Methodical Software Testing Course in Higher Education

https://doi.org/10.3991/ijep.v12i1.26111

Mamdouh Alenezi[1(✉)], Mohammad Akour[1,2]
[1] Computer Science Department, Prince Sultan University, Riyadh, Saudi Arabia
[2] Information Systems Department, Yarmouk University, Irbid, Jordan
malenezi@psu.edu.sa

Abstract—Software testing plays a significant role in developing high-quality software. Over Years, too many companies report that more than 50% of software development cost goes for testing. The main problem here is not about how much testing is conducted to guarantee the quality, the main factor of successful testing is who is doing the testing and how are they conducting the testing. Moreover, testing skills might be started and enriched during undergraduate study. During undergraduate study, students can take very basic skills in testing, their experience will be conducted on a few sets of testing tools and very small software. Many articles and reports highlighted how many recent computer science and software engineering undergraduate students often face obstacles when they start their professional jobs. The reasons are most likely because of the misalignment of the earned skills during their academic school education with what is needed in the industry. In this paper, the authors aim to reduce the gap between what skills are needed in the market and what software testing course is covered in our university. Software testing course is designed and developed for undergraduate students in our university as work on progress, as we believe university-level courses should be updated to match both the well-known standards and the market needs. Moreover, this article summarizes the findings and the lesson learned of using the designed course as a real experiment in university education.

Keywords—engineering education, software testing, ABET, higher education

1 Introduction

Software testing is a major part of teaching software engineering students. However, it is a very challenging topic to be taught from an educational perspective [1]. In teaching software testing, students need to learn different testing levels and techniques, can choose the right technique to apply, assess the quality of their test suites, and write maintainable test code. Moreover, most universities focus on teaching students how to build software, not how break it [2]. A course's setup as mostly delivered lectures can make testing both as a concept and a practice can be very difficult to teach. Software testing in universities needs to have a more practical industry-relevant focus [2].

Educating students on the art of software testing is challenging, for both students and instructors. From the instructors' perspective, it is challenging to keep an up-to-date course with the advances in the field as well as realistic activities and assignments [3]. The other challenge for instructors is the fact that some testing topics are not conceptually straightforward, not easy to demonstrate and generalize, and are not all available in a single textbook [4]. Students on the other hand are more excited about building things, rather than testing them [5]. This lack of motivation in students can be also attributed to the inconsistency between practical and theoretical contents, contents that are taught in the classroom are not the same that are required in the industry, and students have difficulties in understanding the processes and the stages of tests [12].

Software engineers should be aware of the consequences of bugs and defects in software systems and their impact on our society. One of the main responsibilities of software engineers is to make sure that the software works. Software testing has become one of the important skills in software engineers [1, 12, 15]. Inspecting and going through large and complex code bases to find bugs is not an easy task. It needs a deep understanding of manual testing to advanced automated testing techniques. Big tech companies take testing very seriously [1] and require their engineers to master such techniques. Based on the findings of an empirical analysis of knowledge gaps in software engineers, the authors recommend that educators should include more materials on software testing in the SE curriculum [6].

An essential part of any Software Engineering Program is software testing [7, 8, 9]. Clarke et al. [5] highlighted that because there are a lot of topics to be covered in a software engineering program and only a little attention is paid to software testing [7]. Lemos et al. [8] discussed that university instructors do not possess the right skills and expertise to teach students to create more reliable code. Jones [10] proposes as part of the educational experience that each core course in the curriculum should include one or more testing experiences.

Several researchers address the quality improvement of the developed software by undergraduate students from different perspectives. Segura and Staubits [17,18] concentrate on improving students' skills in two specific areas: software usability and programming level respectively. They employed their proposing enhancements as a case study by including a few exercises in the selected curriculum at your universities. The results of their experiments reveal a noticeable appropriation of the knowledge of the group of students who participated in the study. However, in this paper, we address the quality of the developing software from testing perspectives. The authors focus on enhancing the testing course in order to develop the software with high quality, in addition to reducing the gap with the needed industry tester skills. Our testing course has always been designed to be experiential, whereby students applied classroom topics on software systems. Each semester the course was updated to include new topics, new tools, new techniques, new assignments, new activities, and new research results. There are several sources of updates happening to the course including research results, industry practitioners' feedback, and students' feedback. These updates are centered on improving student learning perspectives. The main contributions of this article are providing the details of:

- An undergraduate software testing course structure (topics, activities, and assignments) that has matured over several offerings
- Our experiences structuring the courses around SWEBOK v3, and several feedbacks from the involved students and instructors.
- Design of a course that meets ABET criteria for SE courses.

2 Methodology

The course is designed following the methodology presented in Figure 1. We designed the course based on the educational standards (SWEBOK) and Accreditation body (ABET). Each semester, we take into account the feedback from students, industry experts, and new technologies to improve the content of the course.

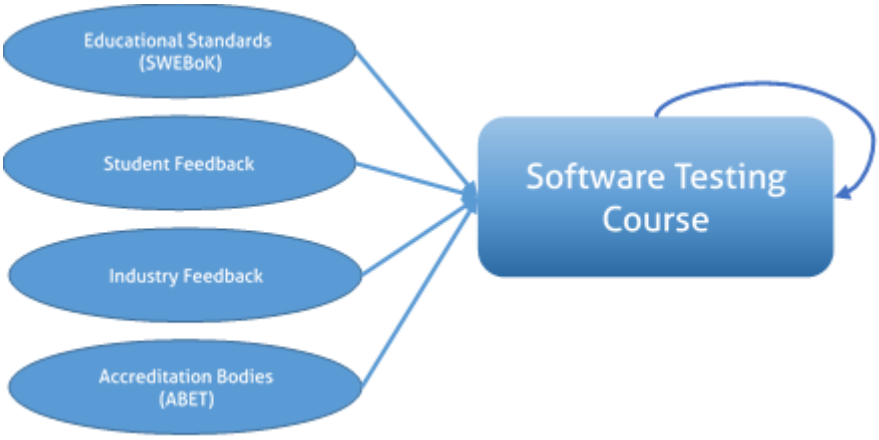

Fig. 1. Methodology to develop and improve the course

The Computer Science program at Prince Sultan University (PSU) is accredited by ABET, and at the current time, we are preparing to submit the ABET readiness document for the SE program. Therefore, the authors designed the course in such a way that matches the Accreditation body (ABET), in addition to following the educational standards (SWEBOK). Each semester, we take into account the feedback from students, industry experts, and new technologies to improve the content of the course.

As shown in Figure 1, four main factors are addressed to make sure the designed course is covering the most important needs from both academic and industry perspectives. For example, the SWEBOK and the ABET criteria and standards are followed to make sure the course is matching a well-known and specialized agency in Software Engineering courses. While the current needs and the lack of required skills are collected from students and industry experts. Authors strive to have a comprehensive view of the best practices in SWEBOK and ABET, and the current/future needed skills from the industry.

2.1 Course details

The undergraduate-level Software Testing course introduces students to the discipline of software testing and quality assurance. The course involves occasional assignments, activities, and a semester-long project. The assignments and activities offer a chance for students to reinforce their understanding of the material from class. The project has small student groups testing an open-source software and applying techniques and tools learned in class to that software.

At PSU, instructors have been teaching a standalone course on software testing for more than 4 years. During all these years, our instructors strived to monitor the engagement of the students, the weaknesses, and the strengths to finalize the content to cope the state of the art testing field. In the designed course, instructors aimed to improve students' attitudes towards testing and make them feel enthusiastic about studied topics. Figure 2 shows the functional dimensional limits of the software testing course. The course website and resources can be accessed at https://malenezi.github.io/malenezi/SE401/. After completing this course, the students will be able to:

1. Recognize software quality assurance and testing as an essential element in the software development life cycle
2. Describe the phases of software quality assurance and testing
3. Develop test plans, identify test conditions, and design test cases
4. Apply a wide variety of testing techniques at various testing levels
5. Compute various metrics from the testing data and interpret them to identify problems in software testing
6. Adequately test a medium software project in a group setting

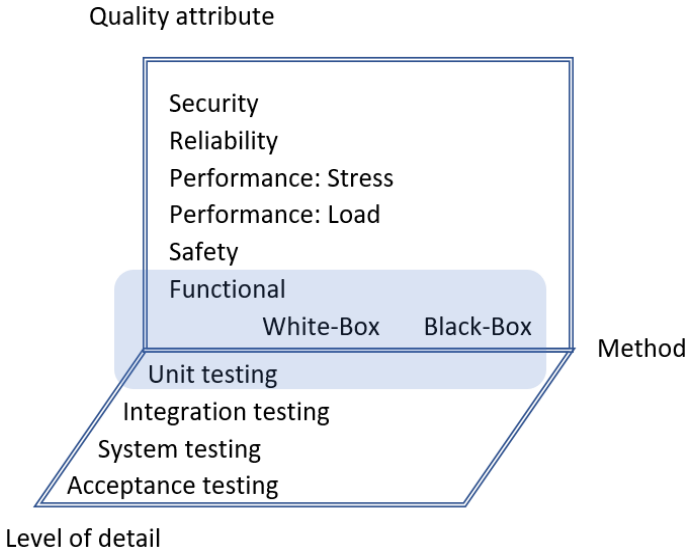

Fig. 2. Dimensions of software testing [11]

The main learning modules taught within the course are:

1. Introduction
2. Software Quality
3. Software Testing Life Cycle
4. Software Testing Plans and Test cases
5. Unit Testing and Junit
6. Black Box Testing
7. White Box Testing
8. Integration, System and Regression Testing
9. Testing Metrics
10. Web and Mobile Testing

2.2 Course assignment and activities

To make sure that the designed course is more than theory-oriented content, the class has several assignments and activities. These tasks are distributed among the course topics. This course requires both a conceptual/theoretical understanding of the main foundations of software testing and the practical aspects which are covered by the activities. Table 1 shows the course organization with regards to topics, assignments, and activities.

Table 1. Software testing course organization

Chapter	Module	Assignment	Activity
1	Introduction	Fundamentals	
2	Software Quality	Quality Engineering	
3	Software Testing Life Cycle	Testing Process and Life Cycle	
4	Software Testing Plans and Test cases		Test Plan and Specification
5	Unit Testing and Junit		JUnit and Ant JUnit and Coverage Testing
6	Black Box Testing		Black Box Testing Equivalence class and boundary value
7	White Box Testing		White Box Testing SpotBugs Complexity analysis and visualization
8	Integration, System and Regression Testing	Integration Testing	
9	Testing Metrics		Code Coverage Analysis
10	Web and Mobile Testing		Web Testing

2.3 Course project

The embedded project aims to allow students to apply the software testing process to a real software product. Since open-source software systems are available to be

downloaded, and some of them have enough technical details, several open-source applications are collected and shared with the students for experimental works. Students work on the project as groups of a maximum of 5 students. The number of students depends on two main factors: the number of students in the course and the size of the software under test. The project is designed to be built incrementally. There are four main phases in the project. The first phase is about test planning where students prepare a test plan to adequately test the software system. The second phase is about test cases where students are required to design and prepare detailed test cases for all the types of tests that have been previously planned: Unit Testing, Integration Testing, and System Testing. The third phase is about executing the test cases that were designed in the previous phase. The last phase is about presenting the reports and analysis in front of the class. Detailed rubrics were developed to assess each phase of the project.

2.4 Software engineering body of knowledge (SWEBOK)

IEEE developed the SWEBOK, which is a guide for the body of knowledge in Software Engineering, to promote the professionalization of Software Engineering. The latest version of the Software Engineering Body of Knowledge (SWEBOK v3) classifies the SE knowledge into 12 Knowledge Areas (KAs), which are themselves broken down into 67 subareas (sub-KAs) in total. Software Testing is chapter number 4 in the SWEBOK. According to SWEBOK v3.0, the software testing topics are the following:

1. Software Testing Fundamentals
2. Test Levels
3. Test Techniques
4. Test-Related Measures
5. Test Process
6. Software Testing Tools

A mapping between SWEBOK v3.0 and the course topics was done to ensure consistency and is presented in Table 2. Each covered topic is mapped to its corresponding one in the SWEBOK.

Table 2. Mapping course topics to SWEBOK v3.0

SWEBOK v3.0	Our Course
1. Software Testing Fundamentals	1. Introduction 2. Software Quality
2. Test Levels	8. Integration, System, and Regression Testing
3. Test Techniques	5. Unit Testing and Junit 6. Black Box Testing 7. White Box Testing
4. Test-Related Measures	9. Testing Metrics
5. Test Process	3. Software Testing Life Cycle 4. Software Testing Plans and Test cases
3. Software Testing Tools	Assignments, Activities, and Project

2.5 ABET

According to the ABET Criteria for Accrediting Engineering Programs (2021 – 2022), Criterion 3 is concerning Student Outcomes. This criterion states that the program must have documented student outcomes that support the program's educational objectives [13, 16]. Attainment of these outcomes prepares graduates to enter the professional practice of engineering. There are seven student outcomes as follows:

1. an ability to identify, formulate, and solve complex engineering problems by applying principles of engineering, science, and mathematics
2. an ability to apply engineering design to produce solutions that meet specified needs with consideration of public health, safety, and welfare, as well as global, cultural, social, environmental, and economic factors
3. an ability to communicate effectively with a range of audiences
4. an ability to recognize ethical and professional responsibilities in engineering situations and make informed judgments, which must consider the impact of engineering solutions in global, economic, environmental, and societal contexts
5. an ability to function effectively on a team whose members together provide leadership, create a collaborative and inclusive environment, establish goals, plan tasks, and meet objectives
6. an ability to develop and conduct appropriate experimentation, analyze and interpret data, and use engineering judgment to draw conclusions
7. an ability to acquire and apply new knowledge as needed, using appropriate learning strategies.

Table 3 shows the mapping between our course learning outcomes and the seven student outcomes. We use different scales to indicate the level of contribution of that learning outcome. The level scale are (I = Introduction, P = Proficient, A = Advanced). Direct and indirect assessments are used to measure these learning outcomes. Direct assessment is according to rubrics to measure students' deliverables such as assignments, activities, exams, presentations, and projects. Indirect assessment is based on the point of view of students and what they learned during the course. Both types of assessments are based on students' learning outcomes that are defined for a specific course. Based on the assessments that are done each semester, some points in the course are modified according to the results. We use four different levels of course learning outcomes (CLOs) assessments. These four levels are (Below Expectations, Developing Expectations, Meeting Expectations, and Above Expectations).

Table 3. The course learning outcomes (CLO) mapped to ABET Student Outcomes (SO)

CLO #	SO1	SO2	SO3	SO4	SO5	SO6	SO7
1	P						
2						A	
3		P	P			A	
4						A	
5	I						
6					P		P

3 Results

To track the success of the course improvement plan, two main assessments are monitored. The first assessment is considered an indirect assessment where the students are asked about the satisfaction of the learning outcomes. An anonymous survey, course exit survey, is distributed to students through the learning management systems. The students are asked to evaluate their satisfaction with each one of the learning outcomes. All enrolled students participated in this survey (105 male students). The second assessment is considered a direct assessment where the students' answers and produced work are evaluated against rubrics [14]. The work includes assignments, activities, exams, and projects. Another feedback comes from industry practitioners and experts. Each semester the material is discussed with at least four experts to seek their feedback about the practicality and usefulness of the materials, tools, and activities. Their feedback is injected into the continuous improvement cycle of the course.

Figure 3 shows the results of the indirect assessments over seven semesters. The results indicate a strong satisfaction of students regarding the course learning outcomes. It is clear that the improvement to the course is helping students get the right skills and knowledge needed to master testing. The survey is done before the final exam to ensure the objectivity of students. All the results are above 75% which is considered the target at our college.

Figure 4 shows the results of the direct assessments over seven semesters. The results indicate that the work produced by students is highly acceptable according to the predefined rubrics. Generally speaking, all learning outcomes are improving over time. Since most of the feedback is coming from students, the improvements are helping students in improving their knowledge and skills throughout semesters.

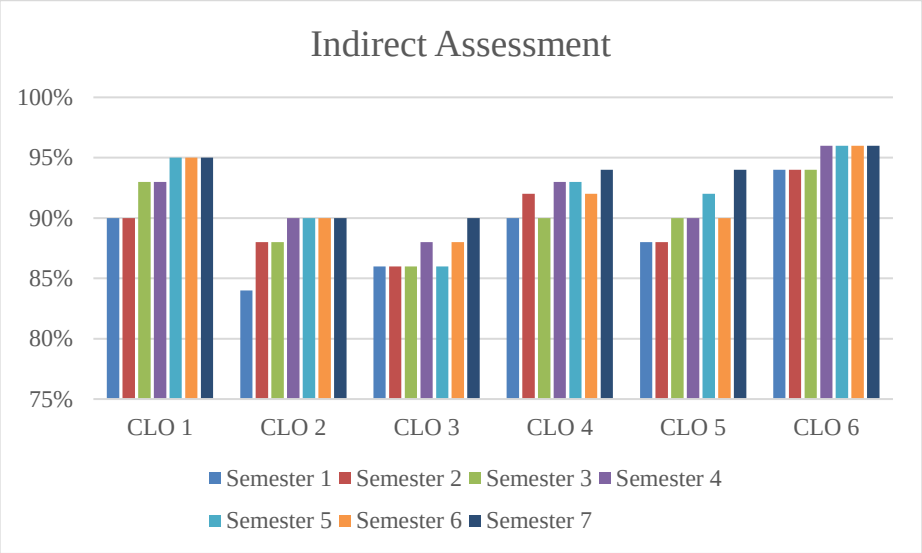

Fig. 3. The results of the indirect assessments over seven semesters

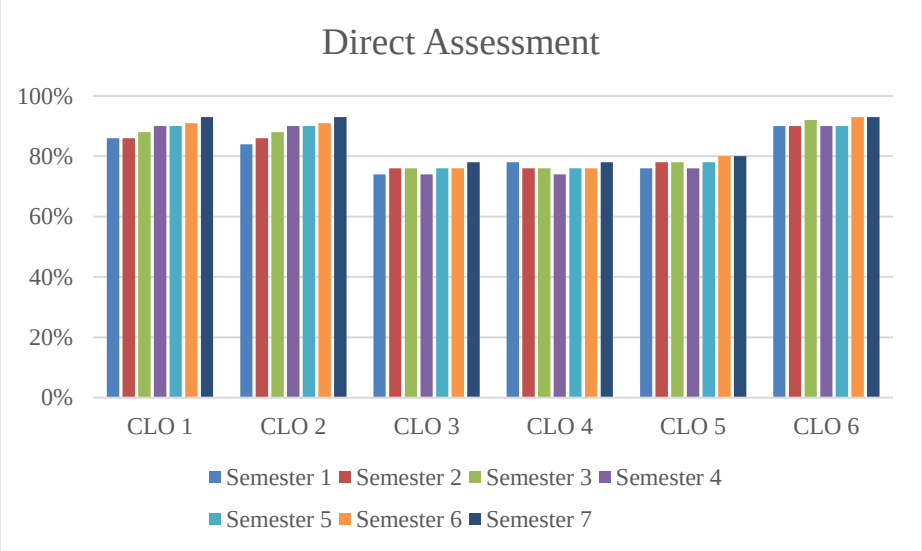

Fig. 4. The results of the indirect assessments over seven semesters

4 Conclusion

The quick growth of topics, tools, and industry needs for software testing raises the question of whether the designed testing courses are sufficient to form a modern student as a specialist that is capable of professionally testing software systems through-

out the software life cycle. To address this question, a software testing course is designed and taught at PSU to contribute to improving PSU students' position in the national and international IT market. As a preliminary step, the author sets out the main visions of software testing, required skills, and national and international quality standards. Moreover, the challenges and students' perspectives are collected to be one of the main feedback sources to come up with the targeted course. The course has evolved according to students' feedback, industry needs, and new technologies. The course evolution has been successful based on both indirect and direct assessments. The industry feedback has been also great since the course graduates are well-received in the industry.

5 Acknowledgment

The authors would like to acknowledge the support of Prince Sultan University for paying the Article Processing Charges (APC) of this publication.

6 References

[1] Aniche, Maurício, Felienne Hermans, and Arie Van Deursen. "Pragmatic software testing education." In Proceedings of the 50th ACM Technical Symposium on Computer Science Education, pp. 414-420. 2019. https://doi.org/10.1145/3287324.3287461

[2] Krutz, Daniel E., Samuel A. Malachowsky, and Thomas Reichlmayr. "Using a real world project in a software testing course." In Proceedings of the 45th ACM technical symposium on Computer science education, pp. 49-54. 2014. https://doi.org/10.1145/2538862.2538955

[3] Garousi, Vahid, and Aditya Mathur. "Current state of the software testing education in north american academia and some recommendations for the new educators." In 2010 23rd IEEE Conference on Software Engineering Education and Training, pp. 89-96. IEEE, 2010. https://doi.org/10.1109/CSEET.2010.29

[4] Timoney, Joseph, Stephen Brown, and Deshi Ye. "Experiences in software testing education: some observations from an international cooperation." In 2008 The 9th International Conference for Young Computer Scientists, pp. 2686-2691. IEEE, 2008. https://doi.org/10.1109/ICYCS.2008.209

[5] Clark, N. "Peer Testing in Software Engineering Projects." In Australasian Computing Education Conference (ACE 2004), vol. 30, pp. 41-48. 2004.

[6] Garousi, Vahid, Gorkem Giray, and Eray Tuzun. "Understanding the knowledge gaps of software engineers: An empirical analysis based on SWEBOK." ACM Transactions on Computing Education (TOCE) 20, no. 1 (2019): 1-33. https://doi.org/10.1145/3360497

[7] Clarke, Peter J., Debra Davis, Tariq M. King, Jairo Pava, and Edward L. Jones. "Integrating testing into software engineering courses supported by a collaborative learning environment." ACM Transactions on Computing Education (TOCE) 14, no. 3 (2014): 1-33. https://doi.org/10.1145/2648787

[8] Lemos, Otávio Augusto Lazzarini, Fábio Fagundes Silveira, Fabiano Cutigi Ferrari, and Alessandro Garcia. "The impact of Software Testing education on code reliability: An empirical assessment." Journal of Systems and Software 137 (2018): 497-511. https://doi.org/10.1016/j.jss.2017.02.042

[9] P. Bourque and R. E. Fairley. 2014. Guide to the software engineering body of knowledge (SWEBOK), version 3.0. IEEE Computer Society Press.
[10] Jones, Edward L. "An experiential approach to incorporating software testing into the computer science curriculum." In 31st Annual Frontiers in Education Conference. Impact on Engineering and Science Education. Conference Proceedings (Cat. No. 01CH37193), vol. 2, pp. F3D-7. IEEE, 2001. https://doi.org/10.1109/fie.2001.963741
[11] Silvis-Cividjian, Natalia. "Awesome Bug Manifesto: Teaching an Engaging and Inspiring Course on Software Testing (Position Paper)." In 2021 Third International Workshop on Software Engineering Education for the Next Generation (SEENG), pp. 16-20. IEEE, 2021. https://doi.org/10.1109/SEENG53126.2021.00010
[12] Valle, Pedro Henrique Dias, Armando Maciel Toda, Ellen Francine Barbosa, and José Carlos Maldonado. "Educational games: A contribution to software testing education." In 2017 IEEE Frontiers in education Conference (FIE), pp. 1-8. IEEE, 2017. https://doi.org/10.1109/FIE.2017.8190470
[13] Meah, Kala, Donald Hake, and Stephen Drew Wilkerson. "A multidisciplinary capstone design project to satisfy abet student outcomes." Education Research International 2020 (2020). https://doi.org/10.1155/2020/9563782
[14] Pejcinovic, Branimir. "Design of Rubrics for Student Outcomes in 2019-2020 ABET Criteria." In 2020 43rd International Convention on Information, Communication and Electronic Technology (MIPRO), pp. 1543-1548. IEEE, 2020. https://doi.org/10.23919/MIPRO48935.2020.9245228
[15] Sedelmaier, Yvonne, and Dieter Landes. "SWEBOS-The Software Engineering Body of Skills." International Journal of Engineering Pedagogy 5, no. 1 (2015). https://doi.org/10.3991/ijep.v5i1.4047
[16] Damaj, Issam, Ashraf Zaher, and Jibran Yousafzai. "Assessment and evaluation framework with successful application in ABET accreditation." Int. J. Eng. Pedagog. 7.3 (2017): 73-91. https://doi.org/10.3991/ijep.v7i3.7262
[17] Segura, Josue. "The Teaching of Usability in Software Development: Case Study in the Computer Engineering Career at the University of Matanzas." Int. J. Eng. Pedagog. 11.1 (2021): 4-15. https://doi.org/10.3991/ijep.v11i1.14837
[18] Staubitz, T., Teusner, R., Meinel, C., & Prakash, N. (2017). Cellular Automata as an Example for Advanced Beginners' Level Coding Exercises in a MOOC on Test Driven Development: Lessons Learned and Suggestions for Improvement. International Journal of Engineering Pedagogy, 7(2). https://doi.org/10.3991/ijep.v7i2.6969

7 Authors

Mamdouh Alenezi is currently the Dean of Quality Assurance and Development at Prince Sultan University. Alenezi received his MS and Ph.D. degrees from DePaul University and North Dakota State University in 2011 and 2014, respectively. Dr. Alenezi is an associate professor in software engineering with a teaching emphasis on software engineering and software security. He participates in organizing several international scientific conferences and editorial boards of well-reputed journals. He has extensive experience in applying data mining and machine learning techniques to solve software engineering problems. He published more than 80 papers. He conducted several research areas and developed predictive models using machine learning to predict fault-prone classes, comprehend source code, and predict the appropriate de-

veloper to be assigned to a newly reported bug. His research focuses on Software Engineering, Software Security, Machine Learning, and Open Source Software Systems. Alenezi served as Chair of the Computer Science Department, the Chief Information Technology Officer, and Dean of Educational Services before he was appointed Dean of Quality Assurance and Development in September 2020.

Mohammad Akour is a Full Professor of Software Engineering at Prince Sultan University (PSU). He got his Bachelor's (2006) and Master's (2008) degree from Yarmouk University in Computer Information Systems with Honor. He joined Yarmouk University as a Lecturer in August 2008 after graduating with his master's in Computer Information Systems. In August 2009, He left Yarmouk University to pursue his Ph.D. in Software Engineering at North Dakota State University (NDSU). He joined Yarmouk University again in April 2013 after graduating with his Ph.D. in Software Engineering from NDSU with Honor. He serves as Keynote Speaker, Organizer, Co-chair, and publicity Chair for several IEEE conferences, and as ERB for more than 10 ISI indexed prestigious journals. He is a member of the International Association of Engineers (IAENG). Akour at Yarmouk University served as Head of accreditation and Quality assurance and then was hired as director of computer and Information Center. In 2018, Akour was hired as Vice Dean of Student Affairs at Yarmouk University.

Article submitted 2021-08-07. Resubmitted 2021-11-03. Final acceptance 2021-11-08. Final version published as submitted by the authors.