International Journal of Engineering Technology and Management Sciences
Website: ijetms.in Issue: 2 Volume No.7 March - April – 2023
DOI:10.46647/ijetms.2023.v07i02.098 ISSN: 2581-4621

A SURVEY ON PERSPECTIVES ON THE GAP BETWEEN THE SOFTWARE INDUSTRY AND THE SOFTWARE ENGINEERING EDUCATION

G.Kishorekumar[1] ,Mrs.P.Uma M.E[2]
[1]PG Scholar,Department of Computer Science and Engineering, Nandha Engineering College(Autonomous),Erode,Tamilnadu,India
[2]Assisstant Professor,Department of Department of Computer Science and Engineering, Nandha Engineering College(Autonomous),Erode,Tamilnadu,India

ABSTRACT
In 1989, the gap between the software industry and software engineering education was first mentioned. Its existence has been regularly reported on since then, and solutions to close it have been proposed. However, after thirty years, all efforts to close the gap have failed. In this study, we argue that the gap between industry and academia exists for a variety of interconnected reasons. To take a broader look at the problem from the standpoint of all related entities, we provide a detailed overview of the profession and identify the entities, extract the causes that stem from these entities, and discuss what each entity should do, report and analyse the results of a questionnaire that was administered to students and recent graduates, highlight the highlights of interviews with students, recent graduates, and academics, We also contribute to finding solutions by considering all entities involved, which allows us to reach out to all of them and find out what they can do to acknowledge and close the gap. Our research concludes that the gap requires constant attention and hard work from all parties involved, and that everyone should be on the lookout for new technologies, learn to embrace changes, and adapt to them in order to keep the gap to a minimum.
KEYWORDS: Software Engineering, Education Gap, Engineering Curriculum

1. INTRODUCTION
1.1 SOFTWARE ENGINEERING
Software engineering education is becoming a distinct and mature discipline. As a result, numerous studies are being conducted to provide guidelines for curriculum design. These guidelines are primarily concerned with core and foundation courses. The current issues in software engineering education programmes are summarized in this paper. It also recommends the following important dimensions as essential components of software engineering education: interdisciplinary skills, practice experience, communication, continuing education skills, and professionalism. These dimensions are not specifically addressed in current guidelines and studies. Although there may be other dimensions to consider in software engineering education, we believe that the ones proposed are critical because software engineering evolves faster than any other engineering discipline. This study also includes an evaluation of these dimensions through a survey of some major universities' undergraduate software engineering programmes.
1.2 EDUCATION GAP
The term "education gap" has two connotations. The first meaning of the education gap is the leap in work experience required to complete your education. This educational disparity is beneficial. Companies have the authority to educate their employees. They encourage and support their employees in obtaining relevant education. Employee education is sometimes paid for by the company. The second meaning of the education gap is when you leave after completing your education or graduating. The gap could be due to personal reasons. Some people take a break in order to improve their concentration and focus. You can use the time between exams or certification courses to study.

DOI:10.46647/ijetms.2023.v07i02.098 ISSN: 2581-4621

1.3 ENGINEERING CURRICULUM

Engineering curricula must be reviewed and improved on a regular basis to ensure the quality and relevance of undergraduate degree programmes. The process of reviewing curriculum, on the other hand, is difficult on many levels and can appear overwhelming to those leading the review and implementing subsequent curriculum changes. Particular challenges include: involving all academic staff in the process to promote ownership of change; developing processes to guide the review toward improvements in the quality of content and students' experiences of being taught; and remaining mindful of contextual factors such as university policy, the needs of external stakeholders, and limited time and money for teaching.

2. LITERATURE SURVEY

2.1 ENGINEERING IMPLEMENTATION IN SOFTWARE ENGINEERING EDUCATION

Jeff Offutt et al. propose in this study Can you envision physics and mechanical engineering together? Could you see ME professors undertaking research in physics? Can you picture a successful ME programme within the walls of a physics department? You probably can't, but that's where it started. Over a century ago, colleges had physics departments and programmes that taught students how to apply physics concepts to create bridges, dams, automobiles, aero planes, and electrical circuits. Over time, physics gave way to subjects like mechanical engineering, civil engineering, electrical engineering, mining engineering, and aerospace engineering. Computer science traditionalists become fervent believers of "individual learning," avoiding cooperation at any costs. This is because teachers are concerned about plagiarism, which is logical considering how easy it is to replicate on computers.

2.2 SOFTWARE COMPLEXITY METRICS SURVEY

Sheng Yu et al. suggest in this study As software development proceeds, the software's size rises to the point that we can no longer readily handle it. Some measures for quantifying software complexity have been presented in recent years. The purpose of this article is to offer a thorough understanding of the software complexity measure. Some of the first and most efficient software complexity measures described and researched include lines of code (LOC), Halstead Complexity Metric (HCM), and Cyclomatic Complexity Metric (CCM). Following that, various alternative techniques based on the aforementioned traditional metrics are described. There is also a comparison and link between these software complexity indicators. The software complexity measure is quickly becoming an important part of software engineering. With so much attention on software quality these days, it's natural to expect the programme complexity measure to be promoted to its appropriate place.

2.3 SOFT SKILLS IN SOFTWARE DEVELOPMENT TEAMS

In this work, Gerardo Matturro and colleagues offered In addition to technical expertise and experience, team members' "soft skills" are critical in software engineering projects. Researchers and practitioners have gained more interested in this area in recent years. The outcomes of a field research in which we interviewed 35 software engineering practitioners from software businesses in Uruguay to learn about the soft skills that the leader and other members of software development teams appreciate the most are presented in this article. As a consequence, team leaders place the most importance on leadership, communication skills, customer orientation, interpersonal skills, and collaboration, while team members place the highest value on analytical, problem-solving, dedication, responsibility, readiness to learn, motivation, and cooperation. According to the data gathered from those interviews, the perspectives of team leaders and team members on the top five most valued soft skills for the role of team leader of a software development team are congruent in that Leadership, Communication skills, Customer orientation, Interpersonal skills, and Teamwork are the top five most valued soft skills for that role.

2.4 PAYING ATTENTION TO SOFTWARE DEVELOPERS IN THEIR EARLY CAREER

Michelle Craig et al. suggest in this study Previous study has found that fresh college graduates who enter the software development business suffer early hurdles owing to considerable discrepancies between their coding expertise in school and what is required of them on the job. To study the

DOI:10.46647/ijetms.2023.v07i02.098 ISSN: 2581-4621

difference between academic and commercial software development, we interviewed twenty early career software engineers with four-year degrees in CS. We give a six-theme analysis of these interviews, including passages written by the developers. We end with suggestions for closing the gap so that our students are better equipped to begin their professions. Many new software engineers, according to our interviews, feel underprepared in a variety of areas, including tool use, communications, cooperation, and working on large, long-lasting, open-scoped, sophisticated software systems.

2.5 A STRATEGIC INVITATION TO BRIDGING THE GAP BETWEEN ENGINEERING SCHOOLS AND INDUSTRY

The long-standing divide (the Gap) between engineering education and industry has been thoroughly established in the literature, according to Kamel Alboaouh et al. in this research. This article addresses the Gap philosophically from three perspectives: the distinction between science and engineering, the reasons of the Gap, and the approach for bridging the Gap. The bachelor's level of education is the emphasis of this study. The gap is created by a lack of input from industry to engineering colleges. This work contributes to the philosophy of engineering education by suggesting a deliberate strategy to bridge the Gap. The planned project is expected to provide the following results: Engineering majors are being reclassified, and the knowledge taught in these institutions is being redesigned to fit market demands. These results might be obtained by first enabling engineers to write about their own industry experiences in engineering journals, allowing schools to develop a grasp of what is going on in the business through their writing. Second, accreditation organisations such as ABET should award engineering schools a higher rating based on their efforts to decrease the gap over time. Finally, PhD research has the potential to improve undergraduate teaching. A PhD student could spend two years working for a firm before returning to school to submit their findings and observations.

2.6 ENGINEERING STUDENTS' VIEWS ON ENGINEERING PRACTICE PREPARATION

Eileen Goold et al. suggest in this study The notion of the phrase global engineer, where the job of the engineer has become rather wide, is a recurrent subject in the engineering scholarly literature. The goal of this study is to investigate engineering students' opinions of their future jobs as well as their readiness for professional practise. This is important information for engineering instructors to know since students who are more interested in professional practise have higher cognitive engagement. Furthermore, many graduating engineers struggle to move into engineering practise. The methodology is multi-method. The degree competences required and gained by electrical engineering students at the Institute of Technology Tallaght Dublin are studied quantitatively. Students' perceptions of their future careers are investigated qualitatively. According to the data, pupils' learning is mainly dependent on academic viewpoints. There are gaps between the skills necessary for engineering practise and the skills gained.

2.7 COLLABORATIONS: BRIDGE THE INDUSTRY-ACADEMY GAP

GAP is used in this paper. Offering a course to industry, as advocated by KATHY BECKMAN et al., does not, in our opinion, constitute as partnership, nor does a mere transfer of monies or an employment opportunity. Collaboration is defined as a joint endeavour in which each partner provides specialised products and services toward a common aim. Such collaborations are proven to be more productive and practical; the key is to adequately organize and communicate openly in order to suit all interests. We've all been a part of fruitful industry-university relationships. We built a cooperation model based on our cumulative experience, which includes rules for organizing an industry advisory board, identifying success criteria, and arranging events and follow-up. This paradigm, as well as thorough descriptions of three successful industry-university relationships, is offered here.

2.8 SEER: LAYING THE GROUNDWORK FOR EDUCATION IN SOFTWARE ENGINEERING

Donald J. Bagert et al. propose in this study There have been a number of unique, trailblazing efforts connected to the progress of software engineering as a profession and an academic field throughout the previous decade. The majorities of these initiatives, however, have either been done or are slated to be completed by 2004, and it is unclear what the software engineering education community should

DOI:10.46647/ijetms.2023.v07i02.098 ISSN: 2581-4621

do next to expand on this effort. This half-day workshop will bring together stakeholders in software engineering education (both academic and industrial) to explore the matter and design a Software Engineering Education Roadmap (SEER) that might offer this community with much-needed direction in the coming years. To begin the conversation before to the workshop, a website and email list for SEER was built, and will be used to share the roadmap generated by the participants as well as to continue the debate following the session. The conclusions of the workshop (as well as the outcomes of future action items) will be made available on the SEER website. By June 2004, the roadmap ideas are expected to be in some sort of "final" shape.

2.9 FIRST-JOB FAILURES OF NEW COLLEGE GRADUATES IN SOFTWARE DEVELOPMENT

Andrew Begel et al. propose in this study how young college graduates acquire their first jobs in software development. In what ways do their school experiences prepare them for their new jobs, and in what ways do they struggle to be productive? We present the findings of a "fly-on-the-wall" observational study of eight recent college graduates in their first six months as Microsoft Corporation software developers. We present findings from 85 hours of on-the-job observation of new software developers, including how to programme, write design specifications, and evidence of persistence strategies for problem solving. We also categorise some of the most common problems encountered by new software developers: communication, collaboration, technical, cognition, and orientation.

2.10 ARE COMPUTER SCIENCE AND ENGINEERING GRADUATES READY FOR THE SOFTWARE INDUSTRY? EXPERIENCES FROM AN INDUSTRIAL INDUSTRY STUDENT TRAINING PROGRAM

Eray Tuzun et al. propose in this paper that the term "software engineering" was coined 50 years ago at a NATO conference in 1968. Most established universities' Computer Science programmes should have covered core software engineering topics by now, with others offering specialized degrees. Nonetheless, many practitioners complain about new software engineering hires lacking in skills. This obvious disparity is widening as the industry's demand for software developers develops. One business method for bridging this gap would be for the sector to provide extra training programmes prior to the recruiting process, which could also help employers assess potential individuals. We explain our experiences and lessons acquired while operating a summer school programme targeted at screening new graduates, introducing them to fundamental skills important to the business and industry, and measuring their attitudes toward learning those skills prior to the employment process in this article. According to our experience, such efforts may help both new recruits and businesses.

2.11 PRIMARY OBJECTIVE: INDUSTRY REQUIREMENTS A SWEBOK-BASED CURRICULUM ADAPTATION

S G Ganesh et al. suggest in this study that Siemens Corporate Development Center India (CT DC IN) creates software solutions for Siemens' industries, energy, health-care, and infrastructure & cities sectors. These applications are often crucial in nature, necessitating the use of software practitioners who are well-versed in software engineering. To boost engineers' proficiency, CT DC IN has established an internal curriculum termed "Foundation Program for Software Engineers" (FOCUS), which is a modified version of IEEE's SWEBOK curriculum. The FOCUS programme has taught approximately 500 engineers in the previous three years. In this experience report, we detail the reason for FOCUS, how it was developed to fit the unique needs of CT DC IN, and how the FOCUS programme was implemented throughout the company.

2.12 HOW TO COMPLETE A LITERATURE REVIEW WITH OPEN SOURCE AND FREE SOFTWARE

Joshua M et al. suggest in this paper A thorough literature review is a necessary component of every academic inquiry since it serves as a firm basis for furthering knowledge. However, conducting literature reviews presents several challenges, including: I a lack of access to the literature due to costs, ii) fragmentation of the literature into many sources, a lack of access and comprehensive coverage in many databases and search engines, and iii) the use of proprietary software lock-in strategies for bibliographic software, which can make porting literature reviews between

DOI:10.46647/ijetms.2023.v07i02.098 ISSN: 2581-4621

organisations difficult and costly. These challenges typically result in low-quality literature reviews conducted by a single researcher who is inexperienced with methods to the same study in other subfields, as well as static evaluations that are frequently lost to the scientific community. This study will expand an open source approach to the application of increasing the quality of literature reviews by giving best practises.

2.13 TRANSFORMING ELECTRONIC ENGINEERING EDUCATION THROUGH A HOLISTIC APPROACH

Anthony A et al. suggest in this paper Our diverse team of educators at the Department of Electrical and Computer Engineering is reinventing what it means to teach and learn. Thanks to a five-year "RED" funding from the National Science Foundation, we are effectively tossing away courses in order to tackle the constraints of the present engineering educational system. When we look at the degree as a whole, we perceive it as an integrated system that fosters cooperation among professors and students, rather than a collection of unconnected courses taught by independent (and isolated) staff. The purpose of this study is to present our new organizational and pedagogical paradigm, which focuses on knowledge integration and weaves thematic content threads throughout the curriculum. We also talk about how we went about adopting the new method, as well as the achievements and setbacks we had along the road. Through this effort, we aspire to be a change agent in engineering education. The problem is exacerbated for ECE students because a holistic understanding is highly dependent on their understanding of key foundational concepts in mathematics and science, which are traditionally taught outside of the ECE department, leaving it up to the students to understand and connect ECE topics and foundational content.

2.14 PROBLEM BASED LEARNING IN THE SOFTWARE ENGINEERING CLASSROOM

Ita Richardson et al. argue in this research that software engineering instructors must teach topics that are often difficult for novice students to understand. As a result, it may be advantageous to examine and use non-traditional teaching approaches that might enhance students' learning. We will look at problem-based learning and how it might help students grasp ideas better in this article. Elements that should be present in 'pure' problem-based learning are presented. Then, one of the writers describes how he employed problem-based learning in a class where students were needed to comprehend information flows using software engineering diagramming techniques in order to evaluate and develop computerized information systems. This problem-based learning class was observed and analyzed by the second author. The analysis presented focuses on problem-based learning factors, how they were implemented in class, and the benefits and drawbacks of using problem-based learning in this manner. Finally, the authors discuss how changing the teaching method for a future class that will use problem-based learning could improve the teaching. This modification is expected to enhance students' learning and experience.

2.15 STUDIES IN SOFTWARE ENGINEERING EDUCATION: TOWARDS AN EVALUABLE MODEL

Instead of conventional lectures, Christopher N et al. offer studio-based education, which stresses a physical "home" for students, problem-based and peer-based learning, and mentorship by academic staff. There have been several attempts to incorporate studio-based learning into software engineering education. Given the practical nature of software engineering, this is sensible in many respects. In contrast, attempts at software studios have largely neglected experiences and philosophy from arts and design studio training. As a result, there is some confusion about what the term "studio" truly means, how well the principles transfer to software engineering, and how effective studios are in reality. Software studios cannot be effectively assessed for their influence on student learning until the term "studio" is defined, and best and worst practices cannot be shared among people who manage studios. We confront this issue head on in this study by undertaking a qualitative investigation into what the term "studio" truly implies in the arts and design. We performed 15 interviews with studio professionals and give an analysis and assessment approach here. Our findings show that, while numerous interconnected factors constitute studio education, it is essentially the people and culture

International Journal of Engineering Technology and Management Sciences
Website: ijetms.in Issue: 2 Volume No.7 March - April – 2023
DOI:10.46647/ijetms.2023.v07i02.098 ISSN: 2581-4621

that create a studio. Digital technology, on the other hand, has the potential to harm studios if not properly recognised.

2.16 AN URGE TO DEVELOP SOFT SKILLS IN SOFTWARE ENGINEERING

For many years, we have examined other areas of software engineering, according to Luiz Fernando Carpets et al. in this research; the missing link in engineering software is the soft skills set, which is vital in the software development process. Soft skills are among the most critical parts of software development, although they are usually overlooked by academics and practitioners. One of the key reasons for this omission is because soft skills are commonly connected with social and personality variables like as teamwork, motivation, dedication, leadership, multi-culturalism, emotions, interpersonal skills, and so on. This editorial is a manifesto proclaiming the significance of soft skills in software engineering, with the intention of directing experts' attention to these issues. The success or failure of a software manager is highly contingent on the performance and creations of others.

2.17 A PROFESSOR'S VIEW OF TRAINING SOFTWARE ENGINEERS USING OPEN-SOURCE SOFTWARE

Traditional Software Engineering (SE) courses usually priorities techniques and concepts in limited, controlled environments: naïve projects used as proof-of-concept rather than full-fledged real-world software systems, according to Gustavo Pinto et al. in this research. Although there are obvious benefits to this technique, it does not pay enough focus to teaching students to cope with complicated, non-trivial legacy software problems. To bridge this gap, new SE courses are employing a varied selection of open-source software (OSS) projects to show how these approaches and principles are applied to actual, non-trivial software systems. To better understand the benefits, obstacles, and potential of this move, we interviewed seven SE instructors who adjusted their academic environment to aspire students to comprehend, manage, and grow OSS systems as part of their SE course.

2.18 EMPIRICAL SOFTWARE ENGINEERING STUDIES CURATING MODEL FOR OPEN-SOURCE SOFTWARE PROJECTS

Juan Andrés Carruther et al. argue in this work that software projects are popular inputs in Empirical Software Engineering (ESE), and they are usually chosen without a clear strategy, resulting in biased samples. To circumvent this problem, researchers prefer to use publically accessible datasets instead of creating their own projects. Some datasets, however, are not updated and contain outdated or obsolete versions of projects. Because of significant changes in development processes and technology throughout time, this may raise some questions about representativeness. The major purpose of this study is to design a method model for creating and maintaining a software project dataset with product quality indicators to help with the development of ESE investigations. Finally, we discovered limited attempts to provide a framework or method model for retaining project temporal validity in datasets. Because ongoing changes in development processes and technology might result in different ESE study conclusions, a dataset of software projects produced several years ago is rarely indicative of newly developed programmes. Lewowski and Madeyski (2020) solved issue in part with their changing dataset tool, but they did not explain how to deal with outdated projects.

2.19 INTEGRATING RP TECHNOLOGY INTO THE TENNESSEE TECH DESIGN AND MANUFACTURING CURRICULUM

Ismail Fidan et al. proposes in these study Rapid improvements in computer technology have produced and will continue to offer new possibilities for teachers who teach CAD/CAM technologies. TTU used the NSF-CCLI-A&I programme funding opportunity to adapt and apply effective Rapid Prototyping (RP) experiences and instructional techniques created and proven at other engineering schools. RP capabilities impact the teaching in the fundamental design and manufacturing curriculum. RP adds excitement and realism to the curriculum by letting students to make physical models directly from CAD data. Engineering data like as fit and limited functional testing, labeling, highlighting, and appearance simulation are all sent through the prototype. Students who grasp the facts of the interaction between CAD tools and design principles will be far more sensitive to the realities of the industrial standard in RP. TTU RP goals have been met by including new hands-on laboratory

International Journal of Engineering Technology and Management Sciences
Website: ijetms.in Issue: 2 Volume No.7 March - April – 2023
DOI:10.46647/ijetms.2023.v07i02.098 ISSN: 2581-4621

experiments into two current junior level mandatory courses: CAD for Technology and CNC Machining Practices.

2.20 EDUCATION OF FUTURE SOFTWARE ENGINEERS: PROBLEM-BASED LEARNING FOR SOFTWARE QUALITY

In this research, Louise R et al. argue that Software Engineering graduates should have both technical and soft skills when they enter the industry. Furthermore, software quality is becoming an increasingly significant subject as a result of educational and business needs. Professors of software engineering must apply their research into the classroom. When all of these aspects are considered together, the speaker may face a tough issue. This study explains how problem-based learning, a prominent pedagogical paradigm in medicine and other fields, may be utilized to fulfill these objectives in a single course session. It outlines a research study in which the implementation of problem-based learning inside an M.Sc. Software Engineering Quality Module is analyzed and compared to stated expectations. The inclusion of a topic in a curriculum, on the other hand, gives no instruction as to how the topic should be taught. The typical form of university-level education involves a lecturer standing in front of students and breaking down the course content into a series of topical lectures. In general, the lecturer makes an effort to encourage student discussion on the topic.

3. COMPARATIVE ANALYSIS

Title	Techniques & Mechanisms	Parameter Analysis	Future Work
Putting the Engineering into Software Engineering Education	Computer science, with its strong roots in mathematics, is usually taught using “convergent thinking,” meaning problems have one answer and successful students should tend toward that answer.	Over a century ago, universities had some departments and programs in physics that taught really practical applications in physics, such as how to use principles from physics to build bridges, dams, cars, airplanes, and electrical circuits.	As software engineering continues to move out of the shadow of CS to establish itself as a separate, independent discipline
A Survey on Metric of Software Complexity	An important trend about the metric of the software complexity is the merging several different methodologies	This article aims at a comprehensive survey of the metric of software complexity. Some classic and efficient software complexity metrics, such as Lines of Codes (LOC), Halstead Complexity Metric (HCM) and Cyclomatic Complexity Metric (CCM), are discussed and analyzed first	In future The software complexity metric is becoming an extremely important part of the software engineering. Nowadays, when more and more attentions are focused on the quality of the software, it’s reasonable to believe that, the software complexity metric will be put on its rightful place.
Soft Skills in Software Development Teams	As a result, Leadership, Communication skills, Customer orientation, Interpersonal skills, and Teamwork are the most	Based on the data collected on those interviews, to perform the specific role of team leader of a software	further work, we are now working with the companies involved in this study to investigate

	valued for team leaders, while Analytic, problem-solving, Commitment, responsibility, Eagerness to learn, Motivation, and Teamwork are the most valued ones for team members.	development team, the point of view of both the team leaders and the team members are coincident in that Leadership, Communication skills, Customer orientation, Interpersonal skills, and Teamwork are the top five most valued soft skills for that role.	
Listening To Early Career Software Developers	We claim that this is due to the fundamental differences between the programming activities typically done in university coursework and the coding done in professional software development environments.	However, the interviewer did know two of the subjects and the initial contact for many of the others was from someone they knew from school. So while minimized, this threat was not eliminated.	In spite of nearly two decades since the gap in industry/academic coding experiences was identified and nearly ten years since researchers made recommendations for curricular changes to address the gap, it is still quite wide.
The Gap Between Engineering Schools and Industry: A Strategic Initiative	This problem demonstrates the reason for our first goal, which is to realign the classifications of knowledge taught by engineering schools to match industry needs.	Reclassification of the majors available in engineering schools and redesigning the knowledge taught in these schools to match industry needs.	In future This will be a valuable source for building an appropriate curriculum based on the knee points of different fields of knowledge.
Engineering Students' Perceptions of their Preparation for Engineering Practice	Students' initiative and willingness to learn is very evident in this study. Students are generally confident that they will make good engineers because they work hard and their education covers the standard necessary engineering material.	the benefits of a portfolio of engineering practice, illustrating the practical realities of modern engineering practice, to first year electronic engineering students at ITTD is currently being investigated	This study investigates engineering students' perceptions of their future careers and their preparation for professional practice.
Collaborations: Closing the Industry–Academia Gap	Clearly defining team member roles and understanding each other's expectations can eliminate many future hardships	Industry advisory boards must be appointed with extreme care or industry university relationships will suffer.	You can thus adjust the collaborative effort as needed and identify future opportunities for collaboration.

International Journal of Engineering Technology and Management Sciences
Website: ijetms.in Issue: 2 Volume No.7 March - April – 2023
DOI:10.46647/ijetms.2023.v07i02.098 ISSN: 2581-4621

SEER: Charting a Roadmap for Software Engineering Education	A website and email list for SEER was created in order to start the discussion before the workshop, will be used both to disseminate the roadmap formulated by the participants and continue the dialog after it.	this topic and to outline a Software Engineering Education Roadmap (SEER) which could potentially provide needed direction for this community over the next several years.	Participants can then disseminate this information back to their specific organizations for whatever future use they see fit.
Struggles of New College Graduates in their First Software Development Job	We report on some common misconceptions of new developers which often frustrate them and hinder them in their jobs, and conclude with recommendations to align Computer Science curricula with the observed needs of new professional developers.	The most in-depth studies of new developer experiences in a professional software context.	Our suggestions for curricular reform are a preface for renewed dialogue between the needs of industry and the goals of computer science education.
Are Computer Science and Engineering Graduates Ready for the Software Industry? Experiences from an Industrial Student Training Program	Our experience suggests that such initiatives can be mutually beneficial for new hires and companies alike.	The summer school and diversified the offerings by adding new, more specialized curricula in the areas of Data Science, Test Engineering, Cyber Security, and Enterprise Software (SAP).	In the future, the company will keep track of performance and turnover of the employees in the software engineering track hired through the regular process versus those hired through the summer school process to better understand the advantages and disadvantages of each.
FOCUS: An Adaptation of a SWEBOK-Based Curriculum for Industry Requirements	The feedback clearly indicates that FOCUS has largely met the organizational requirements. The survey results also point towards certain areas of improvement which are being used to tune the FOCUS program.	our experience with FOCUS (Foundation Curriculum for Software engineers) which is an adoption of a SWEBOK-based curriculum	While it would be useful if SWEBOK, as well as the curricula based on SWEBOK, address these shortcomings in future, an organization planning to immediately use a SWEBOK-based curriculum can consider adding these topics to the curriculum.
How to Perform a Literature Review with Free and Open Source Software	Although there are many types and goals of literature reviews, it is found that all of them	an open source approach will be expanded to the application of	It is concluded, that although there are many types and goals of literature reviews, all of

	can be improved using a tool chain of free and open source software (FOSS) and methods.	improving the quality of literature reviews by providing best practices.	them can be improved using a tool chain of free and open source software and methods.
A Holistic Approach to Transforming Undergraduate Electrical Engineering Education	The program has significantly increased students' face-to-face interactions with industry, and survey results reveal that students and industry alike feel the initiative is valuable and important	Enrollments are up in our new Open Option Projects independent study option, and we are seeing increased interest, and diversity of participants, in our Vertically Integrated Projects program.	Through project-based learning, students are able to see how creativity and innovation distinguish engineering from other disciplines, and how their knowledge is driving our technological future.
Problem Based Learning in the Software Engineering Classroom	Fundamental beliefs will be challenged. Building a comprehensive PBL community requires determination and commitment from all levels – student, faculty and management – to make it work	This problem-based learning class was observed and analyzed by the second author. The analysis presented focuses on the problem-based learning factors, how they were implemented in class, and the strengths and weaknesses of the use of problem-based learning in this way.	The implementation of PBL should be considered as an approach to learning rather than just a technique to support learning
Studios in Software Engineering Education: Towards an Evaluable Model	Now that studios can have their authenticity evaluated, software studios can be properly evaluated to determine how effectively the studio aspects have been implemented.	Finally, intriguing questions can be asked about the differences between studios at separate institutions and whether we can easily identify and transfer elements of one studio to another.	One potential avenue of future research, considering these possible incompatibilities, would be to identify missing or diminished attributes using the model, develop and deploy an artifact in the software studio and see if improving that attribute helps.
A Call to Promote Soft Skills in Software Engineering	Software engineers take pride in the depth of their technical expertise, which separates them from the crowd.	However, computer science and software engineering curricula focus mainly on developing hard skills, thus paying lip services to soft skills. Even the latest guideline for teaching and learning software engineering	The dilemma of having technical knowledge and a managerial position at the same time often forces them to wade in an attempt to fix problems that rightly belong to a subordinate.

International Journal of Engineering Technology and Management Sciences
Website: ijetms.in Issue: 2 Volume No.7 March - April – 2023
DOI:10.46647/ijetms.2023.v07i02.098 ISSN: 2581-4621

Training Software Engineers using Open-Source Software: The Professors' Perspective	To bridge this gap, novel SE courses are leveraging the rich variety of open-source software (OSS) projects to illustrate how these methodologies and concepts are applied to existing, non-trivial software systems.	we interviewed seven SE professors that opted for introducing the process of contributing to an existing, nontrivial OSS project as part of a SE course	For future work, we plan to expand the scope of this study in two ways: first, we plan to conduct additional interviews as a way to refute or validate our additional findings
Open-Source Software Projects Crating Model for Empirical Software Engineering Studies	share an open source software project dataset with their quality metrics. The dataset will be created and maintained with the implementation of the aforementioned model.	The results showed that, researchers most common practices are: making their own guidelines to select projects and using existing software project datasets.	Once the second systematic mapping study is finished, we have two planned studies ahead.
Integrating RP Technology into Tennessee Tech's Design and Manufacturing Curriculum	Currently, the CAD courses are designed to teach engineering design principles using CAD techniques. These courses help prepare students for subsequent process engineering and tool design classes.	The rapid advances in computer technology opened new horizons for the faculty who are teaching in CAD/CAM technologies and will continue to do	Many schools worldwide are committed to providing their students with the innovative tools they need to be successful leaders in their future careers.
Educating Software Engineers of the Future: Software Quality Research through Problem-Based Learning	Software Engineering graduates are expected to enter the workforce with both technical and soft skills.	This project presents how problem-based learning, a pedagogical methodology that is popular in medicine and other disciplines, can be used to accomplish these goals in a single course module.	Introduce research which was being undertaken within Lero and demonstrate how this could potentially be useful to students' in the future.

CONCLUSION

We examined the gap from the perspectives of three key groups: business, academia, and SSE. We examined the disparity from their viewpoints, as well as the factors that we found and those were also described in the literature. A questionnaire is distributed to students and recent graduates to determine the amount of obstacles they have encountered when moving to substantially larger projects. We presented and evaluated its findings and discovered that, while there is no statistical difference between the two groups, there is a statistically significant difference in their responses to the communication challenge and how competent they felt as software engineers.

We conducted online and in-person interviews with students and recent graduates to learn more about these problems. As a consequence, the conclusion was reached that experience is the greatest remedy for closing the gap. They noted that, despite certain obstacles as they progressed to larger projects, they were able to overcome them as they gained more expertise. The interview with academics was also very beneficial since they presented unique suggestions for closing the gap, such as strengthening industry collaboration and giving more exciting projects for students to create.

REFERENCES

[1] P. Kruchten, "Putting the 'engineering' into'software engineering," in Austral. Softw. Eng. Conf., Apr. 2004, pp. 2-8.

[2] S. Yu and S. Zhou, "A review on software complexity metrics," Proc. 2nd IEEE Int. Conf. Inf. Manage. Eng., April 2010, pp. 352-356.

[3] G. Matturro, F. Raschetti, and C. Fontán, "Soft skills in software development teams: A study of team leaders and team members' perspectives," in Proc. IEEE/ACM 8th Int. Workshop Cooperat. Hum. Aspects Softw. Eng., May 2015, pp. 101-104.

[4] M. Craig, P. Conrad, D. Lynch, N. Lee, and L. Anthony, "Listening to Early Career Software Developers," Journal of Computing Science Colleges, vol. 33, pp. 138-149, April 2018.

[5] K. Alboaouh, "Bridging the Gap Between Engineering Schools and Industry: A Strategic Initiative," in IEEE Frontiers Educ. Conf. (FIE), Oct. 2018, pp. 1-6.

[6] E. Goold, "Engineering students' perceptions of their engineering practise preparation," in Proc. 6th Res. Eng. Educ. Symp., Dublin, Ireland, 2015, pp. 1-9.

[7] H. Beckman, N. Coulter, S. Khajenoori, and N. R. Mead, "Collaborations: Closing the Industry-Academia Gap," IEEE Softw., vol. 14, no. 6, Nov. 1997, pp. 49-57.

[8] M. Shaw, "Software engineering education: A roadmap," Proc. Conf. Future Softw. Eng. (ICSE), 2000, pp. 371-380.

[9] A. Begel and B. Simon, "Struggles of New College Graduates in Their First Software Development Job," in Proceedings of the 39th SIGCSE Tech. Symp. Comput. Sci. Educ. (SIGCSE), 2008, pp. 226-230.

[10] E. Tuzun, H. Erdogmus, and I. G. Ozbilgin, "Are computer science and engineering graduates equipped for the software industry? : Experiences from an industrial student training programme," in Proc. 40th International Conference on Softw. Eng., Softw. Eng. Educ. Training (ICSE-SEET), 2018, pp. 68-77.

[11] G. Samarthyam, G. Suryanarayana, A. K. Gupta, and R. Nambiar, "Focus: An adaption of a swebok-based curriculum for industrial requirements," in Proc. 34th Int. Conf. Softw. Eng. (ICSE), Jun. 2012.

[12] J. Yang and J. Wang, "Review on Free and Open Source Software," in IEEE Int. Conf. Service Oper. Logistics, Inform., vol. 1, Oct. 2008, pp. 1044-1049.

[13] "A comprehensive approach to improving undergraduate electrical engineering education," IEEE Access, vol. 5, pp. 8148-8161, Mar. 2017. A. A. Maciejewski, T. W. Chen, Z. S. Byrne, M. A. De Miranda, L. B. S. Mcmeeking, B. M. Notaros, A. Pezeshki, S. Roy, A. M. Leland, M. D.

[14] I. Richardson and Y. Delaney, "Problem-based learning in the software engineering classroom," in Proc. 22nd Conf. Softw. Eng. Educ. Training, pp. 174-181, Feb. 2009.

[15] C. N. Bull, J. Whittle, and L. Cruickshank, "Studios in Software Engineering Education: Towards an Evaluable Model," in Proceedings of the 35th International Conference on Softw. Eng. (ICSE), May 2013, pp. 1063-1072.

[16] "A call to enhance soft skills in software engineering," L. F. Capretz and F. Ahmed, Psychol. Cogn. Sci. J., 4, e1-e3, August 2018.

[17] G. H. L. Pinto, F. F. Filho, I. Steinmacher, and M. A. Gerosa, "Training software engineers using open-source software: The academics' perspective," in Proc. IEEE 30th Conf. Softw. Eng. Educ. Training (CSEE T), Nov. 2017.

"Selecting open source software projects to educate software engineering," T. M. Smith, R. McCartney, S. S. Gokhale, and L. C. Kaczmarczyk, in Proc. 45th ACM Tech. Symp. Comput. Sci. Educ. (SIGCSE), 2014, pp. 397-402.

International Journal of Engineering Technology and Management Sciences
Website: ijetms.in Issue: 2 Volume No.7 March - April – 2023
DOI:10.46647/ijetms.2023.v07i02.098 ISSN: 2581-4621

[19] R. Simpson and T. Storer, "Experimenting with realism in software engineering team projects: An experience report," in Proceedings of the IEEE 30th Conf. Softw. Eng. Educ. Training (CSEE T), Nov. 2017, pp. 87-96.
[20] "Educating software engineers of the future: Software quality research through problem-based learning," in Proc. 24th IEEE-CS Conf. Softw. Eng. Educ. Training (CSEE T), May 2011, pp. 91-100.