A Framework to Evaluate Software Engineering Program Using SWEBOK Version 4

Mohammad Zarour, Mamdouh Alenezi, and Mohammed Akour

Abstract—**Software engineering (SE) has a dynamic nature that makes it challenging to design educational material that can adequately communicate the necessary knowledge and skills to students. The Guide to the Software Engineering Body of Knowledge (SWEBOK) has been updated and a new version has been released to cope with the fast base changes in the domain. Any software engineering program needs to be evaluated from different perspectives, including curricula, teaching methods, lab facilities, and faculty expertise. This paper aims to develop a common framework for evaluating software engineering programs. A case study is conducted to evaluate two aspects of the SE program, namely, the curricula and the expertise of the faculty. The main findings show the lack of coverage in several SWEBOK areas such as maintenance, software process, configuration management, construction, and software engineering economics. Additionally, new SWEBOK areas like software security and software engineering operations have limited course offerings. The study also recognizes some challenges in evaluating the success of bridging academia and industry through capstones and collaborations. Practitioners are recommended to conduct regular reviews of SE curricula in accordance with evolving standards such as the SWEBOK. They should strive to improve the coverage of areas that have not been adequately addressed and to expand course offerings in emerging SWEBOK areas.**

Index Terms—**SWEBOK, Software Engineering Program, Evaluation, Education.**

I. Introduction

Software Engineering (SE) is defined as 'the application of a systematic, disciplined, quantifiable approach to software development, operation, and maintenance' [1]. Since its inception in 1968, SE has evolved and encompassed sub-fields such as programming languages, methodologies, and tools. Software engineering professionals play vital roles in various aspects of software development, including quality assurance, maintenance, and documentation, enabling information technology (IT) to meet business needs effectively.

As IT becomes indispensable for operations, revenue generation, and customer engagement, businesses are increasingly adopting technologies such as e-Commerce, mobile, cloud, and blockchain. In this context, software quality is paramount, as cyber-threats require secure development practices. The growing emphasis on continuous delivery requires the automation of IT processes and the use of software automation tools, which, in turn, requires highly skilled SE professionals to develop, adopt, and maintain these systems. Consequently, SE education plays a crucial role in ensuring the success of both the workforce and the industry. High-quality SE curricula, aligned with industry needs and teaching in-demand skills and knowledge, are essential for student success [2], [3]. However, designing robust and up-to-date SE curricula remains a challenge due to rapid changes in the field.

Undergraduate SE education serves as an entry point into the field, providing multidisciplinary training in areas such as programming, design, testing, and project management. The goal of SE curricula is to equip students with the practical knowledge and skills required in the industry. Despite this objective, SE education has long grappled with providing practical industry-oriented experience [4]. Collaborations with industry partners can offer realistic learning opportunities that enable students to apply their knowledge and skills. Several case studies demonstrate the effectiveness of such partnerships [5].

This study aims to provide an overview of the current state of software engineering education and its alignment with industry practices, specifically in relation to the SWEBOK framework [6], [7]. SWEBOK serves as a foundational framework for defining the essential knowledge areas in software engineering, influencing both educational curricula and professional practices. SWEBOK has evolved significantly since its inception, reflecting the changing landscape of software engineering and the need for continuous adaptation in training and organizational processes. The latest version of SWEBOK V4 [6], incorporates agile, DevOps, and new knowledge areas in software architecture, operations, and security to reflect current industry practices. As the digital environment changes constantly, academic and training programs must adapt to include these elements to equip software engineers for current and future challenges, as outlined in SWEBOK V4 [8]. Hence, recent developments in continuous delivery, automation, and security require updates to the SE curriculum.

Our study emphasizes the need for curriculum updates to reflect current industry practices, as reflected in the latest version of SWEBOK (V4), which includes agile, DevOps, and new knowledge areas in software architecture, operations, and security [9]. We highlight the importance of incorporating emerging trends and technologies, such as continuous delivery, automation, and security, into software engineering curricula.

Manuscript received November 13, 2024; revised February 3, 2025. Date of publication March 20, 2025. Date of current version March 20, 2025. The associate editor prof. Tihana Galinac Grbac has been coordinating the review of this manuscript and approved it for publication.

M. Zarour is with the Faculty of Prince Al-Hussein Bin Abdullah II for Information Technology, Department of Software Engineering, Hashemite University, 13133- Jordan (e-mail: mzarour@hu.edu.jo).

M. Alenezi is with the AI Academy at Tahakoom, Riyadh 12331 Saudi Arabia (email: mkalenezi@tahakom.com).

M. Akour is with the Department of Software Engineering, Prince Sultan University, Riyadh, Saudi Arabia (email: makour@psu.edu.sa).

Digital Object Identifier (DOI): 10.24138/jcomss-2024-0088

We understand that there are other criteria or guidelines available for software engineering education, and our study acknowledges the existence of these. However, we argue that SWEBOK provides a comprehensive and widely accepted framework for software engineering education that can be applied across various educational institutions and industry settings. With regard to the unique characteristics of SWEBOK and its applications, we highlight the following: SWEBOK provides a comprehensive framework for software engineering education, covering a wide range of topics and knowledge areas that are relevant to both introductory and advanced courses. SWEBOK is regularly updated to reflect current industry practices and emerging trends, ensuring that students are prepared for the latest challenges and technologies in the field. SWEBOK emphasizes the importance of agile, DevOps, and security in software engineering, which are critical skills in the modern software industry. Furthermore, SWEBOK provides a common language and set of concepts for software engineering education, facilitating collaboration and communication between educators, students, and industry professionals. Note that adding security as a core topic in the SWEBOK has resulted in integrating security in ABET accreditation of software engineering programs [10].

Although the industry rapidly adopts new technologies and tools, best practices for software development and maintenance are not always followed. Consequently, interest is growing in the use of software standards to address this issue [2]. Universities can play a central role by incorporating best practices and standards into SE curricula. Although SE programs claim to adhere to the SWEBOK and ACM guidelines, few studies evaluate or periodically reassess compliance. Regular updates to SE curricula must align with standards and industry needs. Documenting compliance ensures that the curricula remain current and meet established standards. All this should be done through a clear and repeatable evaluation framework which is developed in this paper. The case study presented in this article has been developed to explain and implement the proposed evaluation framework.

This study evaluates, in its case study, the SE programs at Saudi Arabian universities against SWEBOK V4. SWEBOK model has been frequently used to assess SE programs, as in [7], [11], [12], [13], [14]. This study builds on the foundation laid by [14] in their SECDEP framework to evaluate software engineering curricula.The work in [14] used SWEBOK v3 as a reference to assess the alignment of software engineering courses with industry demands. The current study extends this research by applying the same SECDEP methodology, encompassing all phases of data collection, rating, validation, analysis, and visualization, but utilizing the latest version of SWEBOK (v4). This update aims to provide a more contemporary and comprehensive evaluation of software engineering curricula, considering the evolving landscape of software engineering practices and knowledge. The recommendations aim to improve the outcomes of the SE program. The case study focuses on SWEBOK V4 knowledge areas and their coverage in selected university curricula. The scope is limited to SE programs, excluding non-SE courses and topics. Concentrating on programs addresses a gap in existing research.

The paper is structured as follows. Section 2 reviews the relevant literature. Section 3 presents the general evaluation framework. Section 4 presents a case study that evaluates undergraduate SE curricula in a selected group of universities and proposes possible adjustments to align with the demands of the software market. Section 5 offers recommendations to improve the SE learning outcomes. Section 6 discusses the limitations of our research process and suggests potential avenues for future research.

II. Related Work

Software Engineering Education (SEE) has long focused on developing curricula that keep pace with the evolving needs of the industry. Unfortunately, software engineering curricula cannot change and adopt new technologies in a fast way. Modifying the curriculum to better serve industry needs is a long and tedious process in an academic setting [15]. Research in curriculum and education management is a critical area of study aiming to innovative better teaching methods [16]. In the past decade, there has been a significant body of literature on software engineering (SE) courses and their suitable teaching methods. [17], [18], [19], [20], [21], [22], [23]. This research is a result of international collaboration and co-work among researchers over different countries and in different languages [24]. However, research on SE programs and their alignment with standards and the body of knowledge, such as SWEBOK, has been relatively limited. Therefore, more effort is needed from researchers and stakeholders to improve SE education at the program level [25].

For example, Kitchenham et al. [26] conducted a notable study that examined the coverage of SWEBOK knowledge areas in undergraduate SE programs worldwide. Although most of the programs addressed most of the 15 SWEBOK V.3 knowledge areas, several gaps were identified, specifically in software engineering economics and processes. The study also revealed a lack of coverage of emerging topics such as artificial intelligence, blockchain, and cybersecurity, and recommended that undergraduate SE programs be updated to include these subjects. Radaideh, M. [27], [28] has studied the alignment of a Jordanian university with the SWEBOK KA. The adoption of SWEBOK V3.0 helps identify gaps in the current software engineering curriculum at Jordan University of Science and Technology . The integration of SWEBOK KAs ensures that essential knowledge areas are addressed, enhancing the overall quality of the software engineering program. The evaluation highlights areas where the curriculum aligns well with industry standards and areas needing improvement, guiding future curriculum development

Recently, the use of SWEBOK to enhance the teaching of core software engineering topics has been discussed in several research work. For example, Huang et al. [29] addressed the challenges faced in teaching and learning software engineering economics and reformed the corresponding course based on SWEBOK. Colares et al. [30] manipulated the SWEBOK knowledge area of software process improvement to create a set of content and competencies to be addressed in the course of software process improvement. Qamar and Ikram

[23] adopted the SWEBOK requirement engineering knowledge area to monitor and evaluate the course of requirement engineering.

The analysis and improvement of SE curricula on a global scale is essential to ensure that these programs remain updated with novel approaches and teaching methods. Researchers often investigate the alignment of SE curricula with established models such as SWEBOK or SE2014 in various universities and countries [31], [32], [14], [33], [34], [35], [36], [37]. These researches highlighted the need for software engineers who possess a diverse skill set, including technical and soft skills. To address the global skill gap, which require implementing new engineering methodologies, curriculum renewal, and innovative education and training methods. Unfortunately, no further studies have been documented that re-evaluate the SE curricula to explore if the recommendations of the aforementioned researches have been adopted, and if there are any new gaps. This led us to think of the evaluation process as an iterative, incremental and repetitive process, as discussed in the next section.

As the skills gap in IT is a worldwide problem, academics must not only focus on the coverage of knowledge areas and topics as outlined by SWEBOK or SE2014 but also on delivering these topics in a way that fills the skill gap in the IT industry [38]. To adequately prepare software engineering (SE) students for the industry, it is crucial to include soft skills such as continuous learning, creativity, and solution-oriented thinking in the SE curricula. However, this presents a more significant challenge than aligning the curricula with the SWEBOK and SE2014 guidelines.

The recent literature reflects a growing interest in addressing this issue and proposes solutions to bridge the gap between SE curricula and industry practices [32], [39], [40], [41], [42], [43], [44], [45], [46], [47]. To achieve this objective, collaborations between academia and the software development industry, along with the incorporation of real-world systems and products into the curriculum, have been recognized as crucial strategies. Additionally, career monitoring surveys can provide valuable insights into how well academic programs prepare graduates for the industry [36], [37].

Consequently, this study aims to develop an evaluation framework for the SE program to help evaluate SE programs in any university. The developed framework and the conducted case study will contribute to ongoing efforts to improve software engineering education and ensure that graduates possess the skills necessary for career success.

III. SE Program Evaluation Framework

The SE program evaluation process should be designed as an iterative, incremental, and repetitive process. This is because the SE domain is very dynamic and is rapidly evolving. Hence, a dynamic evaluation framework should provide a tool for assessing the strengths and weaknesses in various dimensions. Using SWEBOK latest version supports developing a robust evaluation as SWEBOK v4 represents the latest advancements in the field. The framework is depicted in Fig. 1. is iterative and incremental in the sense that the evaluation can have various scopes as discussed, next, in the second step of the framework. After completing the first round of the evaluation that covers certain evaluation scope, the evaluator can repeat the evaluation to cover another scope, and so on. The evaluation framework consists of the following steps:

A *Understand SWEBOK v4* First of all, the evaluators who aim to use this evaluation framework should familiarize themselves with the overall structure and organization of SWEBOK v4. It is divided into knowledge areas (KA) that cover various aspects of software engineering, such as requirements, design, development, testing, and maintenance. Within each KA, evaluators need to explore the topics, knowledge elements, and practices outlined. This will give you a comprehensive understanding of the core knowledge and skills that a competent software engineer should possess.

B *Define the evaluation scope* Determine which aspects of the program, as an evaluator, you want to evaluate. It could be the curriculum, teaching methods, faculty expertise, lab facilities, industry collaborations, or student outcomes. The possible evaluation aspects are summarized in Table I. Usually, most of these aspects are overlooked when conducting the evaluation. Although conducting an evaluation that includes all these aspects can be tedious and time-consuming work, it can be done with a sequence of studies or research projects. Then choose specific SWEBOK v4 KAs that best align with your chosen evaluation scope.

C *Develop an evaluation tool* Based on the chosen KAs, identify specific knowledge elements and practices from SWEBOK v4 that can be used as evaluation criteria. For each criterion, define measurable indicators or metrics that allow you to assess how well the program meets the expected standards. This could involve analyzing course syllabi, student projects, graduate surveys, employer feedback, or industry certifications achieved by graduates.

D *Conduct & validate the evaluation* Data collection primarily involved reviewing the corresponding websites for each SE program at the selected universities. This included examining program descriptions, course syllabi, faculty profiles, and any other relevant information available online. The data collected was then evaluated against the SWEBOK v.4 knowledge areas. The evaluation focused on identifying areas of alignment and areas for improvement in each program. To ensure the reliability of the evaluation, inter-rater reliability was assessed by calculating the Kappa coefficient for each SWEBOK knowledge area. This analysis helped minimize subjectivity and ensure consistency in the evaluation process.

E *Report and recommendations* Prepare a report summarizing the evaluation findings, including insights on the program's strengths and weaknesses in relation to SWEBOK v4 standards. Provide recommendations for improvement, such as curriculum adjustments, updates in teaching methods, or industry partnerships for specific KA areas.

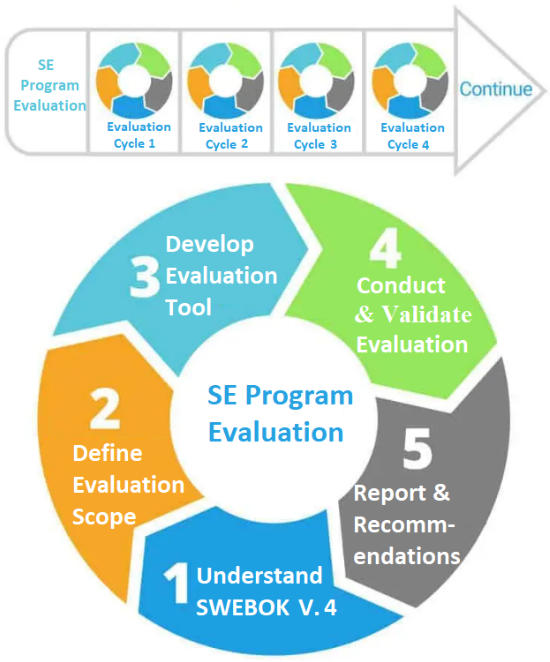

Fig. 1. Software Engineering Programs Evaluation Framework

Note that the evaluation is an iterative and continuous process in which the evaluator may conduct the evaluation periodically to maintain an updated status of the SE program. The evaluators can focus on certain aspects of the SE program in each evaluation. Hence, the evaluator may require different evaluation cycles to cover all aspects.

IV. Case Study

To practice the proposed framework, a case study is conducted to evaluate the curriculum of software engineering programs at Saudi universities. In Saudi Arabia, the demand for software engineering professionals is growing due to the increasing reliance on technology across various sectors. Several universities have established SE programs in response to this demand. Hence, it is crucial to ensure that these programs remain current with the latest developments in the field of software engineering and effectively prepare students for success in the industry.

A. *Understand SWEBOK v4*

The Software Engineering Body of Knowledge (SWEBOK) is a comprehensive guide that defines the core knowledge areas in the field of Software Engineering. SWEBOK has been revised several times since its inception and the latest version, SWEBOK V.4. incorporates new topics and updates to existing ones, making it an essential reference for educators, practitioners, and researchers in the field. Some of the significant changes in SWEBOK V.4 are:

1) New Knowledge Areas: SWEBOK V.4 includes three new knowledge areas, namely, Software Architecture, Software Engineering Operations, and Software Security. These areas are critical for modern software engineering practices and help bridge the gap between software engineering and related fields.
2) Revised knowledge areas: Some of the existing knowledge areas in SWEBOK have been revised to reflect the latest developments in the field. For example, the Software Testing Knowledge Area has been updated to include new testing techniques, such as agile testing and DevOps testing.
3) New topics: SWEBOK V.4 includes several new topics, such as Continuous Delivery, Cloud Computing, Machine Learning, and IoT (Internet of Things). These topics have become increasingly important in modern software engineering and are essential for keeping up with the latest trends in the field.
4) Updated references: SWEBOK V.4 updates the references and resources used in the earlier versions to reflect the latest research and industry practices. This ensures that the SWEBOK remains relevant and up-to-date with the latest developments in the field.

In general, the changes in SWEBOK V.4 reflect the evolving nature of software engineering and the need for a comprehensive and updated guide that can help practitioners, educators, and researchers keep up with the latest developments in the field. Well-known SE curricula guidelines refer to the SWEBOK when specifying core and elective courses, For instance, IEEE/ACM SE2014 curriculum guidelines. Table II, compares the SWEBOK V3 and V4 showing the new knowledge areas that have been added to it. Table III compares the IEEE/ACM SE2014 curricula guidelines and SWEBOK V.4. As can be seen in III, all the SE2014 KA are covered in the SWEBOK guide with at least one KA. This means that the SWEBOK model covers in more detail some KA more than the SE2014 model, for instance, KA 1, KA 2, and KA 6 in SE2014.

TABLE I
SOFTWARE ENGINEERING PROGRAM EVALUATION ASPECTS USING SWEBOK V4

Category	*Aspect*	*Evaluation Focus*
Curriculum	Technical Knowledge	Coverage of SWEBOK KAs, depth and breadth of courses, project-based learning, modern tools and technologies
		Course content, assignments, learning outcomes, projects, industry relevance
Teaching Methods	Active Learning	Use of discussions, group projects, problem-solving activities
		Classroom engagement, student participation, teaching strategies
Faculty Expertise	Qualifications & Experience	Academic credentials, industry experience, relevance to program focus
		Curriculum vitae review, industry partnerships, guest lectures from experts
Facilities & Resources	Computing Labs	Hardware, software, and network resources adequacy, lab environment
		Inventory of equipment, software licenses, lab management practices, accessibility for students
Student Outcomes	Graduate Rates & Employment	Percentage of graduates employed in software engineering field, time to secure employment
		Tracking graduate employment data, employer surveys, career services statistics

This study aims to explore how the new topics have been adopted, and evaluate the Software Engineering programs at the regional level in Saudi Arabia to explore their agility in adopting the new SWEBOK V.4 and cope with the new needs related to the Software Engineering curricula. The same exercise can be done on SE programs offered by universities in other countries using the same framework.

B. Define the Evaluation Scope

For the conducted case study, two aspects have been selected for evaluation, the curricula and the faculty expertise. The curricula of each SE program offered by a Saudi university is identified by reviewing the university's website and available documentation about the program. Similarly, faculty expertise and their domain of knowledge and interest are also reviewed and collected from their university website and personal websites (if available). As mentioned previously, considering all the evaluation aspects can be infeasible in one research work, but it can be achieved over a sequence of continuous evaluations.

C. Define the Evaluation Tool

Once the evaluation scope is defined, the evaluation tool has to be developed based on the latest version of SWEBOK, currently version 4. Hence, for the curricula evaluation tool, the tool consists of all SWEBOK KA and their subareas and the coverage level of each KA in the evaluated curricula. The curriculum evaluators will independently study the courses' descriptions. Whenever possible, the evaluators should communicate with the evaluated SE program coordinator or department chair at the corresponding university asking for more details. The evaluators then rate the courses according to the SWEBOK knowledge areas. Ratings and decisions will be discussed together and compiled. In case of any discrepancies, the reviewers will discuss and resolve them by mutual agreement; this process is depicted in Fig. 2 which outlines the curriculum evaluation process, expanding upon the methodology presented in [14]. A key enhancement is the inclusion of faculty expertise evaluation, providing a more comprehensive assessment of the curriculum's alignment with industry needs and best practices. The rating of each SWEBOK KA is done using a scale of 0-4 as follows:

A. *Scale 0*: No coverage of the knowledge area.
B. *Scale 1*: Some coverage in one course, but no dedicated course for the knowledge area.
C. *Scale 2*: Some coverage in more than one course, but no dedicated course for the knowledge area.
D. *Scale 3*: One dedicated course related to the knowledge area.
E. *Scale 4*: Two or more dedicated courses for the knowledge area.

Evaluating faculty expertise is conducted by reviewing the faculty web-page in each evaluated university, search the web for more information for each faculty member including their own websites, Google Scholar and Linked-In whenever available. The evaluation will adopt the same scale for the KA coverage as follows:

A. *Scale 0*: No faculty member is specialized in the assessed knowledge area.
B. *Scale 1*: A faculty member has some expertise in the evaluated knowledge area (based on his graduate studies majors, declared research interest, publications, and delivered courses).
C. *Scale 2*: More than one faculty member has some expertise in the evaluated knowledge area (based on his graduate studies majors, declared research interest, publications and delivered courses).
D. *Scale 3*: One faculty member is specialized in the evaluated knowledge area (based on his graduate studies majors, declared research interest, publications and delivered courses).
E. *Scale 4*: Two or more faculty members specialized in the evaluated knowledge area (based on his/her graduate studies' majors, declared research interest, publications and delivered courses).

D. Conduct and Validate the Evaluation

Conducting the evaluation means running the evaluation process as part of the evaluation framework, where the evaluation team collects data, rates the SE program and analyzes the findings. Validation the evaluation is discussed in the next section IV-E.

1) Curricula Evaluation - Data Collection: The initial phase of assessing the coverage level of each SWEBOK

TABLE II
KNOWLEDGE AREAS OF SWEBOK VERSION 3 COMPARED TO SWEBOK VERSION 4.

No.	*Knowledge Areas*	*No.*	*Knowledge Areas*
1	Software Requirements	1	Software Requirements
		2	Software Architecture
2	Software Design	3	Software Design
3	Software Construction	4	Software Construction
4	Software Testing	5	Software Testing
		6	Software Engineering operations
5	Software Maintenance	7	Software Maintenance
6	Software Configuration Management	8	Software Configuration Management
7	Software Engineering Management	9	Software Engineering Management
8	Software Engineering Process	10	Software Engineering Process
9	Software Engineering Models and Methods	11	Software Engineering Models and Methods
10	Software Quality	12	Software Quality
		13	Software Security
11	Software Engineering Professional Practice	14	Software Engineering Professional Practice
12	Software Engineering Economics	15	Software Engineering Economics
13	Computing Foundations	16	Computing Foundations
14	Mathematical Foundations	17	Mathematical Foundations
15	Engineering Foundations	18	Engineering Foundations

TABLE III
KNOWLEDGE AREAS OF SWEBOK VERSION 4 COMPARED TO SE2014.

KA#	*SE2014 Knowledge Areas*	*KA#*	*SWEBOK V4 Knowledge Areas*
1	CMP: Computing Essentials	16 4	Computing Foundations Software Construction
2	FND: Mathematical and Engineering Fundamentals	17 18	Mathematical Foundations Engineering Foundations
3	PRF: Professional Practice	14	Software Engineering Professional Practice
4	MAA: Software Modeling and Analysis	11	Software Engineering Models and Methods
5	REQ: Requirements Analysis and Specification	1	Software Requirements
6	DES: Software Design	2 3	Software Design Software Architecture
7	VAV: Software Verification & Validation	5	Software Testing
8	PRO: Software Process	10	Software Engineering Process
9	QUA: Software Quality	12	Software Quality
10	SEC: Security	13	Software security

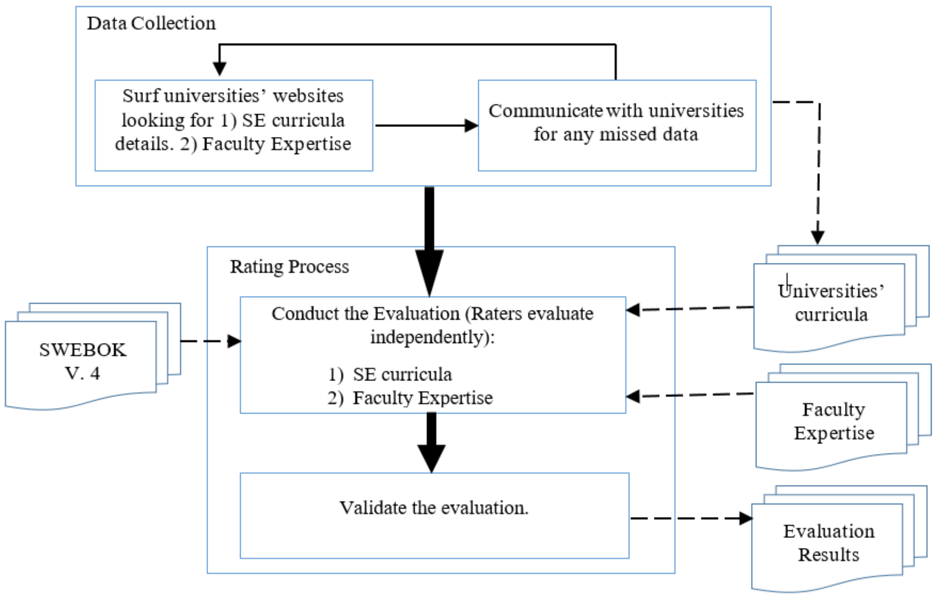
Data Collection
Surf universities' websites looking for 1) SE curricula details. 2) Faculty Expertise
Communicate with universities for any missed data
Rating Process
SWEBOK V. 4
Conduct the Evaluation (Raters evaluate independently):
1) SE curricula
2) Faculty Expertise
Universities' curricula
Faculty Expertise
Validate the evaluation.
Evaluation Results

Fig. 2. Software engineering curricula assessment process.

knowledge area (KA) involves two steps. Firstly, the selection of universities to be included in the evaluation process and gathering adequate information about their software engineering (SE) curricula. This requires identifying academic departments related to computer and information sciences in the relevant universities. Although some universities may not offer dedicated SE programs, they may provide SE courses as part of their computer science programs. Therefore, it is essential to specify the academic programs to be inspected. Secondly, collecting information about the curricula, courses, and their expected outcomes. However, this information may not be readily available on the web, and it may be necessary to contact the corresponding departments to obtain the necessary information. To enhance the evaluation process, information such as study plans, core courses, elective courses, credit hours allocated to each course, credit hours related to the SE discipline, and general education courses can be beneficial.

In Saudi Arabia, there are a total of 34 universities across the country, covering the entire geographical area. Approximately one-fourth of these universities are private, while the remaining are public. Of the 34 universities in Saudi Arabia, only nine have a dedicated software engineering program or department. Four of these universities are public: King Saud University (KSU), King Fahad University for Petroleum and Minerals (KFUPM), Hail University (HU), and Jouf University (JU), while the remaining five are private: Prince Sultan University (PSU), Alfaisal University (Alfaisal), Prince Mohammad University (PMU), University of Business and Technology (UBT), and Alyamamah University (Alyamamah). Table IV provides a summary of the software engineering programs offered by these universities.

TABLE IV
SOFTWARE ENGINEERING PROGRAMS IN SAUDI ARABIA.

Type	*Name*	*Foundation Date*	*SE Programs*
Public	KSU (U1)	1957	BSc, MSc
	KFUPM (U2)	1963	BSc, MSc
	HU (U3)	2005	BSc
	JU (U4)	2005	BSc
Private	PSU (U5)	1999	BSc
	Alfaisal (U6)	2002	BSc
	PMU (U7)	2006	BSc
	AlYammamah (U8)	2001	BSc
	UBT (U9)	2012	BSc

2) Faculty Expertise - Data Collection: Collecting data related to faculty expertise is challenging. The universities' websites that show the faculty members information are incomplete. The websites like google scholar and Linked-In shows details about publications and expertise but not for example their graduate studies majors. due to these limitations, and based on the available information, the faculty expertise evaluation was conducted to two universities, PSU and Al-Faisal Universities.

3) Data Analysis: Once the evaluators complete their evaluation, their findings are tabulated in a table showing, for each SE program in a university, their rating of the KA coverage. Table V shows the rating levels of the courses delivered by Saudi Universities. The collected data are summarized and presented using illustrative charts, for this research, the radar net charts are used (as depicted in Fig. 3). To calculate credit hours, elective courses should be disregarded as their content and frequency of offerings may vary from one semester to another. Similarly, senior project courses are also excluded as they mainly involve the application of knowledge gained in previous courses, and typically do not cover new knowledge areas. However, we will track programs that provide senior projects for cooperative training, as this can indicate which programs equip their students with the necessary industry skills.

Similarly, for faculty expertise evaluation, the review of each faculty expertise and its mapping to the corresponding SWEBOK KA(s) is recorded. Table VI summarizes the evaluation accordingly. The results are depicted in figure 4.

The analysis of the current SE curricula in the various programs/universities showed that:

A. A large number of computing and math foundations courses are offered in various universities. Despite the importance of these two domains which are covered in SWEBOK, we believe that these computing and math courses should be revised and a decision should be made on how to deliver them; as separate courses or merge some topics in one course and design special math and computing courses for SE programs. This will reserve some room for more SE-related courses to be added to the SE curricula.
B. Course offering: some SE courses are offered along with other courses, while it is better to offer them in different semesters, the sequence of SE courses' offerings needs to be revised and sequenced properly.
C. Some SWEBOK knowledge areas are not covered adequately; this includes maintenance, software process, configuration management, construction, and software engineering economics.
D. With regard to the three new knowledge areas, few universities have made a proactive step and offered a dedicated course in software security. The operations knowledge area is still uncovered adequately, neither as part of a core course nor an elective course, while software architecture is partially covered as part of the software design and architecture course in most universities.
E. Although SE programs in Saudi Arabia worked to bridge the gap between academic programs and industrial needs through either a capstone project or cooperative programs with the industry or both, the success of such solutions is not evaluated yet and needs further studies.

The analysis of the faculty expertise in the two universities showed that:

A. There is a critical shortage in knowledge areas (rated as 0 or 1) such as software operations, and software engineering economics knowledge areas. These two KA are uncovered as separate courses or as part of other courses, maybe due to the shortage of expertise.
B. There is a major shortage in several other knowledge areas (rated 2 or 3) like software construction, maintenance, software configuration management, software management, software process.

TABLE V
RATING LEVELS OF THE COURSES DELIVERED BY SAUDI UNIVERSITIES

Area		Universities								
	KA	U1	U2	U3	U4	U5	U6	U7	U8	U9
REQ	Sw Requirements	3	3	3	3	3	3	1	3	3
DES	Sw Design	3	1	2	2	2	2	2	2	2
ARCH	SW Architecture	1	2	2	2	3	2	3	3	2
CST	Sw Construction	3	2	3	3	3	0	0	0	0
TST	Sw Testing	3	3	2	3	3	3	3	3	3
OPS	SW Operations	0	0	0	0	0	0	0	0	0
MNT	Sw Maintenance	3	0	0	3	1	0	1	3	3
CNF	Sw Config. Mgmt.	1	0	0	0	1	1	0	0	0
MGT	SwE Management	3	3	3	3	3	3	0	3	3
PRC	SwE Process	1	1	1	2	1	1	2	2	3
MAM	SwE Models and Methods	2	2	2	2	2	1	2	2	2
QLY	Sw Quality	3	3	3	3	2	2	2	2	4
SEC	SW Security	3	2	2	1	2	2	2	3	2
SEP	SE Professional Practice	4	4	3	3	3	4	1	3	1
SEE	Software Engineering Economics	1	1	0	1	2	0	3	0	0
COF	Computing Foundations	4	4	4	4	4	4	4	4	4
MAF	Mathematical Foundations	4	4	4	4	4	4	4	4	4
ENF	Engineering Foundations	1	1	4	1	0	1	3	0	3

TABLE VI
RATING LEVELS OF THE FACULTY EXPERTISE IN TWO SAUDI UNIVERSITIES

KA	SWBOK KA Desc.	PSU	AlFaisal
REQ	Sw Requirements	3	3
DES	Sw Design	4	4
ARCH	SW Architecture	3	4
CST	Sw Construction	2	2
TST	Sw Testing	4	2
OPS	SW Operations	1	0
MNT	Sw Maintenance	3	2
CNF	Sw Config. Mgmt.	2	2
MGT	SwE Management	2	2
PRC	SwE Process	3	2
MAM	SwE Models and Methods	2	4
QLY	Sw Quality	4	2
SEC	SW Security	4	3
SEP	SE Professional Practice	2	0
SEE	Software Engineering Economics	1	0
COF	Computing Foundations	4	4
MAF	Mathematical Foundations	4	4
ENF	Engineering Foundations	3	4

Note that each department can use this evaluation tool to more deeply delve into its expertise and work to bridge the gap accordingly.

E. Validate the Evaluation

Identifying the knowledge areas to which each course belongs can be a challenging task when evaluating each course. Therefore, determining whether a KA is partially or fully covered by a course is subjective. To measure the level of agreement among evaluators (the authors of this paper), we calculated the inter-rater agreement level using Fliess Kappa analysis. A sample of the kappa analysis is shown in Table VII, where we randomly selected five universities and calculated the kappa coefficient for KAs. The Fliess Kappa coefficient reflects the credibility and robustness of the rating process conducted, as well as the homogeneity in the ratings provided by the raters. The strength of the agreement can be determined based on the classifications given in [12]. Table VII indicates that there is a good level of agreement in almost all KAs. A very good agreement is recorded in software process KA and a moderate agreement is recorded for software operations KA. To further validate the results of the SE curricula evaluation, we plan to send emails to the chair of the department where the software engineering program is delivered at each university. We believe that the emails will be more credible and will gain the recipient's interest if the email includes an attached copy of this research paper that documents the results. By reaching out to the faculty of each SE department, we can gain additional insights that may significantly impact the accuracy of the obtained results. The SE programs will undergo another evaluation process in 2-3 years to explore the updates on the curricula and see to what extent our recommendations are considered.

V. GUIDELINES AND RECOMMENDATIONS

Software engineering programs must adapt to the changing skill requirements in software engineering due to the evolution of the industry's needs, the advent of large language models, cybersecurity, and AI. This is vital due to the growing demand for engineers with a combination of technical and non-technical skills in the local software industry [48], [49], [50]. The following are some examples of these ways:

A. Integration of machine learning and AI: Software engineering programs can integrate machine learning and AI into their curricula to ensure that students have a solid understanding of these technologies. This can include courses on data science, machine learning algorithms, and natural language processing.

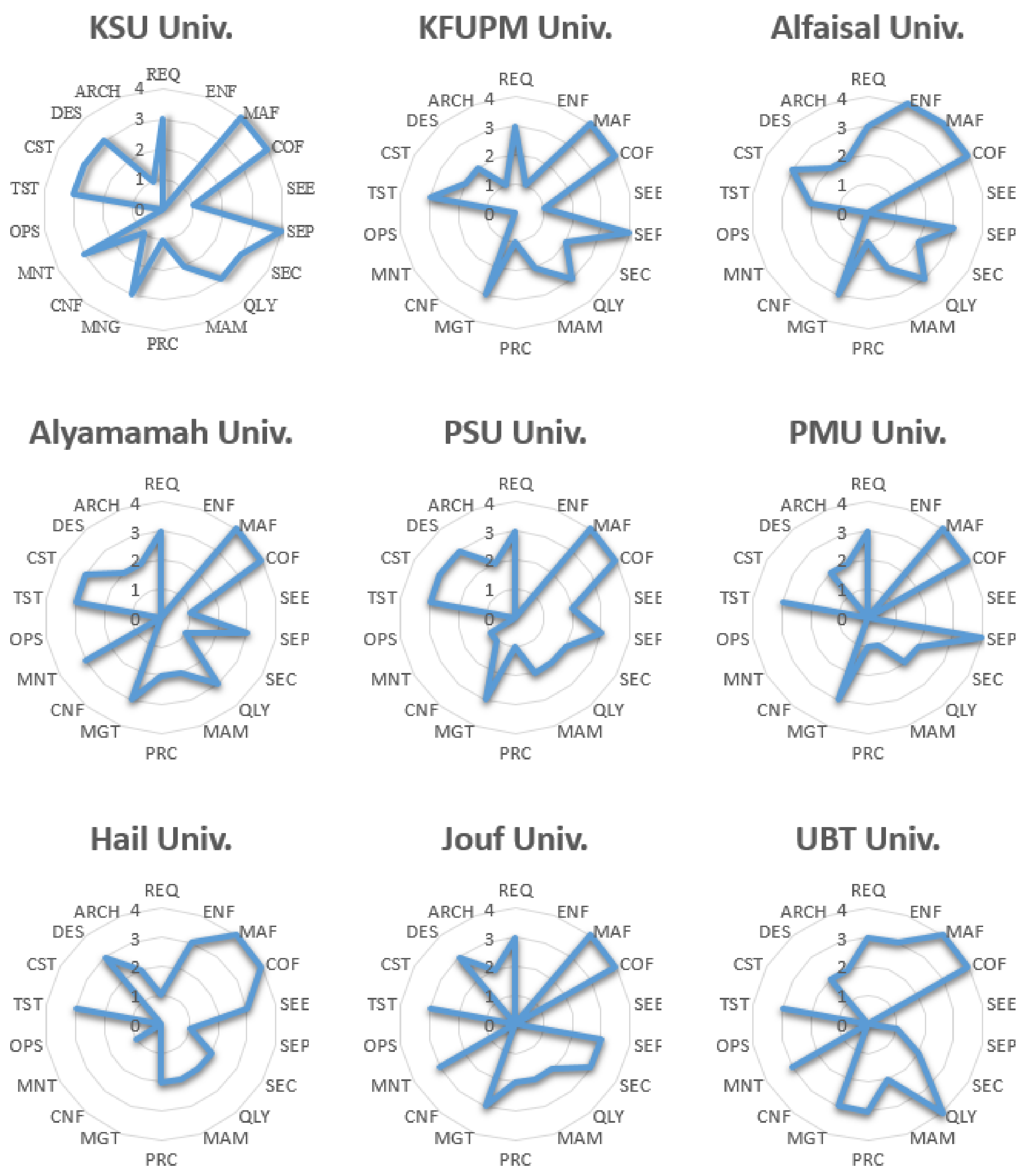

Fig. 3. The evaluation results of the nine studied universities.

B. Integration of cybersecurity into the SE program: cybersecurity plays an integral role in software development. Nowadays developing secure software is crucial and cannot be underestimated. So, software engineering should incorporate security in each phase of the development process.

C. Emphasis on ethics and responsible AI and cybersecurity: As the demand grows for ethical and responsible AI and cybersecurity, software engineering programs can include courses on the ethical and social implications of AI as well as cybersecurity. These courses can cover topics such as bias in AI, privacy concerns, the responsible use of AI, ethical hacker responsibilities and practices.

D. Cloud computing, web, and mobile applications: The market requires engineers who possess the skills to design and develop applications and solutions based on current and emerging technologies. To meet this demand, software engineering programs should of- fer courses in web/mobile design, cloud architectures, and related frameworks to equip students with the necessary competencies.

E. Hands-on experience: Software engineering programs can provide students with hands-on experience with AI and machine learning technologies. This can include projects that involve building AI systems, using AI to solve real-world problems, and working with large datasets.

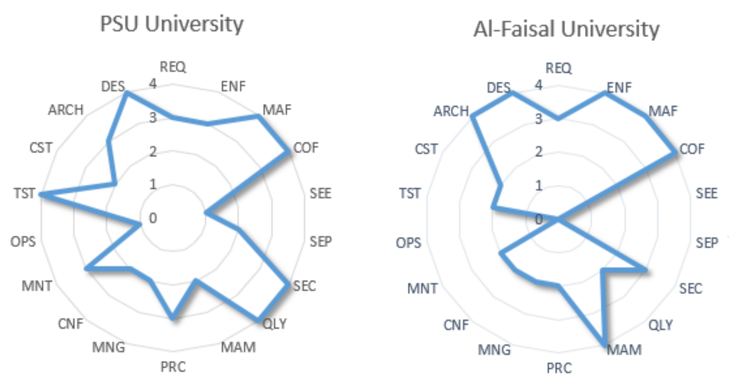

Fig. 4. Faculty Expertise Evaluation for Two Universities.

TABLE VII
INTER-RATER AGREEMENT CALCULATIONS – KAPPA COEFFICIENT.

KA	*K-Value*	*Agreement's strength*
REQ	0.73214	Good
DES	0.75625	Good
ARCH	0.72231	Good
CST	0.73942	Good
TST	0.73077	Good
OPS	0.42308	Moderate
MNT	0.76315	Good
CNF	0.64286	Good
MGT	0.73473	Good
PRC	0.80588	V. Good
MAM	0.76315	Good
QLY	0.73588	Good
SEC	0.74954	Good
SEP	0.73624	Good
SEE	0.76879	Good
COF	0.71519	Good
MAF	0.71519	Good
ENF	0.75838	Good

F. Collaboration with industry: Software engineering programs can collaborate with industry partners to ensure that their curricula are aligned with the latest trends and technologies in the field. These partnerships can also provide students with opportunities for internships, mentorship, and real-world experience.
G. Lifelong learning: Software engineering programs can emphasize the importance of lifelong learning and encourage students to continue developing their skills after graduation. This can include offering professional development courses, providing access to online learning resources, and encouraging participation in industry conferences and events.
H. Testing and quality assurance: Strong skills in testing, validation, and quality assurance are essential to meet market needs. Programs should include specialized courses on test design, automation, and formal verification methods to build expertise in this area.
I. Project management: In addition to technical skills, the Saudi industry requires engineers with a working knowledge of project management, communication, and leadership skills to help deploy software solutions. Programs should provide opportunities to develop these soft skills through team projects, case studies, and possibly dedicated project management courses.
J. Emerging technologies: The fast-changing nature of the Saudi software industry demands lifelong learning skills to keep up with new innovations. Programs should foster a culture of continuous learning and exploration of new technologies to produce adaptable engineers.

VI. Limitations

This study, while providing valuable insights into evaluating software engineering programs, has certain limitations. Firstly, data collection relied heavily on university websites, which may not always provide comprehensive and accurate information about curricula and faculty expertise. We were unable to include universities with websites that were either unavailable or lacked detailed descriptions of their software engineering (SE) programs and courses offered. As a result, we may have excluded some relevant data that could have influenced our findings. Secondly, our classification is based solely on the information obtained from the websites of the universities studied, including program specifications, course descriptions, and the web-pages of the faculty members. Although we attempted to validate our results by obtaining feedback from all the universities studied, this approach may not have captured all the nuances of the SE curricula, courses offered, and all the expertise of the faculty. Throughout our study, we encountered several difficulties, such as incomplete course descriptions or descriptions that did not reflect all the course content, missing information related to faculty expertise, their research interests, list of courses taught, and their majors.

Secondly, the focus was on Saudi Arabian universities which limits the generalizability of the findings to other contexts. However, the framework developed can be applied by any SE program. Additionally, it is worth mentioning that excluding senior projects from the rating presented in this paper may be viewed as a limitation, as most SE students gain opportunities to enhance their SE skills by working on their graduation projects, which involve all phases of the software development life-cycle. However, this decision was made due to the variability of these projects in their topics and scope, making it challenging to rate them accurately. Ultimately, these limitations do not impact the applicability of the results obtained in this research, but rather provide directions for future research and improvements in the methodology.

In addition to the limitations of the case study, the proposed framework itself has some inherent limitations. While SWEBOK is a globally recognized reference for software engineering knowledge, it may not fully capture all aspects of modern software development practices. For instance, emerging technologies and trends might not be adequately reflected in SWEBOK, potentially limiting the framework's ability to assess the preparedness of graduates for the evolving demands of the industry. Furthermore, the SWEBOK knowledge areas are broad, and ensuring comprehensive coverage by evaluators, particularly in specialized areas, can be a significant challenge. The framework also relies on the assumption that the SWEBOK knowledge areas provide a sufficient basis for evaluating the quality of SE education. While SWEBOK is a valuable resource, it is important to acknowledge that it is not an exhaustive or static standard. The continuous evolution of software engineering practices necessitates regular updates to SWEBOK. Therefore, it is crucial to periodically re-evaluate SE curricula in light of SWEBOK updates to ensure alignment with the latest industry standards and best practices.

Finally, the successful implementation of this framework requires significant resources and expertise. Institutions need to invest in training evaluators, gathering data, and conducting the evaluation process. Moreover, ensuring consistent application of the framework across different programs and institutions can be challenging.

VII. Conclusion and Future Directions

In conclusion, this study developed an evaluation framework and applied it to evaluate the software engineering curricula and faculty expertise of selected Saudi universities using the SWEBOK Guide Version 4. The analysis revealed that while the curricula provide adequate coverage of the core software engineering knowledge areas and align with local market needs, certain topics remain underrepresented.

Specifically, the curricula adequately address newly introduced areas in SWEBOK V4 such as software security and professional software engineering practice. This indicates the universities' cognizance of evolving software engineering domains and commitment to adjusting curricula accordingly. However, areas such as software economics, software operations, engineering foundations, software maintenance, and software process require additional coverage in curricula. Based on an analysis of market requirements, increasing the number of testing and project management courses, currently covered by only one course each, would also benefit the curricula. Revision of the curriculum to align with the updated SWEBOK V4 guidelines is also recommended. The suggested actions aim to optimize the curricula to meet both SWEBOK standards and local market demands. Future research could evaluate each SWEBOK knowledge area individually across programs or utilize course hour allocations in curriculum guidelines to quantitatively assess knowledge area coverage. Examining course materials in more detail may also yield in-sights into better alignment with knowledge areas. Other studies could focus on developing quantitative criteria to evaluate software engineering curricula based on metrics such as the number of core courses, general education courses, and hours per course. Analyzing senior cap- stone projects and industry collaborations to identify their strengths, weaknesses, and potential improvements constitutes another area for further research.

In summary, while the evaluated software engineering programs exhibit adequate coverage of core topics and adapt to changes in the field, opportunities remain to strengthen certain under- represented areas and enhance alignment with SWEBOK guidelines through a multifaceted approach. Future research can build upon this study, using various methodologies to gain a more comprehensive understanding of software engineering curricula.

References

[1] ISO/IEC 25010, *ISO/IEC 25010:2011, Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models*, Std., 2011.

[2] Y.-T. Lin, "Effects of flipped learning approaches on students' learning performance in software engineering education," *Sustainability*, vol. 13, no. 17, p. 9849, 2021.

[3] A. Mishra and D. Mishra, "Sustainable software engineering: Curriculum development based on acm/ieee guidelines," *Software Sustainability*, pp. 269–285, 2021.

[4] H. Su, S. Jodis, and H. Zhang, "Providing an integrated software development environment for undergraduate software engineering courses," *Journal of Computing Sciences in Colleges*, vol. 23, no. 2, pp. 143–149, 2007.

[5] J. Maguire and Q. Cutts, "Back to the future: shaping software engineering education with lessons from the past," *ACM Inroads*, vol. 10, no. 4, pp. 30–42, 2019.

[6] P. Bourque and F. R. E. (eds.), Eds., *Software Engineering Course (SWEBOK) V4 (Beta Version)*. Los Alamitos, CA: IEEE Computer Society, 2022.

[7] P. Kamthan and H. Washizaki, "Swebok matters: Report and reflection of a seke panel on the educational and professional implications of swebok," *International Journal of Software Engineering and Knowledge Engineering*, pp. 1–11, 2022.

[8] H. Washizaki, M. I. S. Segura, J. Garbajosa, S. Tockey, and K. Nidiffer, "2. envisioning software engineer training needs in the digital era through the swebok v4 prism," 2023.

[9] P. Singh and L. K. Singh, "Engineering education for development of safety-critical systems," *IEEE Transactions on Education*, vol. 64, no. 4, pp. 398–405, 2021.

[10] W. Schilling, "The state of the practice integrating security in abet accredited software engineering programs," 06 2023.

[11] R. E. Fairley, "A software engineering competency model (swecom)," *IEEE Computer Society*, 2014.

[12] M. Ardis, D. Budgen, G. W. Hislop, J. Offutt, M. Sebern, and W. Visser, "Se 2014: Curriculum guidelines for undergraduate degree programs in software engineering," *Computer*, vol. 48, no. 11, pp. 106–109, 2015.

[13] R. E. D. Fairley, P. Bourque, and J. Keppler, "The impact of swebok version 3 on software engineering education and training," in *2014 IEEE 27th Conference on Software Engineering Education and Training (CSEE&T)*. IEEE, 2014, pp. 192–200.

[14] A. Arrifi, M. Zarour, N. Alomar, Z. Alshaikh, and M. Alsaleh, "Secdep: Software engineering curricula development and evaluation process using swebok," *Information and Software Technology*, vol. 74, pp. 114–126, 2016.

[15] A. Kiselev, "Extracting a body of knowledge as a first step towards defining a united software engineering curriculum guideline," Doctoral Dissertation, Embry-Riddle Aeronautical University, 2023. [Online]. Available: https://commons.erau.edu/edt/743

[16] B. Malik and S. Zafar, "A systematic mapping study on software engineering education," *International Journal of Educational and Pedagogical Sciences*, vol. 6, no. 11, pp. 3343–3353, 2012.

[17] J. Yu, J. Zhang, Y. Chen, N. Wu, Y. Mei, W. Sun, and L. Zhu, "Construction of a resource database of course ideology and politics for software quality assurance and testing course," *structure*, vol. 3, p. 285, 2023.

[18] M. Alenezi and M. Akour, "Methodical software testing course in higher education." *International Journal of Engineering Pedagogy*, vol. 12, no. 1, 2022.

[19] N. Ibrahim, S. A. Halim, and N. A. Saadon, "Learners reflection on collaborative project in project-oriented problem-based learning for software engineering courses," in *AIP Conference Proceedings*, vol. 2433, no. 1. AIP Publishing LLC, 2022, p. 030008.

[20] S. Jiménez, A. Alanis, C. Beltrán, R. Juárez-Ramírez, A. Ramírez-Noriega, and C. Tona, "Usqa: A user story quality analyzer prototype for supporting software engineering students," *Computer Applications in Engineering Education*.

[21] S. Ouhbi, A. Idri, J. L. Fernández-Alemán, and A. Toval, "Requirements engineering education: a systematic mapping study," *Requirements Engineering*, vol. 20, pp. 119–138, 2015.

[22] M. Burch, "The importance of requirements engineering for teaching large visualization courses," in *2020 Fourth International Workshop on Learning from Other Disciplines for Requirements Engineering (D4RE)*. IEEE, 2020, pp. 6–10.

[23] F. Qamar and N. Ikram, "Improving monitoring and evaluation of undergraduate curriculum: A case of software requirements engineering course," *Education and Information Technologies*, pp. 1–29, 12 2024.

[24] S. Alam, S. Zardari, U. Laila, M. Abbas, K. Shaheen, S. Fatima, and S. A. Ahmed, "Swebok-based bibliometric analysis of software engineering," *Journal of Independent Studies and Research Computing*, vol. 22, no. 1, 2024, iSSN (E): 1998-4154.

[25] M. M. Qadir and M. Usman, "Software engineering curriculum: A systematic mapping study," in *2011 Malaysian Conference in Software Engineering*. IEEE, 2011, pp. 269–274.

[26] B. Kitchenham, D. Budgen, P. Brereton, and P. Woodall, "An investigation of software engineering curricula," *Journal of Systems and Software*, vol. 74, no. 3, pp. 325–335, 2005.

[27] M. A. Radaideh, "Benchmarking the software engineering undergraduate program curriculum at jordan university of science and technology with the ieee software engineering body of knowledge (swe knowledge areas #6–10)," in *Advances in Software Engineering, Education, and e-Learning*, H. R. Arabnia, L. Deligiannidis, F. G. Tinetti, and Q.-N. Tran, Eds. Cham: Springer International Publishing, 2021, pp. 85–100.

[28] ——, "Benchmarking the software engineering undergraduate program curriculum at jordan university of science and technology with the ieee software engineering body of knowledge: (software engineering knowledge areas 11-15)," in *2021 International Conference on Computational Science and Computational Intelligence (CSCI)*, 2021, pp. 1043–1049.

[29] J. Huang, J. Zhang, N. Chen *et al.*, "Design and implementation of experimental teaching in software engineering economics," *Frontiers in Educational Research*, vol. 6, no. 25, 2023.

[30] A. F. De Oliveira Colares, J. C. C. Furtado, and S. R. B. Oliveira, "Content and skills for teaching software process improvement in the computer science course: A mapping of acm / ieee, sbc, swebok, cmmi and mr-mps-sw assets," in *2023 IEEE Frontiers in Education Conference (FIE)*, 2023, pp. 1–8.

[31] S. Hanna, H. Jaber, A. Almasalmeh, F. A. Jaber *et al.*, "Reducing the gap between software engineering curricula and software industry in jordan," *Journal of Software Engineering and Applications*, vol. 7, no. 07, p. 602, 2014.

[32] F. Al-Zaghoull, A. Hudaib, and M. Ahed, "Software engineering education in jordan," in *2014 6th International Conference on Computer Science and Information Technology (CSIT)*. IEEE, 2014, pp. 127–132.

[33] C. Watson, K. Blincoe *et al.*, "Attitudes towards software engineering education in the new zealand industry," in *28th Annual Conference of the Australasian Association for Engineering Education (AAEE 2017)*, vol. 785. Australasian Association for Engineering Education Sydney, 2017.

[34] M. Barr and S. W. Nabi, "The development of students' employability skills on a work-based software engineering degree programme," in *2022 IEEE Frontiers in Education Conference (FIE)*. IEEE, 2022, pp. 1–9.

[35] M. Marques, S. F. Ochoa, and M. C. Bastarrica, "Software engineering education in chile-status report," in *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*, 2016, pp. 180–185.

[36] D. Akdur, "Analysis of software engineering skills gap in the industry," *ACM Transactions on Computing Education*, vol. 23, no. 1, pp. 1–28, 2022.

[37] T. Hynninen, A. Knutas, and M. Hujala, "What can we learn from recommendations of early-career engineers? assessing computing and software engineering education using a career monitoring survey," in *Proceedings of the 2022 Conference on United Kingdom & Ireland Computing Education Research*, 2022, pp. 1–7.

[38] W. Groeneveld, J. Vennekens, and K. Aerts, "Identifying non-technical skill gaps in software engineering education: What experts expect but students don't learn," *ACM Transactions on Computing Education (TOCE)*, vol. 22, no. 1, pp. 1–21, 2021.

[39] M. Shkoukani, "Proposed model to find the gap between academic supply and industry demand in software engineering field in jordan," *International Journal of Advanced Computational Engineering and Networking, ISSN (p): 2320*, vol. 2106, 2013.

[40] R. Kauppinen, A. Lagstedt, J. Lindstedt, and O. Rainio, "Software engineering education with industry-mentored projects," *PACIS 2022 Proceedings*, 2022.

[41] K. Hartmann and K. Zhu, "Incorporating learning by doing into the software engineering curriculum," *DEStech Trans. Soc. Sci. Educ. Hum. Sci.*, no. eshd, Jan 2017.

[42] V. Garousi, G. Giray, and E. Tuzun, "Understanding the knowledge gaps of software engineers," *ACM Trans. Comput. Educ.*, vol. 20, no. 1, Nov 2019.

[43] V. Garousi, G. Giray, E. Tuzun, C. Catal, and M. Felderer, "Closing the gap between software engineering education and industrial needs," *IEEE Softw.*, vol. 37, no. 2, pp. 68–77, Mar 2020.

[44] V. Garousi, G. Giray, E. Tüzün, C. Catal, and M. Felderer, "Aligning software engineering education with industrial needs: A meta-analysis," *J. Syst. Softw.*, vol. 156, pp. 65–83, Oct 2019.

[45] L. H. Silva, R. X. Castro, and M. C. Guimaraes, "Supporting real demands in software engineering with a four steps project-based learning approach," in *2021 IEEE/ACM 43rd Int. Conf. Softw. Eng. Softw. Eng. Educ. Train.*, May 2021, pp. 50–59.

[46] G. Liargkovas, A. Papadopoulou, Z. Kotti, and D. Spinellis, "Software engineering education knowledge versus industrial needs," *arXiv:2112.12834v1*, 2021.

[47] C. Gupta and V. Gupta, "C4 skills in the engineering graduate: A study to align software engineering education with market-driven software industry needs," *IEEE Transactions on Education*, vol. 67, no. 1, pp. 31–43, 2024.

[48] M. Alghamlas and R. Alabduljabbar, "Predicting the suitability of it students' skills for the recruitment in saudi labor market," in *2019 2nd International Conference on Computer Applications & Information Security (ICCAIS)*. IEEE, 2019, pp. 1–5.

[49] M. Almaliki, "Software engineering in saudi arabia: A bibliometric assessment," *IEEE Access*, vol. 9, pp. 17 245–17 255, 2021.

[50] F. S. Altuwaijri and M. A. Ferrario, "Awareness and perception of agile in saudi software industry," in *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS)*. IEEE, 2021, pp. 10–18.

Mohammad Zarour holds a Ph.D. in Software Engineering since June 2009 from École de Technologie Supérieure (ETS) – Université du Québec (Montréal, Canada). A master degree in Computer Science in 1998 from University of Jordan. He is currently a faculty member at Hashemite University, Jordan. He has over twelve years of experience in teaching in a university environment including (Petra Unviersity, Jordan, and Prince Sultan Unviersity, KSA). Dr. Zarour worked as a chief Technical Advisor at one of the UNDP programmes in King AbdulAziz City of Science and Technology in Riyadh for 3 years. He also has several years of industry experience in information systems development and process improvement. His research interests include software process quality, software product quality, cost estimation, data mining and UX. He has tens of peer-reviewed publications.

Mamdouh Alenezi is a Professor, a prominent figure in software engineering and artificial intelligence, leads the AI Academy at Tahakom, a key player in Saudi Arabia's tech industry. His pioneering research in AI and software engineering has had a transformative impact, driving innovation and positive change across sectors globally. Professor Alenezi's work bridges the gap between technology and society, emphasizing excellence, innovation, and societal improvement.

Mohammed Akour is a Professor in Software Engineering. He got his Bachelor (2006) and Master (2008) degree from Yarmouk University in Computer Information Systems with Honor. He joined Yarmouk University as a Lecturer in August 2008 after graduating with his master in Computer Information Systems. He joined Yarmouk University again in April 2013 after graduating with his PhD in Software Engineering from NDSU with Honor. He serves as Keynote Speaker, Organizer, a Co-chair and publicity Chair for several IEEE conferences, and as ERB for more than 10 ISI indexed prestigious journals. He is a member of the International Association of Engineers. (IAENG). Dr. Akour at Yarmouk University served as Head of accreditation and Quality assurance for two years and then was hired in 2017 as director of computer and Information Center. In 2018, Dr. Akour was hired as vice Dean of Student Affairs at Yarmouk University. In 2019, Dr. Akour got a sabbatical leave and joined Al Yamamah University -Riyadh Saudi Arabia- as an associate professor in Software Engineering, he served as a Software Engineering program coordinator. In 2021, Dr. Akour joins Prince Sultan University as professor in Software Engineering. Akour served as Computer Science department chair 2023-2024, and now he is the Software Engineering department chair.