of SE topics from our study. For RQ 1.2, we compare how less and more experienced software engineers perceive the importance of SE topics. For RQ 1.3, we compare the importance of SE topics from our dataset with empirical data from the literature, as synthesized in the recent SLR study [14].

*4.2.1 RQ 1.1: Importance Data of SE Topics from Our Study.* Figure 7 shows the importance ratings of the participants for each KA as a violin plot [39]. The thicker part means the values in that section of the violin have higher frequency, and the thinner part implies lower frequency. The white dot in each violin represents the median for the corresponding KA. Construction and design



RIGHTSLINK

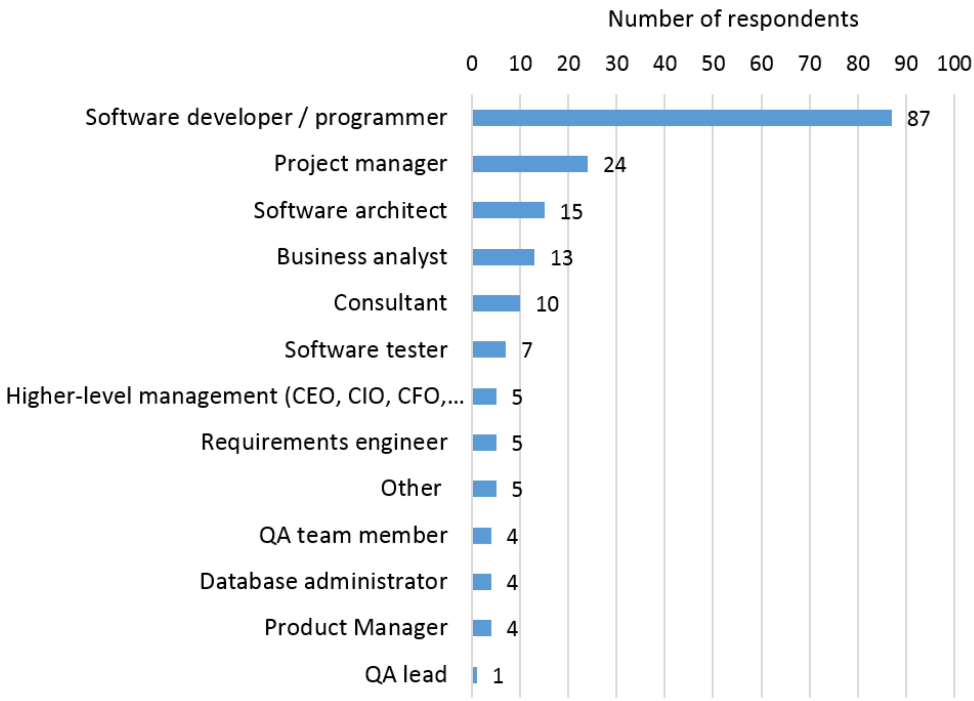



Fig. 6. Roles undertaken by the participants (selected all that applied, n = 129).

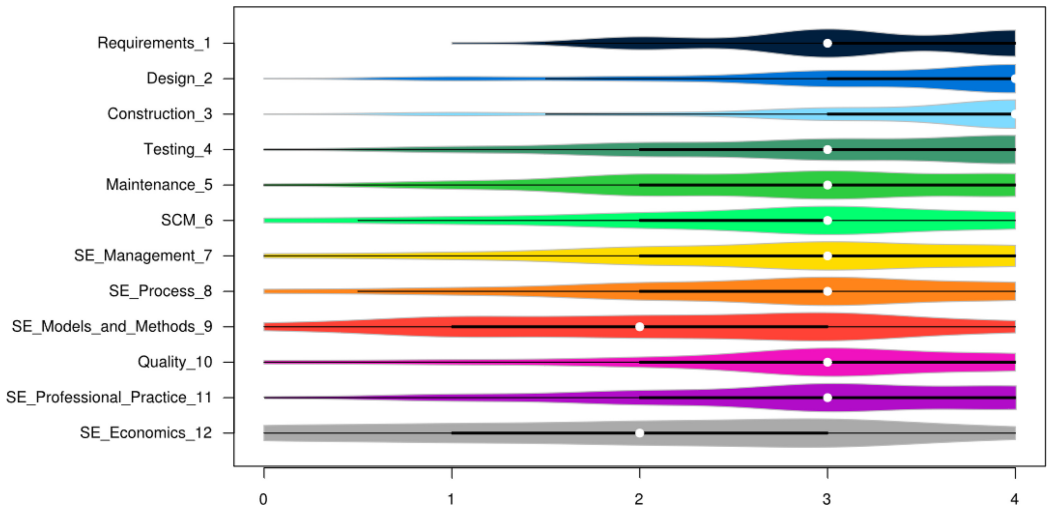



Fig. 7. Importance data of SE KAs, visualized as a violin plot (x-axis: responses to the Likert scale from 0 = "completely useless" to 4 = "critical to my work"; y-axis: SWEBOK SE KAs).

are the two top KAs with the highest importance ratings. Moreover, the data distribution for these two KAs is mostly above "(3)-Very useful" and the median is four. The least important KAs have been SE economics and SE models and methods with a median of two. The data distribution of these two KAs is scattered (the values are not highly concentrated around the median), which



RIGHTSLINK

Table 5. The KAs Whose Importance Are Perceived Differently by Less and More Experienced Practitioners

| Rank | KA | Mean Difference | Std. Deviation | t |
|---|---|---|---|---|
| 1 | SE Economics | 0.51 | 0.23 | 2.21 |
| 2 | SE Management | 0.41 | 0.20 | 1.99 |
| 3 | SE Professional Practice | 0.40 | 0.23 | 2.21 |

means that the perceived importance of these KAs varies too much. The other eight KAs have been generally rated as "(3)-Very useful." As shown in Figure 7, the data distribution for these KAs is less scattered compared to SE economics and SE models and methods.

*4.2.2 RQ 1.2: Comparing the SE Topic Importance Perceptions of Less and More Experienced Practitioners.* The level of experience might affect which KAs are being perceived as important. As software engineers gain more experience, they may assume more and diverse responsibilities. With these responsibilities, they can experience more aspects of SE. To understand the impact of experience on perceptions of KA importance, we divided our dataset into two groups in the following ways: (1) less experienced software engineers having 1 to 10 years of working experience and (2) more experienced software engineers having more than 10 years of working experience. Then we conducted independent samples t-test for comparing the importance of KAs perceived by less (n = 84) and more experienced (n = 45) practitioners. Independent samples t-test is an inferential statistical test that determines whether there is a statistically significant difference between the means in two separate groups.

There are three KAs for which the importance levels are significantly different between less and more experienced practitioners. As shown in Table 5, more experienced practitioners consider SE economics, SE management, and SE professional practice as more important compared to less experienced ones. The reason for this can be that software engineers deal with economical and managerial aspects of SE more as they gain more experience and take over more responsibilities. In addition, software engineers, like most of the other professions, should use their soft skills more as they gain more managerial responsibilities [40, 41]. The SE professional practice KA includes knowledge on group dynamics, psychology, and communication skills, which are very important for software engineers with managerial responsibilities.

*4.2.3 RQ 1.3: Comparing the Importance Data from Our Dataset with Empirical Data from the Literature.* In our earlier work [14], we conducted a meta-analysis of 33 studies reporting empirical findings (mostly based on survey data) on the alignment of SE education with industry needs. In that meta-analysis study [14], we mapped the importance data of KAs reported in those studies to SWEBOK KAs. By doing so, we were able to unify data from 12 countries and over 4,000 data points. To be able to unify data from various studies, we normalized the data of importance in each paper to the range of [0, 1].

It would be interesting and insightful to compare data from our current study with the unified data reported in [14]. Such a comparison provides insights on the KA importance data as gathered from our survey participants versus 4,000+ data points in [14]. Also, given the Turkish focus of our dataset, it could provide hints on any possible similarities or differences in how the importance of SE KAs is assessed in different geographical locations.

To be able to compare data from our dataset with the unified data reported in [14], we normalized the importance ratings of this study. We divided the ratings by four (since our Likert scale is from zero to four) and obtained the normalized ratings between zero and one. Figure 8 shows the



RIGHTSLINK

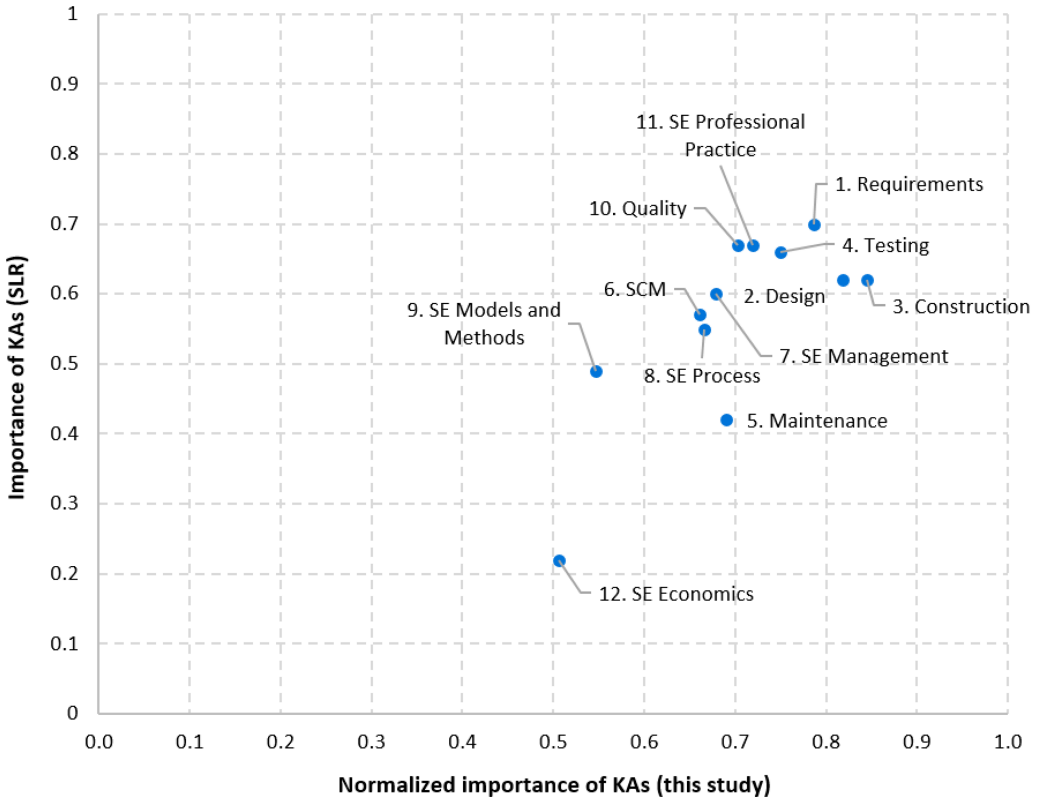



Fig. 8. Comparison of our importance data with the empirical data from the literature (based on the data in [14]).

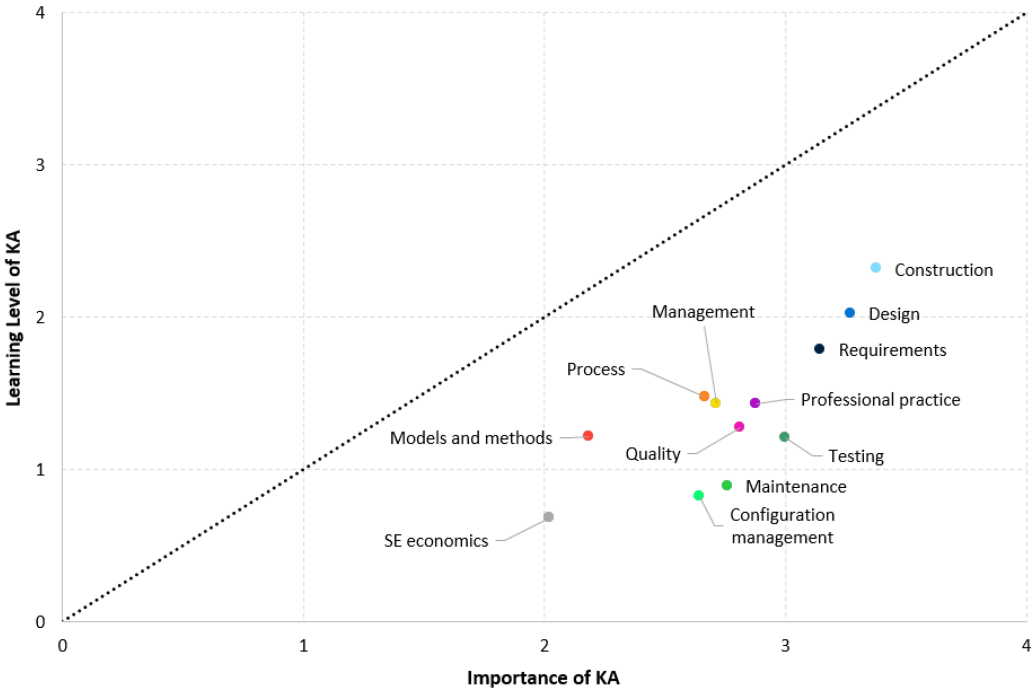
importance ratings from the data obtained in this study along the x-axis and the ratings reported in the literature (data from [14]) along the y-axis. When we compare the normalized importance ratings of this study with the unified, normalized importance ratings present in the literature (reported by the SLR in [14]), it is interesting that we can see a high correlation between these two. The Pearson correlation coefficient of the two data series is 0.76 and statistically significant, meaning that the findings obtained in this study are consistent with the findings obtained in the SLR study [14]. This denotes that there are similarities in how the importance of SE KAs is assessed in our population of Turkish software engineers versus the aggregated dataset if 4,000+ data points in [14], which included survey data from software engineers in many different countries (e.g., United States and Canada).

### 4.3 RQ 2: Learning Topics in University versus Their Importance at Work

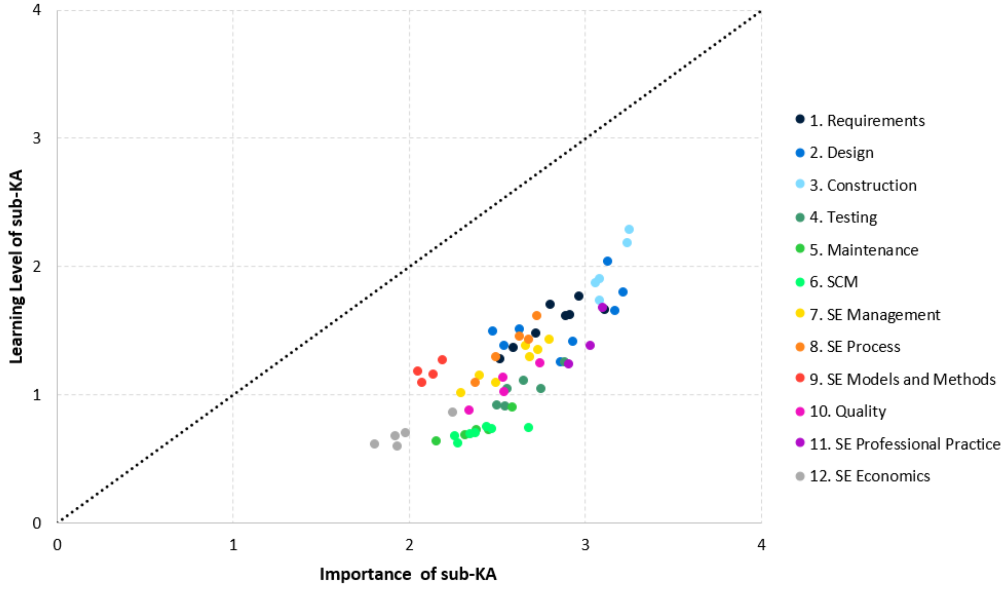
We looked at the provided data (importance and learning levels) for each individual SWEBOK KA and its sub-KAs. To get a broad picture on the opinions and for ease of presentation, our first analysis was to average the values of all the 129 data points to get a single metric value for each KA and sub-KA. Based on the above approach, the two scatter plots in Figure 9 depict the importance/usage of the topics at work versus how much they were learned in university education. The plots show both the coarse-grained (one value for each KA in Figure 9(a)) and fine-grained values (per sub-KA in Figure 9(b)). For example, the eight sub-KAs of the requirements



RIGHTSLINK




(a)




(b)

Fig. 9. Perception of respondents about importance of KA at work versus how much they learned them in university, as scatter plots: (a) in KA level; (b) in sub-KA level.



RIGHTSLINK

KA are shown as individual points in Figure 9(b). We discuss next the most notable observations from Figure 9.

According to the KA-level data in Figure 9(a), we observe that, among the 12 KAs of SWEBOK, construction, design, and then requirements skills, in order, are the most important/used at work, and also the most taught in university settings. The Pearson correlation coefficient of the two data series is 0.78. This measure denotes that, in general, the higher the importance/usage ratio of a KA, the higher the perception of its learning in university; i.e., if a respondent considers a KA more important, he or she thinks he or she has learned it quite well, and vice versa. This can be seen as a general positive comment for university programs that these students had studied in.

On the other hand, the participants, in general, have learned less about topics such as SE economics and software configuration management (SCM) in their university education (the two KAs with the least learning level in Figure 9(a)), although the importance of these KAs has been rated as "moderately" or "very" useful at work by many participants. Such observations have implications for university educators to include more materials from those KAs in university courses. We had also observed such a situation and had made necessary recommendations in our earlier studies [27, 28].

Furthermore, we see in Figure 9(b) that the "cluster" of sub-KAs for each KA is not that spread apart and is relatively close to each other. This denotes that the participants rated the importance and learning levels of sub-KAs of a single KA quite similarly. From another viewpoint, among the SWEBOK KAs, we do not see any topic in general that has been taught too much in university but not used that often in practice. All the points in both plots of Figure 9 are under the 45-degree line, denoting that importance/usage ratios of all topics in general are higher than their degrees of learning in university. Also, the average perceptions of learning in most cases are below two out of four, which denotes the shortage of university learning in almost all topics. However, this is as expected and not a negative issue per se, since, given the large spectrum of topics and concepts in SE, covering all of them in detail is quite challenging (if not impossible) in university programs. As a result, most graduates usually learn the basics of many topics in university and improve their skills on their own in the workplace.

### 4.4 RQ 3: Knowledge Gaps: Areas That Need More Training

For RQ 3.1, we present the knowledge data from our study. For RQ 3.2, we compare how less and more experienced software engineers perceive their knowledge gaps on SE topics. For RQ 3.3, we compare the knowledge gap data from our dataset with empirical data from the literature, as synthesized in the recent SLR study [14].

*4.4.1 RQ 3.1: Knowledge Gap Data from Our Study.* As the next step, we wanted to quantitatively assess the knowledge gap for each KA to determine which areas need more training in university programs. First, we followed Lethbridge's approach [15, 16] and enriched it using paired-samples t-test. Table 6 shows the KAs along with the corresponding average knowledge gap. Our paired-samples t-test showed that there is a statistically significant difference between the importance of and learning level in each KA (Sig. (two-tailed) $< 0.05$).

Second, we used Kitchenham's approach [17] to identify knowledge gaps quantitatively. Table 7 shows the KAs ordered by knowledge gaps.

We used Spearman's rank-order correlation to measure the strength and direction of association between two rankings. The Spearman correlation coefficient between the rankings shown in Table 6 and Table 7 is 0.678 and there is a statistically significant association between these two rankings ($p = 0.015 < 0.05$). In line with statistical evidence, we can clearly see that software



RIGHTSLINK

Table 6. SWEBOK KAs Sorted by Mean of Knowledge Gap Based on Paired Samples t-Test

| Rank | KA | Mean of Knowledge Gap | Std. Deviation | t |
|---|---|---|---|---|
| 1 | Maintenance | 1.87 | 1.20 | 17.66 |
| 2 | Configuration Management (SCM) | 1.82 | 1.25 | 16.52 |
| 3 | Testing | 1.79 | 1.37 | 14.81 |
| 4 | Quality | 1.53 | 1.32 | 13.12 |
| 5 | SE Professional Practice | 1.44 | 1.42 | 11.49 |
| 6 | Requirements | 1.36 | 1.27 | 12.16 |
| 7 | SE Economics | 1.34 | 1.39 | 10.92 |
| 8 | SE Management | 1.28 | 1.37 | 10.61 |
| 9 | Design | 1.25 | 1.24 | 11.40 |
| 10 | SE Process | 1.19 | 1.34 | 10.11 |
| 11 | Construction | 1.06 | 1.18 | 10.24 |
| 12 | SE Models and Methods | 0.97 | 1.37 | 8.01 |

Table 7. SWEBOK KAs Sorted by Percentage (%) of Knowledge Gap Based on Approach by Kitchenham et al. [17]

| Rank | KA | Knowledge Gap (%) |
|---|---|---|
| 1 | SE Economics | 39% |
| 2 | Maintenance | 39% |
| 3 | Configuration Management (SCM) | 36% |
| 4 | Testing | 25% |
| 5 | Quality | 22% |
| 6 | SE Professional Practice | 22% |
| 7 | SE Models and Methods | 18% |
| 8 | SE Process | 16% |
| 9 | SE Management | 12% |
| 10 | Requirements | 9% |
| 11 | Design | 5% |
| 12 | Construction | 3% |

maintenance, SCM, and software testing all rank in the top five of both analyses and could be considered the three largest knowledge gaps as a result.

The most observable difference in the results we obtained is that while "SE economics" is ranked first as the KA with the largest knowledge gap in Table 7, its rank is seventh in Table 6. To understand the reason behind this, we plotted the numbers used to calculate the knowledge gap based on Lethbridge's [15, 16] and Kitchenham's approach [17] as seen in Figure 10(a) and Figure 10(b), respectively. In both figures, we calculated the knowledge gap as the difference between importance and learning level (means in Figure 10(a) and percentages in Figure 10(b)) and sorted the KAs in descending order according to knowledge gap.

Kitchenham's approach [17] involves grouping the scores of importance into binary groups, namely "% of participants scored greater than zero" and "% of participants scored zero" according to the Likert scale described in Table 3. Similarly, the scores of learning level are also consolidated into two groups, namely "% of participants scored greater than zero" and "% of participants scored



RIGHTSLINK

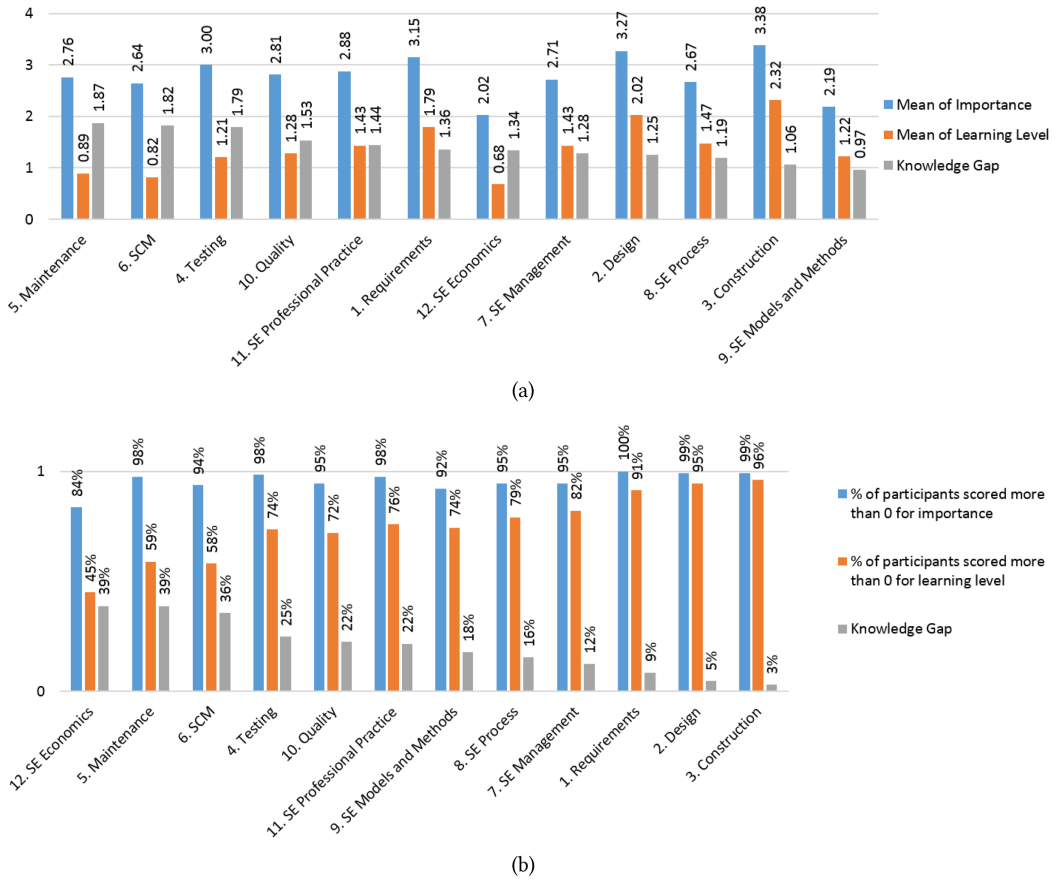



Fig. 10. Importance, learning level, and knowledge gap calculated per KA based on (a) Lethbridge's approach [15, 16] and (b) Kitchenham's approach [17].

zero." This approach does not discriminate if the respondent had an answer of (1) or (4) according to the Likert scale shown in Table 3.

Unlike Kitchenham's approach [17], Lethbridge's [15, 16] analysis calls for calculating the mean of the scores and the scores in each group. The importance of SE economics is rated as 2.02 out of 4 in Figure 10(a), whereas it is rated as 84% in Figure 10(b). This means that even though 84% of the participants consider SE economics as occasionally or more important, its average is relatively low according to Lethbridge's analysis [15, 16]. Moreover, only 45% of participants reported an at least basic level of learning for the SE economics KA, while the learning level is 0.68 out of 4. The knowledge gap increases with a higher importance rate and lower rate of learning level (refer to Section 3.5 for formulas). Since we obtained a relatively high importance rate (84%) based on Kitchenham's analysis [17] compared to a relatively low mean of importance (2.02 out of 4) based on Lethbridge's approach [15, 16] and a relatively low learning level (45%) based on Kitchenham's analysis [17], the knowledge gap ranking of the SE economics KA varied between these two analysis approaches. In this article, we focused on the top three common KAs with the highest knowledge gaps according to these two approaches.

The violin plot [39] in Figure 11 shows the data distribution and median for each KA. We can clearly see that the gap measure for all the KAs is negative, thus denoting that, for all the KAs, the



RIGHTSLINK

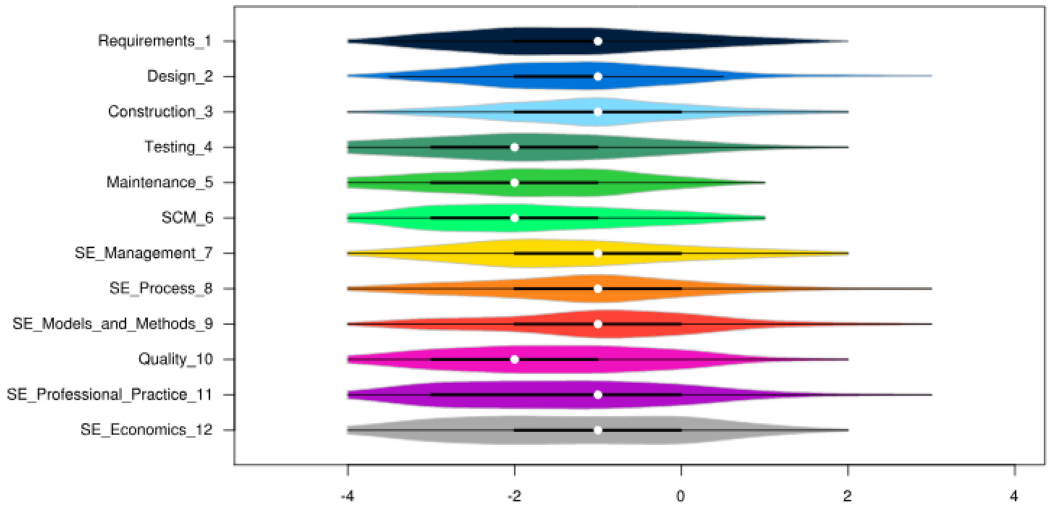



Fig. 11. Knowledge gaps of different KAs as a violin plot (x-axis: knowledge gap; y-axis: SWEBOK SE KAs).

general perception of respondents is that their university education level for each and every KA is "less" than the extent to which it is used in workplace. Moreover, the data distribution also shows that most of the participants agree on these knowledge gaps since data are highly concentrated on the left-hand side of Figure 11, below the zero value in the x-axis. Quality, SCM, maintenance, and testing are the KAs with the lowest medians, and hence the KAs with the highest knowledge gaps.

In our discussions with several software engineers, we determined the root cause of such a perception to be the large spectrum and variability of SE technologies. For example, if a developer is fluent in C++ and Java based on solid university education in those languages, when he or she is given the task to maintain a Python code base, he or she has to learn new things about programming in Python. In general, we see that university education in SE cannot possibly cover all the SE topics in great depth, but instead should cover the fundamentals and instill the habit of self-learning in new engineers. Many practitioners also believe this, as discussed in various blogs and conferences, e.g., "Software engineers must continuously learn and integrate" [42]. We also observe from the data that, for software engineers, university education is important but not usually enough, as also mentioned by other sources [43].

To gain insight on how SE education can cover the fundamentals more, we furthermore looked into the sub-KAs of each SWEBOK KA and calculated the quantitative knowledge gap level for each of the 67 sub-KAs. Table 8 displays the results of paired-samples t-test for the top 20 sub-KAs with the largest knowledge gaps.

We can see that 16 sub-KAs out of 20 are under SCM (seven sub-KAs), Software testing (five sub-KAs), and Software maintenance (four sub-KAs). Two sub-KAs ranked as 8th and 12th are from SE professional practice. One sub-KA from design is ranked as 16th and another sub-KA is ranked as 20th from the quality KA.

Table 9 shows the analysis done for sub-KAs using Kitchenham's approach [17]. The first 10 sub-KAs in Table 9 are listed under two KAs, i.e., SCM (6 sub-KAs) and maintenance (4 sub-KAs). SE economics (four sub-KAs), testing (one sub-KA), quality (one sub-KA), and professional practice (two sub-KAs) are the additional KAs, some of whose sub-KAs are listed in the top 20.

The Spearman correlation coefficient between the rankings shown in Table 8 and Table 9 is 0.702 and statistically significant ($p = 0.000 < 0.05$). Therefore, we can conclude the rankings obtained



RIGHTSLINK

Table 8. Top 20 SWEBOK Sub-KAs with the Largest Knowledge Gap Sorted by Mean of Knowledge Gap Based on Paired Samples t-Test

| Rank | Sub-KA | Mean of Knowledge Gap | Std. Deviation | t | Ranking in Table 9 |
|---|---|---|---|---|---|
| 1 | 6.6. Software release management and delivery | 1.95 | 1.37 | 16.11 | 6 |
| 2 | 6.1. Management of the SCM process | 1.74 | 1.22 | 16.24 | 12 |
| 3 | 5.2. Key issues in software maintenance | 1.74 | 1.21 | 16.32 | 2 |
| 4 | 4.5. Test process | 1.71 | 1.30 | 14.88 | - |
| 5 | 6.3. Software configuration control | 1.70 | 1.31 | 14.67 | 9 |
| 6 | 5.1. Software maintenance fundamentals | 1.69 | 1.33 | 14.39 | 14 |
| 7 | 6.2. Software configuration identification | 1.68 | 1.24 | 15.44 | 5 |
| 8 | 11.1. Professionalism | 1.67 | 1.48 | 12.85 | 19 |
| 9 | 5.3. Maintenance process | 1.67 | 1.28 | 14.76 | 8 |
| 10 | 6.7. SCM tools | 1.66 | 1.29 | 14.61 | 3 |
| 11 | 6.5. Software configuration auditing | 1.66 | 1.28 | 14.68 | 1 |
| 12 | 11.2. Group dynamics and psychology | 1.65 | 1.59 | 11.76 | 20 |
| 13 | 4.4. Test-related measures | 1.64 | 1.37 | 13.59 | 13 |
| 14 | 5.4. Techniques for maintenance | 1.64 | 1.23 | 15.09 | 4 |
| 15 | 4.1. Software testing fundamentals | 1.64 | 1.32 | 14.11 | - |
| 16 | 2.5. Software design quality analysis and evaluation | 1.61 | 1.50 | 12.20 | - |
| 17 | 4.6. Software testing tools | 1.59 | 1.46 | 12.40 | - |
| 18 | 6.4. Software configuration status accounting | 1.59 | 1.30 | 13.92 | 10 |
| 19 | 4.3. Test techniques | 1.54 | 1.39 | 12.64 | - |
| 20 | 10.3. Practical considerations | 1.53 | 1.36 | 12.72 | - |

from two different analyses are statistically consistent. There are 14 sub-KAs in common in Table 8 and Table 9. For comparison purposes, we provided the rankings provided by each approach in Table 8 and Table 9 as the last column. These 14 sub-KAs are classified under four KAs, i.e., SCM, maintenance, testing, and professional practice. A noticeable difference is that Table 9 contains four sub-KAs from SE economics, while Table 8 does not have any. We believe this difference can be explained with the interpretation of knowledge gap calculation between Lethbridge's [15, 16] and Kitchenham's approach [17] as we discuss in Section 4.4.1.

*4.4.2 RQ 3.2: Comparing the Knowledge Gaps of Less and More Experienced Practitioners.* Software engineers may realize lack of knowledge in different SE topics as they gain more experience and take over more diverse responsibilities. To understand the impact of experience on knowledge gap, we divided our dataset into two groups in the following ways (as in RQ 1.2): (1) less experienced software engineers having 1 to 10 years of working experience and (2) more experienced software engineers having more than 10 years of working experience. Then we conducted paired samples t-test for comparing knowledge gaps of less (n = 84) and more experienced (n = 45) practitioners.

Table 10 displays the rank, mean of knowledge gap, standard deviation, and t value for two groups. The arrows in the second rank column show the increase and decrease in ranking compared to the first rank column. The last column shows the difference between the means of two groups for each KA. More experienced practitioners think that there is more gap between their knowledge obtained from universities and needs in industry except for construction KA (all the values in the column named "mean difference" are positive except for construction KA).



RIGHTSLINK

Table 9. SWEBOK Sub-KAs Sorted by % of Knowledge Gap Based on Approach by Kitchenham et al. [17]

| Rank | Sub-KA | Knowledge Gap (%) | Ranking in Table 8 |
|---|---|---|---|
| 1 | 6.5. Software configuration auditing | 48% | 11 |
| 2 | 5.2. Key issues in software maintenance | 45% | 3 |
| 3 | 6.7. SCM tools | 45% | 10 |
| 4 | 5.4. Techniques for maintenance | 45% | 14 |
| 5 | 6.2. Software configuration identification | 45% | 7 |
| 6 | 6.6. Software release management and delivery | 44% | 1 |
| 7 | 5.5. Software maintenance tools | 44% | - |
| 8 | 5.3. Maintenance process | 43% | 9 |
| 9 | 6.3. Software configuration control | 43% | 5 |
| 10 | 6.4. Software configuration status accounting | 43% | 18 |
| 11 | 12.5. Practical considerations | 42% | - |
| 12 | 6.1. Management of the SCM process | 41% | 2 |
| 13 | 4.4. Test-related measures | 36% | 13 |
| 14 | 5.1. Software maintenance fundamentals | 36% | 6 |
| 15 | 12.1. SE economics fundamentals | 36% | - |
| 16 | 10.4. Software quality tools | 36% | - |
| 17 | 12.4. Economic analysis methods | 36% | - |
| 18 | 12.2. Lifecycle economics | 35% | - |
| 19 | 11.1. Professionalism | 34% | 8 |
| 20 | 11.2. Group dynamics and psychology | 34% | 12 |

Table 10. Knowledge Gaps of Less and More Experienced Software Engineers Based on Paired Samples t-Test

| | Knowledge Gap of Software Engineers Having 1 to 10 Years of Working Experience (n = 84) | | | | Knowledge Gap of Software Engineers Having More Than 10 Years of Working Experience (n = 45) | | | | |
|---|---|---|---|---|---|---|---|---|---|
| KA | Rank | Mean of Knowledge Gap | Std. Deviation | t | Rank | Mean of Knowledge Gap | Std. Deviation | t | Mean Difference |
| Maintenance | 1 | 1.74 | 1.26 | 12.62 | 1 – | 2.11 | 1.05 | 13.50 | 0.37 |
| Configuration Man. (SCM) | 2 | 1.73 | 1.19 | 13.34 | 4 ↓ | 2.00 | 1.37 | 9.83 | 0.27 |
| Testing | 3 | 1.65 | 1.31 | 11.55 | 2 ↑ | 2.04 | 1.46 | 9.39 | 0.39 |
| Quality | 4 | 1.40 | 1.25 | 10.28 | 8 ↓ | 1.78 | 1.44 | 8.26 | 0.37 |
| Construction | 5 | 1.17 | 1.12 | 9.57 | 12 ↓ | 0.87 | 1.27 | 4.57 | −0.30 |
| Design | 6 | 1.14 | 1.18 | 8.85 | 10 ↓ | 1.44 | 1.34 | 7.23 | 0.30 |
| SE Professional Practice | 7 | 1.13 | 1.44 | 7.21 | 3 ↑ | 2.02 | 1.22 | 11.16 | 0.89 |
| Requirements | 8 | 1.12 | 1.31 | 7.82 | 7 ↑ | 1.80 | 1.06 | 11.42 | 0.68 |
| SE Economics | 9 | 1.07 | 1.40 | 7.00 | 6 ↑ | 1.84 | 1.24 | 9.96 | 0.77 |
| SE Process | 10 | 1.01 | 1.38 | 6.70 | 9 ↑ | 1.53 | 1.20 | 8.58 | 0.52 |
| SE Management | 11 | 0.92 | 1.33 | 6.33 | 5 ↑ | 1.96 | 1.19 | 11.06 | 1.04 |
| SE Models and Methods | 12 | 0.82 | 1.31 | 5.75 | 11 ↑ | 1.24 | 1.46 | 5.70 | 0.42 |



RIGHTSLINK

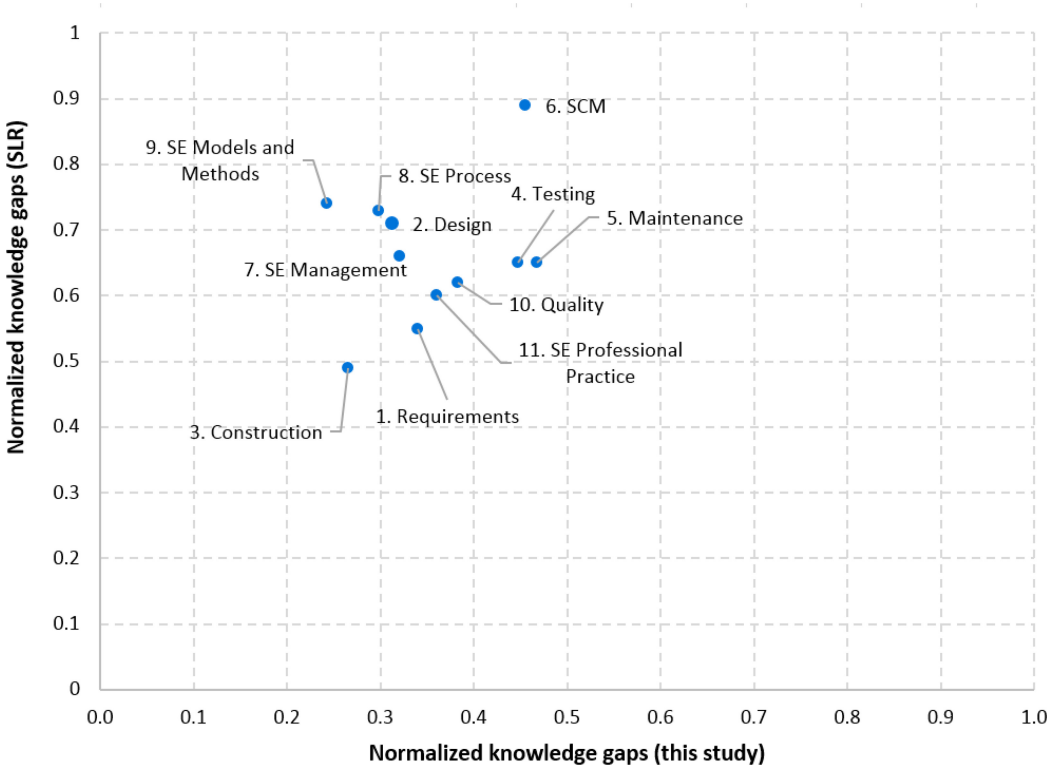



Fig. 12. Comparison of our knowledge gap data with the empirical data from the literature (using data from [14]).

Maintenance is the KA with the largest knowledge gap for both groups. The top three KAs for which knowledge gaps are the largest are SE management, SE professional practice, and SE economics, with 1.04, 0.89, and 0.77 mean differences, respectively. As we point out in RQ 1.2, the reason for this can be that software engineers deal with economical and managerial aspects of SE more as they gain more experience and take over more responsibilities. In addition, software engineers, like most of the other professions, should use their soft skills more as they gain more managerial responsibilities [40, 41]. The SE professional practice KA includes knowledge on group dynamics, psychology, and communication skills, which are very important for software engineers with managerial responsibilities.

*4.4.3 RQ 3.3: Comparing the Knowledge Gaps from Our Dataset with Empirical Data from the Literature.* To normalize the knowledge gap ratings of this study, we divided the ratings by four and obtained the normalized ratings between zero and one. Figure 12 shows the knowledge gap ratings from the data obtained in this study in the x-axis and the ratings reported in the literature in the y-axis. Note that we do not have the SE economics KA in Figure 12, since we do not have a knowledge gap data for this KA in the SLR.

To get the empirical data of knowledge gaps from the papers reported in the literature, we used the data as synthesized in a recent SLR study [14]. We should note that the knowledge-gap data were not measured in the same way in each study in the literature, as the scales and exactly what was asked for varied between the studies. However, we found a way in the SLR [14] to somewhat "harmonize" and synthesize them. For details, refer to [14].



RIGHTSLINK

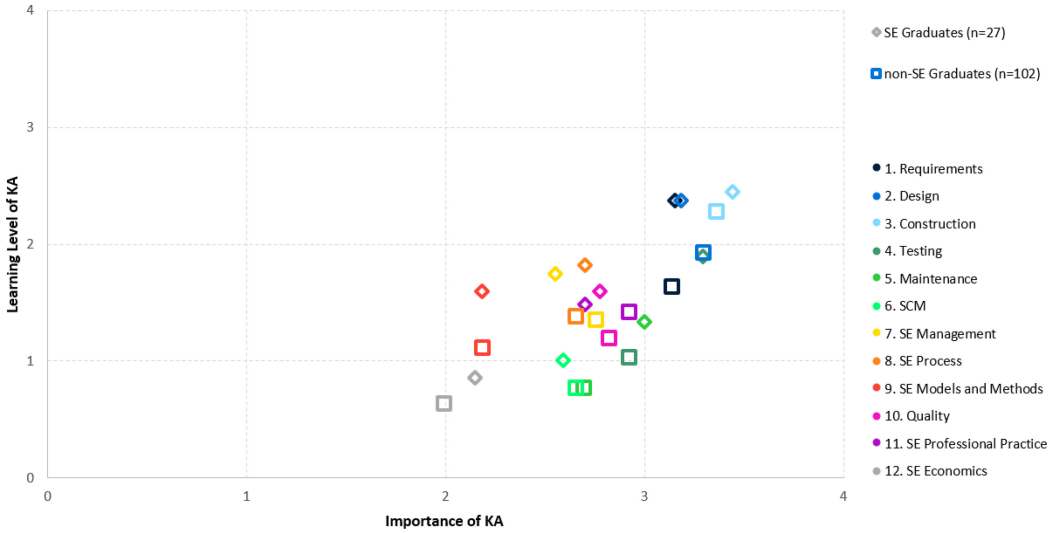



Fig. 13. Perception of respondents about importance of KAs at work versus how much they learned them in university: SE graduates versus non-SE graduates.

When we compare the knowledge gap ratings of this study with the unified, normalized knowledge gap scores present in the literature (reported by the SLR in [14]), we can see a low correlation between these two. The Pearson correlation coefficient of the two data series is 0.26. One of the reasons behind this inconsistency might be the various methods used to calculate knowledge gap in the primary studies in our SLR. As we pointed out in Section 4.4.1, different methods might yield different results, as seen for the SE economics KA in this study. Therefore, it is important to use different methods and conduct a comparative analysis on the results, as we did in this study. We have not seen such a comparative analysis in the primary studies covered in the SLR we conducted.

### 4.5 RQ 4: Comparing the Data for Graduates with SE versus Non-SE Degrees

The majority of software engineers working in the field today have either CS or CE degrees. With the increasing importance of software, SE undergraduate degrees have been established starting from the 1990s. We wanted to see whether SE departments are more successful in delivering education compared to CS, CE, and the other departments. To this end, we conducted independent samples t-test for comparing the importance and learning level of KAs and sub-KAs perceived by practitioners with (n = 27) and without an SE degree (n = 102). Independent samples t-test is an inferential statistical test that determines whether there is a statistically significant difference between the means in two separate groups. According to the results, there is not any significant difference between the importance of KAs and sub-KAs perceived by SEs with and without an SE degree. In other words, all participants have a similar perception on the importance of topics independent from their degree.

When it comes to learning levels, in general, learning levels of SE graduates are higher than the graduates from the other departments for all of the KAs. As shown in Figure 13, all the "◊" signs (which denote SE graduates) are above the square ("□") signs (which denote non-SE grads) with the same color according to the y-axis, i.e., learning level. Therefore, we can see that the learning levels for each KA are better for SE grads compared to the rest.

There are five KAs for which the learning levels are significantly different between practitioners with and without an SE degree. As shown in Table 11, practitioners with an SE degree think that



RIGHTSLINK

Table 11. The KAs for Which SEs with an SE Degree Think They Obtained Better Education

| Rank | KA | Mean Difference | Std. Deviation | t |
|---|---|---|---|---|
| 1 | Testing | 0.86 | 0.21 | 4.16 |
| 2 | Requirements | 0.73 | 0.21 | 3.49 |
| 3 | Maintenance | 0.56 | 0.19 | 2.88 |
| 4 | SE Models and Methods | 0.47 | 0.22 | 2.16 |
| 5 | Design | 0.44 | 0.20 | 2.19 |

they were educated better in software testing, software requirements, software maintenance, SE models and methods, and software design. This might be explained by the fact that according to SE curriculum guidelines [26], there should be dedicated classes for "Software Requirements Engineering," "Software Testing and Quality Assurance," and "Software Design." In CS and CE programs, these kinds of classes are either nonexistent or optional classes.

When we comment on this analysis in conjunction with the findings of RQ 1 and RQ 2, we can say that SE departments place more emphasis on software testing and maintenance, which are identified with the highest knowledge gaps. On the other hand, SCM is not addressed differently in SE departments than other computing-related departments.

## 5 DISCUSSIONS

In the following subsections, we summarize the implication of our study and discuss threats to validity.

### 5.1 Implication of Findings and Recommendations for Educators, New Graduates, and Managers

We believe that our dataset and findings have actionable implications for the university educators and researchers, new graduates, and managers who hire new graduates. This is true not only in the region under study but also beyond. Although our sample population size was not that large and training/skill profiles of graduates and software engineers would vary in different parts of the world, readers will agree that getting survey data from large populations is not trivial and usually very effort intensive [44]. While our survey and its dataset provided one "snapshot" from one specific region (Turkey) on this important issue, our approach can be replicated in other contexts to assess the knowledge gaps of software engineers.

This survey can potentially be used by education researchers as a starting point to conduct further research in knowledge deficiencies for new graduates in SE. Unlike previous studies in this area, our survey results include fine-grained details about the knowledge gaps in the sub-KA level of SWEBOK that is shown in Table 8 and Table 9.

By becoming aware of the general trends and what skills they will most likely need, new graduates will be able to sharpen their skills in those areas that were not well covered by their university program. These knowledge gaps can potentially be addressed by different methods such as internships, part-time jobs, certificate programs, or self-study.

Likewise, this survey can potentially help the managers to have an initial set of expectations of new graduates. The hiring managers can consider the general need for further training of new hires in the sub-KAs that have the highest knowledge gap. In one of our earlier studies, we presented an industrial summer school program [45] conducted by industry to address potential knowledge gaps in university education.



RIGHTSLINK

Based on our dataset, requirements engineering, design, and construction are perceived as the most important KAs in SE in industry. In the same way, the majority of curricula in universities today mainly focus on these three KAs. While this is a good sign for educating competent SEs for industry, we recommend that educators should address the knowledge gaps in their curricula. Based on our presented findings in Table 6 and Table 7, the top three core knowledge gaps are mainly concentrated on three KAs in SWEBOK; software maintenance, software configuration management (SCM), and software testing; 16 and 13 out of the top 20 sub-KAs with the highest knowledge gap belong to these three KAs according to Table 8 and Table 9, respectively.

In terms of curriculum design decisions, we would prefer not to advocate a certain curriculum design decision with respect to whether universities should focus on teaching topics that are found most important in positions new graduates hold or those topics important in positions that software engineers hold later in their career. However, we believe that it makes sense for university programs to focus slightly more on the former type of skills and topics, and to also instill skills for lifelong learning [46, 47], so that software engineers can continue to learn new topics/technologies as they encounter them in their career. After all, universities cannot teach *everything* to students.

In general, the universities are expected to provide holistic education, and not expected to give a fully customized training that is customized for any individual company's business process. However, despite the fact that the practices of implementing these would change from company to company, the general principles and best practices are largely the same. We argue that there is still a great benefit to teach the basics of these three KAs in universities. In the following subsections, we provide our recommendations in software maintenance, software configuration management, and software testing KAs.

*5.1.1 Software Maintenance.* Software maintenance is an important and inevitable activity ensuring that software continues to satisfy its objective over its intended lifetime [48]. The participants of our survey identified a high knowledge gap on this KA as shown in Table 6 and Table 7. Moreover, the sub-KAs of the maintenance KA are identified within the top 20 sub-KAs with the highest knowledge gaps as displayed in Table 8 and Table 9. In line with these findings, it has been pointed out in the literature that most SE courses involve developing software from scratch, whereas software engineers heavily work on the maintenance of software systems in the industry [49, 50]. To address this gap, there have been some efforts to design an SE course centered on a maintenance project [51]. The biggest challenge for teaching software maintenance has been identified as finding proper projects and codebases appropriate for undergraduate students [51]. To overcome this challenge, the use of open source software [50, 51] and outputs of previous years' students' projects [49] have been proposed. Based on the findings of [50], it is possible to use large-size open source software in course projects. On the other hand, the authors of [49] preferred to use a medium-sized, student-produced codebase with real software bugs.

When it comes to the main topics of software maintenance, students can be introduced with the fundamentals (role and scope, need for software maintenance, definitions, and categories of maintenance as covered by Section 5.1 of SWEBOK) and the key issues (technical and management issues identified by Section 5.2 of SWEBOK). Maintenance process and activities (Section 5.3 of SWEBOK) can be introduced by using a standard such as the "Software Maintenance Maturity Model" [52]. After building a base with core concepts, students can apply the techniques (Section 5.4 of SWEBOK) by using the tools (Section 5.5 of SWEBOK) for maintaining software. A catalog of refactoring techniques proposed by Fowler [53] can be used as structured teaching material. These techniques can be applied by students in course projects as presented in [54]. There are also some SE books [55] that organize the content of an SE course around maintenance. [55] proposes a flow of topics that consists of basic concepts, impact analysis, maintenance of software applying some techniques and using some tools, and verification of the changes made.



RIGHTSLINK

In summary, some researchers have identified the knowledge gap in the software maintenance KA and put some efforts into filling this gap. These efforts can be organized around the titles proposed by SWEBOK to enrich the current SE courses with a body of knowledge on software maintenance.

*5.1.2 Software Configuration Management (SCM).* Due to its nature, through the lifecycle of a software, many different kinds of changes occur. These changes have important consequences to the success of software; thus, they need to be explicitly managed with an SCM process. According to ISO/IEC/IEEE 24765 [56], "SCM is a discipline applying technical and administrative direction and surveillance to: (1) identify and document the functional and physical characteristics of a configuration item, (2) control changes to those characteristics, (3) record and report change processing and implementation status and verify compliance with specified requirements."

Although it is a crucial process in SE, the participants of our survey identified a high knowledge gap regarding SCM as shown in Table 6 and Table 7. In a more fine-grained level, 7 out of the top 20 sub-KAs with the highest knowledge gaps as displayed in Table 8 and Table 9 are related to SCM. These seven sub-KAs correspond to all sub-KAs of SCM. Since almost all of the programming assignments and term projects have only one version (final version), the concepts related to SCM are almost never taught in universities [57]. On the contrary, in industry, almost all software has versions, and these versions have to be explicitly maintained.

The core of the SCM process is handled by version control systems such as Git, SVN, Microsoft TFS, and AWS Code Commit. Although their specific functionality and the commands that they provide to the users might be different, the general logic of the version control systems is very similar from one tool to another. The concepts of pushing a commit, check-in, and check-out from a version control system, baselines, labels, and tags are shared among all the tools. So we strongly believe that students should be introduced with the fundamentals of software configuration identification (Section 6.2 of SWEBOK), software configuration control (Section 6.3 of SWEBOK), software configuration status accounting (Section 6.4 of SWEBOK), software configuration auditing (Section 6.5 of SWEBOK), and software configuration management tools (Section 6.7 of SWEBOK) using version control systems.

According to the survey respondents, the highest knowledge gap is measured in the software release management and delivery sub-KA (Section 6.6 of SWEBOK). With the recent increase in the industry of shorter releases and agile software development trends, the products are expected to have more frequent deliveries. This is addressed by DevOps frameworks and tools [58, 59], which is rarely taught in universities.

In the SE education literature, we only have found one example of a separate software configuration management class. Asklund and Bendix [57] reported their experiences on giving a 7-week undergraduate-level full course dedicated to SCM. The majority of the concepts in SCM are closely related to software maintenance (IEEE 14764 describes software configuration management as a critical element of the maintenance process [60]) and the knowledge gaps could be addressed with a single class related to these two subjects.

*5.1.3 Software Testing.* According to SWEBOK [24], software testing is the dynamic verification of a program to check if the program behaves expectedly using a finite set of test cases. The main goal of this verification activity is to identify the potential defects in a program.

Although it is a crucial process in SE and closely related to the quality of the software produced, the participants of our survey identified a large knowledge gap on Software Testing as shown in Table 6 and Table 7. In a more fine-grained level, 5 out of the top 20 sub-KAs with the highest knowledge gaps as displayed in Table 8 are related to software testing.



RIGHTSLINK

Generally, the majority of curricula in universities today mainly focus on the initial phases of the software development lifecycle, namely: requirements engineering, design, and implementation [61]. The testing-related knowledge is usually acquired while working on solutions related to an implementation-oriented course. On the contrary, in the industry, testing is considered a profession on its own, and there is an official title called "software test engineer." According to a survey conducted in North America [62], 6 out of 15 universities provide software testing courses, and there is an increasing trend in providing testing-related courses.

In the preparation of a software testing class, students can be introduced with the fundamentals (testing-related methodology, key issues, and relationship of testing to other activities as covered by Section 4.1 of SWEBOK) and the test processes and different levels of testing (as identified by Section 4.5 and Section 4.2 of SWEBOK). After building a base with core concepts, students can apply the test techniques (Section 4.3 of SWEBOK) by using the software testing tools (Section 4.6 of SWEBOK) for testing software.

In short, this knowledge gap can be addressed by Software Testing or Software Verification and Validation classes that are taught preferably by industry professionals as adjunct faculty.

### 5.2 Limitations and Potential Threats to Validity

Our study is subject to a number of threats to validity categorized in the following ways: (1) internal validity is the extent to which a piece of evidence supports a claim about cause and effect, and (2) external validity is the applicability of the conclusions to other environments.

*5.2.1 Internal Validity.* In our survey design, one of the important decisions is the selection of the knowledge categories. In this study, we used SWEBOK, a well-known and established standard in the industry. To minimize the possible misunderstanding of the sub-KAs in SWEBOK, we provided a summary of SWEBOK KA descriptions in the survey as described in Section 3.2. To further check the consistency of our survey, we initially conducted a pilot survey study with five engineers. Based on the feedback received from this pilot study, we performed minor modifications in our survey.

After we received the responses, we conducted reliability analysis in order to check the internal consistency of the responses. These analyses are performed by one of the authors and later reviewed by the remaining two authors. Related to the reliability analysis, Cronbach's alpha revealed high internal consistency of the responses about the importance of and perceived learning in all knowledge areas with 0.980 and 0.986 values, respectively.

Another piece of evidence for internal consistency can be the correlation between the responses given for a KA and its corresponding sub-KAs. The participants rated the importance and learning level for each KA and its corresponding sub-KAs separately (as shown in Figure 3 for the requirements KA). We expect that there should be a statistically significant correlation between the ratings of a KA and its corresponding sub-KAs. To assess this, we used the Pearson correlation coefficient to calculate correlations of importance and learning level of each sub-KA with its corresponding parent KA. All of the correlations are statistically significant at the 0.01 level (two-tailed). Pearson correlation coefficient values range from 0.408 to 0.903 for importance and 0.524 to 0.922 for learning level. This supports the internal consistency of our data. In addition, such a consistency can be an input for researchers on survey design for the same purpose. They might only assess the importance and learning level based on KAs and keep the size of the survey limited if they do not require data at a more detailed level. This might increase the number of data points, since the number of questions impacts participation in (and completion of) surveys.

We used parametric analysis to analyze our data since it is perfectly appropriate to summarize the ratings generated from Likert scales using means and standard deviations, and it is perfectly



RIGHTSLINK

appropriate to use parametric techniques [63], like paired and independent samples t-test in our case. [63] reports that it is also perfectly appropriate to use parametric techniques to obtain a more powerful and nuanced analysis of data. It is obvious that the distances between each pair of points on our 5-point scale are not equal. Parametric analysis treats these distances as equal and this is appropriate as stated by many researchers. Moreover, we do not use the means as absolute values; rather, we rank the means and draw conclusions by comparing these means. Another threat to validity can be associated with the calculation of knowledge gap. It is obvious that the corresponding items of importance perception and learning level, for instance, "3-Very useful" and "3-Learned a lot," are not equal. Therefore, taking the difference of importance perception and learning level is not correct from a mathematical perspective. On the other hand, we again do not use the differences as absolute values and use rankings to draw conclusions. In addition, we also compared our results on knowledge gap with another method [17] present in the literature and drew conclusions based on two methods. We did not use nonparametric techniques since we accepted the "intervalist position" [63] in this study and used parametric techniques, which are consistent with this view.

Some researchers claim that data obtained via Likert scale should be normally distributed if they are analyzed using parametric techniques [34]. Some researchers oppose this and claim that parametric statistics can be used with nonnormal distributions [35]. We checked Skewness and Kurtosis z-values [64] for each dataset obtained from the questions using a Likert scale and saw that most of the responses are normally distributed.

*5.2.2 External Validity.* Related to the external validity threats, the profile and the representativeness of the participants are important issues. In our study, the majority of the participants (118 out of 129) are located in Turkey and all of them received a bachelor's degree from a university in Turkey. However, we can argue that the majority of the participants are working in companies that have established partnerships with international companies, which means that the participants' backgrounds and their required knowledge level for different skills are similar compared to the international participants. At the same time, we should accept that cultural issues vary across countries and can have an impact on the results.

Another potential threat to validity in this study is the selection bias in participant selection. We tried to mitigate this by randomizing participant selection by combining convenient sampling and snowball sampling. We reached our industry network via email and social media accounts. Convenience sampling has some drawbacks, like its potential for leading to a homogeneous sampling frame and producing results that have limited generalizability to the broader population [32]. To minimize these drawbacks, we also employed snowball sampling and tried to reach a more heterogeneous sample via our industry network. In snowball sampling, referrals may not be effective in identifying diverse participants and only identify participants meeting specific characteristics. We also observe that convenience sampling is the dominant survey and experimental approach in SE [31]. Snowball sampling can also be effective in increasing the size of the sample [32].

Nearly half of the participants (51.2%) had five or fewer years of experience; the majority had under 10 years of experience (65.1%). This might have some impact on the results drawn out of our dataset. We can expect that some topics would be more important as one's career progresses. We addressed this in Section 4.2.2 and saw that more experienced practitioners (10+ years of experience) tend to consider three KAs (SE economics, SE management, and SE professional practice) significantly more important than less experienced ones ($\leq$10 years of experience). This difference supports the fact that less experienced practitioners likely have not been in roles that require some of the key SE skills, and hence their experience has an impact on their importance perception. Therefore, we should accept that our analysis reflects the opinions of less experienced software engineers. The positive side of this can be that recent graduates may recall what was covered during their university education better.



RIGHTSLINK

Another threat to validity can be that perceived importance at work may be impacted by what was stressed in degree programs of less experienced practitioners. The reason for this can be that they might have a good memory of what was covered in their education and they likely have not been in roles that require some of the key SE skills at work. We can address this threat with our analysis for RQ 4. Although there are overlaps between the courses delivered in SE and non-SE departments (CS, CE, etc.), SE departments offer more specific courses addressing SE topics in a more detailed way. On the other hand, according to our results, there is not any significant difference between the importance of KAs and sub-KAs perceived by SEs with and without an SE degree. In other words, all participants have a similar perception on the importance of topics independent from their degree, and hence we can conclude that the impact of the curricula on importance perception should be limited.

*5.2.3 Limitations.* We were able to compare the importance and learning-level perceptions of SE and non-SE graduates as presented in Section 4.5. Some computing degrees, such as CS and CE, include courses that relate to SE. We could not explore the difference between those with degrees in SE, CS, and CE since we do not have sufficient data points to allow a comparison. Moreover, we could have analyzed the difference between graduates of departments related and not related to computing if we had sufficient data points. These are the limitations of this study and can be the subject of our further work.

SWEBOK is one of the few comprehensive resources that classifies SE topics and is still expected to undergo revisions (we used the third version). We used SWEBOK KAs and sub-KAs to represent SE topics. This is another limitation of this study since we cannot claim that SWEBOK covers all SE topics or has a perfect classification. It is difficult to cover all SE topics since the field is very dynamic and new topics are emerging quite regularly, e.g., Agile methods, domain-specific languages, DevOps, test-driven development, and continuous integration. We could have formed a taxonomy, but it would be difficult to validate such a taxonomy. Regarding the classification, SWEBOK's sub-KAs were not formed to have a standard level of detail. While some sub-KAs, such as "test levels," have a narrow scope, some others, such as "SE methods," cover a much wider scope. Also, the sub-KA "practical considerations" under the requirements KA seems fuzzy or broad. We also realize that this could be a drawback of SWEBOK, which could possibly limit the implications of our study's findings. On the other hand, we can observe that the same problems would still exist in the case of any custom taxonomies formed via interviews [22] or other research methods [23] (as we presented briefly in Section 2).

Despite its drawbacks, we can see that SWEBOK is also being used for surveying SE practices and curricula assessment and development. One of the authors of this article surveyed SE practices used in Turkey and classified based on SWEBOK KAs [65]. [66] classified testing techniques based on SWEBOK's testing sub-KA. [67] surveyed games used for SE education and classified the findings based on SWEBOK. [68] proposed an SE curricula development and evaluation process using SWEBOK.

Another limitation is that we did not separate learning SE topics in university education versus "on the job." It is obvious that university education often focuses on fundamental SE topics and many software engineers learn a wealth of additional SE knowledge "after" they graduate and while working in companies.

## 6 CONCLUSIONS AND FUTURE WORK

As discussed earlier, the work reported in this article is part of a larger hybrid approach that we have developed over the past few years, to analyze and improve SE courses and programs. Our hybrid approach is based on two important inputs: SE body-of-knowledge frameworks (the two



RIGHTSLINK

well-known ones being the ACM SE 2014 and SWEBOK) and opinions about graduates' skills and industrial needs. Focusing on the former aspect related to SE body-of-knowledge frameworks, we reported two studies based on SWEBOK in two recent papers [27, 28], and the work in this article contributed to the latter aspect.

Based on our survey dataset, this article aims to raise awareness on the notion of quality in educating the software engineers of tomorrow by identifying the topics that are the most important in practice. We quantified the extent of learning of each SWEBOK KA in university settings versus their extent of use in industry by software engineers. Unlike previous related studies, we provided the knowledge gap information on a fine-grained level that is a sub-KA level in SWEBOK. We also quantified the knowledge gaps in each KA to identify the areas that need more training. We identified three key KAs (software maintenance, software configuration management, and software testing) with the highest knowledge gap according to the survey results. We have provided actionable recommendations for each KA to potentially address the knowledge gaps. Moreover, we also observed that the graduates with an SE degree obtain better education in five KAs (testing, requirements, maintenance, SE models and methods, and design) compared to the rest of the participants (who have a degree in a computing or noncomputing discipline except SE).

In terms of the road ahead, since we have observed the large number of studies on skill profiles of software engineers, we would like to raise the need for systematic syntheses and reviews in this important area. This study is not the first and certainly is not the last work in analyzing skill profiles of software engineers. Thus, we would like other researchers and practitioners to consider our approach and replicate or extend it in future studies. Further upcoming studies with larger datasets could provide more generalizable insights into this important issue.

## REFERENCES


[1] T. C. Lethbridge. 2000. Priorities for the education and training of software engineers. *Journal of Systems and Software* 53, 1 (2000), 53–71.

[2] A. Radermacher and G. Walia. 2013. Gaps between industry expectations and the abilities of graduates. In *Proceeding of ACM Technical Symposium on Computer Science Education*, 525–530.

[3] L. Kappelman, M. C. Jones, V. Johnson, E. R. McLean, and K. Boonme. 2016. Skills for success at different stages of an IT professional's career. *Communications of the ACM* 59, 8 (2016), 64–70.

[4] M. V. Genuchten. 1991. Why is software late? An empirical study of reasons for delay in software development. *IEEE Transactions on Software Engineering* 17, 6 (1991), 582–590.

[5] S. McConnell. 2010. What does 10x mean? Measuring variations in programmer productivity. In *Making Software: What Really Works, and Why We Believe It*, A. Oram and G. Wilson, Eds. O'Reilly Media.

[6] A. Bednarz. 2011. How IBM started grading its developers' productivity. http://www.networkworld.com/article/2182958/software/how-ibm-started-grading-its-developers–productivity.html. Last accessed: Jan. 2019.

[7] IBM. IBM launches new initiatives to advance developer skills. https://www-03.ibm.com/press/in/en/pressrelease/49969.wss. Last accessed: Jan. 2019.

[8] IBM. The value of training. https://www-304.ibm.com/services/learning/pdfs/IBMTraining-TheValueofTraining.pdf. Last accessed: Jan. 2019.

[9] M. J. Lutz, J. F. Naveda, and J. R. Vallino. 2014. Undergraduate software engineering: Addressing the needs of professional software development. *Queue* 12, 6 (2014), 30–39.

[10] R. Dewar. 2014. The education of embedded systems software engineers: Failures and fixes. *Embedded Systems Design Magazine*. https://www.embedded.com/the-education-of-embedded-systems-software-engineers-failures-and-fixes. Last accessed: Jan. 2019.

[11] Peter J. Denning. 2018. The computing profession. *Communications of the ACM* 61, 3 (2018), 33–35.

[12] Evans Data Corp. 2015. Global Developer Population and Demographic Study. https://evansdata.com/reports/viewRelease.php?reportID=9. Last accessed: Jan. 2019.

[13] D. Costa. 2012. STEM labor shortages? Microsoft report distorts reality about computing occupations. http://www.epi.org/publication/pm195-stem-labor-shortages-microsoft-report-distorts/. Last accessed: Jan. 2019.

[14] V. Garousi, G. Giray, E. Tuzun, C. Catal, and M. Felderer. 2019. Closing the gap between software engineering education and industrial needs. *IEEE Software*. DOI: 10.1109/MS.2018.2880823




RIGHTSLINK

[15] T. C. Lethbridge. 1998. The relevance of software education: A survey and some recommendations. *Annals of Software Engineering*, 6, 1–4 (1998), 91–110.
[16] T. C. Lethbridge. 2000. What knowledge is important to a software professional? *Computer* 33, 5 (2000), 44–50.
[17] B. Kitchenham, D. Budgen, P. Brereton, and P. Woodall. 2005. An investigation of software engineering curricula. *Journal of Systems and Software* 74, 3 (2005), 325–335.
[18] R. Colomo-Palacios, C. Casado-Lumbreras, P. Soto-Acosta, F. J. GarcíA-PeñAlvo, and E. Tovar-Caro. 2013. Competence gaps in software personnel: A multi-organizational study. *Computers in Human Behavior* 29, 2 (2013), 456–461.
[19] A. Radermacher, G. Walia, and D. Knudson. 2014. Investigating the skill gap between graduating students and industry expectations. In *Companion Proceedings of International Conference on Software Engineering*, 291–300.
[20] A. Radermacher, G. Walia, and D. Knudson. 2015. Missed expectations: Where CS students fall short in the software industry. *CrossTalk: The Journal of Defense Software Engineering* 28, 1 (2015), 4–8.
[21] J. Liebenberg, M. Huisman, and E. Mentz. 2015. Software: University courses versus workplace practice. *Industry and Higher Education* 29, 3 (2015), 221–235.
[22] M. Stevens and R. Norman. 2016. Industry expectations of soft skills in IT graduates: A regional survey. In *Proceedings of the Australasian Computer Science Week Multiconference*. ACM.
[23] M. Exter, S. Caskurlu, and T. Fernandez. 2018. Comparing computing professionals' perceptions of importance of skills and knowledge on the job and coverage in undergraduate experiences. *ACM Transactions on Computing Education (TOCE)* 18, 4, Article 21 (2018), 34 pages.
[24] P. Bourque and R. E. Fairley. 2014. Guide to the software engineering body of knowledge (SWEBOK), version 3.0. IEEE Computer Society Press.
[25] R. v. Solingen and E. Berghout. 1999. The Goal/Question/Metric Method: A Practical Guide for Quality Improvement of Software Development. McGraw-Hill.
[26] ACM. 2014. Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering (Software Engineering 2014). http://www.acm.org/binaries/content/assets/education/se2014.pdf. Last accessed: Jan 2019.
[27] V. Garousi, A. Mishra, and A. Yazici. Assessment and improvement of the Software Engineering curriculum using the SWEBOK: A case study in a Turkish program. In *11th European Computer Science Summit*.
[28] G. Giray, E. Tüzün, and V. Garousi. 2016. Assessment of the Turkish software engineering university programs using the SWEBOK guide. In *Proceedings of the Turkish National Software Engineering Symposium (in Turkish)*, 574–585.
[29] V. Garousi, G. Giray, and E. Tüzün. Survey questions: Skills important to software professionals based on SWEBOK (both in English and Turkish). https://doi.org/10.5281/zenodo.546594. Last accessed: Mar. 2017.
[30] V. Garousi, G. Giray, and E. Tüzün. 2017. Survey raw data: Skills important to software professionals based on SWEBOK. http://doi.org/10.5281/zenodo.584068. Last accessed: May 2017.
[31] D. I. K. Sjoeberg, J. E. Hannay, O. Hansen, V. B. Kampenes, A. Karahasanovic, N. K. Liborg, and A. C. Rekdal. 2005. A survey of controlled experiments in software engineering. *IEEE Transactions on Software Engineering* 31 (2005), 733–753.
[32] M. A. Valerio, N. Rodriguez, P. Winkler, J. Lopez, M. Dennison, Y. Liang, and B. J. Turner. 2016. Comparing two sampling methods to engage hard-to-reach communities in research priority setting. *BMC Medical Research Methodology* 16, 1 (2016), 146. DOI: 10.1186/s12874-016-0242-z
[33] Leo A. Goodman. 1961. Snowball sampling. *Annals of Mathematical Statistics* 32, 1 (1961), 148–170.
[34] Gerald van Belle. 2008. *Statistical Rules of Thumb*. 2nd edition. Wiley-Interscience.
[35] Geoff Norman. 2010. Likert scales, levels of measurement and the "laws" of statistics. *Advances in Health Sciences Education* 15, 5 (2010), 625–632.
[36] Susan Jamieson. 2004. Likert scales: How to (ab) use them. *Medical Education* 38, 12 (2004), 1217–1218.
[37] Thomas R. Knapp. 1990. Treating ordinal scales as interval scales: An attempt to resolve the controversy. *Nursing Research* 39, 2 (1990), 121–123.
[38] Norman H. Anderson. 1961. Scales and statistics: Parametric and nonparametric. *Psychological Bulletin* 58, 4 (1961), 305–316.
[39] Jerry L. Hintze and Ray D. Nelson. 1998. Violin plots: A box plot-density trace synergism. *American Statistician* 52, 2 (1998), 181–184.
[40] Robert L. Katz. 1955. Skills of an effective administrator. *Harvard Business Review* 33, 1 (1955), 33–42.
[41] Peter G. Northouse. 2018. *Leadership: Theory and Practice*. Sage Publications.
[42] R. Roumeliotis. 2015. Software engineers must continuously learn and integrate: Four ways programmers can thrive in their careers. https://www.oreilly.com/ideas/software-engineer-developer-coding-architecture-mobile-open-source. Last accessed: Dec. 2018.
[43] John L. Miller. 2016. Education Is Important, but Nothing Beats Experience in a Software Engineer. www.forbes.com/sites/quora/2016/08/25/education-is-important-but-nothing-beats-experience-in-a-software-engineer. Last accessed: Dec. 2018.



RIGHTSLINK

[44] H. Nekkanti and S. S. V. R. Reddy. 2016. *Surveys in Software Engineering: A Systematic Literature Review and Interview Study*. MSc thesis, Blekinge Institute of Technology, Sweden.
[45] E. Tuzun, H. Erdogmus, and I. G. Ozbilgin. 2018. Are computer science and engineering graduates ready for the software industry? Experiences from an industrial student training program. In *Proceedings of International Conference on Software Engineering: Software Engineering Education and Training*, 68–77.
[46] Barbara Kehm. 2015. The challenge of lifelong learning for higher education. *International Higher Education* 22 (2015), 5–7. DOI: https://doi.org/10.6017/ihe.2001.22.6906
[47] Reinhard Baran. 2011. Computer science aspects in lifelong learning. In *Proceeding of the 7th International Conference on Next Generation Web Services Practices*. IEEE, 476–480.
[48] D. D. Walden and G. J. Roedler (Eds.). 2015. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*. 4th edition. INCOSE.
[49] Claudia Szabo. 2014. Student projects are not throwaways: Teaching practical software maintenance in a software engineering course. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education (SIGCSE'14)*. ACM, New York, NY, 55–60. DOI: https://doi.org/10.1145/2538862.2538965
[50] M. Petrenko, D. Poshyvanyk, V. Rajlich, and J. Buchta. 2007. Teaching software evolution in open cource. *Computer* 40, 11 (2007) 25–31. DOI: 10.1109/MC.2007.402
[51] James H. Andrews and Hanan L. Lutfiyya. 2000. Experiences with a software maintenance project course. *IEEE Transactions on Education* 43, 4 (2000), 383–388.
[52] A. April, J. Huffman Hayes, A. Abran, and R. Dumke. 2005. Software maintenance maturity model (SMmm): The software maintenance process model. *Journal of Software Maintenance and Evolution: Research and Practice* 17, 3 (2005), 197–223.
[53] Martin Fowler. 2018. *Refactoring: Improving the Design of Existing Code*. 2nd edition. Addison-Wesley Professional.
[54] Shamsa Abid, Hamid Abdul Basit, and Naveed Arshad. 2015. Reflections on teaching refactoring: A tale of two projects. In *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education*. ACM, 225–230.
[55] Vaclav Rajlich. 2011. *Software Engineering: The Current Practice*. CRC Press.
[56] ISO/IEC/IEEE 24765:2010 Systems and Software Engineering—Vocabulary, *ISO/ IEC/IEEE*, 2010.
[57] Ulf Asklund and Lars Bendix. 2005. A Software Configuration Management Course.
[58] C. Ebert, G. Gallardo, J. Hernantes, and N. Serrano. 2016. DevOps. *IEEE Software* 33, 3 (2016), 94–100. DOI: 10.1109/MS.2016.68
[59] Mik Kersten. 2018. A Cambrian explosion of DevOps Tools. *IEEE Software* 35, 2 (2018), 14–17. DOI: 10.1109/MS.2018.1661330
[60] ISO/IEC/IEEE International Standard for Software Engineering - Software Life Cycle Processes – Maintenance. 2006. In ISO/IEC 14764:2006 (E) IEEE Std 14764-2006 Revision of IEEE Std 1219-1998, 1–58. DOI: 10.1109/IEEESTD.2006.235774
[61] T. Astigarraga, E. M. Dow, C. Lara, R. Prewitt, and M. R. Ward. 2010. The emerging role of software testing in curricula. In *Transforming Engineering Education: Creating Interdisciplinary Skills for Complex Global Environments*. IEEE, 1–26.
[62] V. Garousi and A. Mathur. 2010. Current state of the software testing education in North American academia and some recommendations for the new educators. In *23rd IEEE Conference on Software Engineering Education and Training (CSEE&T'10)*. IEEE, 89–96.
[63] James Carifio and Rocco Perla. 2008. Resolving the 50-year debate around using and misusing Likert scales. *Medical Education*, 42, 12 (2008), 1150–1152.
[64] David P. Doane and Lori E. Seward. 2011. Measuring skewness: a forgotten statistic? *Journal of Statistics Education* 19, 2 (2011).
[65] V. Garousi, A. Coşkunçay, A. Betin-Can, and O. Demirörs. 2015. A survey of software engineering practices in turkey. *Journal of Systems and Software* 108 (2015), 148–177.
[66] N. Juristo, A. Moreno, S. Vegas, and M. Solari. 2006, In search of what we experimentally know about unit testing. *IEEE Software* 23, 6 (2006), 72–80.
[67] C. Caulfield, J. C. Xia, D. Veal, and S. Maj. 2011. A systematic survey of games used for software engineering education. *Modern Applied Science* 5, 6 (2011), 28–43.
[68] A. Alarifi, M. Zarour, N. Alomar, Z. Alshaikh, and M. Alsaleh. 2016. SECDEP: Software engineering curricula development and evaluation process using SWEBOK. *Information and Software Technology* 74 (2016), 114–126.

Received January 2018; revised June 2019; accepted July 2019



RIGHTSLINK