T. MAAYUF

ATILIM UNIVERSITY 2024

UNDERSTANDING THE SKILL GAPS OF SOFTWARE ENGINEERING STUDENTS

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
ATILIM UNIVERSITY

TALAL MOHAMED MAAYUF

A MASTER OF SCIENCE THESIS
IN
THE DEPARTMENT OF COMPUTER ENGINEERING

JUNE 2024

UNDERSTANDING THE SKILL GAPS OF SOFTWARE ENGINEERING STUDENTS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
ATILIM UNIVERSITY

BY

TALAL MOHAMED MAAYUF

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
THE DEPARTMENT OF COMPUTER ENGINEERING

JUNE 2024

Approval of the Graduate School of Natural and Applied Sciences, Atilim University.

Prof. Dr. Ender KESKİNKILIÇ
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of **Master of Science in Computer Engineering, Atılım University**.

Prof. Dr. Gökhan ŞENGÜL
Head of Department

This is to certify that we have read the thesis UNDERSTANDING THE SKILL GAPS OF SOFTWARE ENGINEERING STUDENTS submitted by TALAL MAAYUF and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Asst. Prof. Dr. Damla TOPALLI
Supervisor

Examining Committee Members:

Assoc. Prof. Dr. Cansu Çiğdem EKİN
Computer Engineering Department, Atılım University

Asst. Prof. Dr. Damla TOPALLI
Computer Engineering Department, Atılım University

Prof. Dr. Nergiz Erçil ÇAĞILTAY
Software Engineering Department, Çankaya University

Date: *24.06.2024*

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name : TALAL MOHAMED MAAYUF

Signature :

ABSTRACT

UNDERSTANDING THE SKILL GAPS OF SOFTWARE ENGINEERING STUDENTS

Maayuf, Talal

Department of Computer Engineering

Supervisor: Asst. Prof. Dr. Damla Topallı

June 2024, 85 pages

The fast development in the software engineering industry means that the academic institutions need to keep their curricula up-to-date to meet the changing demands of the job market. This thesis investigates the gap between what software engineering students learn in the university and the skills they need to possess in the software engineering field. The focus is on the emerging technologies like Artificial Intelligence (AI), Machine Learning (ML), Cloud Computing and DevOps, as well as established guidelines like the Software Engineering Body of Knowledge (SWEBOK). This thesis takes a deep dive into the university's software engineering curriculum. It compares the curriculum with SWEBOK standards and analyzes job postings to identify the most demanded skills in the software engineering industry. Additionally, a survey was conducted with senior software engineering students to understand how prepared they feel to tackle the challenges of today's tech landscape. The findings reveal significant gaps between the skills taught in the university and those required by employers. Particularly, emerging technologies and soft skills are underrepresented in the curriculum. The analysis provides practical recommendations for updating the curriculum to better align with industry needs. By making these improvements, educational institutions can better prepare their graduates for the workforce, increasing their employability and readiness to contribute effectively to the software engineering field. This thesis aims to contribute to the ongoing conversation about curriculum development in software engineering education. It offers a model for aligning academic programs with industry expectations, emphasizing the need for adaptive curricula that evolve with technological progress. This ensures that graduates possess the relevant skills needed to succeed in a competitive job market.

Keywords: Skill Gap, Software Engineering Education, Software Engineering Industry, Emerging Technologies

ÖZ

YAZILIM MÜHENDİSLİĞİ ÖĞRENCİLERİ İÇİN SEKTÖR AKADEMİ ARASINDAKİ BOŞLUĞUN ANLAŞILMASI

Maayuf, Talal

Bilgisayar Mühendisliği Bölümü

Tez Yöneticisi : Dr. Öğr. Üyesi Damla Topallı

Haziran 2024, 85 sayfa

Yazılım mühendisliği endüstrisindeki hızlı gelişmeler, akademik kurumların müfredatlarını iş piyasasının değişen taleplerini karşılamak için güncel tutmalarını gerektirmektedir. Bu tez, üniversitede yazılım mühendisliği öğrencilerinin öğrendikleri ile yazılım mühendisliği alanında sahip olmaları gereken beceriler arasındaki farkı araştırmaktadır. Odak noktası, Yapay Zeka (AI), Makine Öğrenimi (ML), Bulut Bilişim ve DevOps gibi gelişen teknolojiler ile Yazılım Mühendisliği Bilgi Tabanı (SWEBOK) gibi yerleşik yönergeler üzerinedir. Bu çalışmada bir üniversitenin yazılım mühendisliği müfredatını derinlemesine incelenmiştir. Çalışma ilgili müfredatı SWEBOK standartları ile karşılaştırmakta ve yazılım mühendisliği endüstrisinde en çok talep edilen becerileri belirlemek için iş ilanlarını analiz etmektedir. Ayrıca, günümüz teknoloji ortamının zorluklarıyla başa çıkmaya ne kadar hazır olduklarını anlamak için son sınıf yazılım mühendisliği öğrencileri ile bir anket yapılmıştır. Bulgular, üniversitede öğretilen beceriler ile işverenlerin talep ettiği beceriler arasında önemli boşluklar olduğunu ortaya koymaktadır. Özellikle, gelişen teknolojiler ve yumuşak beceriler müfredatta yetersiz temsil edilmektedir. Analiz, müfredatı endüstri ihtiyaçları ile daha iyi uyumlu hale getirmek için pratik öneriler sunmaktadır. Bu iyileştirmeler yapıldığında, eğitim kurumları mezunlarını iş gücüne daha iyi hazırlayabilir, istihdam edilebilirliklerini ve yazılım mühendisliği alanına etkili bir şekilde katkıda bulunma kapasitelerini artırabilir. Bu tez, yazılım mühendisliği eğitiminde müfredat geliştirme konusundaki süregelen tartışmalara katkıda bulunmayı amaçlamaktadır. Akademik programların endüstri beklentileri ile uyumlu hale getirilmesi için bir model sunmakta ve teknolojik ilerlemelerle birlikte evrilen uyumlu müfredatların önemini vurgulamaktadır. Sonuçların mezunların rekabetçi iş piyasasında başarılı olmak için gerekli olan ilgili becerilere sahip olmalarını sağlamaya katkısı olacağı beklenmektedir.

Anahtar Kelimeler: Beceri Boşluğu, Yazılım Mühendisliği Eğitimi, Yazılım Mühendisliği Sektörü, Gelişen Teknolojiler

To my family

ACKNOWLEDGMENTS

I wish to extend my heartfelt gratitude to my supervisor, Asst. Prof. Dr. Damla TOPALLI, for her unwavering support, invaluable guidance, and insightful feedback throughout the course of my thesis. Her encouragement and expertise have been pivotal in shaping this work.

I am also deeply thankful to the members of my thesis examination committee, Assoc. Prof. Dr. Cansu Çiğdem EKİN and Prof. Dr. Nergiz ÇAĞILTAY, for their valuable input and constructive criticism, which have greatly enhanced the quality of this thesis.

Embarking on this academic journey after nearly 20 years in the workforce was not an easy decision. I am profoundly grateful to my family for their unwavering love and support throughout this challenging endeavor. Their belief in me has been a constant source of motivation and strength, enabling me to navigate and succeed in this academic pursuit while shaping both my research endeavors and personal growth.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS/ABBREVIATIONS

SWEBOK	Software Engineering Body of Knowledge
AI	Artificial Intelligence
ML	Machine Learning
IoT	Internet of Things
PBL	Project Based Learning
SPSS	Statistical Package for the Social Sciences

CHAPTER 1

INTRODUCTION

This chapter provides an overview of the study, outlining the key motivations and background that drive this research. It delves into the industry-academia gap in software engineering education, highlights the significance of the Software Engineering Body of Knowledge (SWEBOK), and sets the stage for the research objectives and methodology.

1.1 Background and Motivation

This section explores the context and reasons behind the study. It discusses the existing disparity between academic curricula and industry requirements in software engineering, emphasizing the need to align educational programs with emerging technological trends.

1.1.1 Industry-Academia Gap

The misalignment of university curricula and industry needs in software engineering is a long-standing problem. Specially, if we take a look into emerging technologies like AI, Machine Learning (ML), Big Data, Cloud Computing, Blockchain and DevOps[1], [2]. The rapidly evolving landscape of the software engineering profession has historically posed a challenge for academia, the software engineering industry is very dynamic and we have new technologies, platforms or methodologies on a yearly basis. And hence, graduates are frequently illiterate in practice even after receiving excellent theoretical background [3], [4].

Indeed, other studies have also pointed out the specific locations where this educational discrepancy is most apparent. One example is noted by this study of Oguz D and Oguz K, while universities do provide a solid theoretical basis, they often lack in being able to deliver realistic projects that are representative for industry contexts [1]. The disparity means that graduates may be missing critical, soft skills such as how to function well in a team, experience with project management and the ability to cope when confronted with real-world software development challenges [1] In addition, existing academic programs do not fully sit the industry's need for skills in new technologies [5]. Slow When it Comes to Technology, many studies have found that courses fail to keep pace with new trends in technology – such as Artificial Intelligence, Machine learning, Big data analytics and Cloud computing [2]. This delay in the updating of curricula thus widens this gap leaving graduates partially trained and unable to apply up-to date knowledge and skills [2], [6].

1.1.2 Importance of SWEBOK

Software Engineering Body of Knowledge (SWEBOK) is important, since it helps to identify and fill in these gaps within software engineering education. A standard way to define the core knowledge areas required for the profession of software engineering is given in SWEBOK. It provides a reference for the development of curricula to assure that educational programs address all essential aspects of software engineering [7]. Its target market is everywhere software engineering is done, and its purpose is to promote the vision of a well-formed discipline (Boundaries of Software Engineering with respect to other fields such as computer science, management training in project management or risk and mathematics). It characterizes what can be expected from software engineers, defining a frame of reference for both curriculum development and professional certification [7].

SWEBOK seeks to provide topical access to the Software Engineering Body of Knowledge thereby making it easier for readers, educators and practitioners to view software engineering as a discipline and further enable better theory building in research. This would provide a more systematic gateway to fill the gap between academic training and Industry needs as knowledge acquired by students is expected to be current and relevant for professional practices [7]. It seems as if just by designing the syllabus in tune with what SWEBOK lays down might help various academic institutions to bridge the gap between their curricula and industry requirements. It allows the alignment of skills and knowledge imparted in software engineering education worldwide, so it not only addresses standardization issues but can also elevate employability by filling up gaps/supplying finally demanded by industry like capabilities [3], [7].

1.2 Research Objectives

1.2.1 Gap Analysis

The aim of this study and the main goal of this investigation is to explore the specific knowledge gaps and skills in current curricula in Software Engineering. In fact, how this misalignment is significant between what academia teaches and software engineering industry needs are well documented [1], [3] Moreover, some studies indicate that although the theoretical knowledge is covered as needed and effectively, practical skills or real-life scenarios are generally not sufficient 9, 5. For example, students are taught about data structures and algorithms but there is no opportunity for them to practice those concepts in large-scale software projects that mimic industry standards [3], [8]. Therefore, it is important to bridge the gap in order to boost the employability of graduates.

1.2.2 Emerging Technologies Focus

Another primary goal would be to drive home the importance of integrating new technologies into their curriculum. Artificial Intelligence, Machine Learning, Big Data, Cloud Computing, Blockchain and DevOps which are the already leading technologies in the software engineering field. Nevertheless, mainstream academic programs are lagging far behind in incorporating these advancements into their offerings [2], [5]. This research also aims to highlight the imperative of revising curricula with these technologies in mind so that graduates receive preparation appropriate for current and future industry needs [2], [6] .

1.2.3 Data Sources and Methodology

A comprehensive gap analysis will be conducted using multiple data sources including academic curricula, job advertisements and student surveys. first, the analysis of academic curricula will focus on analyzing software engineering programs to determine how effectively they teach required skills and knowledge as required by the industry [3], [8] . Then, we will analyze job postings to identify what skills and technologies that software engineering graduates should have for them to be more attractive in the market [2], [6] . Senior students will also be surveyed to ascertain what skills and knowledge they believe that they have gained during their course of study compared with those which are needed in the workforce [3], [6] . Both qualitative and quantitative methodologies will be used to analyze this data. All heavy with qualitative research, will have thematic analysis of curricula and job postings Qualitative Analysis Quantifies gaps in survey data using highly methodical statistical methods [2], [6], [8].

1.3 Structure of the Thesis

This thesis is organized as follows: in Chapter 2 literature review section examines the current state of software engineering education, the integration of SWEBOK, and analyzes industry requirements and emerging technologies. It discusses the gaps identified between academia and industry and provides recommendations for curriculum improvement. In Chapter 3, methodology of the study is presented the research objectives and questions are defined, data collection from academic curricula, job postings, and student surveys as well as the type of qualitative and quantitative analysis used. In Chapter 4, the results of an analysis of academic curricula, job market analysis and survey data are provided. Chapter 5 interprets the findings, discussing what these might mean for academia and industry including implications. Finally, in Chapter 6 Conclusion, the main findings of the project are summarized, practical tips for curriculum development are given, reflection is made on how significant our research was and future work is discussed.

CHAPTER 2

LITERATURE REVIEW

This chapter examines existing literature to provide a comprehensive understanding of the current state of software engineering education, industry requirements, and emerging technologies. It identifies gaps between academic training and industry needs and offers recommendations for curriculum improvement.

2.1 Current State of Software Engineering Education

This section reviews the traditional curriculum content in software engineering programs and the use of SWEBOK in education. It highlights the strengths and weaknesses of current educational practices in preparing students for the software engineering profession.

2.1.1 Traditional Curriculum Content

Most traditional curricula in software engineering are designed to prepare the graduate professionally with a balanced knowledge base between both theoretical and practical aspects of computing. The course content is usually designed to cover large components in the field, which could include programming languages and data structures and algorithms as well computer architecture, operating systems, databases principles, software engineering practices and network systems [1], [8].

The main goal of this curriculum is to teach the basics of programming knowledge and skills onto students, setting the groundwork for further software development and engineering theory. For example, programming languages like Java, C++ and Python are taught in which is essential for creating software. They also learn algorithms and data structures, which are essential in creating efficient code. Undergraduate courses in computer architecture and operating systems offer the essential knowledge of software hardware interface allowing students to find better performance or even handle system level problems [1], [8].

Nevertheless, as thorough as that information is, the traditional curricula often lack in several regards. Major misses in detail, however, are it put too little emphasis on theoretical applications and real life examples. Needless to say, theoretical knowledge is important but countless programs exist that overemphasize this and underexpose students on much-needed practical skills for the software engineering workplace. This lack of coverage is especially problematic as the technology landscape evolves with new domains like AI and ML, IoT, Big Data, Blockchain to name a few [1], [9], [10].

2.1.2 Use of SWEBOK in Education

Software Engineering Body of Knowledge (SWEBOK) is a vital resource that seeks to standardize the knowledge areas in software engineering. Commonly used in academic courses as an assurance that course curricula are well rounded and cover a broad understanding of the subject. SWEBOK structure the knowledge areas of software requirements, design, construction, testing, maintenance, configuration management and engineering process as well as tools and methods [5], [7].

Universities adopt SWEBOK in their programs to get the students on a common platform where everyone learns about software engineering concepts in a structured format. In turn, this helps ensure educational outcomes more closely align with market requirements and increases the likelihood of graduate employability. This is exemplified from the focus in SWEBOK on software requirements and design, which recognizes that the ability to accurately capture and maintain user needs is an industry-critical skill [1], [5], [7], [9].

In addition, SWEBOK introduces students to the various aspects of software testing and maintenance which is a next important practice once the system is deployed and being used in operational purpose. These are often underrepresented in traditional curricula, which would usually focus more on the development phase. When integrating SWEBOK, educational programs provide a more comprehensive and well-rounded view of software engineering lifecycle [1], [7], [10], [11].

SWEBOK has its advantages, but it is not that easier to implement in academic programs. One of its biggest concerns today is the gap between the theoretical aspects as laid down in SWEBOK and how the industry requires such skills in practical real-life situation. If there is a caution about the body of knowledge it is exactly that— as valuable and rich in depth theory as SWEBOK may be, it does not always translate to practical real world competencies that students can leverage to execute their job role. This suggests that a gap between the software engineering theory as captured by SWEBOK and the practice of industry needs to be addressed with real-world projects in academic programs [1], [5], [7], [10], [11].

2.2 Industry Requirements and Emerging Technologies

This section explores the demand for emerging technologies in the software engineering industry and presents case studies of successful curriculum integration.

2.2.1 Demand for Emerging Technologies

Recent years have seen an explosion in the volume of emerging technologies that are being demanded by software engineering. Analysis of Job Postings and Industry Surveys Point to Skills Shortage in AI, ML, Big Data, Cloud Computing, Blockchain and DevOps Due to their disruptive nature they are impacting industries and creating new pain points that have increasing demand for professionals capable of working in these business areas [10], [11], [12], [13].

As an example, most job postings include AI and ML as integers reflecting the fact that intelligent systems are being developed & every decision now is backed by data. Like python, Big Data skills are in high demand as more and more companies are finding the necessity of processing large amounts of data. As many companies pushing their architecture to cloud, the Cloud Computing skills are also in high demand — no organization want to build and maintain infrastructure in parallel which should be scalable and flexible enough on failover /unpredictable situations. Another Blockchain technology: with secure transaction and decentralized systems is also providing the high job potential in future. [5], [8], [9], [14].

Similarly, DevOps is designed to encourage communication links between development and IT operations enabling better software quality for a higher degree of continuous delivery. Nowadays, these technologies have been integrated into software engineering practices in a way that, to contemporary software engineers; they are mandatory knowledge [1], [3], [7], [9], [15].

2.2.2 Case Studies

Many universities have implemented new technologies into their curricula, but these implementation successes provide valuable case studies and demonstrate best practices for curriculum development. For instance, the collaboration of Technical University of Munich (TUM) with the industry has enabled TUM to offer AI/ML courses co-taught by industrial experts, enabling their students to get concrete and up-to-date knowledge in this data intensive field, etc [1], [9].

This is clearly an improvement, and one obvious example of how things could be different can again be drawn from the United States, the University of California Berkeley has a complete range of courses – including Big Data and Cloud Computing. The courses are developed in conjunction with industry partners like Google and Amazon, ensuring that the academic content matches the needs in the market place. Through this consortium model, students will be ready and able to successfully meet the rigors of a highly competitive job market upon their graduation [3], [5], [12].

Also, the Blockchain technology had already made their appearance onto some software engineering programs at Carnegie Mellon University (CMU). Curriculum includes hands-on projects, industry collaboration with blockchain startups for real world experience. This method also improves employability by providing students with essential high demand skills [10], [11], [13] .

Here, these case-studies show how the industry and academia have to join hands for real-world curriculum development. Employing emerging technologies in their courses, and collaborating with practitioners can help institutions prepare graduate students to be able to function effectively within the ever-changing software engineering sector [1], [5], [9], [14], [15].

2.3 Identified Gaps Between Academia and Industry

This section identifies gaps between academic training and industry needs, focusing on technical and non-technical skills.

2.3.1 Technical Skills Gap

Software Engineering programs in academics fail to provide industry-relevant technical skills. This gap is more pronounced when it comes to upcoming technologies like AI, ML, Big Data, Cloud Computing, Blockchain and DevOps. Even though universities impart the best theoretical knowledge they could, students are often seen wanting for practical experience and industry-oriented hands-on skills that are a quintessential need in today's world. Courses like the fundamentals of programming languages and algorithms are taught in many academic programs, but it is rare for advanced subjects such as deploying machine learning models, managing cloud infrastructure or implementing a blockchain technology to be properly explained [4], [16].

The knowledge of industry standards tools like Docker for containerization, Jenkins for integration as well as AWS cloud services is evidently missing or not part of the academic curriculum which demonstrates a major technical skills gap [8] In addition, industry surveys have indicated that employers are seeking individuals who not only possess fundamental knowledge but also have some experience working with contemporary development environments and tools [6] .

Also the speed at which technology changes require that curricula are kept up to scratch with whatever skills industry needs. Most universities are unable to adapt their programs quickly to keep up with the help of new technologies and methodologies, leading many graduates that will be not really completely ready for today's market [17]. This gap implies that graduates might be strong in coding and theory, but they lag in using their skills to complex real-world problems like creating AI solutions at scale or managing data science projects [15] .

However, the training to what is now industry accepted software development methods and tools are noticeably lacking. It is necessary to combine practical training with sound theoretical knowledge in the universities for students' preparation before the technical requirements of the industry [11] .

2.3.2 Non-Technical Skills Gap

Non-technical are just as important as technical skills for success in the software engineering field. Many of these competencies, like teamwork, communication skills, problem-solving and project management are often missing or marginalized in higher education curricula even though employers place a high value on them. By the same token other studies found that proficiency is needed not just in digital communications tools and skills but equally in effective team working capability and ability to communicate complex concepts in simple terms [18] .

Most software development projects require some amount of teamwork. But fewer programs allow students to work together in that way, as they would need to do on the job. Such gap will also result in graduates with little experience on collaborative coding practices, Git, or agile methodologies which stress teamwork and iterative development [19].

We need decent communication skills especially when it comes to requirements gathering and stakeholder management and technical solution presentation to non-technical stakeholders. Graduates from degree programs have excellent technical skills but lack in soft-skills and communication experience for those working closely with customers or cross-functional teams [20].

Also, problem-solving and critical thinking are core skills, that are required for every aspect of software engineering. Although these skills are indirectly fostered in technical courses, direct formation and practice is often overlooked. Employers are looking for people who can solve complex problems, develop new ideas and face unexpected challenges; skills that cannot always be taught fully within a traditional academic framework [9] .

Increased recognition is also being given to soft skills including leadership and adaptability, which are similarly important in the realization of a successful career as a software engineer. Still, the curriculums gave software engineering programs low marks in those areas [21] .

Academic programs can be effective in cumulatively have graduates that address technical and non-technical skill gaps so that they are holistic professionals and will fit the multifaceted demands of a role in software engineering [22] .

2.4 Recommendations for Curriculum Improvement

This section offers strategies for improving software engineering curricula, including integrating emerging technologies, enhancing practical skills, fostering non-technical skills, and adopting innovative teaching approaches.

2.4.1 Integrating Emerging Technologies

In order to bridge the gap between what is taught in academia and what is actually used in that industry, it is necessary to make sure that latest emerging technologies must be included as part of the curriculum on AI, ML, Big Data etc. along with DevOps as well. There are number of strategies to achieve this integration. One way this can be achieved is by further extension universities need to leverage on the changing trends and update their course offerings accordingly that includes technology. Using specialized subjects that cover practical aspects of these technologies can serve to arm students with skills they need. As Nesterenko O. stresses in his study that the need to be better oriented along industry expectations by introducing new courses like AI, blockchain, and data visualization in curriculum [23].

Also, partnerships with industry could provide students with real and emerging industry technology platforms. They can be guest lecturers, internships, research projects and so on. Such efforts ensure that students remain current with the most recent industrial developments and even get experience first-hand of new technologies as well [22].

2.4.2 Enhancing Practical Skills

Providing students with practical hands-on experiences is equally important to sheer theoretical knowledge for their preparation for real-world work. This is being done by introduction of subject to new methodology like the use project based learning (PBL) in subjects. PBL is designed in such a manner that it lets student work on real-world problems, which helps them sharpen their problem-solving skills and get practical exposure. Ouhbi et al. in their study to the integration of PBL [17], due to the fact that it provides students with a better understanding of theoretical concepts being linked with their practical use.

Furthermore, requiring internships and co-op programs as part of the degree process can greatly increase students' hands-on experience. These courses help students to work in the industry, utilize their skills and learn professional culture efficiently. The paper "Software Engineering" identifies two important needs interns can fill: being a connection between academia and industry [20].

Fostering Non-Technical Skills

Technical training combined with ongoing development of soft skills empowers graduates to thrive in professional environments. Employers often speak of the value of soft skills like teamwork, communication and project management which are in surprisingly short supply within curricula. Universities ought to incorporate courses and activities which aim to provide these skills in their curriculum.

Integrating soft skills development with existing course work is a great way to do so. Such project-based courses can include teamwork and communication exercises, which forces students to learn these skills while being directly involved with those in need of improvement. The developers' soft skills form part of the job market's expectations, and integrating them into software engineering graduates' studies is recommended by Qamar et al. in their research [18].

In addition, universities can hold workshops and seminars on leadership, presents negotiation, communication. Some of these sessions could be carried out by people in the industry who can give individual guidance and an idea of what is expected from the workplace. For example, Hills et al states that initiatives like these can help develop students´ softkills for a successful career future [19].

2.4.3 Additional Recommendations

Innovative teaching approaches like flipped classrooms and blended learning also need to be considered by universities to make the experience of learning a pleasant one. Such practices are assumed to offer increased interactivity as a remote learning environment, and are mentioned in a research by Garouso et al. [24]. Finally, the implementation of permanent feed-back from the industry ensures that an education programme always stays up to date with respect to the particular field [25].

With these strategies in place, academic programs will produce graduates that not only have the technical abilities but also the hands-on and soft skills necessary for success in software engineering.

CHAPTER 3

METHODOLOGY

This chapter details all the steps involved in this study including how data was collected, analyzed and interpreted. This section will outline the research design, methods of data collection and analysis, including but not limited to information on participants and data-generating instruments as well as procedures. The goal is to make the study systematic, reproducible and transparent. The methodology of this thesis is presented in Figure 3.1.

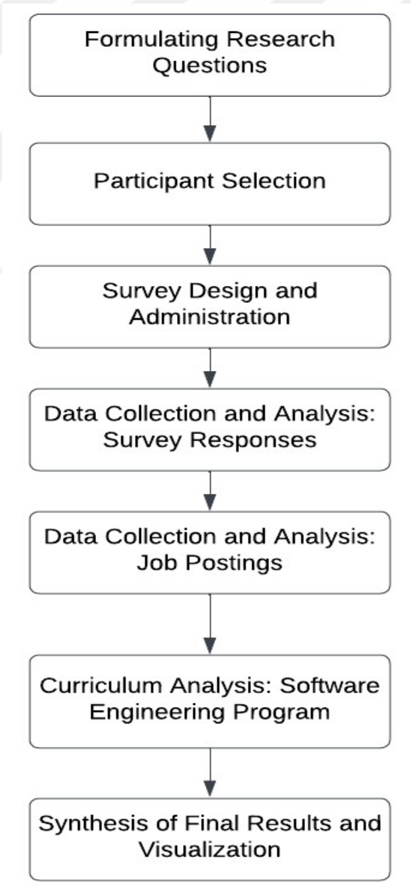

Figure 3.1 Methodology Process

3.1 Research Questions:

The research questions addressed within this study include the following:

RQ1. How can the software engineering curriculum in a university provide students with skills they can immediately put to use in software development jobs?

RQ2. How do senior software engineering students perceive their readiness to navigate the difficulties arising from new technologies in software engineering sector?

RQ3. What new technical skills and competencies, especially in newly released technologies, are most commonly requested in software engineering job advertisements and how well does that align with the curriculum at a university?

RQ4. How might the software engineering curriculum be changed or added to, in order to more accurately reflect the demands that software engineering will have on our graduates when they enter the job market a few years from now?

3.2 Research Design:

This study uses mixed method to understand the skill gap between academic curricula and industrial requirement in software engineering by using both quantitative and qualitative data. The following are the key elements of the Study Design:

- *Survey:* To capture quantitative information on student exposure to SWEBOK knowledge areas, technical skills preparedness and soft skills preparedness.
- Curriculum analysis: To evaluate the alignment of software engineering curriculum with both SWEBOK knowledge areas and industrial needs.
- Job Postings: What are the skills and knowledge areas required in entry level software engineering job postings?

3.3 Participants

The experiment consists of the final year students studying software engineering at a University in Turkey. These students were selected as they are close to being graduated and hence have a better sense of industry preparedness. Demographics of Participants are as follows;

Year of Study: 3rd and 4th year undergraduate engineering students

Major: Mostly Software Engineering Department

3.4 Data Collection Methods

- **Surveys**: A structured survey containing questions regarding familiarity with SWEBOK knowledge areas, technical skills preparedness and soft skills preparedness was given to students. The survey was comprised of Likert scale questions as well as open-ended questions to allow for both quantitative and qualitative data.
  - **SWEBOK Knowledge Areas Familiarity**: Students Student assessed their knowledge on a scale from 1 (Not at all familiar) to 5 (Extremely familiar) for various SWEBOK knowledge areas.
  - **Technical Skills Preparedness**: Students assessed how well they were prepared in key technical skills from 1 (Not at all prepared) to 5 (Extremely well prepared).
  - **Soft Skills Preparedness**: Students assessed how well they were prepared in key soft skills from 1 (Not at all prepared) to 5 (Extremely well prepared).
- **Curriculum Analysis**: The software engineering course curriculum at Atilim University was analyzed for the matching with SWEBOK areas. This included analysis of course content, syllabi and learning objectives.
- **Job Postings Analysis**: Entry-level software engineering jobs postings from LinkedIn job portal were collected, these postings were analyzed to determine the skill set demanded in greatest frequency and also areas of knowledge. The analysis consisted of the classification of their reference and the frequency with which they appeared through skill and technology.

3.5 Instruments

- **Survey Questionnaire**: The survey instrument includes questions to capture familiarity with SWEBOK knowledge areas, technical skills preparedness, and soft skills preparedness.

- **Curriculum Documents:** Course syllabi, learning outcomes and program descriptions from the software engineering curriculum at Atilim University.
- **Job Postings**: Information scraped from job sites encompassing the skills, technologies and competencies necessary for positions as an entry-level software engineering.

3.6 Procedures

Survey Administration: The survey was administered to senior-year software engineering students; Data were collected through an online survey software to streamline the process of typing responses and summarizing them.

Curriculum Documentation: Curriculum documents were examined to explore the match between the content and SWEBOK knowledge areas. The analysis aimed to document gaps and strengths in the curriculum.

Job Posting Analysis: Job postings were scraped from LinkedIn. Posts were then categorized by topic to identify the most common requested skills and knowledge areas. These numbers indicate the frequency of each skill, technology that was listed to show what is in demand with industry.

Data Analysis:

- **Quantitative Data Analysis:** Descriptive statistics (mean, standard deviation) were used to summarize the survey responses. The data was analyzed using SPSS (Statistical Package for the Social Sciences) to identify areas where students felt more or less prepared
- **Qualitative Data Analysis:** Open-ended survey responses were analyzed using thematic analysis to identify common themes and insights related to curriculum gaps and strengths.
- **Curriculum and Job Postings Analysis:** The content of the curriculum was compared with the skills and knowledge areas extracted from job postings. This comparison allowed us to understand the alignment provided by our academic formation and where we could fill in the gaps towards industry needs.

The study also seeks to gain a broader perspective on the skill gaps by recommending how software engineering education can be improved taking these recommendations for curricular enhancements that are more strongly in line with industry needs.

CHAPTER 4

RESULTS

4.1 Participant Demographics

4.1.1 Year of Study

The majority of participants are in their 4th year of study as shown in the Figure 4.1, indicating that most respondents are nearing the completion of their programs and are likely more aware of their preparedness for the industry.

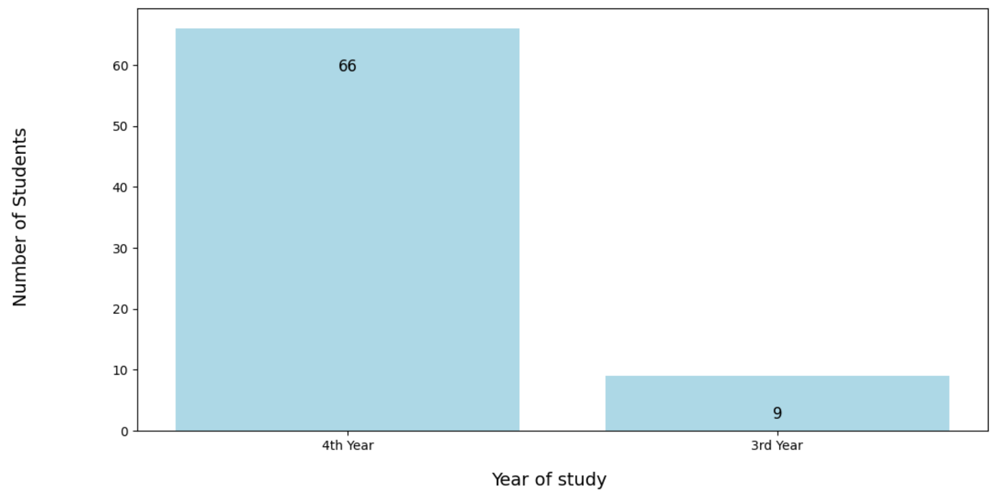

Figure 4.1Year of Study of the participants

4.1.2 Degree Program

Most participants are enrolled in the Software Engineering program as shown in Figure 4.2, with a few in Computer Engineering and even fewer in Electrical & Electronics Engineering who are double major students in Software Engineering. This focus on Software Engineering students aligns well with the aim of assessing preparedness for software engineering roles.

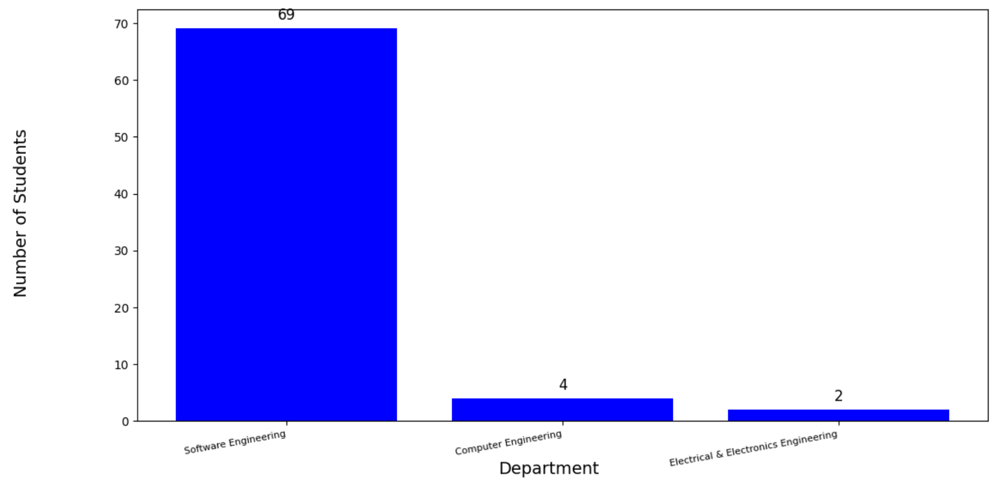

Figure 4.2 Degree Program of the participants

4.1.3 Internship Experience:

A significant majority (approximately 85%) of participants have had internship experience as shown in Figure 4.3, while about 15% have not had any internship experience. This high level of internship engagement suggests that most students have practical industry experience to complement their academic learning.

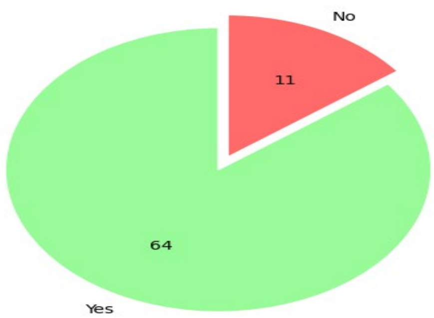

Figure 4.3 Internship Experience of the participants

4.2 Descriptive Statistics

4.2.1 SWEBOK Knowledge Area Familiarity

The data indicates that students generally feel moderately familiar with most SWEBOK knowledge areas as in Table 4.1 with higher familiarity reported in software design (Mean = 3.47) and Models and Methods (Mean = 3.32). Conversely, areas such as software testing (Mean = 2.92), software maintenance (Mean = 2.85), and configuration management (Mean = 2.96) exhibit lower familiarity levels. These findings suggest a need for greater emphasis on these areas within the curriculum to ensure comprehensive student preparedness.

Table 4.1 Descriptive statistics of SWEBOK Knowledge Area Familiarity

SWEBOK Knowledge Areas	N	Minimum	Maximum	Mean	Std. Deviation
Requirements Engineering	75	1	5	3.15	1.12
Software Design	75	1	5	3.47	0.99
Software Construction	75	1	5	3.17	1.14
Software Testing	75	1	5	2.92	1.10
Software Maintenance	75	1	5	2.85	1.10
Configuration Management	75	1	5	2.96	1.01
Engineering Management	75	1	5	3.21	0.92
Engineering Process	75	1	5	3.23	0.95
Models and Methods	75	1	5	3.32	0.98
Software Quality	75	1	5	3.21	1.08

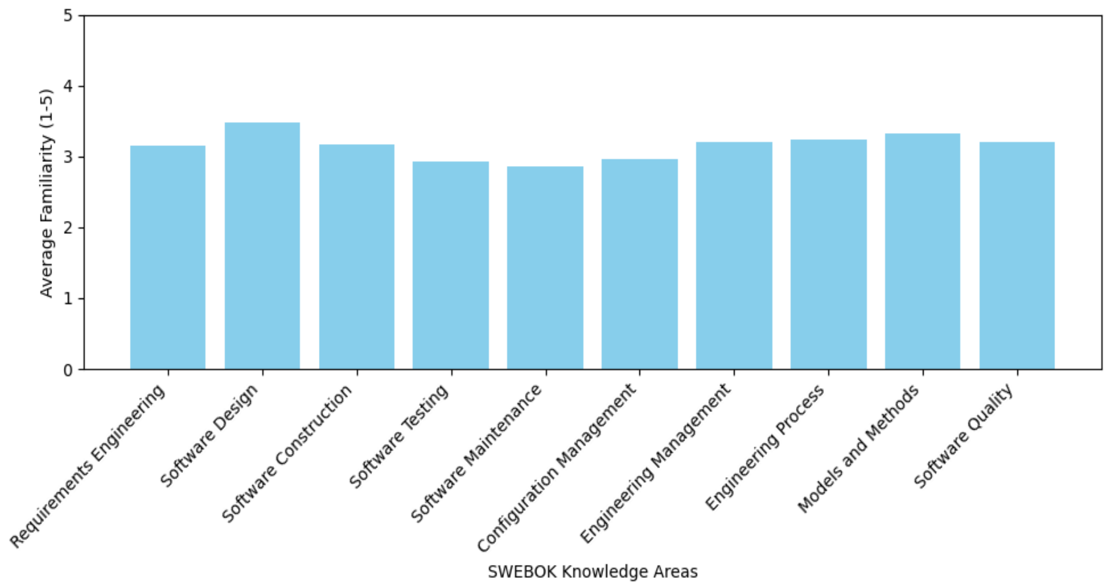

Figure 4.4 Average Familiarity with SWEBOK Knowledge Areas

4.2.2 Technical Skills Preparedness.

As seen in Table 4.2, students reported a high level of preparedness in core programming languages (Mean = 3.85) and object-oriented programming concepts (Mean = 3.78). However, lower levels of preparedness were observed in areas such as web development frameworks (Mean = 2.69), cloud computing (Mean = 2.05), and Basic understanding of AI/ML concepts (Mean = 2.94). These findings highlight the necessity for enhanced curricular focus on modern frameworks, cloud technologies, and AI/ML to align student skills with industry demands. Average preparedness in technical skills is given in Figure 4.5.

Table 4.2 Descriptive statistics Technical Skills Preparedness

Technical Skills: Level of Preparedness	N	Minimum	Maximum	Mean	Std. Deviation
Proficiency in core programming languages	75	2	5	3.85	0.78
Object-Oriented Programming (OOP) concepts	75	2	5	3.78	0.85
Version control systems (e.g., Git)	75	1	5	3.10	1.15

Table 4.2 (continued)

Web development frameworks (e.g., React, Angular)	75	1	5	2.69	1.34
Cloud Computing (e.g., AWS, Azure)	75	1	5	2.05	1.11
Basic understanding of AI/ML concepts	75	1	5	2.94	1.26

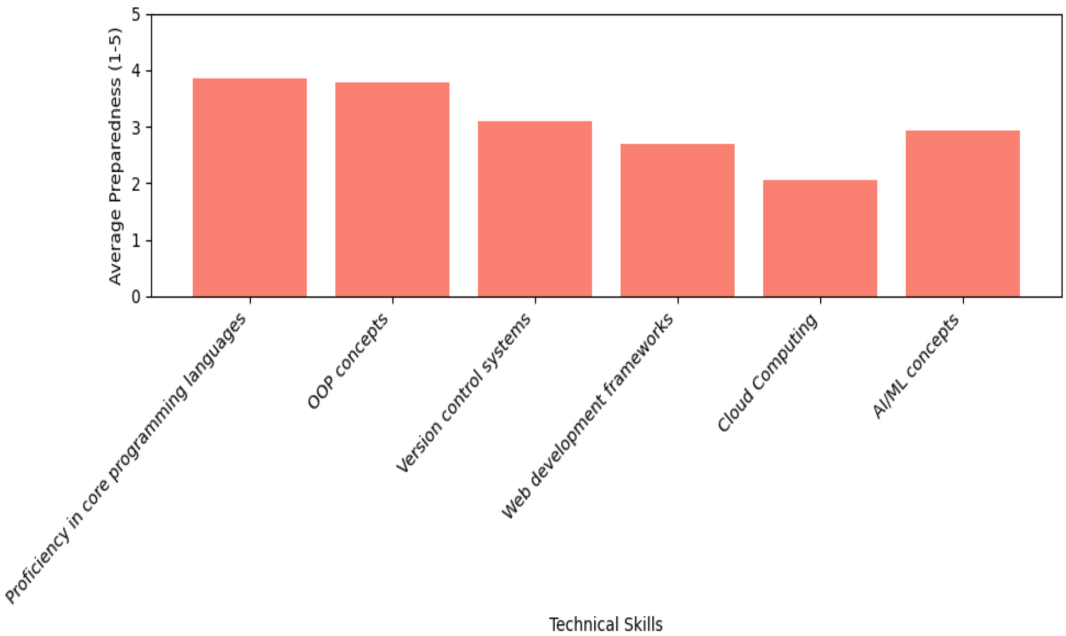

Figure 4.5 Average Preparedness in Technical Skills

4.2.3 Soft Skills Preparedness

The analysis indicates that students generally feel well-prepared in essential soft skills as in Table 4.3, particularly in teamwork and collaboration (Mean = 4.08), and problem-solving and critical thinking (Mean = 4.01). Written and oral communication skills (Mean = 3.79) and time management and adaptability (Mean = 3.91) also show strong levels of preparedness. These results suggest that the current curriculum adequately addresses the development of these critical soft skills, though continuous emphasis is necessary to maintain and enhance these competencies. Figure 4.6 shows descriptive statistics of soft skill preparedness.

Table 4.4 Descriptive statistics of Soft Skills Preparedness

Soft Skills: Level of Preparedness	**N**	**Minimum**	**Maximum**	**Mean**	**Std. Deviation**
Written and oral communication	75	2	5	3.79	1.02
Problem-solving and critical thinking	75	2	5	4.01	0.89
Teamwork and collaboration	75	2	5	4.08	0.93
Time management and adaptability	75	2	5	3.91	0.92

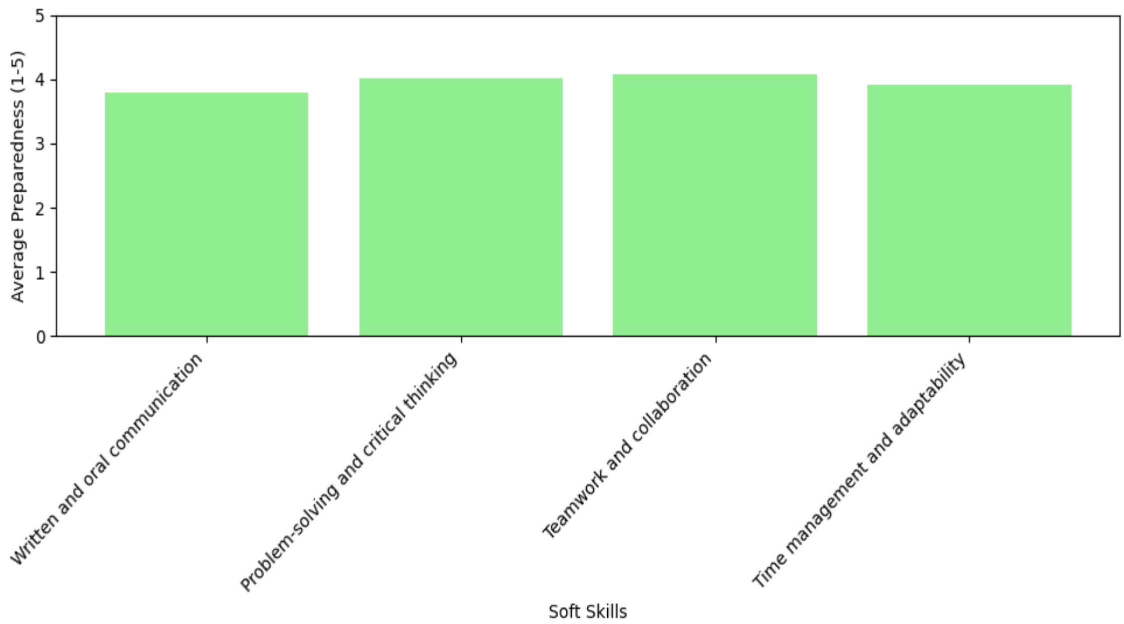

Figure 4.6 Average Preparedness in Soft Skills

4.2.4 Curriculum Alignment with SWEBOK Knowledge

The descriptive statistics reveal that the curriculum alignment with SWEBOK knowledge areas is generally moderate as in Table 4.4 and Figure 4.7. The mean values for most knowledge areas fall between 3.04 and 3.28, indicating a satisfactory alignment but with room for improvement. Notably, Software Testing (Mean = 2.84) and Software Maintenance (Mean = 2.77) have lower mean values, suggesting that these areas are less adequately covered in the curriculum. The variability in students' responses, as indicated by the standard deviations, highlights the need for a more consistent approach to integrating these topics into the curriculum.

Table 4.5 Descriptive statistics of Curriculum Alignment with SWEBOK Knowledge

SWEBOK Knowledge Areas	N	Minimum	Maximum	Mean	Std. Deviation
Requirements Engineering	75	1	5	3.04	1.10
Software Design	75	1	5	3.28	1.03
Software Construction	75	1	5	3.15	0.98
Software Testing	75	1	5	2.84	1.13
Software Maintenance	75	1	5	2.77	1.16
Software Configuration Management	75	1	5	2.92	1.08
Software Engineering Management	75	1	5	3.07	1.07
Software Engineering Process	75	1	5	3.17	1.02
Software Engineering Models and Methods	75	1	5	3.21	1.06
Software Quality	75	1	5	3.05	1.06

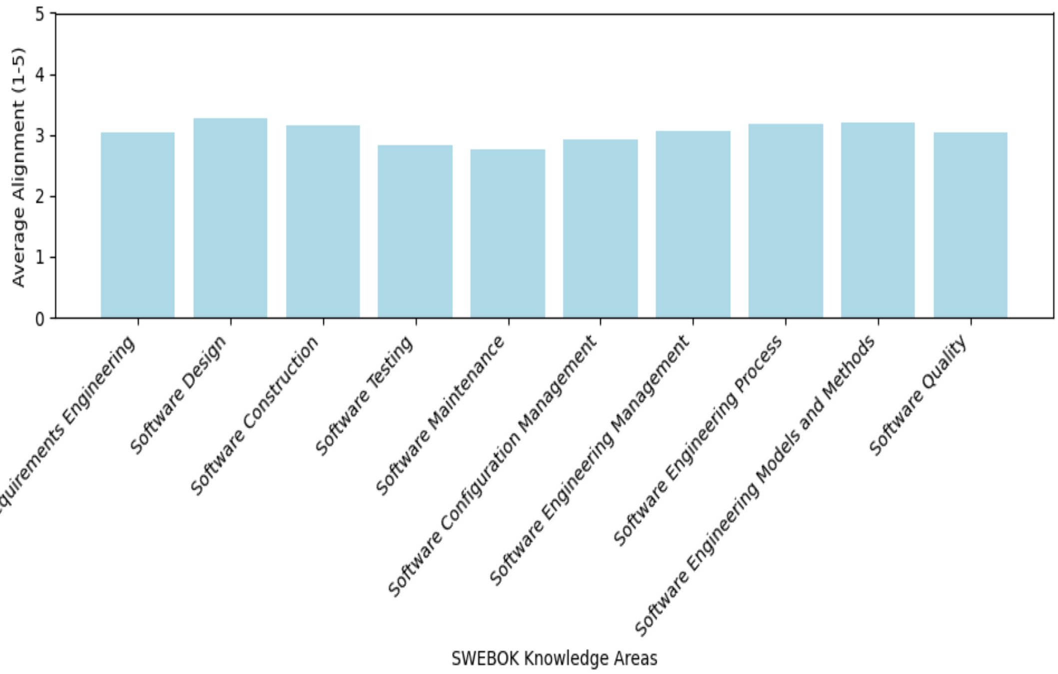

Figure 4.7 Curriculum Alignment with SWEBOK Knowledge

4.2.5 Experience Impact on Skill Development

The data in Table 4.5 shows that internships and work experience are viewed as the most effective in developing essential skills (Mean = 4.12), followed closely by internships and co-ops (Mean = 3.95) and personal projects (Mean = 3.92). Academic coursework and online courses are also considered helpful but to a slightly lesser extent. These findings underscore the importance of practical, hands-on experiences in complementing academic learning.

Table 4.6 Descriptive statistics of Experience Impact on Skill Development

Experience Impact	N	Minimum	Maximum	Mean	Std. Deviation
Internships and Coops	75	1	5	3.95	1.14
Academic Coursework	75	1	5	3.01	0.98
Personal Projects	75	1	5	3.92	1.01

Table 4.6 (continued)

Internships/Work Experience	75	1	5	4.12	1.01
Online Courses/Tutorials	75	2	5	3.4	0.90

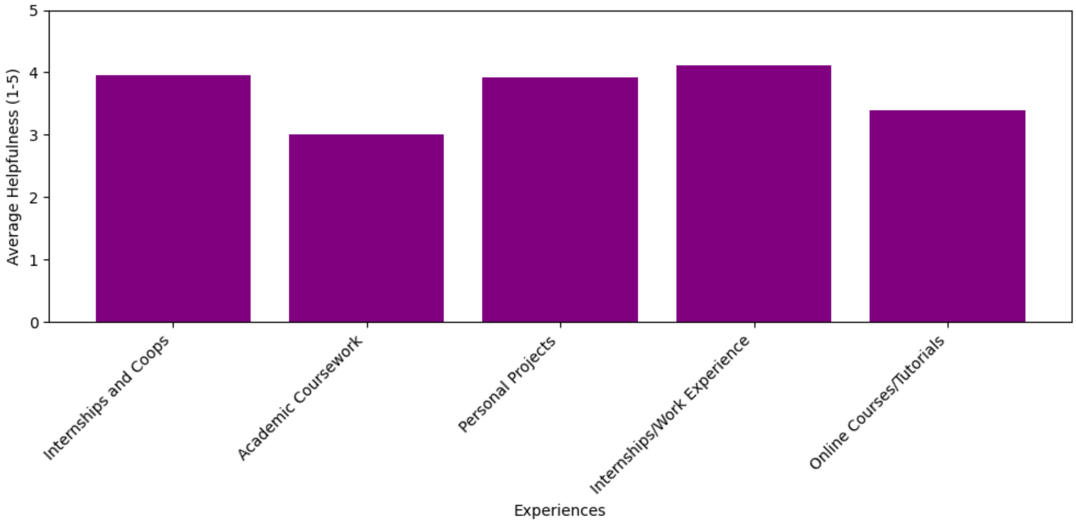

Figure 4.8 Experience Impact on Skill Development

4.2.6 Preparedness for Entry-Level Roles.

The mean value of 3.49 indicates that students feel moderately prepared for entry-level software engineering roles as stated in Table (4-9). The standard deviation of 0.79 suggests there is some variation in the responses, with some students feeling more prepared than others. This suggests that while the curriculum provides a solid foundation, there is still room for improvement in preparing students for the practical demands of the industry.

Table 4.7 Descriptive statistics of Overall Preparedness for Entry-Level

	N	Minimum	Maximum	Mean	Std. Deviation
Overall Preparedness	75	1	5	3.49	0.79

4.3 Analysis of the Impact of Internship Experience

4.3.1 The Impact of Internship Experience on Technical Skill Preparedness.

To determine the effect of internship experience on the perceived preparedness of technical skills, descriptive statistics were computed for each technical skill, reporting the means and standard deviations for both groups (students with internship experience and those without) as shown in Table 4.7. The results indicated that students with internship experience generally reported higher levels of preparedness in most technical skills, though the differences were not substantial.

To further investigate these differences, independent samples t-tests were conducted. Levene's test for equality of variances was performed to check the assumption of equal variances, with significant values ($p < 0.05$) indicating unequal variances. The t-tests compared the means of the two groups for each technical skill. As shown in the table (4-11) none of the t-tests revealed a significant difference between the groups (all p-values > 0.05), indicating no statistically significant difference in technical skill preparedness between students with and without internship experience.

Additionally, effect size measures were calculated, including Cohen's d, Hedges' g (corrected for small sample sizes), and Glass's delta. Hedges' g was selected as the most appropriate effect size measure due to its adjustment for small sample sizes, providing a more accurate estimate. As shown in the table (4-12) the effect sizes, as measured by Hedges' g, were generally small, with confidence intervals including zero, indicating a lack of practical significance in the differences observed. Specifically, for programming languages, Hedges' g was 0.127 (95% CI: -0.507, 0.760); for OOP concepts, Hedges' g was 0.439 (95% CI: -0.200, 1.074); for version control, Hedges' g was 0.185 (95% CI: -0.450, 0.818); for web development frameworks, Hedges' g was 0.128 (95% CI: -0.506, 0.761); for cloud computing, Hedges' g was -0.053 (95% CI: -0.686, 0.580); and for AI/ML basics, Hedges' g was -0.049 (95% CI: -0.682, 0.585).

In conclusion, the analysis demonstrates that there is no statistically significant impact of internship experience on the preparedness for technical skills such as programming languages, OOP concepts, version control, web development frameworks, cloud computing, and AI/ML basics. The effect sizes are small to negligible, indicating that any observed differences are not practically significant.

Table 4.8 Group Statistics of the Impact of Internship Experience on Technical Skill Preparedness

Internship Experience (Yes/No)		N	Mean	Std. Deviation	Std. Error Mean
Preparedness Skill Programming Languages	Yes	64	3.83	0.75	0.09
	No	11	3.73	1.01	0.30
Preparedness Skill OOP Concepts	Yes	64	3.83	0.79	0.10
	No	11	3.45	1.13	0.34
Preparedness Skill Version Control	Yes	64	3.13	1.15	0.14
	No	11	2.91	1.22	0.37
Preparedness Skill Web Development frameworks	Yes	64	2.72	1.34	0.17
	No	11	2.55	1.37	0.41
Preparedness Skill Cloud Computing	Yes	64	2.03	1.14	0.14
	No	11	2.09	0.94	0.29
Preparedness Skill AI/ML Basics	Yes	64	2.94	1.22	0.15
	No	11	3.00	1.55	0.47

Table 4.9 Independent Samples Test of the Impact of Internship Experience on Technical Skill Preparedness

		Levene's Test for Equality of Variances		t-test for Equality of Means				
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference

Table 4.9 (continued)

Preparedness Skill Programming Languages	Equal variances assumed	2.97	0.09	0.39	73.00	0.70	0.10	0.26
	Equal variances not assumed			0.32	11.95	0.76	0.10	0.32
Preparedness Skill OOP Concepts	Equal variances assumed	4.53	0.04	1.36	73.00	0.18	0.37	0.28
	Equal variances not assumed			1.06	11.73	0.31	0.37	0.35
Preparedness Skill Version Control	Equal variances assumed	0.11	0.74	0.57	73.00	0.57	0.22	0.38
	Equal variances not assumed			0.55	13.22	0.59	0.22	0.40
Preparedness Skill Web Development frameworks	Equal variances assumed	0.01	0.91	0.40	73.00	0.69	0.17	0.44
	Equal variances not assumed			0.39	13.50	0.70	0.17	0.45
Preparedness Skill Cloud Computing	Equal variances assumed	0.88	0.35	-0.16	73.00	0.87	-0.06	0.36
	Equal variances not assumed			-0.19	15.49	0.85	-0.06	0.32
Preparedness Skill AI/ML Basics	Equal variances assumed	1.65	0.20	-0.15	73.00	0.88	-0.06	0.42
	Equal variances not assumed			-0.13	12.22	0.90	-0.06	0.49

Table 4.10 Independent Samples Effect Sizes of the Impact of Internship Experience on Technical Skill Preparedness

		Standardizer	Point Estimate	95% Confidence Interval	
				Lower	Upper
Preparedness Skill Programming Languages	Cohen's d	0.79	0.13	-0.51	0.77
	Hedges' correction	0.80	0.13	-0.51	0.76
	Glass's delta	1.01	0.10	-0.54	0.74
Preparedness Skill OOP Concepts	Cohen's d	0.84	0.44	-0.20	1.09
	Hedges' correction	0.85	0.44	-0.20	1.07
	Glass's delta	1.13	0.33	-0.33	0.98
Preparedness Skill Version Control	Cohen's d	1.16	0.19	-0.46	0.83
	Hedges' correction	1.17	0.19	-0.45	0.82
	Glass's delta	1.22	0.18	-0.47	0.82
Preparedness Skill Web Development frameworks	Cohen's d	1.34	0.13	-0.51	0.77
	Hedges' correction	1.36	0.13	-0.51	0.76
	Glass's delta	1.37	0.13	-0.52	0.77
Preparedness Skill Cloud Computing	Cohen's d	1.12	-0.05	-0.69	0.59
	Hedges' correction	1.13	-0.05	-0.69	0.58
	Glass's delta	0.94	-0.06	-0.70	0.58
Preparedness Skill AI/ML Basics	Cohen's d	1.27	-0.05	-0.69	0.59
	Hedges' correction	1.28	-0.05	-0.68	0.59
	Glass's delta	1.55	-0.04	-0.68	0.60

4.3.2 The Impact of Internship Experience on Soft Skill Preparedness

To determine the effect of internship experience on the perceived preparedness of soft skills, descriptive statistics were computed for each soft skill, reporting the means and standard deviations for both groups (students with internship experience and those without) as shown in table (4-13). The results indicated that students with internship experience generally reported higher levels of preparedness in most soft skills, though the differences were not substantial.

To further investigate these differences, independent samples t-tests were conducted. Levene's test for equality of variances was performed to check the assumption of equal variances, with significant values ($p < 0.05$) indicating unequal variances. The t-tests compared the means of the two groups for each soft skill. As shown in the Table 4.10 none of the t-tests revealed a significant difference between the groups (all p-values > 0.05), indicating no statistically significant difference in soft skill preparedness between students with and without internship experience.

Additionally, effect size measures were calculated, including Cohen's d, Hedges' g (corrected for small sample sizes), and Glass's delta. Hedges' g was selected as the most appropriate effect size measure due to its adjustment for small sample sizes, providing a more accurate estimate. As shown in the table (4-15) the effect sizes, as measured by Hedges' g, were generally small, with confidence intervals including zero, indicating a lack of practical significance in the differences observed. Specifically, for communication skills, Hedges' g was 0.380 (95% CI: -0.258, 1.014); for problem-solving and critical thinking skills, Hedges' g was 0.135 (95% CI: -0.499, 0.768); for teamwork and collaboration skills, Hedges' g was 0.100 (95% CI: -0.534, 0.733); and for time management and adaptability skills, Hedges' g was 0.460 (95% CI: -0.179, 1.095).

In conclusion, the analysis demonstrates that there is no statistically significant impact of internship experience on the preparedness for soft skills such as communication, problem-solving and critical thinking, teamwork and collaboration, and time management and adaptability. The effect sizes are small to negligible, indicating that any observed differences are not practically significant.

Table 4.11 Group Statistics of the Impact of Internship Experience on Soft Skill Preparedness

Internship Experience (Yes/No)		N	Mean	Std. Deviation	Std. Error Mean
Preparedness Skill Communication	Yes	64	3.84	1.00	0.12
	No	11	3.45	1.13	0.34
Preparedness Skill Problem Solving critical thinking	Yes	64	4.03	0.89	0.11
	No	11	3.91	0.94	0.29
Preparedness Skill Teamwork collaboration	Yes	64	4.09	0.92	0.12
	No	11	4.00	1.00	0.30
Preparedness Skill Time Management Adaptability	Yes	64	3.97	0.93	0.12
	No	11	3.55	0.82	0.25

Table 4.12 Independent Samples Test of the Impact of Internship Experience on Soft Skill Preparedness

		Levene's Test for Equality of Variances		t-test for Equality of Means				
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference
Preparedness Skill Communication	Equal variances assumed	0.55	0.46	1.18	73.00	0.24	0.39	0.33
	Equal variances not assumed			1.08	12.82	0.30	0.39	0.36
Preparedness Skill Problem Solving critical thinking	Equal variances assumed	0.00	0.95	0.42	73.00	0.68	0.12	0.29
	Equal variances not assumed			0.40	13.24	0.70	0.12	0.31

Table 4.12 (continued)

Preparedness Skill Teamwork collaboration	Equal variances assumed	0.00	0.96	0.31	73.00	0.76	0.09	0.30
	Equal variances not assumed			0.29	13.08	0.78	0.09	0.32
Preparedness Skill Time Management and Adaptability	Equal variances assumed	0.00	0.98	1.42	73.00	0.16	0.42	0.30
	Equal variances not assumed			1.55	14.74	0.14	0.42	0.27

Table 4.13 Independent Samples Effect Sizes of the Impact of Internship Experience on Soft Skill Preparedness

		Standardizer	**Point Estimate**	**95% Confidence Interval**	
				Lower	**Upper**
Preparedness Skill Communication	Cohen's d	1.02	0.38	-0.26	1.03
	Hedges' correction	1.03	0.38	-0.26	1.01
	Glass's delta	1.13	0.35	-0.32	0.99
Preparedness Skill Problem Solving critical thinking	Cohen's d	0.90	0.14	-0.51	0.78
	Hedges' correction	0.91	0.14	-0.50	0.77
	Glass's delta	0.94	0.13	-0.52	0.77
Preparedness Skill Teamwork collaboration	Cohen's d	0.93	0.10	-0.54	0.74
	Hedges' correction	0.94	0.10	-0.53	0.73
	Glass's delta	1.00	0.09	-0.55	0.73

Table 4.13 (continued)

Preparedness Skill Time Management Adaptability	Cohen's d	0.91	0.46	-0.18	1.11
	Hedges' correction	0.92	0.46	-0.18	1.10
	Glass's delta	0.82	0.52	-0.17	1.18

4.4 Analysis SWEBOK Knowledge Areas against Curriculum Alignment.

This analysis presents the mean preparedness of students in various SWEBOK Knowledge Areas alongside the perceived effectiveness of the curriculum in addressing those areas. The table 4.13 highlights the differences between students' preparedness and the curriculum's effectiveness, providing insights into how well the educational content aligns with students' self-assessed readiness. The gap column indicates the difference between the preparedness and effectiveness scores, revealing areas where the curriculum might be enhanced to better meet industry standards and student expectations.

Table 4.14 Analysis SWEBOK Knowledge Areas against Curriculum Alignment

SWEBOK Knowledge Area	Preparedness	Effectiveness	Gap
Requirements Engineering	3.15	3.04	0.11
Software Design	3.47	3.28	0.19
Software Construction	3.17	3.15	0.02
Software Testing	2.92	2.84	0.08
Software Maintenance	2.85	2.77	0.08
Software Configuration Management	2.96	2.92	0.04
Software Engineering Management	3.21	3.07	0.14
Software Engineering Process	3.23	3.17	0.06
Software Engineering Models and Methods	3.32	3.21	0.11

Table 4.14 (continued)

Software Quality	3.21	3.05	0.16

Requirements Engineering shows a mean preparedness score for students of 3.15, while the perceived effectiveness of the curriculum is 3.04, resulting in a small positive gap of 0.11. This indicates that students feel slightly more prepared in requirements engineering than the curriculum's effectiveness rating suggests, pointing to a reasonable alignment between curriculum content and student preparedness. In Software Design, students have a mean preparedness score of 3.47, compared to a curriculum effectiveness rating of 3.28, yielding a gap of 0.19. This larger positive gap suggests that students feel more confident in their software design skills than the curriculum coverage indicates. This might reflect students' overestimation of their own preparedness or an underestimation of the curriculum's comprehensiveness. For Software Construction, the preparedness score is 3.17, closely aligned with the curriculum effectiveness score of 3.15, resulting in a minimal gap of 0.02. This close alignment suggests that the curriculum and student preparedness in software construction are well-matched. In Software Testing, students' preparedness score is 2.92, slightly higher than the curriculum effectiveness score of 2.84, resulting in a small gap of 0.08. This small positive gap indicates a reasonable alignment between the curriculum and student preparedness in software testing. For Software Maintenance, the preparedness score of 2.85 compared to an effectiveness score of 2.77 results in a gap of 0.08. Similar to software testing, this small gap suggests that the curriculum content and student preparedness in software maintenance are reasonably aligned. In Software Configuration Management, with a preparedness score of 2.96 and an effectiveness score of 2.92, the gap is a minimal 0.04. This close match indicates a good alignment between the curriculum and student preparedness in software configuration management. For Software Engineering Management, the preparedness score is 3.21, while the curriculum effectiveness is rated at 3.07, creating a gap of 0.14. This noticeable positive gap suggests that students feel more prepared in software engineering management than the curriculum's effectiveness rating implies, possibly indicating that students gain additional experience or knowledge outside the curriculum. In Software Engineering Process, students' preparedness score is 3.23, compared to a curriculum effectiveness score of 3.17, resulting in a small gap of 0.06. This small gap indicates a reasonable alignment between the curriculum and student preparedness in software engineering processes. For Software Engineering Models and Methods, the preparedness score is 3.32, while the effectiveness score is 3.21, yielding a gap of 0.11. This positive gap suggests that students feel somewhat more prepared in software engineering models and methods than the curriculum's effectiveness indicates, reflecting a slight overconfidence in their own preparedness. Finally, in Software Quality, with a preparedness score of 3.21 and an effectiveness score of 3.05, the resulting gap is 0.16. This positive gap suggests that students feel more prepared in software quality than the curriculum's effectiveness rating indicates, which might reflect students gaining additional knowledge or experience outside the curriculum.

4.5 Correlation matrix of technical skills preparedness and skill acquisition method.

The following table (4-17) presents the correlation coefficients between students' self-assessed preparedness in various technical skills and different methods of skill acquisition. This analysis helps identify the effectiveness of various learning methods in enhancing technical skills.

Table 4.15 Correlation matrix of technical skills preparedness and skill acquisition method

Technical Skill	Online Courses	Personal Projects	Academic Coursework	Internships/Work Experience
Programming Languages	-0.06	-0.20	-0.10	-0.07
OOP Concepts	-0.07	-0.11	-0.18	0.05
Version Control	0.09	0.21	-0.24	0.10

Table 4.15 (continued)

Web Development Frameworks	0.23	0.23	-0.25	0.16
Cloud Computing	0.01	0.19	-0.20	0.25
AI/ML Basics	-0.08	-0.03	-0.05	-0.08

For Programming Languages, all methods show weak negative correlations: Online Courses (-0.06), Personal Projects (-0.20), Academic Coursework (-0.10), and Internships/Work Experience (-0.07). This indicates that these methods have minimal impact on preparedness in programming languages. In the case of OOP Concepts, there are weak negative correlations for Online Courses (-0.07), Personal Projects (-0.11), and Academic Coursework (-0.18), while Internships/Work Experience shows a minimal positive correlation (0.05). This suggests that while internships might provide some benefit, other methods do not significantly contribute to preparedness.

Moving to Version Control, there are minimal to moderate positive correlations for Online Courses (0.09), Personal Projects (0.21), and Internships/Work Experience (0.10), but a moderate negative correlation for Academic Coursework (-0.24). This pattern indicates that personal projects, in particular, are beneficial for preparedness in version control. For Web Development Frameworks, both Online Courses and Personal Projects show moderate positive correlations (0.23), while Internships/Work Experience has a weak positive correlation (0.16). However, Academic Coursework has a moderate negative correlation (-0.25), highlighting the effectiveness of online courses and personal projects in this area.

Regarding Cloud Computing, there is a minimal positive correlation with Online Courses (0.01), a weak positive correlation with Personal Projects (0.19), a weak negative correlation with Academic Coursework (-0.20), and a moderate positive correlation with Internships/Work Experience (0.25). This suggests that internships are especially valuable for preparedness in cloud computing. Finally, for AI/ML Basics, there are weak negative correlations across all methods: Online Courses (-0.08), Personal Projects (-0.03), Academic Coursework (-0.05), and Internships/Work Experience (-0.08), indicating minimal impact on preparedness from these methods.

Based on that we can say that the Internships/Work Experience show moderate positive correlations with preparedness in cloud computing and web development frameworks, but minimal correlations with other skills. Online Courses/Tutorials exhibit a moderate positive correlation with preparedness in web development frameworks, but minimal to weak negative correlations with other skills. Personal Projects show moderate positive correlations with preparedness in version control and web development frameworks, but weak correlations with other skills. Academic Coursework, on the other hand, shows moderate negative correlations with preparedness in version control, web development frameworks, and other skills.

4.6 Correlation matrix of soft skills preparedness and skill acquisition method.

The following Table 4.15 presents the correlation coefficients between students' self-assessed preparedness in various soft skills and different methods of skill acquisition. This analysis helps identify the effectiveness of various learning methods in enhancing soft skills.

Table 4.16 Correlation matrix of soft skills preparedness and skill acquisition method

Soft Skill	**Online Courses**	**Personal Projects**	**Academic Coursework**	**Internships/Work Experience**
Communication	-0.02	-0.03	-0.24	0.03
Problem-Solving/Critical Thinking	-0.14	-0.07	-0.20	0.01
Teamwork/Collaboration	-0.02	-0.05	-0.19	-0.04
Time Management/Adaptability	-0.09	-0.06	-0.27	0.09

For Communication, online courses and personal projects both show minimal negative correlations (-0.02 and -0.03, respectively), while academic coursework shows a moderate negative correlation (-0.24). In contrast, internships/work experience show a minimal positive correlation (0.03). This suggests that academic coursework has a more substantial negative impact on communication skills, while internships provide a slight positive effect. When examining Problem-Solving/Critical Thinking, online courses (-0.14) and academic coursework (-0.20) exhibit weak negative correlations, and personal projects show a minimal negative correlation (-0.07). Internships, on the other hand, have a minimal positive correlation (0.01), indicating that all methods generally have a negative impact except for a slight positive influence from internships. For Teamwork/Collaboration, all methods show minimal to weak negative correlations: online courses (-0.02), personal projects (-0.05), academic coursework (-0.19), and internships (-0.04), indicating limited overall impact on developing this skill.

In the area of Time Management/Adaptability, online courses (-0.09) and personal projects (-0.06) show weak and minimal negative correlations, respectively, while academic coursework has a moderate negative correlation (-0.27). Internships/work experience, however, show a weak positive correlation (0.09), suggesting that internships may contribute positively to developing time management and adaptability skills, whereas academic coursework tends to have a more substantial negative impact. Based on that we can say that the Internships/Work Experience show minimal to weak positive correlations with preparedness in soft skills, indicating they might have a slight positive impact. Conversely, Online Courses/Tutorials exhibit weak negative correlations with preparedness in most soft skills, indicating they might not be very effective for improving these skills. Personal Projects show very weak negative correlations with preparedness in soft skills, indicating limited impact. On the other hand, Academic Coursework shows weak to moderate negative correlations with preparedness in most soft skills, indicating that traditional academic coursework might not be the most effective method for developing these skills.

4.7 Identification of Knowledge and Skill Gaps.

Students reported several areas where they feel the curriculum could be improved as shown in Figure 4.9. Commonly mentioned gaps include:

- Advanced software testing techniques
- In-depth software maintenance practices
- Practical experience with configuration management tools
- Emerging technologies such as AI/ML and cloud computing

These insights suggest that while the curriculum provides a solid foundation in many areas, it needs to adapt to the evolving technological landscape and industry demands.

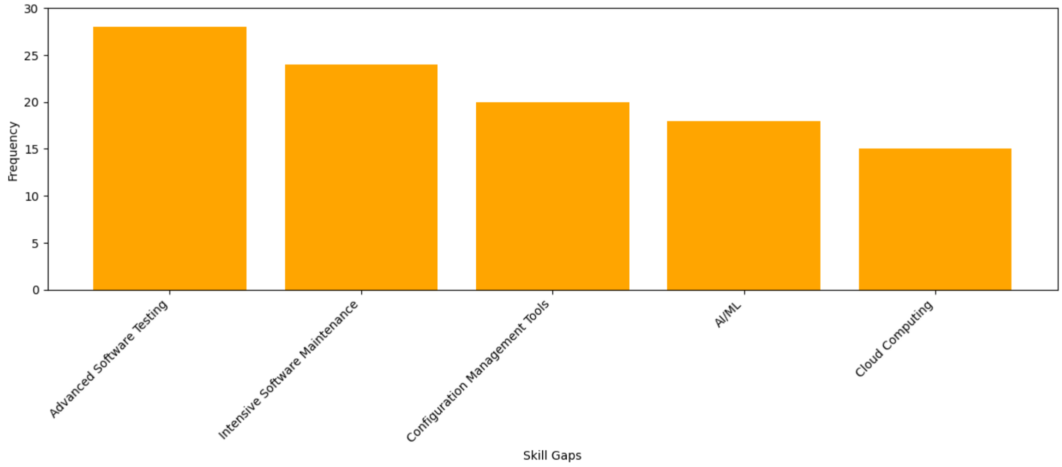

Figure 4.9 Frequency of Identified knowledge and Skill Gaps

4.8 Program Strengths

This subsection highlights the aspects of the software engineering curriculum that students found most valuable in preparing them for industry expectations. Understanding the strengths can help in reinforcing these areas and building upon them.

As shown in Figure 4.10 Students identified several strengths of the current curriculum, including:

- Strong foundation in core programming languages and OOP concepts
- Effective teamwork and collaboration projects
- Comprehensive problem-solving and critical thinking exercises
- Practical assignments that simulate real-world software engineering challenges

These strengths suggest that the curriculum effectively prepares students in several key areas, providing a solid basis for their professional development.

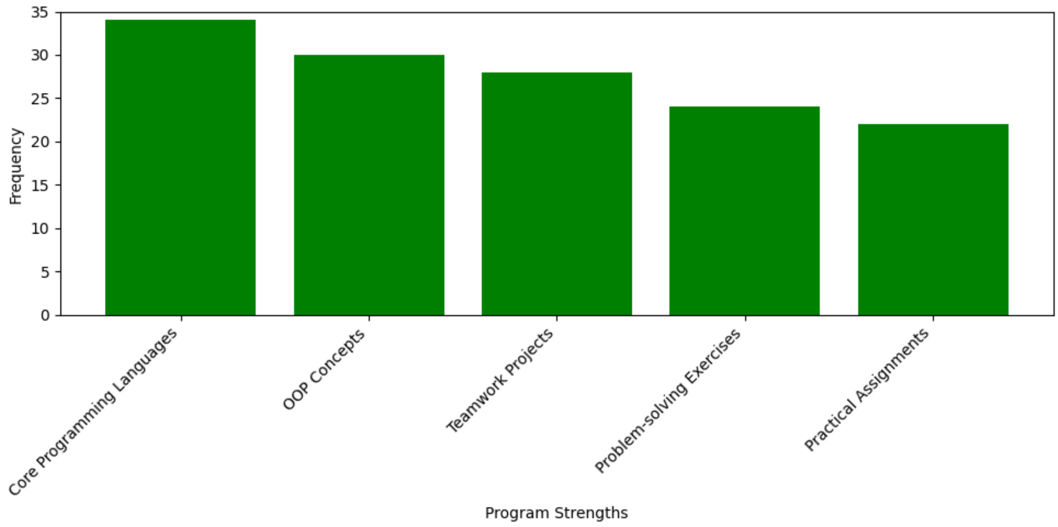

Figure 4.10 Frequency of Identified program strengths

4.9 University Support

As in Table 4.16 the responses indicate that 46.7% of the students (35 out of 75) feel that the university provides sufficient guidance and resources to secure relevant internships or work experience. However, 53.3% (40 out of 75) of the students do not feel sufficiently supported. This suggests a need for the university to improve its support mechanisms to ensure all students have access to the necessary resources for securing practical experiences.

Table 4.17 University Support Analysis

University Support	**Count**
Yes	35
No	40
Total	75

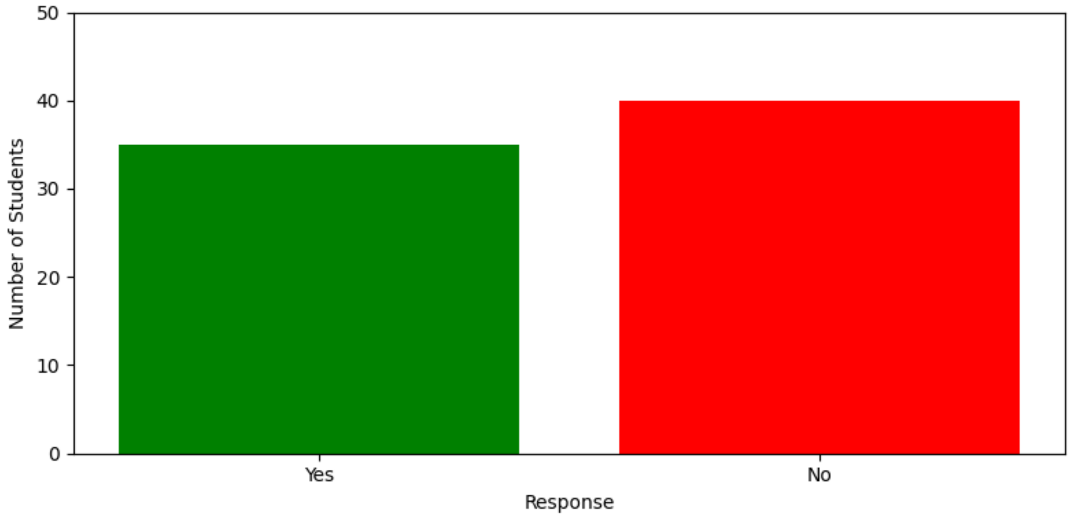

Figure 4.11 University support for Internships and work experience

4.10 Awareness of emerging technologies and its importance in education.

4.10.1 Awareness of AI and ML.

The responses shown in Table 4.17 indicate that 63.5% of the students (47 out of 74) are familiar with the role of AI and ML in software development, 24.3% (18 out of 74) are somewhat familiar, and 12.2% (9 out of 74) are not familiar. These results highlight that a significant portion of students are aware of AI and ML, but there is still a considerable number who are not fully familiar with these technologies, suggesting a need for enhanced focus on these topics within the curriculum.

Table 4.18 Awareness of AI and ML

Response	Count
Yes	47
Somewhat	18
No	9

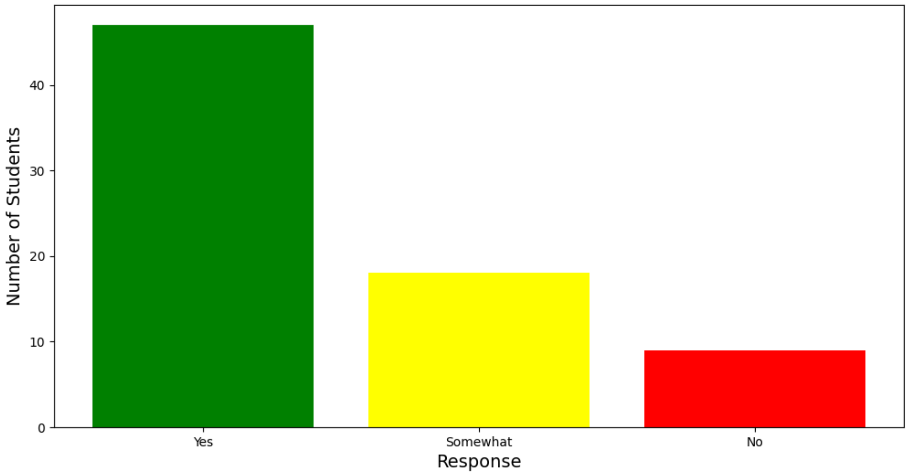

Figure 4.12 Awareness for emerging technologies

4.10.2 Importance of AI and ML in education.

The responses in Table 4.18 indicate that 89.3% of the students (67 out of 75) believe that incorporating AI and ML concepts is important in software engineering education, 5.3% (4 out of 75) do not believe it is important, and 5.3% (4 out of 75) somewhat agree. These findings suggest that a majority of students recognize the importance of AI and ML, indicating a strong need for further emphasis and integration of these technologies within the curriculum.

Table 4.19 Importance of AI and ML in education

Response	**Count**
Yes	67
Somewhat	4
No	4

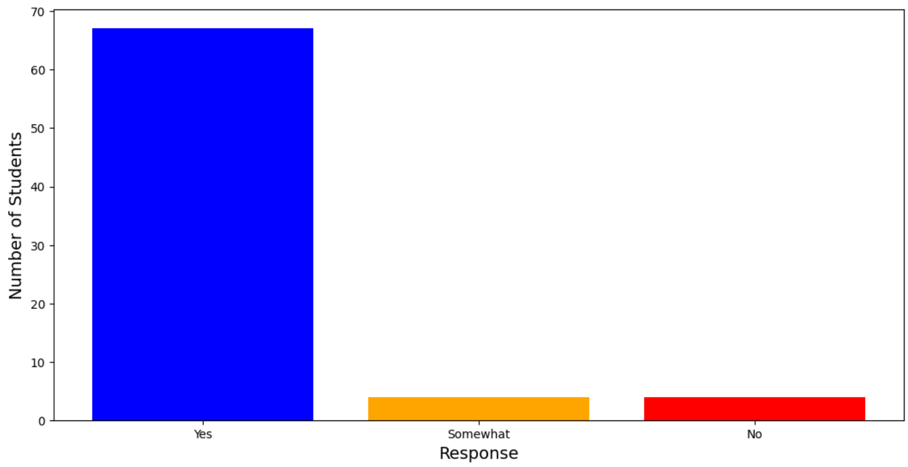

Figure 4.13 Importance of emerging technologies in education

4.11 Software Engineering Curriculum and Job Market Analysis.

4.11.1 Curriculum Analysis.

The Software Engineering curriculum at in a University is structured to provide a comprehensive education over eight semesters, encompassing a broad range of essential topics, where it integrates a balanced mix of programming and computing fundamentals, mathematics and physics, software engineering core subjects, and additional areas such as language and career planning courses. This structure ensures a holistic educational experience designed to equip students with both technical and non-technical skills.

The core software engineering courses provide a robust foundation in essential software engineering principles and practices. The details of these compulsory courses are provided in Appendix B, covering topics such as software process models, requirements engineering, system modeling, architectural design, software testing, and project management. The following are key courses included in the curriculum:

- SE112: Introduction to Software Engineering
- SE221: Software Requirements Engineering
- SE212: Human-Computer Interaction
- SE345: Software Quality Assurance

- SE321: Object-Oriented Analysis and Design
- SE324: Software Project Management and Economics
- SE328: Algorithms and Optimization Methods
- SE322: Software Architecture
- SE344: Systems Software Validation and Testing

Each of these courses plays a critical role in developing students' expertise in software engineering and aligning with industry standards.

4.11.2 Analysis of Recent Job Postings

To comprehensively analyze the Software Engineering curriculum at Atilim University, we need to compare the curriculum content with the competencies and skills required in the job market. This analysis involves examining job postings to identify the most frequently required skills and knowledge areas. And the results of this analysis was as explained below:

Programming Languages:

Python: Mentioned in 16 job postings (37.2%). Courses such as Computer Programming I & II, and Object-Oriented Programming cover these skills.

Java: Mentioned in 11 job postings (25.6%). Java is typically covered in Object-Oriented Programming and related courses.

Web Development Technologies:

React: Mentioned in 3 job postings (7.0%). This indicates a need for stronger focus on modern web development frameworks.

JavaScript: Mentioned in 3 job postings (7.0%). Essential for front-end development, should be integrated into relevant courses.

Software Development Practices:

Git: Mentioned in 11 job postings (25.6%). Version control is a crucial skill, covered under courses like Software Quality Assurance.

Agile Methodologies: Mentioned in 2 job postings (4.7%). Agile principles should be emphasized in project management and software development courses.

Cloud Computing:

AWS: Mentioned in 11 job postings (25.6%). Cloud computing concepts could be incorporated into existing courses or offered as specialized electives.

Data Science and AI:

Data Science: Mentioned in 11 job postings (25.6%). Data science topics can be integrated into software engineering courses or offered as electives.

Machine Learning: Mentioned in 8 job postings (18.6%). Courses on AI and ML should be introduced to address this demand.

Database Management:

SQL: Mentioned in 8 job postings (18.6%). Covered in courses like Database Design and Management.

Software Testing and Quality Assurance:

Software Testing: Covered under Systems Software Validation and Testing and Software Quality Assurance. The job market demands a solid understanding of these areas.

Soft Skills:

Communication Skills: Mentioned in 15 job postings (34.9%). Emphasized through English for Occupational Purposes and project-based courses.

Problem Solving: Mentioned in 3 job postings (7.0%). Critical thinking and problem-solving skills are integral to the software engineering curriculum.

DevOps and CI/CD:

DevOps: Mentioned in 5 job postings (11.6%). Introduction of DevOps practices and tools in the curriculum can better prepare students for industry roles.

Containerization and Orchestration:

Docker: Mentioned in 8 job postings (18.6%).

Kubernetes: Mentioned in 4 job postings (9.3%). These technologies should be introduced in advanced courses or electives.

4.11.3 Evaluation of the Curriculum Against Industry Needs.

Alignment with SWEBOK

The Software Engineering curriculum at Atilim University exhibits a strong alignment with the Software Engineering Body of Knowledge (SWEBOK) across several critical areas as explained below:

Software Requirements:

Course: SE221 - Software Requirements Engineering

This course covers the essential aspects of requirements engineering, including requirements elicitation, analysis, specification, and validation. It ensures that students understand the processes for identifying and managing software requirements from various stakeholders.

Software Design:

Course: SE321 - Object-Oriented Analysis and Design

Students learn fundamental principles of object-oriented design, including UML modeling, design patterns, and architectural design. This course ensures a thorough understanding of designing robust and scalable software systems.

Software Construction:

Course: SE112 - Introduction to Software Engineering

This introductory course includes topics on software construction techniques, coding standards, and best practices, ensuring that students are well-prepared for practical software development tasks.

Software Testing:

Course: SE344 - Systems Software Validation and Testing

This course provides comprehensive coverage of software testing methodologies, including unit testing, integration testing, system testing, and test automation, aligning closely with SWEBOK's testing knowledge area.

Software Maintenance:

Course: SE112 - Introduction to Software Engineering

Topics on software evolution and maintenance are included to ensure that students understand the challenges and strategies associated with maintaining software systems over time.

Software Configuration Management:

Course: SE345 - Software Quality Assurance

This course includes essential topics on configuration management, version control systems, and the management of software changes, which are crucial for maintaining the integrity and traceability of software products.

Software Engineering Management:

Course: SE324 - Software Project Management and Economics

Students are introduced to project management principles, including project planning, resource allocation, risk management, and cost estimation, ensuring they are equipped with the skills to manage software projects effectively.

Software Engineering Process:

Course: SE112 - Introduction to Software Engineering

This course covers various software process models, including agile, waterfall, and iterative processes, helping students understand the frameworks within which software development occurs.

Software Quality:

Course: SE345 - Software Quality Assurance

This course focuses on the principles of software quality assurance, metrics, and standards, ensuring that students understand how to measure and improve the quality of software products.

Software Engineering Models and Methods:

Course: SE328 - Algorithms and Optimization Methods

This course teaches students about algorithm design, optimization techniques, and computational complexity, providing a strong foundation in the theoretical underpinnings of software engineering methods.
By aligning these courses with SWEBOK's knowledge areas, the curriculum ensures that students receive a comprehensive education that meets internationally recognized standards for software engineering practice.

Alignment with Skills and Knowledge Required by the Software Industry
The analysis of job postings reveals the following key areas of demand in the industry:
Programming Skills:

The curriculum includes extensive training in programming languages and paradigms through courses such as Computer Programming I & II, Object-Oriented Programming, and Study of Programming Languages. These courses ensure that students are proficient in languages like Java, C++, and Python, which are frequently listed in job postings.

Web Development Technologies:

React: Mentioned in job postings, indicating a need for stronger focus on modern web development frameworks.

JavaScript: Essential for front-end development, should be integrated into relevant courses.

Software Development Practices:

Git: Version control is a crucial skill, covered under courses like Software Quality Assurance.

Agile Methodologies: Agile principles should be emphasized in project management and software development courses.

Cloud Computing:

AWS: Cloud computing concepts could be incorporated into existing courses or offered as specialized electives.

Data Science and AI:

Data Science: Data science topics can be integrated into software engineering courses or offered as electives.

Machine Learning: Courses on AI and ML should be introduced to address this demand.

Database Management:

SQL: Covered in courses like Database Design and Management.

Software Testing and Quality Assurance:

Covered under Systems Software Validation and Testing and Software Quality Assurance. The job market demands a solid understanding of these areas.

Soft Skills:

Communication Skills: Emphasized through English for Occupational Purposes and project-based courses.

Problem Solving: Critical thinking and problem-solving skills are integral to the software engineering curriculum.

DevOps and CI/CD:

Introduction of DevOps practices and tools in the curriculum can better prepare students for industry roles.

Containerization and Orchestration:

Docker and **Kubernetes**: These technologies should be introduced in advanced courses or electives.

4.11.4 SWOT analysis.

A SWOT analysis (Strengths, Weaknesses, Opportunities, Threats) was conducted as given in Figure 4.14 based on student feedback, job market analysis, and curriculum mapping.

Strengths:

- **Comprehensive Coverage of Fundamentals:** The curriculum covers fundamental concepts in software engineering, providing a strong foundation.
- **Practical Components:** Inclusion of practical experiences through summer practices and project courses enhances hands-on learning.
- **Emphasis on Language Skills:** Courses focused on English for Academic and Occupational Purposes ensure students have strong communication skills.

Weaknesses:

- **Limited Coverage of Emerging Technologies:** The curriculum lacks sufficient focus on emerging technologies such as AI, ML, and cloud computing.
- **Insufficient Focus on Real-World Applications:** There is a need for more real-world applications and industry practices, particularly in newer technologies.
- **Lack of Interdisciplinary Courses:** The curriculum could benefit from integrating interdisciplinary courses that connect software engineering with other fields.

Opportunities:

- **Industry Partnerships:** Opportunities exist to develop partnerships with tech companies to enhance practical learning and internships.
- **Integration of Latest Technologies:** The curriculum can be updated to include courses on the latest technologies and methodologies.
- **Leveraging Online Learning Platforms:** Online courses and tutorials can supplement classroom learning and keep students updated with current trends.

Threats:

- **Rapid Technological Changes:** The fast pace of technological advancement could quickly render some course content obsolete.

- **Global Competition:** Competition from other institutions that are quicker to adapt their curricula to industry changes.

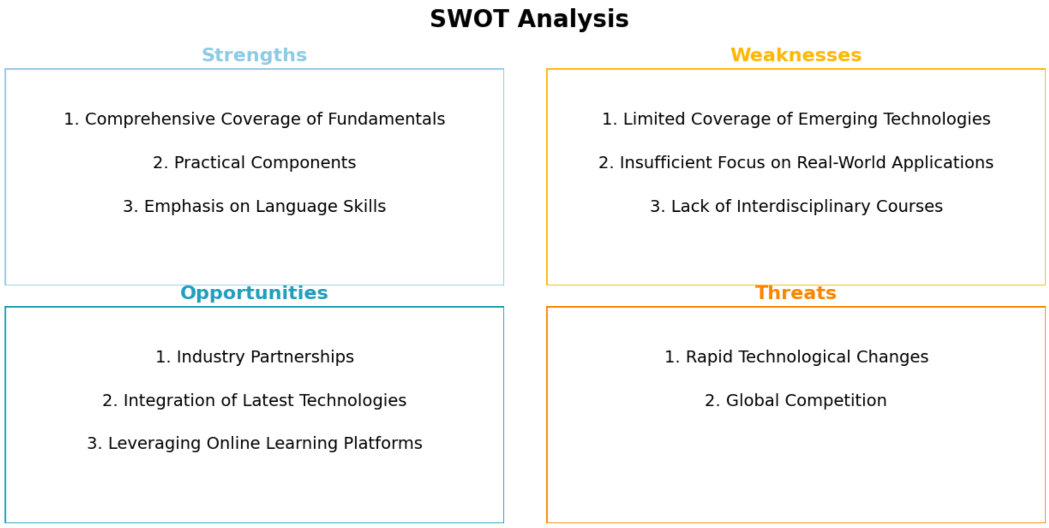

Figure 4.14 Curriculum of software engineering SWOT analysis

CHAPTER 5

DISCUSSION

This chapter presents a comprehensive discussion of the results obtained from the analysis of survey data, curriculum alignment with the Software Engineering Body of Knowledge (SWEBOK), and job posting analysis. It systematically addresses the findings, providing a detailed interpretation of the results, their relation to existing research, and their implications for both academia and the software engineering industry. This discussion contributes to the overall objective of the thesis, which aims to identify and bridge the skill gaps between software engineering education and the competencies required in the software engineering job market.

5.1 Descriptive Statistics

The data indicate that students feel moderately familiar with most SWEBOK knowledge areas. Higher familiarity was reported in software design (Mean = 3.47) and requirements engineering (Mean = 3.15). Conversely, areas such as software testing (Mean = 2.92), software maintenance (Mean = 2.85), and configuration management (Mean = 2.96) exhibit lower familiarity levels. These findings suggest a need for greater emphasis on software testing, maintenance, and configuration management within the curriculum. The relatively high familiarity in software design and requirements engineering suggests these areas are well-covered. However, the curriculum needs to improve in other areas to ensure comprehensive student preparedness. Improving the curriculum coverage in these areas can ensure that students are comprehensively prepared for the demands of the software engineering industry. These findings are consistent with the study by Alabbadi and Qureshi (2016), which emphasizes the importance of hands-on experience in these areas [11] . For academia, this indicates the need for curriculum adjustments to incorporate more practical exercises and industry-aligned training. For the software engineering industry, it highlights the importance of providing feedback to educational institutions to bridge these gaps.

Technical Skills Preparedness

Students reported high levels of preparedness in core programming languages (Mean = 3.85) and object-oriented programming concepts (Mean = 3.78). However, lower preparedness levels were noted in areas such as web development frameworks (Mean = 2.69), cloud computing (Mean = 2.05), and AI/ML concepts (Mean = 2.94). The high preparedness in core programming languages and object-oriented programming concepts indicates that the curriculum effectively teaches these fundamental skills. However, the lower preparedness in modern web development frameworks, cloud computing, and AI/ML highlights a significant gap. These skills are increasingly demanded in the job market, and their integration into the curriculum is crucial. This finding is consistent with prior studies that emphasize the need for updating curricula to include these modern technologies [3], [26] .

For academia, this means updating course content to reflect current industry practices and emerging technologies. For the software engineering industry, it underscores the need for ongoing professional development and training programs to keep skills current.

Soft Skills Preparedness

Students generally feel well-prepared in essential soft skills, particularly in teamwork and collaboration (Mean = 4.08) and problem-solving and critical thinking (Mean = 4.01). Written and oral communication skills (Mean = 3.79) and time management and adaptability (Mean = 3.91) also show strong levels of preparedness. The high preparedness in soft skills suggests that the current curriculum effectively addresses the development of these critical competencies. Soft skills are crucial for professional success in software engineering, and the curriculum's emphasis on teamwork, problem-solving, communication, and time management is commendable. Maintaining and enhancing these competencies through continuous emphasis and practical experiences is necessary to ensure students' readiness for professional challenges. This aligns with the findings of Groenveld W and Vennekens. (2022), who highlight the importance of these skills in professional settings [5]

For academia, this emphasizes the importance of incorporating team-based projects and communication skills training into the curriculum. For the software engineering industry, it highlights the value of these skills in the workplace and the need for ongoing support and development.

Curriculum Alignment with SWEBOK Knowledge Areas

The descriptive statistics reveal that the curriculum alignment with SWEBOK knowledge areas is generally moderate. The mean values for most knowledge areas fall between 3.04 and 3.28, indicating satisfactory alignment but with room for improvement. Notably, software testing (Mean = 2.84) and software maintenance (Mean = 2.77) have lower mean values, suggesting these areas are less adequately covered in the curriculum. The variability in students' responses highlights the need for a more consistent approach to integrating these topics into the curriculum. Enhancing coverage of software testing and maintenance will improve alignment with SWEBOK and better prepare students for the industry. This will ensure students are equipped with the necessary skills and knowledge, aligning their education more closely with industry standards. This finding is supported by research from Kuusinen K and Albertsen S . (2010) on the importance of aligning academic programs with professional standards[27] . For academia, this suggests the need for continuous curriculum reviews and updates to ensure alignment with professional standards. For the software engineering industry, it underscores the importance of collaborating with educational institutions to ensure that curricula meet industry needs.

Experience Impact on Skill Development

Internships and work experience are viewed as the most effective in developing essential skills (Mean = 4.12), followed closely by personal projects (Mean = 3.92) and internships/co-ops (Mean = 3.95). Academic coursework and online courses are also considered helpful but to a slightly lesser extent. These findings underscore the importance of practical, hands-on experiences in complementing academic learning. Internships, co-ops, and personal projects provide real-world experience that is crucial for developing technical and soft skills. The curriculum should incorporate more opportunities for students to engage in practical experiences through industry partnerships and project-based learning to better prepare them for the workforce. This is consistent with the findings of Saiedian in his research paper (2018), which emphasize the critical role of practical experience in skill development [3] . For academia, this means establishing stronger partnerships with industry to provide students with practical experience opportunities. For the software engineering industry, it highlights the importance of offering internships and co-op programs to develop future talent.

Overall Preparedness for Entry-Level Roles

The mean value of 3.49 indicates that students feel moderately prepared for entry-level software engineering roles. The standard deviation of 0.79 suggests some variation in responses, with some students feeling more prepared than others. While the curriculum provides a solid foundation, there is room for improvement in preparing students for the practical demands of the industry. Enhancing the curriculum to include more practical experiences and updated content on emerging technologies will help improve overall preparedness and ensure students are ready to meet industry expectations. This aligns with the findings of Garousi et al. (2018), which highlight the importance of aligning academic training with industry requirements [28] . For academia, this suggests the need for continuous curriculum updates and the inclusion of practical experiences. For the software engineering industry, it emphasizes the need for clear communication of job requirements to educational institutions.

Impact of Internship Experience on Skill Preparedness

The analysis showed that the mean score for perceived technical skill preparedness was not significantly higher for students with internship experience compared to those without. For programming languages, students with internship experience had a mean score of 3.83 (SD = 0.747), while those without had a mean score of 3.73 (SD = 1.009). The T-test for equality of means revealed no statistically significant difference ($t(73) = 0.392$, $p = 0.696$). Similarly, for soft skills, students with internship experience generally reported higher mean scores, but none of the differences were statistically significant. For communication skills, the mean score was 3.84 (SD = 0.996) for those with internship experience and 3.45 (SD = 1.128) for those without, with no statistically significant difference ($t(73) = 1.175$, $p = 0.244$). These findings indicate that internship experience does not significantly influence students' perceived preparedness of either technical or soft skills. Despite these results, the small effect sizes suggest that while differences may not be statistically significant, there could be practical implications that warrant further exploration. This evidence underscores the need to carefully evaluate the structure and content of internship programs within the academic curriculum to ensure they effectively contribute to students' skill development. While previous studies, such as Mishra A. and Mishra D. (2016), emphasize the value of internships in skill acquisition, our findings suggest that more targeted and robust internship experiences may be necessary to achieve significant improvements in perceived skill preparedness. For academia, this means developing more effective internship programs and stronger partnerships with industry. For the software engineering industry, it highlights the importance of providing meaningful internship experiences that truly enhance job readiness in graduates.

SWEBOK Knowledge Areas vs. Curriculum Alignment

The table in the results file highlights the differences between students' preparedness and the curriculum's effectiveness in various SWEBOK knowledge areas. For example, requirements engineering shows a small positive gap of 0.11, while software design shows a larger positive gap of 0.19. Software construction, software testing, and software maintenance have smaller gaps, indicating reasonable alignment between the curriculum and student preparedness in these areas. The small positive gaps in several knowledge areas suggest that students feel slightly more prepared than the curriculum's effectiveness rating implies. This might reflect students' confidence in their abilities or areas where the curriculum is effectively preparing them. However, the gaps also indicate areas where the curriculum could be enhanced to better meet industry standards and student expectations. Focusing on improving curriculum content in areas with larger gaps, such as software design and quality, will help align educational outcomes with industry needs. This conclusion is supported by Alabbadi and Qureshi (2016) [11] . For academia, this suggests the need for targeted improvements in specific knowledge areas. For the software engineering industry, it underscores the importance of providing input to academia to ensure that curricula align with industry needs.

Correlation Matrix of Technical Skills Preparedness and Skill Acquisition Method

The correlation analysis shows that internships and work experience are positively correlated with preparedness in several technical skills, particularly in web development frameworks (0.23) and cloud computing (0.25). Academic coursework, on the other hand, shows moderate negative correlations with preparedness in version control (-0.24) and web development frameworks (-0.25). These results suggest that practical, hands-on experiences such as internships and personal projects are more effective in enhancing technical skill preparedness than traditional academic coursework. This aligns with existing research that emphasizes the importance of experiential learning in technical education. Enhancing the curriculum with more practical projects and industry-related tasks will likely improve students' technical skill preparedness. This finding aligns with the research by Kamthan and Shamir (2019) [16]. For academia, this indicates the need to integrate more experiential learning opportunities into the curriculum. For the software engineering industry, it highlights the value of providing students with hands-on project experience.

Correlation Matrix of Soft Skills Preparedness and Skill Acquisition Method

The correlation analysis shows minimal to weak positive correlations for internships and work experience with preparedness in soft skills, indicating they might have a slight positive impact. Conversely, academic coursework shows weak to moderate negative correlations with preparedness in most soft skills. These findings highlight the importance of practical experiences in developing soft skills. While academic coursework is essential, it appears less effective in preparing students for the interpersonal and communicative demands of the industry. Integrating more real-world projects and internships into the curriculum can provide students with the necessary soft skills for their professional careers. This conclusion is consistent with the findings of Karimi and Pina (2016) [30]. For academia, this suggests the need for more emphasis on practical experiences that develop soft skills. For the software engineering industry, it underscores the value of internships and real-world projects in developing these skills.

Identification of Knowledge and Skill Gaps

Students reported several areas where they feel the curriculum could be improved, including advanced software testing techniques, in-depth software maintenance practices, practical experience with configuration management tools, and emerging technologies such as AI/ML and cloud computing. These insights suggest that while the curriculum provides a solid foundation in many areas, it needs to adapt to the evolving technological landscape and industry demands. Addressing these gaps by incorporating advanced and emerging technologies into the curriculum will better prepare students for the software engineering job market. This need for adaptation is echoed in the study Felix et al. (2020) [2]. For academia, this means updating curricula to include advanced and emerging technologies. For the software engineering industry, it highlights the importance of communicating technological trends to educational institutions.

Program Strengths

Students identified several strengths of the current curriculum, including a strong foundation in core programming languages and OOP concepts, effective teamwork and collaboration projects, comprehensive problem-solving and critical thinking exercises, and practical assignments that simulate real-world software engineering challenges. These strengths suggest that the curriculum effectively prepares students in several key areas, providing a solid basis for their professional development. Building on these strengths while addressing identified gaps will enhance the overall effectiveness of the curriculum. This is supported by Chiru et al. [31]. For academia, this means leveraging strengths while addressing gaps. For the software engineering industry, it underscores the value of these core skills in professional settings.

University Support

The responses indicate that 46.7% of the students feel that the university provides sufficient guidance and resources to secure relevant internships or work experience, while 53.3% do not feel sufficiently supported. This suggests a need for the university to improve its support mechanisms to ensure all students have access to the necessary resources for securing practical experiences. Enhancing career services and establishing stronger industry partnerships can help bridge this gap and provide better support for students. This finding aligns with the research that conducted by Garousi et al. (2018) [32]. For academia, this means enhancing support services and industry partnerships. For the software engineering industry, it highlights the importance of providing opportunities for student engagement and development.

Awareness of Emerging Technologies and Its Importance in Education

The responses indicate that 63.5% of the students are familiar with the role of AI and ML in software development, while 24.3% are somewhat familiar, and 12.2% are not familiar. Additionally, 89.3% of the students believe that incorporating AI and ML concepts is important in software engineering education. These findings highlight a significant awareness among students of the importance of AI and ML in software development. However, the presence of students who are not fully familiar with these technologies suggests a need for enhanced focus on these topics within the curriculum. Integrating AI and ML concepts more comprehensively into the curriculum will better prepare students for future industry demands. This is in line with the findings of Ntinda et al. (2019) [26]. For academia, this means integrating AI and ML more thoroughly into the curriculum. For the software engineering industry, it emphasizes the importance of these technologies in current and future job roles.

Software Engineering Curriculum and Job Market Analysis

Curriculum Analysis

The curriculum analysis shows that while the core areas are well-covered, there are gaps in the coverage of emerging technologies and practical skills. The curriculum includes comprehensive courses on programming languages, data structures, and algorithms, but lacks sufficient focus on cloud computing, AI/ML, and modern web development frameworks. Addressing these gaps by incorporating updated content and practical projects related to these technologies will better prepare students for the job market. This conclusion is supported by O'Leary et al. (2017) [33]. For academia, this suggests the need for curriculum updates to include emerging technologies. For the software engineering industry, it emphasizes the importance of communicating technological needs to educational institutions.

Analysis of Recent Job Postings

The analysis of recent job postings reveals that employers are looking for skills in programming languages (Python, Java), web development technologies (React, JavaScript), cloud computing (AWS), and AI/ML. The current curriculum covers some of these skills but needs to include more comprehensive courses on modern web development frameworks, cloud computing, and AI/ML to meet job market demands. Enhancing the curriculum to align with these job market trends will improve students' employability and readiness for industry roles. This finding aligns with the research by Daneva et al. (2017) [10]. For academia, this means aligning curricula with job market demands. For the software engineering industry, it underscores the importance of clearly articulating skill requirements in job postings.

Evaluation of the Curriculum Against Industry Needs

The evaluation of the curriculum against industry needs indicates a moderate alignment, with a need for improvement in areas such as software testing, maintenance, and configuration management. The curriculum should incorporate more practical experiences and industry-related tasks to ensure students are well-prepared for the demands of the software engineering job market. Integrating internships, industry projects, and certifications can bridge the gap between academic knowledge and industry requirements. This conclusion is supported by Ayofe and Ajetola (2018) [14]. For academia, this suggests the need for curriculum adjustments to meet industry needs. For the software engineering industry, it emphasizes the importance of providing input to academic programs to ensure alignment.

SWOT Analysis

The SWOT analysis of the curriculum reveals several key insights. Strengths include a robust foundation in core programming languages and object-oriented programming concepts, effective teamwork and collaboration projects, comprehensive problem-solving and critical thinking exercises, and practical assignments that simulate real-world software engineering challenges. However, weaknesses are apparent in the gaps in coverage of emerging technologies such as AI/ML and cloud computing, insufficient focus on software testing, maintenance, and configuration management, and a need for more practical, hands-on experiences. Opportunities exist in updating content to include emerging technologies, integrating more practical projects and industry-related tasks, enhancing career services and industry partnerships, and offering more internship opportunities. Conversely, threats involve rapid technological advancements that necessitate continuous curriculum updates, potential misalignment with fast-evolving industry demands, and the risk of not adequately preparing students for current industry standards and technologies. This comprehensive SWOT analysis, conducted as part of this thesis, provides a clear direction for curriculum improvements and underscores the importance of ongoing dialogue between academia and the software engineering industry to ensure curricula remain relevant.

CHAPTER 6

CONCLUSION

This comprehensive analysis of the survey data from senior students in the software Engineering program at Atilim University has revealed several critical insights into their preparedness for the software engineering industry. Most SWEBOK knowledge areas were moderately familiar to students, with software design and requirements engineering generating greater familiarity whilst software testing, maintenance and configuration management being lesser known. Also, they are quite confident in basic programming languages and OOP concepts. But they have low preparedness usually on Web Development Frameworks, Concepts of Cloud Computing and AI (Artificial Intelligence), ML which also suggests the curriculum is bit week on content in these areas. In addition, the survey results indicated that students perceive themselves as adequately prepared in basic soft skills such as teamwork and collaboration or problem-solving and critical thinking. However, written communication and oral communication could do with some building up as well as time management skills and adaptability.

It was also the strongest of all associated with intern experiences. People are more likely to view internships and personal projects as the most beneficial experiences in obtaining relevant skills. However, a significant segment of students indicate that the university does not adequately help them find internships in their field; suggesting an area where the institution can improve. And even though many students rightly understood or were at least familiar with AI and ML, a far from negligible portion of the student body have less knowledge in those areas which should give reason to put an emphasis on these subjects more throughout the curriculum. Overall, students feel moderately prepared for entry-level software engineering roles, with some variation in the responses, suggesting that not all students feel equally prepared As these results.

clearly signify, there are many consequences for higher education, especially in term of academic curricular design. The first is a need to shore up the curriculum where reported familiarity and preparedness were low. The testing, maintenance and configuration management blocks within software engineering courses need to be enhanced! Also, the syllabus must include modern web technology frameworks like Cloud Computing and AI/ML to train students for immediate employability [34], [35]. This can be ensured by integrating such themes in both the coursework and practical projects.

In addition, the low to moderate familiarity with SWEBOK knowledge areas and technical skills indicate that theoretical knowledge only is not enough. Hence, it the need of the hour to introduce practical hands-on experiences via labs, workshops & real-time projects in order for students to move from theoretical knowledge straight into applied practice. Industry related experience such as that received from a co-op or internship program can better prepare students to assume professional positions after graduation [32], [36]. Practical work integrated learning experiences can be connected through establishing partnerships with local industries.

Additionally, in view of the speed with which technology is evolving, it also becomes necessary that academic programs reflect these changes as much as possible. The curriculum needs to include the latest technologies like AI, ML, Big Data, Cloud Computing and should be subject to regular reviews. This would help graduates to not only be knowledgeable of current technologies, but also learn how to adapt as future advancements arise [29], [33]. Having elective courses or specializations in these can provide students with more specialized knowledge and skills.

Additionally, in a rapidly changing technological environment, ongoing professional development is absolutely necessary. This can be helped by an industry willing to create programs, workshops and certifications for students as well as current employees. This helps ensure a skilled and adaptable workforce [29], [33]. Offering lifelong learning opportunities so employees remain up-to-date with industry trends

Finally, then there is an element of industry participation whereby professional bodies can contribute to developing curriculum by determining which are the competencies or even knowledge that has relevance in the current job market. Such engagement makes it possible to ensure that graduates have the necessary skills for employability [24], [37].

6.1 Limitations and Future Work

There were a number of limitations that should be acknowledged. The size of the sample used in the study is relatively small, which included senior students from Atilim University only. What is being captured below may not be capturing much of the larger student population or what is happening at other institutions. Future research should involve a larger and more varied groups of participants for improved generalization. Moreover, self-reported data need to be interpreted with caution as they may lead to social desirability bias through overestimation or underestimation of skills and knowledge. Self-reports paired with objective assessments might more accurately reflect student readiness. Finally, the data pertain to the University and are not necessarily generalizable to other higher education institutions or contexts. The specific context and curriculum at this University possibly affect the results.

Future research, therefore, should build on these limitations and endeavor to explore other areas fully. Moreover, including students from a wide-range of institutions and increasing the sample size will produce more generalizable results regarding academic-industry gap in SE education. A larger sample size can allow more trends and differences among various academic settings or regions to be found. Long-term monitoring of students from their academic roots into the early career stage would also deliver useful information about how effective curricular changes are overall, and ultimately whether these changes predict professional success. With longitudinal data, we can see how confident graduates feel entering the workforce and what areas of their educational experience were most useful or lacking.

There is also a need for more research to be carried out on the uptake of non-technical skills as part of a curriculum and their readiness for industry. For example, project management, communication and teamwork. Future research can look at various pedagogical styles and curricular frameworks to assess which ones are most effective in developing such competencies. Furthermore, investigating how continuous professional development and lifelong learning contribute to narrowing the skills gap

has potential as a further avenue of alignment between education and industry. This is to give an idea of how support with continuous education and training might greatly influence above-average progression done earlier in his/her career, as well a competency sustainable over time.
In various sectors in Software Engineering the requirement could differ. This focus would likely be appropriate for follow-up studies that could explore postsecondary preparation in the pipeline to particular industries – finance, healthcare, entertainment (for example) and customize curriculum recommendations accordingly. In addition, with technology changing so quickly, research needs to keep up on how new technologies are being introduced into the curriculum. As perspectives shift, this ensures that academic programs stay fresh and students are learning the most current content. Work in this direction would complement the new analysis found here to develop a more nuanced and organic theory of how we might best help students prepare for careers as software engineers. This holistic style of teaching could solve both the issues today — that we already missed out a lot in our curriculum and are not skilled enough, as well as guarantee future readiness bringing up students who can adapt to whatever comes their way quickly with increasing rapid changes in tech landscape. Through ongoing collaboration with industry, the integration of new technologies, and a focus on attaining a broader range of skills including both technical and non-technical competencies, educational programs can make significant strides in building more ready graduates.

Finally, future work needs to be directed on these lines so that superior educational strategies are formulated and the academic training provided aligns well with what is in practice/expected by industry in software engineering. A graduated approach to something like that will work toward producing some of the highest caliber software engineers available: top-notch, well-adjusted entry-levels who can get a job done and innovate by turning ideas into workflow.

REFERENCES

[1] D. Oguz and K. Oguz, “Perspectives on the Gap Between the Software Industry and the Software Engineering Education,” *IEEE Access*, vol. 7, pp. 117527–117543, 2019.

[2] F. Dobslaw, K. Angelin, L. Öberg, and A. Ahmad, “The Gap between Higher Education and the Software Industry — A Case Study on Technology Differences,” in ECSEE 2023: European Conference on Software Engineering Education, Jun. 2023, pp. 11–21.

[3] H. Saiedian, “Bridging Academic Software Engineering Education and Industrial Needs*,” Computer Science Education*, vol. 12, no. 1–2, pp. 5–9, Mar. 2002.

[4] O. Cico, L. Jaccheri, A. Nguyen-Duc, and H. Zhang, “Exploring the intersection between software industry and Software Engineering education - A systematic mapping of Software Engineering Trends,” *Journal of Systems and Software*, vol. 172, p. 110736, Jul. 2020.

[5] 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*, vol. 22, no. 1, pp. 1–21, Mar. 2022.

[6] A. Siddique, G. M. Butt, A. Zahid, Q. N. Naveed, and M. T.-H. Alouane, “Analyzing Software Industry Trends to Improve Curriculum,” *IEEE Access*, vol. 12, pp. 22510–22523, Jan. 2024.

[7] P. Bourque and R. E. Fairley, “SWEBOK : guide to the software engineering body of knowledge”. Los Alamitos, Ca: *IEEE Computer Society*, 2014.

[8] S. Ludi and J. Collofello, “An analysis of the gap between the knowledge and skills learned in academic software engineering course projects and those required in real: projects,” in *Frontiers in Education Conference*, IEEE, Nov. 2002, vol.1, p. T2D-8.

[9] C. Sarin, “Analyzing Skill Gap between Higher Education and Employability,” *Research Journal of Humanities and Social Sciences*, vol. 10, no. 3, p. 941, 2019.

[10] M. Daneva, C. Wang, and P. Hoener, “What the Job Market Wants from Requirements Engineers? An Empirical Analysis of Online Job Ads from the Netherlands,” in *International Symposium on Empirical Software Engineering and Measurement*, IEEE, Nov. 2017, pp. 448–453.

[11] A. Alabbadi and R. Qureshi, “The Proposed Methods to Improve Teaching of Software Engineering,” *Modern Education and Computer Science*, vol. 7, pp. 13–21, 2016.

[12] N. R. Aljohani, A. Aslam, A. O. Khadidos, and S.-U. Hassan, “Bridging the skill gap between the acquired university curriculum and the requirements of the job market: A data-driven analysis of scientific literature,” *Journal of Innovation & Knowledge*, vol. 7, no. 3, Jul. 2022.

[13] D. Jackson, “Modelling graduate skill transfer from university to the workplace,” *Journal of Education and Work*, vol. 29, no. 2, pp. 199–231, Apr. 2014.

[14] A. Ayofe and A. Ajetola, “Exploration of the Gap Between Computer Science Curriculum and Industrial I.T Skills Requirements,” *International Journal of Computer Science and Information Security, IJCSIS*, vol. 4, no. 1, 2, 2009.

[15] M. A. Mardis *et al.*, “Assessing alignment between information technology educational opportunities, professional requirements, and industry demands,” *Education and Information Technologies*, vol. 23, no. 4, pp. 1547–1584, Dec. 2017.

[16] P. Kamthan and N. Shahmir, “On Software Projects in Academia and Industry from a Perspective of Software Engineering Education,” in *International Conference on Computational Science and Computational Intelligence (CSCI)*, IEEE, Dec. 2019, pp. 34-39.

[17] S. Ouhbi, A. Idri, J. L. Fernández-Alemán, and A. Toval, “Requirements engineering education: a systematic mapping study,” *Requirements Engineering*, vol. 20, no. 2, pp. 119–138, Nov. 2013.

[18] N. Qamar, M. Hafiza, F. Kiran, S. Ahmad, B. Saeed, and Abid, “Software Engineering Fresh Graduates Career Choices and Software Industry Demands: An Empirical Analysis,” *IJCSNS International Journal of Computer Science and Network Security*, vol. 20, no. 12, p. 72, 2020.

[19] J. M. Hills, G. Robertson, R. Walker, M. A. Adey, and I. Nixon, “Bridging the Gap Between Degree Programme Curricula and Employability Through

Implementation of Work-related Learning," *Teaching in Higher Education*, vol. 8, no. 2, pp. 211–231, Apr. 2003.

[20] P. Naur and B. Randell, Eds., "Software engineering: report on a conference sponsored by the NATO Science Committee," in *NATO SOFTWARE ENGINEERING CONFERENCE*, Scientific Affairs Division, NATO, 1969.

[21] O. Nesterenko, "Technological trends & software engineering education: a systematic review study," *Problems in programming*, no. 3–4, pp. 107–116, Dec. 2022.

[22] I. Gimenes, L. Barroca, and E. Barbosa, "The future of human resources qualifications in Software Engineering - meeting demands from industry and benefiting from educational and technological advances," in *Brazilian Symposium on Software Engineering (SBES)*, IEEE, 2012, pp. 181–185.

[23] O. Nesterenko, "Computing Education & Technological Trends: A Systematic Review Study," in *the 13th International Scientific and Practical Programming Conference UkrPROG 2022*, CEUR Workshop Proceedings, 2022, pp. 101–112.

[24] V. Garousi, K. Petersen, and B. Ozkan, "Challenges and best practices in industry-academia collaborations in software engineering: A systematic literature review," *Information and Software Technology*, vol. 79, pp. 106–127, Nov. 2016.

[25] B. Suri, N. Jatana, and M. Tomer, "Towards Advancement of Education in Software Engineering," in *2015 IEEE 3rd International Conference on MOOCs, Innovation and Technology in Education (MITE)*, IEEE, 2015, pp. 208–212.

[26] M. Ntinda, M. Apiola, and E. Sutinen, "Mind the Gap: Aligning Software Engineering Education and Industry in Namibia," in *2021 IST-Africa Conference (IST-Africa)*, IEEE, May 2021, pp. 1–8.

[27] K. Kuusinen and S. Albertsen, "Industry-Academy Collaboration in Teaching DevOps and Continuous Delivery to Software Engineering Students: Towards Improved Industrial Relevance in Higher Education," in *IEEE/ACM International Conference on Software Engineering: Software Engineering and Education Track (ICSE-SEET)*, IEEE, May 2019, pp. 23-27.

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

[29] A. Mishra and D. Mishra, "Industry Oriented Advanced Software Engineering Education Curriculum," *Croatian Journal of Education*, vol. 14, no. 3, Apr. 2012.

[30] H. Karimi and A. Pina, "Strategically Addressing the Soft Skills Gap Among STEM Undergraduates," *Journal of Research in STEM Education*, vol. 7, no. 1, pp. 21–46, Jul. 2021.

[31] C. Chiru, S. G. Ciuchete, G. G. L. Sztruten), and E. P. Sandor), "A Cross Country Study on University Graduates Key Competencies. An Employer's Perspective," *Procedia - Social and Behavioral Sciences*, vol. 46, pp. 4258–4262, 2012.

[32] V. Garousi, G. Giray, E. Tuzun, C. Catal, and M. Felderer, "Closing the Gap Between Software Engineering Education and Industrial Needs," *IEEE Software*, vol. 37, no. 2, pp. 68–77, Mar. 2020.

[33] C. O'Leary, D. Lawless, D. Gordon, L. Haifeng, and K. Bechkoum, "Developing a Software Engineering Curriculum for the Emerging Software Industry in China," in *19th Conference on Software Engineering Education and Training, CSEE and T 2006*, IEEE, May 2006.

[34] E. Marasco *et al.*, "A collaborative autoethnographic analysis of industry-academia collaboration for software engineering education development," in *Proceedings of the Canadian Engineering Education Association (CEEA)*, Canadian Engineering Education Association, Nov. 2022.

[35] A. Almaleh, M. A. Aslam, K. Saeedi, and N. R. Aljohani, "Align My Curriculum: A Framework to Bridge the Gap between Acquired University Curriculum and Required Market Skills," *Sustainability*, vol. 11, no. 9, pp. 1–13, May 2019.

[36] F. Gurcan and C. Köse, "Analysis of software engineering industry needs and trends: implications for education," *International Journal of Engineering Education*, vol. 33, no. 4, pp. 1361–1368, Jan. 2017.

[37] R. Monteiro, M. Souza, S. Oliveira, C. Portela, and C. Lobato, "The Diversity of Gamification Evaluation in the Software Engineering Education and Industry: Trends, Comparisons and Gaps," in *Proceedings of the 43rd International Conference on Software Engineering: Joint Track on Software Engineering Education and Training*, IEEE, May 2021.

APPENDIX A

SURVEY TO UNDERSTAND THE SKILL GAPS

Purpose: This survey aims to evaluate how well the Software Engineering program at [University Name] prepares students for the demands of the software industry. We'll focus on skills outlined in the Software Engineering Body of Knowledge (SWEBOK) as well as emerging technologies.

Objectives: This research will help us identify skill gaps, assess curriculum effectiveness, and gather recommendations to enhance our program.

Target Audience: This survey is designed for senior-level software engineering students (4th year and above).

Confidentiality: All responses will be anonymous and used only for research purposes.

Section 1: Participant Demographics

- Year of Study:
  - 3rd Year
  - 4th Year
- Degree Program:
  - Bachelor of Science in Software Engineering
  - Master of Science in Software Engineering (if applicable)
- Internship/Work Experience:
  - Yes
  - No
  - If Yes, briefly describe the type of experience:

Section 2: SWEBOK and Emerging Technologies Skill Assessment

- **SWEBOK Knowledge Areas:** Please indicate your level of familiarity with each SWEBOK Knowledge Area, using a scale of 1 (Not at all familiar) to 5 (Extremely familiar):
  - Requirements Engineering
  - Software Design

- Software Construction
- Software Testing
- Software Maintenance
- Software Configuration Management
- Software Engineering Management
- Software Engineering Process
- Software Engineering Models and Methods
- Software Quality

- **Industry-Essential Skills:** Please rate your current level of preparedness in the following skills, using a scale of 1 (Not at all prepared) to 5 (Extremely well prepared):
  - **Technical Skills**
    - Proficiency in core programming languages (e.g., Java, Python, C++)
    - Object-Oriented Programming (OOP) concepts
    - Version control systems (e.g., Git)
    - Web development frameworks (e.g., React, Angular, Django)
    - Cloud Computing (e.g., AWS, Azure)
    - Basic understanding of AI/ML concepts
  - **Soft Skills**
    - Written and oral communication
    - Problem-solving and critical thinking
    - Teamwork and collaboration
    - Time management and adaptability
- **Skill Acquisition:** For skills where you indicated a preparedness level lower than 4, please select the primary method(s) through which you acquired knowledge:
  - Academic Coursework
  - Personal Projects
  - Internships/Work Experience

- Online Courses/Tutorials
- Other (Please specify):

Section 3: Curriculum and Skill Gaps

- **Curriculum Alignment:** Rate how effectively your software engineering program has addressed each SWEBOK Knowledge Area, using a scale of 1 (Not Addressed) to 5 (Very Well Addressed):
  - Requirements Engineering
  - Software Design
  - Software Construction
  - Software Testing
  - Software Maintenance
  - Software Configuration Management
  - Software Engineering Management
  - Software Engineering Process
  - Software Engineering Models and Methods
  - Software Quality
- **Skill Gaps:** List any SWEBOK-aligned skills or emerging technologies that you feel were underemphasized in your program.
- **Program Strengths:** Which aspects of your software engineering curriculum do you feel were most valuable in preparing you for industry expectations?

Section 4: Bridging the Gap

- **Experience Impact:** Rate the effectiveness of the following in developing essential skills and knowledge of emerging technologies, using a scale of 1 (Not at all helpful) to 5 (Extremely helpful):
  - Internships/Co-ops
  - Academic Coursework
  - Personal Projects
  - Internships/Work Experience
  - Online Courses/Tutorials
- **University Support:** Do you feel the university provides sufficient guidance and resources to help students secure relevant internships or work experience?

- Yes
- No
- Please elaborate:

Section 5: Emerging Technologies

- **Awareness:** Are you familiar with the role of Artificial Intelligence (AI) and Machine Learning (ML) in the context of software development?
    - Yes
    - No
    - Somewhat
- **Importance:** Do you believe that incorporating concepts of AI and ML is important in software engineering education?
    - Yes
    - No
    - Somewhat
    - Please explain:

Section 6: Overall Preparedness and Recommendations

- **Self-Assessment:** Please rate your overall level of preparedness for an entry-level software engineering role, using a scale of 1 (Poorly Prepared) to 5 (Extremely Well-Prepared).
- **Recommendations:** What changes or additions to the Software Engineering curriculum would you recommend to better address industry needs and bridge identified skill gaps?

Section 7: Soft Skills

- **Soft Skill Preparedness:** Rate your current level of preparedness in the following soft skills essential for software engineers, using a scale of 1 (Not at all prepared) to 5 (Extremely well-prepared):
    - Technical Communication (Written and Oral)
    - Problem-Solving
    - Teamwork and Collaboration
    - Self-Learning and Adaptability
    - Time Management
    - Professionalism (Work ethic, reliability, and taking responsibility)

APPENDIX B.1.
CURRICULUM OF A SOFTWARE ENGINEERING DEPARTMENT IN A UNIVERSITY

1. Semester

Course Code	Course Name	ECTS
CMPE113	Computer Programming I	4
HIST111	Principles of Atatürk and History of Turkish Revolution I (in English)	2
PHYS101	General Physics I	6
ENG101	English For Academic Purposes I	3.5
CMPE109	Fundamentals of Computing	2.5
MATH151	Calculus I	7
CHE105	General Chemistry	5
KRY111	Career Planning	1
Total Credits:		31

2. Semester

Course Code	Course Name	ECTS
HIST112	Principles of Atatürk and History of Turkish Revolution II (in English)	2
PHYS102	General Physics II	6
HIST221	History of Civilization	3
ENG102	English for Academic Purposes II	3.5
SE112	Introduction to Software Engineering	3.5
CMPE114	Computer Programming II	5
MATH152	Calculus II	7
Total Credits:		30

3. Semester

Course Code	Course Name	ECTS
MATH275	Linear Algebra	6
ENG201	English for Academic Purposes III	3
SE221	Software Requirements Engineering	6
CMPE251	Discrete Computational Structures	7
CMPE225	Object-Oriented Programming	8
Total Credits:		30

4. Semester

Course Code	Course Name	ECTS
MATH276	Differential Equations	6
IE220	Probability and Statistics	5
ENG202	English for Academic Purposes IV	3
SE212	Human-Computer Interaction	8
CMPE226	Data Structures	8
Total Credits:		30

5. Semester

Course Code	Course Name	ECTS
SE399	Summer Practice I	6
CMPE325	Study of Programming Languages	6
CMPE341	Database Design and Management	7
SE345	Software Quality Assurance	7
SE321	Object-Oriented Analysis and Design	7

ENG301	English for Occupational Purposes I	3
Total Credits:		30

6. Semester

Course Code	Course Name	ECTS
SE324	Software Project Management and Economics	5
SE328	Algorithms and Optimization Methods	5
SE322	Software Architecture	5
SE344	Systems Software Validation and Testing	6
CMPE338	Computer Networks and Operating Systems	6
ENG302	English for Occupational Purposes II	3
Total Credits:		30

7. Semester

Course Code	Course Name	ECTS
TURK401	Turkish Language I	2
SE493	Project Orientation	5
SE499	Summer Practice II	10
	Area Elective	5
	Area Elective	5
	Area Elective	5
	General Elective	4
	General Elective	4
Total Credits:		30

8. Semester

Course Code	Course Name	ECTS
TURK402	Turkish Language II	2
SE494	Senior Project	9
	Area Elective	5
	Area Elective	5
	Area Elective (Multidisciplinary Courses)	5
	General Elective	4
Total Credits:		30

APPENDIX B

CONTENT OF THE COMPULSORY SE COURSES IN THE UNDERGRADUATE SE CURRICULUM

The Software Engineering Curriculum in a university is provided in the following link: **https://www.atilim.edu.tr/en/se/page/2284/curriculum**

1. SE112 Introduction to Software Engineering

Course Details and Content:

Computer Software and its types, Software Engineering discipline and its fundamental concepts, Software Process models, Requirements Engineering concepts, system modeling, architectural design, design and implementation, software testing, software evolution and maintenance, project management, quality and configuration management.

Weekly Subject Distribution:

Week	Subjects
1	Computer Software and its types
2	Software Engineering discipline and its fundamental concepts
3	Project Management and Planning (Lab: Pert Charts and Gantt Diagrams)
4	Software Requirements
5	Requirements Engineering Concepts (Lab: Requirements Gathering)
6	Software Design (High-Level) (Lab: Drawing Use-Case diagrams using a CASE tool)
7	Software Design (Low-Level) (Lab: Drawing Activity diagrams using a CASE tool)
8	Software Development
9	Testing
10	Metrics
11	Software Evolution and Maintenance
12	Process Models (Lab: Designing mockups using a CASE tool)
13	Process Models

14	Final Examination Period

2. **SE221 Software Requirements Engineering**

Course Details and Content:

Basics of software requirement, requirements from the customer's perspective, applications in requirements engineering, the role of requirements analyst, forming product vision and project scope, understanding customer and user requirements, documenting requirements, risk reduction through prototyping, setting requirement priorities, validating

Weekly Subject Distribution:

Week	Subjects
1	Software Process and SDLC
2	Software Requirements and Types
3	Requirements Engineering Processes
4	Problem analysis and Requirements Inception
5	Requirements Elicitation
6	Requirements Analysis and Specification
7	Use Case Models
8	Requirements Negotiation and Prioritization
9	User Interface Design
10	Data Modelling
11	Requirements Verification and Validation
12	Requirements Management, Tracing requirements and managing change

3. **SE212 Human-Computer Interaction**

Course Details and Content:

Basic principles of user interfaces, human capabilities and limitations; usability paradigms and principles, user and task analysis, design process, prototyping and evaluation, color and typography, new user interface technologies.

Weekly Subject Distribution:

Week	Subjects
1	Basics of Human Computer Interaction
2	How Human Mind Works I
3	How Human Mind Works II
4	Interaction and Interaction Styles
5	Page Design
6	Navigation
7	Color
8	Typography
9	Accessibility
10	Identifying Needs and Establishing Requirements
11	User Testing
12	Ubiquitous Computing, Wearables, AR & VR

4. SE345 Software Quality Assurance

Course Details and Content:

Introduction to software quality and assurance; software quality metrics; construction of software quality assurance; configuration management; software validation and verification; reviews, inspection and audits; software process improvement models; software testing strategies and testing techniques; defect reporting and removal; software

Weekly Subject Distribution:

Week	Subjects
1	Introduction to software quality and assurance
2	Software quality factors
3	Overview of components of software quality assurance system
4	Integrating quality activities in project life cycle
5	Reviews, Inspection and Audits, Procedures and work instructions

6	Software Testing Strategies
7	Software Testing Implementation
8	Configuration Management
9	Software Quality Metrics
10	Software Quality Metrics
11	Cost of software quality
12	SQA process standards and SQA unit

5. SE321 Object-Oriented Analysis and Design

Course Details and Content:
Fundamentals of object-orientation, object-oriented (OO) modeling using UML, Experimentation in OO analysis: identification of use cases and objects, experimentation in OO design: class hierarchies, implementation in OO programming, design pattern (overview), case study to reinforce the underlying concepts.

Weekly Subject Distribution:

Week	Subjects
1	Information Systems-What are they?
2	Challenges in IS Development Meeting the Challenges
3	Fundamentals of Object-orientation
4	Modeling Concepts
5	Requirement Elicitation
6	Requirement Modeling
7	Refining Requirement Modeling
8	Object Interaction
9	Specifying Operations, Specifying Control
10	System Architecture
11	System Design
12	Detailed Design
13	Design Pattern

6. SE324 Software Project Management and Economics

Course Details and Content:

Introduction to software project management; overview of project planning; selection of appropriate project approach; activity planning, resource allocation, project tracking and control; contract management; managing people and organizing teams; software quality assurance; configuration management; various tools of software project management;

Weekly Subject Distribution:

Week	Subjects
1	Introduction to Software Project Management
2	Intro to PM
3	Project Evaluation and Programme Management
4	Organizational Structure
5	Defining Projects
6	Software Effort Estimation
7	Software Effort Estimation
8	Project Networks
9	Project Networks
10	Schedule Resources and Costs
11	Reduce Duration
12	Managing Risk
13	Project Manager, Teams, Virtual Project Management
14	Agile Project Management

7. SE328 Algorithms and Optimization Methods

Course Details and Content:

Design and analysis of algorithms; mathematical complexity of algorithms; master theorem; decrease-and-conquer; divide-and-conquer; transform-and-conquer; introduction to some optimization techniques; dynamic programming; greedy technique; iterative improvement; coping with limitations of algorithm power.

Weekly Subject Distribution:

Week	Subjects
1	The role of algorithms and Getting Started
2	Growth of Functions, Asymptotic Notation
3	Solving Recurrences: Substitution Method
4	Solving Recurrences: Recursion-Tree Method, Master's Method
5	Brute Force and Exhaustive Search
6	Decrease-and-Conquer
7	Divide-and-Conquer
8	Transform-and-Conquer
9	Dynamic Programming
10	Dynamic Programming
11	Greedy Algorithms
12	Greedy Algorithms
13	Iterative Improvement: The Simplex Method
14	Limitations of Algorithm Power, Coping with the Limitations of Algorithm Power, P, NP, NP-Complete Problems

8. SE322 Software Architecture

Course Details and Content:

Introduction to software architecture, architecture business cycle, creating an architecture, introducing a case study, understanding and achieving quality, design, document and reconstruct software architecture, methods in architecture evaluation, quantitative approach to architecture design decision making, software product lines

Weekly Subject Distribution:

Week	Subjects
1	Understanding Software Architecture
2	Introducing Case Study
3	Software Quality Attributes
4	Middleware Architectures and Technology
5	Software Architecture Process
6	Documenting a Software Architecture

7	Case Study Design
8	Software Product Lines
9	Aspect Oriented Architectures
10	Model-Driven Architecture
11	Service Oriented Architecture & Technologies
12	Semantics Web
13	Software Agents

9. SE344 Systems Software Validation and Testing

Course Details and Content:

Software testing and its essentials, risk management, verification and validation, static V&V techniques and their comparison, software testing strategy and techniques, software testing tools, configuration management, software measurement and metrics related with testing.

Weekly Subject Distribution:

Week	Subjects
1	Giriş ve bilgilendirme
2	Basic Concepts
3	Basic Concepts
4	Unit Testing
5	Control Flow Testing
6	Control Flow Testing
7	Data Flow Testing
8	Midterm Exam
9	Data Flow Testing
10	Integration Testing
11	System Testing Categories
12	Functional Testing
13	Functional Testing