Preface

We are living in an evolving society, where we can feel the changes in every aspect of our lives: communication, transportation, healthcare, education, career building, financial management, entertainment, social networking, and so on. Part of the driving forces of the current evolution of human society is information technology, where software products play an important role in shaping the way we live and how the society works. Accordingly, the functions and the qualities provided by software products could affect everyone in our society.

Software engineering is the discipline we follow to produce software products. The discipline itself and how well it is being followed could affect software products, which in turn could affect our lives. Software engineering education is the process we educate and train professional software engineers to deliver disciplined software engineering knowledge and skills. Therefore, software engineering education has an indirect impact on our evolving society.

Currently, software engineering education has two major challenges. On the one hand, software development is a fast-changing area. New methods and new technologies emerge every year. As a result, the education of software engineering is generally considered not to be keeping pace with the development of software engineering in industry. On the other hand, lecture-based software engineering education hardly engages and convinces students. Students often view software engineering principles as mere academic concepts and do not know how to use them in practice. In some cases, students even consider software engineering courses less interesting, less valuable, and less practical. However, the reality is computer science graduates often find that software engineering knowledge and skills are more in demand after they join the industry.

Ideally, most of the software engineering principles should be learned through real-world industry practices. However, given the limited resources in academia, it is not an easy task to deliver industry-standard knowledge and skills, especially non-technical knowledge and skills, in a software engineering classroom. For example, most software engineering educators are computer scientists instead of communication or management experts; they often have difficulties finding resources to support the delivery of communication skills, team-working skills, management skills, and so on. Accordingly, there is a gap between the industry expectations and what academia can provide in software engineering education.

This book presents the recent advances in software engineering education to overcome the aforementioned challenges. Through assembling the current best practices in software engineering education, this book intends to provide guidelines for software engineering educators to improve their curricula and instruct computer science students with ready-to-apply software engineering knowledge and skills.

This book aims to be useful for both research and teaching. The target audience will be software engineering researchers and educators in computer science programs, software engineering programs, or information technology programs of higher education institutions.

The book is organized into 10 sections, 24 chapters, which cover *developing project management skills*, *encouraging collaborations and teamwork*, *supporting communications*, *improving soft skills*, *promoting project-based learning*, *engaging classroom games*, *experiencing case-based teaching and problem-based learning*, *meeting industry expectations*, *using open-source tools*, and *adopting digital learning*. Below, we provide a brief summary of the 24 chapters.

BOOK LAYOUT

Sections

1. Developing Project Management Skills (Chapters 1-2)
2. Encouraging Collaborations and Teamwork (Chapters 3-5)
3. Supporting Communications (Chapters 6-7)
4. Improving Soft Skills (Chapters 8-11)
5. Promoting Project-Based Learning (Chapters 12-14)
6. Engaging Classroom Games (Chapters 15-16)
7. Experiencing Case-Based Teaching and Problem-Based Learning (Chapters 17-18)
8. Meeting Industry Expectations (Chapters 19-20)
9. Using Open-Source Tools (Chapters 21-22)
10. Adopting Digital Learning (Chapter 23-24)

Chapters

In chapter 1, Kasi Periyasamy describes his project management course, which has been used as a vehicle to deliver nontechnical skills to software engineering students. Various approaches, such as teamwork and conflict resolution, are used to improve students' managerial skills. Through team projects, domain knowledge and development skills are also delivered to students.

In chapter 2, Marcos Martínez et al. present their approach and experience of teaching leadership and team management skills in computer science program. Their coaching-based approach is realized through workshops, where students play the major roles and the instructor acts as a coach. Through participating in these workshops, students could also practice other nontechnical skills, such as team-working skills and communication skills.

In chapter 3, Pankaj Kamthan discusses collaborations in software engineering courses. Various interactions between students and other stakeholders are discussed. Specifically, the use of Social Web, such as wiki, is discussed to support collaborations and communications in an agile course project. In addition, the author proposes a conceptual collaboration model to represent human-human interactions.

In chapter 4, Ann Gates et al. describe their approach to teaching team working skills following a framework called TOSE (Team-Oriented Software Engineering). TOSE allows students to develop team-working skills through cooperative learning that encourages continuous improvement of team functions during the software development process. Trained through this approach, students could be increasingly more knowledgeable and become effective software engineering practitioners.

In chapter 5, Mirjana Ivanovic et al. present their experience of encouraging students to communicate and cooperate through Web 2.0 and social networking. Quantitative analysis is performed to evaluate their teaching approach. These experiences could be shared by other instructors who are going prepare their students with distributed working skills.

In chapter 6, Marcel Fouda Ndjodo and Virginie Blanche Ngah analyze their approach to improving students' communication skills in requirement elicitation. To communicate efficiently with the clients, the authors propose a five-step (know, comprehend, apply, analyze, and synthesize) approach to enriching students linguistic knowledge and improving their communication skills, interpretation skills, and representation skills.

In chapter 7, Damith Rajapakse explains how peer feedback is used as a communication mechanism in his software engineering course. Peer feedback has been used for team meetings, code reviews, and peer mentoring. Overall, peer feedback is demonstrated to be an effective mechanism to improve the communications and collaborations among developers.

In chapter 8, Jocelyn Armarego details a series of interventions used in a software engineering course aimed at better teaching employability skills to students. The purpose of putting the interventions in place is to better align the capabilities of graduates of a software engineering course with employer expectations in the area of soft skills. The three approaches tried in this research are the Cognitive Apprenticeship model, the Problem-Based Learning, and the Studio approach. This chapter also provides extensive reports on student reactions to the interventions.

In chapter 9, Yvonne Sedelmaier and Dieter Landes discuss practicing soft skills through a project-based didactical approach. One of the interesting parts about their approach is that each team contains both undergraduate students and graduate students, where graduates work as team leaders and are responsible for scheduling and project management, and undergraduate students are responsible for implementation. The project-based didactical approach is proven to be successful in delivering both technical skills and soft skills.

In chapter 10, Marco Kuhrmann et al. propose teaching soft skills in software engineering classes through controlled experiments, a new teaching approach that combines lectures, practical experiments, and discussions. Two experiments on group dynamics and global software development are implemented to demonstrate this approach. This teaching approach is proved to be an effective method to deliver soft skills.

In chapter 11, Lynette Johns-Boast discusses how large-scale group projects can help students develop skills that are necessary to work in the industry environment. After describing the motivation and background, the author presents her teaching approach and teaching experience, which include problem-based learning, teamwork, and assessment.

In chapter 12, Luis Alves et al. describe their project-based approach in teaching software engineering. In their classes, students work in teams to solve real-world problems. Through working on real-world projects, students interact with real-world clients and practice management skills and quality controls. One specific feature of their project is that the students' documents are assigned with ISBN, which makes the documents official and easy for future reference.

In chapter 13, Ezequiel Scott et al. describe integrating their team coaching method called Agile Coach with Scrum, an agile software development framework, in a capstone software engineering course. The development team uses Virtual Scrum, a prototype tool that aims to help students set up a virtual working environment. They conclude that their teaching strategy may facilitate students to integrate themselves in the software industry environment.

In chapter 14, Marc Lainez et al. provide their experience of teaching agile software development in project-based environment. Specifically, students are asked to follow lightweight agile process to implement a mobile app. Besides technical skills, this project also helps students develop cross-disciplinary skills such as modeling, teamwork, planning, management, and communication. This teaching approach is proven to be successful because agile-based development encourages communication and collaboration.

In chapter 15, Sakgasit Ramingwong and Lachana Ramingwong describe their one-day software development life cycle game, which is used to help students get an initial understanding of the complexity of software process. In this game, students are asked to build a pseudo-software product, a board house. Through playing this game, students understand the importance of communication, teamwork, and customer relations. They also learn basic concepts of project management.

In chapter 16, Elizabeth Monsalve et al. illustrate their teaching experience of using SimulES-W, a computer-based board and card game, in their software engineering course. Through playing the game, students could learn software engineering knowledge and practice software development skills. Statistical studies are also performed to evaluate the effectiveness of this game-based teaching approach, which is proved to be as effective as the regular teaching approach.

In chapter 17, Salamah Salamah et al. present their case-study-based approach to engaging students to learn software engineering concepts. Their case study can provide students with the development experience of a full software life cycle. In particular, they talk about one case study, DigitalHome, a Web-based system that allows home users to manage devices that control the environment of a home remotely. Through working on these case studies, students could learn software engineering principles that are not easily obtained through regular lectures.

In chapter 18, Oisin Cawley et al. review problem-based learning approach, especially its application in software engineering education. They describe two case studies. The students' experience, instructors' experience, and the assessment are also presented in this chapter. They conclude that problem-based learning is an important approach to educating computer science graduates to meet industry expectations.

In chapter 19, Bonnie MacKellar et al. introduce a Client-Oriented Open Source Software (CO-FOSS) model for undergraduate software engineering courses in order to bridge the academia-industry gap. The model has been applied to three universities, which are diverse in terms of size, demographics of students, and curricula. Four case studies are presented in this chapter, which demonstrate that CO-FOSS model can be flexibly adapted in software engineering education based on different needs.

In chapter 20, Paolo Ciancarini and Stefano Russo present the rationale and the experience of teaching software architecture in industrial and academic contexts. The authors compare two different environments and two different expectations and provide suggestions to reduce the gap between industry and academia. Future research directions in this area are also presented.

In chapter 21, Jagadeesh Nandigam and Venkat Gudivada describe their experience of using industry-standard open-source tools in their software engineering classes. The open-source tools and open-source data are the backbones of their class structure. Each learning objective is presented with the corresponding open-source data/tools, which make this approach easy to be replicated by other educators.

In chapter 22, Liguo Yu et al. provide their experience of applying open-source tools and open-source data in their graduate level software engineering course. The using of open-source tools and open-source data is incorporated into five team projects. One interesting project is to estimate the development cost of Linux kernel. Through working on these projects, students could develop industry-expected skills, such as measuring, configuration management, and estimation.

In chapter 23, Yujian Fu presents the experience of using cyber-enabled environment to improve student collaboration skills. The study was performed on software engineering courses of both undergraduates and graduates. The technology-based teaching is implemented on Blackboard, which supports Yahoo Messenger, Google Talk, video, and audio demo. Their study demonstrates that technology-based teaching could engage students, especially those students who are less motivated and less prepared for college.

In chapter 24, Zuhoor Abdullah Salim Al-Khanjari talks about distance learning in software engineering. The author provides a review on software engineering education and e-learning technology. It explores the need to adopt software engineering e-learning model to help facilitators/instructors prepare and manage online software engineering courses. This chapter also addresses how e-learning environment could simplify the application of the constructivist learning model towards software engineering education.

This book is an assembly of education research and classroom experiences collected from educators around the world. Some techniques and experiences have been proven successful and some are still under experiment and refinement. I hope the readers will find this book useful and inspiring in adapting software engineering education with new challenges.

Liguo Yu
Indiana University South Bend, USA