# Software Engineering Carlo Ghezzi

The book is about a very active research field in software engineering. In modern society, the fact of the world's high reliance on software requires the system's robustness, i.e., continual availability and satisfactory service quality. This requirement gives rise to the popularity of the research on the self-adaptive software in open environment. There are some academic conferences dedicated to this field. But there is a lack of monographs about the topic. We believe such need is unmet in marketplace. By publishing the book, it can help bridge the gap and bring benefits to readers thereof. Key Features: The topic is well-motivated, interesting and actively studied worldwide The research represents as the state-of-the-art in the field The technical part of the book is rigidly evaluated The theoretical part of the book is sound and proved The organization and presentation of the book will be double-checked by professional scholars

Software Engineering Economics is an invaluable guide to determining software costs, applying the fundamental concepts of microeconomics to software engineering, and utilizing economic analysis in software engineering decision making.

This book discusses how model-based approaches can improve the daily practice of software professionals. This is known as Model-Driven Software Engineering (MDSE) or, simply, Model-Driven Engineering (MDE). MDSE practices have proved to increase efficiency and effectiveness in software development, as demonstrated by various quantitative and qualitative studies. MDSE adoption in the software industry is foreseen to grow exponentially in the near future, e.g., due to the convergence of software development and business analysis. The aim

of this book is to provide you with an agile and flexible tool to introduce you to the MDSE world, thus allowing you to quickly understand its basic principles and techniques and to choose the right set of MDSE instruments for your needs so that you can start to benefit from MDSE right away. The book is organized into two main parts. The first part discusses the foundations of MDSE in terms of basic concepts (i.e., models and transformations), driving principles, application scenarios, and current standards, like the well-known MDA initiative proposed by OMG (Object Management Group) as well as the practices on how to integrate MDSE in existing development processes. The second part deals with the technical aspects of MDSE, spanning from the basics on when and how to build a domain-specific modeling language, to the description of Model-to-Text and Model-to-Model transformations, and the tools that support the management of MDSE projects. The second edition of the book features: a set of completely new topics, including: full example of the creation of a new modeling language (IFML), discussion of modeling issues and approaches in specific domains, like business process modeling, user interaction modeling, and enterprise architecture complete revision of examples, figures, and text, for improving readability, understandability, and coherence better formulation of definitions, dependencies between concepts and ideas addition of a complete index of book content In addition to the contents of the book, more resources are provided on the book's website http://www.mdse-book.com, including the examples presented in the book. A major challenge for modern software systems is to become more cost-effective, while being versatile, flexible, resilient, energy-efficient, customizable, and configurable when reacting to run-time changes that may occur within the system itself, its environment or requirements. One of the most promising approaches to achieving such properties is to equip the software system

with self-adaptation capabilities. Despite recent advances in this area, one key aspect that remains to be tackled in depth is the provision of assurances. Originating from a Dagstuhl seminar held in December 2013, this book constitutes the third volume in the series “Software Engineering for Self-Adaptive Systems”, and looks specifically into the provision of assurances. Opening with an overview chapter on Research Challenges, the book presents 13 further chapters written and carefully reviewed by internationally leading researchers in the field. The book is divided into topical sections on research challenges, evaluation, integration and coordination, and reference architectures and platforms.

This book constitutes the strictly refereed post-workshop proceedings of the International Workshop on Requirements Targeting Software and Systems Engineering, RTSE '97, held in Bernried, Germany in October 1997. The 15 revised full papers presented in the book were carefully revised and reviewed for inclusion in the book. Among the authors are internationally leading researchers. The book is divided in sections on foundations of software engineering, methodology, evaluation and case studies, and tool support and prototyping.

This open access book includes contributions by leading researchers and industry thought leaders on various topics related to the essence of software engineering and their application in industrial projects. It offers a broad overview of research findings dealing with current practical software engineering issues and also pointers to potential future developments. Celebrating the 20th anniversary of adesso AG, adesso gathered some of the pioneers of software engineering including Manfred Broy, Ivar Jacobson and Carlo Ghezzi at a special symposium, where they presented their thoughts about latest software engineering research and which are part of this book. This way it offers readers a concise overview of the essence of

software engineering, providing valuable insights into the latest methodological research findings and adesso's experience applying these results in real-world projects.

This open access book includes contributions by leading researchers and industry thought leaders on various topics related to the essence of software engineering and their application in industrial projects. It offers a broad overview of research findings dealing with current practical software engineering issues and also pointers to potential future developments.Celebrating the 20th anniversary of adesso AG, adesso gathered some of the pioneers of software engineering including Manfred Broy, Ivar Jacobson and Carlo Ghezzi at a special symposium, where they presented their thoughts about latest software engineering research and which are part of this book. This way it offers readers a concise overview of the essence of software engineering, providing valuable insights into the latest methodological research findings and adesso's experience applying these results in real-world projects. This work was published by Saint Philip Street Press pursuant to a Creative Commons license permitting commercial use. All rights not granted by the work's license are retained by the author or authors.

Here, the authors strive to change the way logic and discrete math are taught in computer science and mathematics: while many books treat logic simply as another topic of study, this one is unique in its willingness to go one step further. The book traets logic as a basic tool which may be applied in essentially every other area.

This open access book aims to set an agenda for research and action in the field of Digital Humanism through short essays written by selected thinkers from a variety of disciplines, including computer science, philosophy, education, law, economics, history, anthropology,

political science, and sociology. This initiative emerged from the Vienna Manifesto on Digital Humanism and the associated lecture series. Digital Humanism deals with the complex relationships between people and machines in digital times. It acknowledges the potential of information technology. At the same time, it points to societal threats such as privacy violations and ethical concerns around artificial intelligence, automation and loss of jobs, ongoing monopolization on the Web, and sovereignty. Digital Humanism aims to address these topics with a sense of urgency but with a constructive mindset. The book argues for a Digital Humanism that analyses and, most importantly, influences the complex interplay of technology and humankind toward a better society and life while fully respecting universal human rights. It is a call to shaping technologies in accordance with human values and needs.

This book accompanies the event "Matinée with Carlo Ghezzi - from Programming Languages to Software Engineering", held on June 3, 2012, in Zurich, Switzerland, collocated with ICSE 2012. The event gave friends and colleagues an opportunity to celebrate Carlo's many achievements and accomplishments.Carlo Ghezzi is a professor and chair of Software Engineering at the Politecnico di Milano, Italy, and an adjunct professor at the Università della Svizzera italiana, Switzerland.He received his Dr. Eng. degree in Electrical Engineering from the Politecnico di Milano, where he spent most of his professional life, as assistant, associate, and full professor. He is a 1999 ACM fellow and a 2006 IEEE fellow, as well as a member of Istituto Lombardo Accademia di Scienze e Lettere. In 2008, he has been awarded a prestigious Advanced Investigators Grant from the European Research Council, funding the SMScom project.

A novel, model-driven approach to security requirements engineering that focuses on socio-

technical systems rather than merely technical systems. Security requirements engineering is especially challenging because designers must consider not just the software under design but also interactions among people, organizations, hardware, and software. Taking this broader perspective means designing a secure socio-technical system rather than a merely technical system. This book presents a novel, model-driven approach to designing secure socio-technical systems. It introduces the Socio-Technical Modeling Language (STS-ML) and presents a freely available software tool, STS-Tool, that supports this design approach through graphical modeling, automated reasoning capabilities to verify the models constructed, and the automatic derivation of security requirements documents. After an introduction to security requirements engineering and an overview of computer and information security, the book presents the STS-ML modeling language, introducing the modeling concepts used, explaining how to use STS-ML within the STS method for security requirements, and providing guidelines for the creation of models. The book then puts the STS approach into practice, introducing the STS-Tool and presenting two case studies from industry: an online collaborative platform and an e-Government system. Finally, the book considers other methods that can be used in conjunction with the STS method or that constitute an alternative to it. The book is suitable for course use or as a reference for practitioners. Exercises, review questions, and problems appear at the end of each chapter.

"This volume contains the proceedings of the fourth European Software Engineering Conference. It contains 6 invited papers and 27 contributed papers selected from more than 135 submissions. The volume has a mixtureof themes. Some, such as software engineering and computer supported collaborative work, are forward-looking and anticipate future

developments; others, such as systems engineering, are more concerned with reports of practical industrial applications. Some topics, such as software reuse, reflect the fact that some of the concerns first raised in 1969 when software engineering was born remain unsolved problems. The contributed papers are organized under the following headings: requirements specification, environments, systems engineering, distributed software engineering, real-time systems, software engineering and computer supported collaborative work, software reuse, software process, and formal aspects of software engineering."--PUBLISHER'S WEBSITE.

This text combines a practical, hands-on approach to programming with the introduction of sound theoretical support focused on teaching the construction of high-quality software. A major feature of the book is the use of Design by Contract.

This book provides essential insights on the adoption of modern software engineering practices at large companies producing software-intensive systems, where hundreds or even thousands of engineers collaborate to deliver on new systems and new versions of already deployed ones. It is based on the findings collected and lessons learned at the Software Center (SC), a unique collaboration between research and industry, with Chalmers University of Technology, Gothenburg University and Malmö University as academic partners and Ericsson, AB Volvo, Volvo Car Corporation, Saab Electronic Defense Systems, Grundfos, Axis Communications, Jeppesen (Boeing) and Sony Mobile as industrial partners. The 17 chapters present the “Stairway to Heaven” model, which represents the typical evolution path companies move through as they develop and mature their software engineering capabilities. The chapters describe theoretical frameworks, conceptual models and, most importantly, the industrial experiences gained by the partner companies in applying novel software engineering

techniques. The book's structure consists of six parts. Part I describes the model in detail and presents an overview of lessons learned in the collaboration between industry and academia. Part II deals with the first step of the Stairway to Heaven, in which R&D adopts agile work practices. Part III of the book combines the next two phases, i.e., continuous integration (CI) and continuous delivery (CD), as they are closely intertwined. Part IV is concerned with the highest level, referred to as "R&D as an innovation system," while Part V addresses a topic that is separate from the Stairway to Heaven and yet critically important in large organizations: organizational performance metrics that capture data, and visualizations of the status of software assets, defects and teams. Lastly, Part VI presents the perspectives of two of the SC partner companies. The book is intended for practitioners and professionals in the software-intensive systems industry, providing concrete models, frameworks and case studies that show the specific challenges that the partner companies encountered, their approaches to overcoming them, and the results. Researchers will gain valuable insights on the problems faced by large software companies, and on how to effectively tackle them in the context of successful cooperation projects.

The dependence on quality software in all areas of life is what makes software engineering a key discipline for today's society. Thus, over the last few decades it has been increasingly recognized that it is particularly important to demonstrate the value of software engineering methods in real-world environments, a task which is the focus of empirical software engineering. One of the leading protagonists of this discipline worldwide is Prof. Dr. Dr. h.c. Dieter Rombach, who dedicated his entire career to empirical software engineering. For his many important contributions to the field he has received numerous awards and recognitions,

including the U.S. National Science Foundation's Presidential Young Investigator Award and the Cross of the Order of Merit of the Federal Republic of Germany. He is a Fellow of both the ACM and the IEEE Computer Society. This book, published in honor of his 60th birthday, is dedicated to Dieter Rombach and his contributions to software engineering in general, as well as to empirical software engineering in particular. This book presents invited contributions from a number of the most internationally renowned software engineering researchers like Victor Basili, Barry Boehm, Manfred Broy, Carlo Ghezzi, Michael Jackson, Leon Osterweil, and, of course, by Dieter Rombach himself. Several key experts from the Fraunhofer IESE, the institute founded and led by Dieter Rombach, also contributed to the book. The contributions summarize some of the most important trends in software engineering today and outline a vision for the future of the field. The book is structured into three main parts. The first part focuses on the classical foundations of software engineering, such as notations, architecture, and processes, while the second addresses empirical software engineering in particular as the core field of Dieter Rombach's contributions. Finally, the third part discusses a broad vision for the future of software engineering.

This tutorial book presents an augmented selection of the material presented at the Software Engineering Education and Training Track at the International Conference on Software Engineering, ICSE 2005, held in St. Louis, MO, USA in May 2005. The 12 tutorial lectures presented cover software engineering education, state of the art and practice: creativity and rigor, challenges for industries and academia, as well as future directions.

A new, quantitative architecture simulation approach to software design that circumvents costly testing cycles by modeling quality of service in early design states. Too often, software

designers lack an understanding of the effect of design decisions on such quality attributes as performance and reliability. This necessitates costly trial-and-error testing cycles, delaying or complicating rollout. This book presents a new, quantitative architecture simulation approach to software design, which allows software engineers to model quality of service in early design stages. It presents the first simulator for software architectures, Palladio, and shows students and professionals how to model reusable, parametrized components and configured, deployed systems in order to analyze service attributes. The text details the key concepts of Palladio's domain-specific modeling language for software architecture quality and presents the corresponding development stage. It describes how quality information can be used to calibrate architecture models from which detailed simulation models are automatically derived for quality predictions. Readers will learn how to approach systematically questions about scalability, hardware resources, and efficiency. The text features a running example to illustrate tasks and methods as well as three case studies from industry. Each chapter ends with exercises, suggestions for further reading, and “takeaways” that summarize the key points of the chapter. The simulator can be downloaded from a companion website, which offers additional material. The book can be used in graduate courses on software architecture, quality engineering, or performance engineering. It will also be an essential resource for software architects and software engineers and for practitioners who want to apply Palladio in industrial settings.

This book presents the thoroughly refereed and revised post-workshop proceedings of the 17th Monterey Workshop, held in Oxford, UK, in March 2012. The workshop explored the challenges associated with the Development, Operation and Management of Large-Scale

complex IT Systems. The 21 revised full papers presented were significantly extended and improved by the insights gained from the productive and lively discussions at the workshop, and the feedback from the post-workshop peer reviews.

The carefully reviewed papers in this state-of-the-art survey describe a wide range of approaches coming from different strands of software engineering, and look forward to future challenges facing this ever-resurgent and exacting field of research.

This book explores research from the researchers' perspective: why to engage in research, what methods to follow, how to operate in daily life, what the responsibilities are, how to engage with society, and the ethical issues confronting professionals in their day-to-day research. The book systematically discusses what every student should be told when entering academic or industrial research so that they can avoid going through the painful process of learning by personal experience and lots of errors. Rather than being technical, it is philosophical and sometimes even anecdotal, combining factual information and commonly accepted knowledge on research and its methods, while at the same time clearly distinguishing between objective and factual concepts and data, and subjective considerations. The book is about scientific research in general and as such holds true for any scientific field. However, it is fair to say that the different fields differ in their research cultures and in their eco-systems. The book reflects the author's experience accumulated over almost 50 years of teaching graduate courses and lecturing in doctoral symposia at Politecnico di Milano, University of Zurich, TU Wien, Peking University, and at various conferences, and of academic research in informatics (also known as computer science). This book is mainly intended for students who are considering research as a possible career option; for in-progress researchers who have

entered doctoral programs; and for junior postdoctoral researchers. It will also appeal to senior researchers involved in mentoring students and junior researchers.

This volume focuses on current and future trends in the interplay between software engineering and artificial intelligence.This interplay is now critical to the success of both disciplines, and it also affects a wide range of subject areas. The articles in this volume survey the significant work that has been accomplished, describe the state of the art, analyze the current trends, and predict which future directions have the most potential for success. Areas covered include requirements engineering, real-time systems, reuse technology, development environments and meta-environments, process representations, safety-critical systems, and metrics and measures for processes and products.

The first course in software engineering is the most critical. Education must start from an understanding of the heart of software development, from familiar ground that is common to all software development endeavors. This book is an in-depth introduction to software engineering that uses a systematic, universal kernel to teach the essential elements of all software engineering methods. This kernel, Essence, is a vocabulary for defining methods and practices. Essence was envisioned and originally created by Ivar Jacobson and his colleagues, developed by Software Engineering Method and Theory (SEMAT) and approved by The Object Management Group (OMG) as a standard in 2014. Essence is a practice-independent framework for thinking and reasoning about the practices we have and the practices we need. Essence establishes a shared and standard understanding of what is at the heart of software development. Essence is

agnostic to any particular method, lifecycle independent, programming language independent, concise, scalable, extensible, and formally specified. Essence frees the practices from their method prisons. The first part of the book describes Essence, the essential elements to work with, the essential things to do and the essential competencies you need when developing software. The other three parts describe more and more advanced use cases of Essence. Using real but manageable examples, it covers the fundamentals of Essence and the innovative use of serious games to support software engineering. It also explains how current practices such as user stories, use cases, Scrum, and micro-services can be described using Essence, and illustrates how their activities can be represented using the Essence notions of cards and checklists. The fourth part of the book offers a vision how Essence can be scaled to support large, complex systems engineering. Essence is supported by an ecosystem developed and maintained by a community of experienced people worldwide. From this ecosystem, professors and students can select what they need and create their own way of working, thus learning how to create ONE way of working that matches the particular situation and needs.

This book provides selective, in-depth coverage of the fundamentals of software engineering by stressing principles and methods through rigorous formal and informal approaches. In contrast to other books which are based on the lifecycle model of software development, the authors emphasize identifying and applying fundamental

principles that are applicable throughout the software lifecycle. This emphasis enables readers to respond to the rapid changes in technology that are common today. Principles and techniques are emphasized rather than specific tools--users learn why particular techniques should or should not be used. Understanding the principles and techniques on which tools are based makes mastering a variety of specific tools easier. KEY TOPICS: The authors discuss principles such as design, specification, verification, production, management and tools. Now coverage includes: more detailed analysis and explanation of object-oriented techniques; the use of Unified Modeling Language (UML); requirements analysis and software architecture; Model checking--a technique that provides automatic support to the human activity of software verification; GQM--used to evaluate software quality and help improve the software process; Z specification language. MARKET: For software engineers.

The book is concerned with the broad topic of software engineering. It comprises the proceedings of the European Software Engineering Conference (ESEC) held at the University of Warwick in the United Kingdom in September 1989 and its primary purpose is to summarise the state of the art in software engineering as represented by the papers at that conference. The material covers both submitted papers and a number of invited papers given at the conference. The topics covered include: metrics and measurement, software process modelling, formal methods including their use in industry, software configuration management, software development environments, and

requirements engineering. The book is most likely to be of interest to researchers and professionals working in the field of software development. The primary value of the book is that it gives an up-to-date treatment of its subject material and includes some interesting discussions of the transfer of research ideas into industrial practice.
Object-Oriented Software Engineering: An Agile Unified Methodology by David Kung presents a step-by-step methodology that integrates modeling and design, UML, patterns, test-driven development, quality assurance, configuration management, and agile principles throughout the life cycle. The overall approach is casual and easy to follow, with many practical examples that show the theory at work. The author uses his experiences as well as real-world stories to help the reader understand software design principles, patterns, and other software engineering concepts. The book also provides stimulating exercises that go far beyond the type of question that can be answered by simply copying portions of the text.
This book is an introduction to graph transformation as a foundation to model-based software engineering at the level of both individual systems and domain-specific modelling languages. The first part of the book presents the fundamentals in a precise, yet largely informal way. Besides serving as prerequisite for describing the applications in the second part, it also provides a comprehensive and systematic survey of the concepts, notations and techniques of graph transformation. The second part presents and discusses a range of applications to both model-based software engineering and

domain-specific language engineering. The variety of these applications demonstrates how broadly graphs and graph transformations can be used to model, analyse and implement complex software systems and languages. This is the first textbook that explains the most commonly used concepts, notations, techniques and applications of graph transformation without focusing on one particular mathematical representation or implementation approach. Emphasising the research and engineering methodologies used, it will be a valuable resource for graduate students, practitioners and researchers in software engineering, foundations of programming and formal methods.

SEAFOOD 2009: Enabling Global Partnerships to Deliver on Business Needs

Companies have been outsourcing areas of software development work for many years, either because of the engineering challenges or because the outsourced aspect is not central to their core business. A profound transformation has been a?ecting this model over recent years: a massive transfer of development - tivities from the USA and Europe to a skilled labor force in service-providing countries. This transformation has been driven by the demands of a global bu- ness climate seeking to increase the value delivery of IT investment. However, the ability to realize this value can prove problematic in practice. Of particular concern are the hidden costs of globally distributed models of working, such as understanding and communicating the true business needs across organizational and cultural boundaries. To address such issues, o?shore outsourcing requires di?erent support from in-

housedevelopmentandthismeansadaptingfamiliartechniques,processesand tools to this setting, as well as perhaps creating innovative new ones. Coupled with this industry transformation there is hence a pressing need to re-examine thosesoftwareengineeringapproachesthateither facilitate orimpede this model of working. With an inevitable focus on the economy in 2009, business decisions regarding the sourcing of software development projects will come under close scrutiny. It will become increasingly critical to design global partnerships that both clarify cost/bene?ts and enable delivery on business needs.

This book constitutes the refereed proceedings of the 9th International Conference on Fundamental Approaches to Software Engineering, FASE 2006, held in Vienna, Austria in March 2006 as part of ETAPS. The 27 revised full papers, two tool papers presented together with two invited papers were carefully reviewed and selected from 166 submissions. The papers are organized in topical sections.

Software has long been perceived as complex, at least within Software Engineering circles. We have been living in a recognised state of crisis since the first NATO Software Engineering conference in 1968. Time and again we have been proven unable to engineer reliable software as easily/cheaply as we imagined. Cost overruns and expensive failures are the norm. The problem is fundamentally one of complexity: software is fundamentally complex because it must be precise. Problems that appear to be specified quite easily in plain language become far more complex when written in a more formal notation, such as computer code. Comparisons with other engineering disciplines are deceptive. One cannot easily increase the

factor of safety of software in the same way that one could in building a steel structure, for example. Software is typically built assuming perfection, often without adequate safety nets in case the unthinkable happens. In such circumstances it should not be surprising to find out that (seemingly) minor errors have the potential to cause entire software systems to collapse. The goal of this book is to uncover techniques that will aid in overcoming complexity and enable us to produce reliable, dependable computer systems that will operate as intended, and yet are produced on-time, in budget, and are evolvable, both over time and at run time. We hope that the contributions in this book will aid in understanding the nature of software complexity and provide guidance for the control or avoidance of complexity in the engineering of complex software systems.

A high-level introduction to new technologies andmethods in the field of software engineering Recent years have witnessed rapid evolution of software engineering methodologies, and until now, there has been no single-source introduction to emerging technologies in the field. Written by a panel of experts and divided into four clear parts, Emerging Methods, Technologies, and Process Management in SoftwareEngineering covers: Software Architectures – Evolution of software composition mechanisms; compositionality in software product lines; and teaching design patterns Emerging Methods – The impact of agent-oriented software engineering in service-oriented computing; testing object-oriented software; the UML and formal methods; and modern Web application development Technologies for Software Evolution – Migrating to Web services and software evolution analysis and visualization Process Management – Empirical experimentation in software engineering and foundations of agile methods Emerging Methods, Technologies, and Process Management in Software

Engineering is a one-stop resource for software engineering practitioners and professionals, and also serves as an ideal textbook for undergraduate and graduate students alike.
This is a valuepack for Software Engineering undergraduate courses. Fundamentals of Software Engineering: International Edition, 2/Eis Appropriate for both undergraduate and graduate introductory software engineering courses found in Computer Science and Computer Engineering departments. This text provides selective, in-depth coverage of the fundamentals of software engineering by stressing principles and methods through rigorous formal and informal approaches. The authors emphasize, identify, and apply fundamental principles that are applicable throughout the software lifecycle, in contrast to other texts which are based in the lifecycle model of software development. This emphasis enables students to respond to the rapid changes in technology that are common today. Also included is How to Break Softwarewhich takes a very applied and non-rigid approach to teaching how to test software for common bugs. It is a departure from conventional testing in which testers prepare a written test plan and then use it as a script when testing the software.Instead of relying on a rigid plan, it should be intelligence, insight, experience and a nose for where the bugs are hiding that guide testers. This book helps testers develop this insight. The techniques presented in this book not only allow testers to go off-script, they encourage them to do so. Don't blindly follow a document that may be out of date and that was written before the product was even testable. Instead, use your head Open your eyes Think a little, test a little and then think a little more.