Preface

In the last six decades, many advanced software engineering approaches, techniques, and tools have been introduced to solve various problems related to the development and maintenance of complex large-scale software systems. Some of these advancements have been applied to practice with various degrees of success, including the acceptance of some methods and tools as de facto technologies by software industries. For example, with the introduction of the advanced CASE tools, developers are able to reduce the time to market and minimise the errors made during the software development stage.

Software systems play a major role in contemporary society and penetrate almost all aspects of modern life, ranging from social network systems to safety-critical systems. Although many advancements have been made in software engineering, there are emerging challenges in software engineering due to the increasing complexity of user requirements and operating environments. The produced software is sometimes unable to fulfill user requirements, delivered late or delivered at an unacceptable quality, and unable to cope with requirement changes after certain periods of time. This book highlights some significant research on emerging advancements and technologies in software engineering.

The aim of this book is to present some of the advances in software engineering research and practice. In Section 1, the authors discuss several topics related to software requirements. Understanding the software requirements is an important task in identifying the correct requirements for a large and complex software system.

Section 1 discusses the issues and new approaches in requirements engineering, which is one of the important topics in software engineering. The authors share their experiences obtained from real-life projects:

- “What, Why, Who, When and How of Software Requirements”: Linda Westfall provides an exhaustive coverage of the major elements of requirements engineering such as the definition of software requirements, the benefits of having the right requirements, and the requirements engineering process. This chapter provides a quick guide to quality software requirements.
- “Critical Issues in Requirements Engineering Education” – Rafia Naz Memon, Rodina Ahmad, and Siti Salwah Salim investigate the current state of requirements engineering education and identify the issues in teaching software engineering. Based on the identified issues, the authors recommend the suitable teaching approaches and propose potential topics for further research in requirements engineering education.
- “When the Wisdom of Communication is Vital during the Requirements Elicitation Process: Lessons Learnt through Industry Experience” – Sabrina Ahmad, Noor Azilah Muda, Maslita Abd. Aziz, and Emaliana Kasmuri present the lessons learnt from the application of the requirement elicitation process for three different government agencies. The authors highlight the best practices in requirements elicitation and suggest alternative options during the process.
Section 2 presents the advancements in software design and construction. The chapters cover a number of issues related to model coordination, quality-of-service in relation to code bases, limitations of design patterns, service discovery in real-time systems, and the verification and specification of aspect-oriented systems:

- **“State-of-the-Art Concepts and Future Directions in Modelling Coordination”** – Abdelhamid Abdelhadi Mansor and Wan M. N. Wan-Kadir present the current work on modelling coordination based on the coordination features. The chapter shows that recent research has yielded significant contributions to coordination models and languages, and to the capture of QoS requirements as first class aspects in open distributed software. The authors emphasise that the impact of coordination models on the engineering of complex systems will have profound implications for methodologies and software processes, as well as for related research.

- **“Low-Overhead Development of Scalable Resource-Efficient Software Systems”** – Wei-Chih Huang and William Knottenbelt criticise the use of traditional software engineering pipelines that leads to either a small code base which cannot guarantee quality-of-service or multiple manually-optimised code bases, which are hard to maintain. The vision of this chapter is to propose a means to achieve both: a small, easily maintained code base, together with automated mechanisms to support improved quality-of-service delivery at run-time according to the specific execution environment and application context.

- **“An Empirical Study of the Effect of Design Patterns on Class Structural Quality”** – Liguo Yu and Srini Ramaswamy discuss how the inappropriate use of design patterns may unnecessarily increase program complexity, such as class structural quality. This chapter studies six components from five open-source Java projects and empirically investigates if the design patterns can affect the class structural quality. The authors find that pattern-involved classes are more complex than pattern-free classes, and recommend the cautious use of design patterns to avoid unnecessary increases in class complexity and decreases in class structural quality.

- **“Service Discovery Framework for Distributed Embedded Real-Time Systems”** – Furkh Zeshan, Radziah Mohamad, and Mohammad Nazir Ahmad present a framework for distributed embedded real-time systems for the automatic discovery of services based on the context and quality-of-service information. The proposed framework enables the user to efficiently and automatically search the services running on the embedded devices. The results of the experiment indicate the significance of the proposed work.

- **“An Algebraic Approach for the Specification and Verification of Aspect-Oriented Systems”** – Arsène Sabas, Subash Shankar, Virginie Wiels, John-Jules Ch. Meyer, and Michel Boyer present an algebraic specification approach which proposes a solution to the lack of modular formal reasoning, specification, and verification of aspect-oriented systems. This approach is based on category theory and algebraic specification techniques due to their formality, their modularity benefits, and their high levels of abstraction. The authors also present a prevention mechanism for three aspect fault types that can cause undesirable behaviours in an aspect-oriented system.

The chapters in Section 3 present recent advancements in model-driven development, which is one of the widely accepted approaches in software engineering. The authors aim to improve the specification and model transformation process:
• “Supporting Model-Driven Development: Key Concepts and Support Approaches” – Rita Suzana Pitangueira Maciel, Ana Patrícia F. Magalhães Mascarenhas, Ramon Araújo Gomes, and João Pedro D. B. de Queiroz provide an overview of the model-driven development process and propose an approach to support the model-driven software process, which they call the Model-Driven Process-Centered Software Engineering Environment (MoDErNE). MoDErNE consists of a metamodel and tool that facilitate the integrated specification and enactment of model-driven software processes.

• “A Model-Driven Solution for the Automatic Generation of Executable Code from Business Process Models” – Javier Fabra, Valeria de Castro, Verónica Andrea Bollati, Pedro Álvarez, and Esperanza Marcos integrate the service-oriented development method and a framework for the development and execution of business processes for the automatic code generation of applications based on service-oriented architecture applications. They also develop an automated tool to support the model transformation.

• “Modelling Platform-Independent and Platform-Specific Service Architectures with UML and the ArchiMeDeS Framework” – Marcos López Sanz and Esperanza Marcos propose UML profiles that consist of a set of metamodels that can be used in the modelling of service-oriented software. Their framework is used to systematically guide the model transformation processes using the proposed UML profiles. The feasibility of the proposed approach is validated using the GESiMED case study.

Section 4 presents the advancements in agile methods including security concerns, knowledge documentation, and project size issues:

• “Back to Basics: In Support of Agile Development” – Roy Morien provides insights into the factors causing failure or leading to success in a software project. The author discusses the issues related to the common misunderstanding of “project size.” The author also criticises the traditional waterfall approach and emphasises the need to adopt the agile approach.

• “Integrating Security into Agile Models: Scrum, Feature-Driven Development, and eXtreme Programming” – Imran Ghani, Adila Firdaus Bt Arbain, Zulkarnain Azham, Nor Izzaty Yasin, and Seung Ryul Jeung report the performance of intensive studies into the suitability of original agile methods for developing secure software. Their studies conclude that the original agile methods such as Scrum, feature-driven development, and eXtreme Programming do not properly guide the agile teams to develop secure software.

• “Agile Development Processes and Knowledge Documentation” – Eran Rubin and Hillel Rubin discuss and highlight the importance of documentation for agile development. The authors propose an active documentation approach that attempts to overcome the key limitation of agile methods, which is the “lack of documentation.” With the help of their active approach, agile methods can incorporate agile documentation.

Section 5 presents the recent advancements in software quality and testing, including test scenario generation, and highlights the importance of regulations and compliance:
• “An Integrated Secure Software Engineering Approach for Functional, Collaborative, and Information Concerns” – J.A. Pavlich-Mariscal, Steve Berhe, A. De la Rosa Algarín, and S. Demurjian explore a secure software engineering approach that spans functional (object-oriented), collaborative (sharing), and information (Web modelling and exchange) concerns in support of role-based, discretionary, and mandatory access control. Their findings illustrate that the proposed modelling and generation allows security changes to have less of an impact on an application.
• “Lessons from Practices and Standards in Safety-Critical and Regulated Sectors” – William G. Tuohey provides a comprehensive overview and comparison of a number of software engineering standards specific to safety-critical and regulated sectors. The author emphasises that Software Level C of RTCA/DO-178C (2011) represents a good baseline standard for software developments. A number of high priority improvement measures are identified in the chapter, the early implementation of which is likely to maximise the primary benefit of return on the investment in software process improvement.
• “The Role of Compliance and Conformance in Software Engineering” – José C. Delgado discusses the importance of compliance and conformance as the fundamental concepts for the interoperability problem of software artifacts. The author proposes a multidimensional framework and argues that the use of compliance and conformance can help establish partial interoperability among software artifacts.
• “T-Way Testing Strategies: Issues, Challenges, and Practices” – Kamal Z. Zamli, Abdul Rahman Al-Sewari, Mohammed I. Younis, and Rozmie Razif Othman emphasise the importance of software testing due to the increased possibility of faults in software. The authors review the fundamental elements of T-Way testing strategies and highlight the issues relevant to the current state-of-the-art and the current practices in the field.
• “An Improved Model-Based Technique for Generating Test Scenarios from UML Class Diagrams” – Oluwatolani Oluwagbemi and Hishammuddin Asmuni discuss the importance of test scenario generation. The authors propose a novel approach that automatically generates test scenarios by extracting the attributes, operations, and semantics of a class diagram. The authors present the performance evaluation of the proposed approach, leading to significant improvement in the generation of test scenarios.

Section 6 presents advancements in software quality measurement. Due to the increasing complexity of software, it is important for software quality to be measured early in the lifecycle. The measurement results are crucial in supporting decision-making at the various levels of development.

• “A Methodology for Model-Based Reliability Estimation” – Muhammad Adham Isa and Dayang Norhayati Abang Jawawi argue that existing reliability models are not expressive enough and do not provide adequate consistence-modelling mechanisms to allow developers to estimate the reliability parameter values. Consequently, the reliability estimation using those parameters is usually oversimplified. The authors propose a model-based reliability estimation methodology that supports the reliability estimation during the design stage. The methodology consists of a detailed description of the estimation process and the employed reliability model.
• “Non-Intrusive Adaptation of System Execution Traces for Performance Analysis of Software Systems” – Manjula Peiris and James H. Hill present the non-intrusive performance analysis approach based on information from the system execution traces. They propose a framework called the System Execution Trace Adaptation Framework that obtains the required properties by adapting system execution traces and the corresponding dataflow model. The proposed approach is successfully applied in two selected case studies (i.e., Apache ANT and DAnCE).

• “Code Clone Detection and Analysis in Open Source Applications” – Al-Fahim Mubarak-Ali, Shahida Sulaiman, Sharifah Mashita Syed-Mohamad, and Zhencang Xing review the existing code clone detection approaches and tools. They perform an empirical evaluation of two code clone detection tools, namely the generic pipeline model and the enhanced generic pipeline model, on three selected open source applications.

• “Important Issues in Software Fault Prediction: A Road Map” – Golnoush Abaei and Ali Selamat present an overview of the issues in software fault prediction and highlight the research challenges in this area. The authors present a general scheme of the steps that any researcher faces when he/she wants to conduct software fault prediction work. Different parameters such as software metrics, machine-learning techniques, and performance evaluation metrics are illustrated and compared in order to highlight the factors that have an effect on building the finest prediction models.

• “Building Defect Prediction Models in Practice” – Rudolf Ramler, Johannes Himmelbauer, and Thomas Natschläger present a defect prediction model and discuss the application of the prediction model in a real-world project. The authors provide rich discussion and insights gained by their recent experiences in data mining and defect prediction projects in the context of large software systems. One of these projects serves as an illustrative use-case throughout the chapter.

Section 7 presents the advancements in software management and evolution, including knowledge-based software engineering and software evolution visualization.

• “Knowware-Based Software Engineering: An Overview of its Origin, Essence, Core Techniques, and Future Development” – RuQian Lu and Zhi Jin propose the concept of knowledge-based software engineering called “Knowware-based software engineering.” The concept originates in their experiences in finding new techniques for knowledge-based software engineering while performing PROMIS. The key point of PROMIS is to automatically generate applications by separating the development of domain knowledge. This is achieved by automatically acquiring and summarising domain knowledge based on the pseudo-natural language.

• “Software Evolution Visualisation: Status, Challenges, and Research Directions” – Renato Lima Novais and Manoel Gomes de Mendonça Neto perform an exhaustive study and discuss the current state-of-the-art and challenges in software evolution visualisation. The authors provide useful guidelines, solutions, and recommendations to circumvent the problems in this area.

This book is designed to be used by software engineers, developers, quality-assurance personnel, testers, researchers, teachers, coaches, and students of software engineering. This is a handbook on software engineering for practitioners or researchers at all levels of expertise from the novice to the expert.
In the preparation of this book, we received many high quality contributions in response to our call for chapters. The number of contributions indicates that software engineering is a very promising area in research and is a sign of continuous improvement in practice. We are very grateful for the contributions and would like to thank all the authors for their efforts.

Imran Ghani  
*Universiti Teknologi Malaysia, Malaysia*

Wan Mohd Nasir Wan Kadir  
*Universiti Teknologi Malaysia, Malaysia*

Mohd Nazir Ahmad  
*Universiti Teknologi Malaysia, Malaysia*