# Undergraduate Topics in Computer Science

'Undergraduate Topics in Computer Science' (UTiCS) delivers high-quality instructional content for undergraduates studying in all areas of computing and information science. From core foundational and theoretical material to final-year topics and applications, UTiCS books take a fresh, concise, and modern approach and are ideal for self-study or for a one- or two-semester course. The texts are all authored by established experts in their fields, reviewed by an international advisory board, and contain numerous examples and problems, many of which include fully worked solutions.

The UTiCS concept relies on high-quality, concise books in softback format, and generally a maximum of 275–300 pages. For undergraduate textbooks that are likely to be longer, more expository, Springer continues to offer the highly regarded Texts in Computer Science series, to which we refer potential authors.

Gerard O'Regan

# Concise Guide to Software Engineering

## From Fundamentals to Application Methods

Second Edition

Springer

Gerard O'Regan
University of Central Asia
Naryn, Kyrgyzstan

ISSN 1863-7310 ISSN 2197-1781 (electronic)
Undergraduate Topics in Computer Science
ISBN 978-3-031-07815-6 ISBN 978-3-031-07816-3 (eBook)
https://doi.org/10.1007/978-3-031-07816-3



This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

*To*
*Past and present members of the Formal Methods Group (Foundations and Methods Group) at Trinity College Dublin, Ireland.*

# Preface

## Overview

The objective of this book is to provide a concise introduction to the software engineering field to students and practitioners. The principles of software engineering are discussed, and the goal is to give the reader a grasp of the fundamentals of the software engineering field, as well as guidance on how to apply the theory in an industrial environment.

## Organization and Features

Chapter 1 presents a broad overview of software engineering and discusses various software lifecycles and the activities in software development. We discuss requirements gathering and specification, software design, implementation, testing and maintenance. The lightweight Agile methodology is introduced, and it has become very popular in industry.

Chapter 2 discusses the professional responsibilities of software engineers. Engineers have a responsibility to ensure that the products that they design and develop are built to the highest possible standards and are safe for the public to use. Engineers must behave ethically in their dealings with their clients, and they need to adhere to the code of ethics of the professional engineering body.

Chapter 3 discusses ethical software engineering where the ethical software engineer needs to examine both the technical and the ethical dimensions of decisions that affect wider society. We discuss the Volkswagen emissions scandal where engineers installed a "defeat device" to enable cars to pass an emissions test.

Chapter 4 introduces project management for traditional software engineering, and we discuss project estimation, project planning and scheduling, project monitoring and control, risk management, managing communication and change, and managing project quality.

Chapter 5 discusses requirements engineering and discusses activities such as requirements gathering, requirements elicitation, requirements analysis, requirements management, and requirements verification and validation.

Chapter 6 discusses software design and development, where software design is the blueprint of the solution to be developed. It is concerned with the high-level architecture of the system, as well as the detailed design that describes the algorithms and functionality of the individual programs. The detailed design is then implemented in a programming language such as C++ or Java. We discuss software development topics such as software reuse, customized-off-the-shelf software (COTS), and open-source software development.

Chapter 7 discusses software inspections, which play an important role in building quality into a product. The well-known Fagan inspection process that was developed at IBM in the 1970s is discussed, as well as lighter review and walk-through methodologies.

Chapter 8 is concerned with software testing and discusses the various types of testing that may be carried out during the project. We discuss test planning, test case definition, test environment set-up, test execution, test tracking, test metrics, test reporting, and testing in an e-commerce environment.

Chapter 9 discusses ethics and privacy where professional ethics are a code of conduct that governs how members of a profession deal with each other and with third parties. It expresses ideals of human behaviour, and the fundamental values of the organization, and is an indication of its professionalism. Privacy is defined as "the right to be left alone," and specifies there should be no intrusion upon seclusion, and no public disclosure of private facts or false information.

Chapter 10 is concerned with metrics and problem-solving, and this includes a discussion of the balanced score card which assists in identifying appropriate metrics for the organization. The goal, question, metrics (GQM) approach is discussed, and this allows appropriate metrics related to the organization goals to be defined. A selection of sample metrics for an organization is presented, and problem-solving tools such as fishbone diagrams, Pareto charts, trend charts are discussed.

Chapter 11 is concerned with the selection and management of a software supplier. It discusses how candidate suppliers may be identified, formally evaluated against defined selection criteria, and how the appropriate supplier is selected. We discuss how the selected supplier is managed during the project.

Chapter 12 discusses software configuration management and discusses the fundamental concept of a baseline. Configuration management is concerned with identifying those deliverables that must be subject to change control and controlling changes to them.

Chapter 13 discusses software quality assurance and the importance of process quality. It is a premise in the quality field that good processes and conformance to them is essential for the delivery of high-quality product, and this chapter discusses audits, and describes how they are carried out.

Chapter 14 discusses the Agile methodology which is a popular lightweight approach to software development. Agile provides opportunities to assess the direction of a project throughout the development lifecycle and ongoing changes to requirements are considered normal in the Agile world. It has a strong collaborative style of working, and it advocates adaptive planning and evolutionary development.

Chapter 15 discusses software reliability and dependability and covers topics such as software reliability and software reliability models; the cleanroom methodology, system availability; safety and security critical systems; and dependability engineering.

Chapter 16 discusses formal methods, which consist of a set of mathematical techniques to specify and derive a program from its specification. Formal methods may be employed to rigorously state the requirements of the proposed system. They may be employed to derive a program from its mathematical specification, and they may be used to provide a rigorous proof that the implemented program satisfies its specification. They have been mainly applied to the safety critical field.

Chapter 17 presents the Z specification language, which is one of the more popular formal methods. It was developed at the Programming Research Group at Oxford University in the early 1980s. Z specifications are mathematical, and the use of mathematics ensures precision and allows inconsistencies and gaps in the specification to be identified. Theorem provers may be employed to demonstrate that the software implementation meets its specification.

Chapter 18 presents the unified modelling language (UML), which is a visual modelling language for software systems, and I used to present several views of the system architecture. It was developed at Rational Corporation as a notation for modelling object-oriented systems. We present various UML diagrams such as use case diagrams, sequence diagrams, and activity diagrams.

Chapter 19 discusses software process improvement. It begins with a discussion of a software process and discusses the benefits that may be gained from a software process improvement initiative. Various models that support software process improvement are discussed, and these include the Capability Maturity Model Integration (CMMI), ISO 9000, Personal Software Process (PSP), and Team Software Process (TSP).

Chapter 20 gives an overview of the CMMI model and discusses its five maturity levels and their constituent process areas. We discuss both the staged and continuous representations of the CMMI and SCAMPI appraisals that indicate the extent to which the CMMI has been implemented in the organization, as well as identifying opportunities for improvement.

Chapter 21 discusses various tools to support the various software engineering activities. The focus is first to define the process, and then to find tools to support the process. Tools to support project management are discussed as well as tools to support requirements engineering, configuration management, design and development activities, and software testing.

Chapter 22 discusses innovation in the software field including miscellaneous topics such as distributed systems, service-oriented architecture, software as a service, cloud computing and embedded systems. We discuss the need for innovation in software engineering and discuss some recent innovations such as aspect-oriented software engineering.

Chapter 23 is concerned with the application of the legal system to the computing field. This includes the protection of intellectual property such as patents, copyright, trademarks and trade secrets, and the resolution of disputes between parties.

Chapter 24 discusses cybersecurity and cybercrime. Cybercrime is a crime that involves a computer and a network. The computer may be the vehicle by which the crime was conducted, or it may be the target of the crime. Cybersecurity is concerned with the ability of a computer system to protect itself from attacks, and there are several characteristics of security such as confidentiality, integrity, and availability.

Chapter 25 is the concluding chapter in which we summarize the journey that we have travelled in this book.

## Audience

The main audience of this book is computer science students who are interested in learning about software engineering and in learning on how to build high-quality and reliable software on time and on budget. It will also be of interest to industrialists including software engineers, quality professionals and software managers, as well as the motivated general reader.

## Acknowledgments

I am deeply indebted to family and friends who supported my efforts in this endeavour, and my thanks, as always, to the team at Springer. This book is dedicated to present and past members of the Formal Methods Group (Foundations and Methods Group) at Trinity College Dublin where the author spent several happy years. I would especially like to thank Dr. Mícheál Mac An Airchinnigh, Dr. Andrew Butterfield, Dr. Hugh Gibbons, Dr. Arthur Hughes, Alexis Donnelly, Dara Gallagher, Eoin McDonnell, Gradamir Starovic, and Glenn Strong.

Cork, Ireland Gerard O'Regan

# Contents

# List of Figures

# List of Tables