

# Preface to the Second Edition

This is the second edition of the textbook in which most of the concepts introduced in the first edition are extended and updated, and a significant amount of new material has been added. While preserving the original intent of focusing on **software specification**, this edition emphasizes the practice of formal methods for **specification and verification activities** for different types of software systems and at different stages of developing the software systems. This expanded view is reinforced both in the organization of the book and in the presentation of its contents. The primary driving force for writing the second edition came from Springer-Verlag, London, who expressed a great desire and strong interest in catering to the growing needs of students and researchers in the area of Formal Software Engineering.

---

## Background and Motivation

Although during the initial stages of formal methods research there was only a marginal use of formal methods in industry, new languages, techniques and tools developed during early 1990s have spurred great interest in adapting formal methods in industries. In fact, the 1990s witnessed an explosion of new developments in formal methods research. NASA Langley Research Center was the first hub of formal methods research and practice. The researchers at Langley focused on large scale commercial projects that are suitable for injecting formal methods. They felt that the industries were reluctant to use formal methods because of inadequate tools, inadequate background, and lack of adequate examples. However, this situation started changing gradually during 1995–2004 when new directions of research and development of tools in the three areas *Software Specification Methods*, *Model Checking*, and *Theorem Proving* provided a great spur for formal development activity in industries. Most notably, software engineers at nuclear power stations, aerospace and transportation industries used formal methods to formally specify and verify the properties of *safety critical* parts in systems. In 1998, the fully automatic driverless subway was launched in Paris Metro and, in 2006, the fully automatic driverless shuttle servicing the various terminals at Roissy Airport, Paris was launched. With success stories such as these

arose a desire in academia and industries to learn formal methods more systematically. In order to choose a method that is appropriate for a specific application that demands concepts such as *causality*, *concurrency*, and *conflict avoidance* a certain level of expertise in formal methods education is necessary. Getting to know that it is possible to mix different abstractions from different languages to model heterogeneous systems is an asset for an efficient development process. Nowadays formal methods often are bundled up with tools, many available as open source software, to support architectural principles of generality and orthogonality. In view of these spectrum of changes and success stories, software engineers now have several case studies to learn from and choose languages and methods with a rich repertoire of appropriate concepts for their intended applications. In keeping up with this trend this second edition is offered. In writing this second edition, the expectation is that formal methods will be well integrated into the teaching of software engineering programs. In this hope, topics related to the integration of formal methods in software development process are discussed quite early in the text and are followed by presentations of abstraction principles, formalism definitions, notations of formalism, and a wide variety of fairly detailed specification examples.

---

## What is New in the Second Edition?

Old material has been updated to improve both content and presentation. Some chapters in the first edition of the text have undergone extensive revisions. In some cases, an old chapter has been split into two or more chapters and in each of them extensive new material have been added. New chapters that discuss Object-Z, B-Method, and Calculus of Communicating Systems have been added. The entire book has been structured into six parts. The distinguishing features of this restructured and expanded second edition are as follows.

**Part I** The first part of the book introduces specification fundamentals. The material is presented in four chapters. An elaborate introduction to the role of specification is followed by discussions on specification activities and specification qualities. The first part concludes with a discussion on abstraction principles, illustrated with a domain abstraction example.

**Part II** The second part introduces the basics of formalism, automata notations used in formal languages, study extensions to the basic automata notation, and concludes with a discussion on the classification of formal specification techniques. This material is presented in four chapters. The chapters that discuss automata and extended state machine notations are almost self-contained. A variety of examples that arise in software construction are taken up for formal modeling using different variations of state machines. Top-down and bottom-up constructions of formal models, their sequential and parallel compositions are discussed and illustrated with examples.

**Part III** The third part of the book is entirely devoted to logic. Propositional logic, predicate logic, and temporal logic are treated in three separate chapters. The presentation focuses on introducing the logics as formal languages, and hence introduces their syntax, semantics, and reasoning methods in succession. The expressive power of predicate logic is illustrated for representing knowledge, policies, as well as serving as axiomatic system for program verification. For the latter purpose, Hoare axioms are presented and illustrated by verifying simple sequential programs. Temporal logic chapter gives a detailed discussion of the syntax, and semantics of linear temporal logic. Many examples from reactive systems and concurrent systems are chosen to emphasize the expressivity of the logic languages in specifying such systems and their properties. A discussion of axiomatic proof method and model checking are included.

**Part IV** Most of the model-based specification languages are based on set theory and first-order predicate logic. Therefore, it is essential to have a strong background in set theory and relations. This part of the book includes one chapter on set theory and relations. Most parts of this chapter are retained from the previous edition of the book.

**Part V** Three specification methods are discussed to illustrate the property-oriented approach to specifications. Two of the chapters, *Algebraic Specifications* and *Larch*, are left unchanged. The chapter *Calculus of Communicating Systems* is new and it discusses Milner's algebraic approach to specifying communication and concurrency. Some examples discussed in Temporal Logic chapter are drawn in here to strike a comparison between the two approaches. An effort has been made to make the discussion in this chapter simple, rigorous, and self-contained.

**Part VI** This part is devoted to model-based specification techniques. Four such techniques are described in detail. These are VDM-SL, Z, Object-Z and the B-Method. Material on VDM-SL and Z are retained from the previous edition of the book, while the bibliographic references have been updated. Two new chapters have been introduced, one for Object-Z and another for the B-Method. The material for new chapters are presented in the same style as in the old chapters. Also, the two new chapters include extensive examples and case studies, and provide a detailed tutorial of the techniques introduced in those chapters.

---

## How to Use the Book

In the second edition of the book, we have added considerable new material and we have also restructured the chapters into various parts. Consequently, those who have used the first edition may see a different layout of the book. The book includes several different specification techniques grouped into various categories. In addition, it also includes chapters with necessary mathematical background for these techniques. Because of the diverse nature of these techniques, the book can be used by different groups of people for different purposes. Below we suggest a few streams of course offerings to fit different curriculum needs.

1. Chapters in Part I are required for further reading of the book.
2. Based on Part I, Part II, Chaps. 9, 10 of Part III, and Part IV a one-semester undergraduate course within a software engineering program can be given. This course is intended to be an Introduction to Formal Software Engineering Methods. The course can be extended into another semester by covering the material from one of the four specification languages discussed in Chaps. 16 through 19, and choosing a project in which the students would write a complete specification and analyze the specification. The examples and case studies given in these chapters would help the students to achieve this goal.
3. Chapters in Part II, Part III, and Chap. 15 from Part V can be offered as a one-semester course for senior undergraduate students or first year graduate students in computer science and computer engineering programs. This course is intended to be an Introduction to Formal Methods.
4. Parts V and VI are devoted to various formal specification techniques. Each chapter in these parts gives a thorough tutorial of one specification technique. Together with the mathematical fundamentals described in Part IV, each chapter in Parts V and VI can be individually used to teach a one-semester course on a particular specification technique. The course will introduce the formal method in some depth, choosing appropriate tools suggested in the bibliographic notes of these chapters. It is suitable to teach this course at senior undergraduate level or at the graduate level provided that the students are exposed to some of the mathematical fundamentals described in Parts I through IV before taking this course. Alternately, a quick overview of the fundamentals can be covered in few weeks and the rest of the semester can be spent on the syntax and semantics of the chosen specification technique.
5. An advanced graduate-level course can be taught using any one of the techniques discussed in Parts V and VI with emphasis on developing complete specification for a fairly large problem. This would involve refinement, proof obligation, and implementation.
6. Another option would be to teach an advanced graduate-level course that requires the students to critically compare the techniques in each group and write a report. For example, one course could be taught on model-based specification techniques, all chapters in Part VI. Students in this course will get an in-depth understanding of the techniques and also would be able to choose the appropriate technique for a given problem.
7. Practitioners of formal methods, especially those who use formal methods for industrial applications, can use this book as a reference. In particular, the chapters in Parts V and VI have been written in such a way that a practitioner who is familiar with one technique can quickly jump start with another technique with little time. The examples and case studies in each chapter in these two parts provide sufficient information for a practitioner to start writing the specification for a new application without much preparation time.

---

## Intended Audience

This book is written to serve as a text book for students in Software Engineering, Computer Science, Computer Engineering and Information Systems Engineering. Software professionals who want to familiarize themselves with formal methods can use this book as a

good reference. The wide coverage of various formal specification techniques and the tutorial nature of descriptions of each individual technique make the book as a good resource for formal methods, all in one place. The bibliographic notes given at the end of each chapter provokes the reader to expand their horizon beyond the materials discussed in that chapter along with information on tool support.

---

## Acknowledgments

Our sincere thanks go to the editorial board of Springer-Verlag, London whose persistent persuasion gave us sufficient motivation to engage in this venture. Many people have assisted us in bringing out the second edition of this book. First of all, those who have helped us during the first edition deserve a second round of applause. During the extensive revisions and additions to the second edition we received great support from Lei Feng, Pankaj Goyal, Naseem Ibrahim, Diep Mai, Ka Lok Man, Mubarak Mohammad, Shiri Nematollah, and Olga Ormandjieva. We express our sincere thanks for their dedication and timely support.



<http://www.springer.com/978-0-85729-276-6>

Specification of Software Systems

Alagar, V.S.; Periyasamy, K.

2011, XXVI, 646 p., Hardcover

ISBN: 978-0-85729-276-6