Preface

This book addresses software test and analysis in the context of an overall effort to achieve quality. It is designed for use as a primary textbook for a course in software test and analysis or as a supplementary text in a software engineering course, and as a resource for software developers.

The main characteristics of this book are:

- It assumes that the reader's goal is to achieve a suitable balance of cost, schedule, and quality. It is not oriented toward critical systems for which ultra-high reliability must be obtained regardless of cost, nor will it be helpful if one's aim is to cut cost or schedule regardless of consequence.

- It presents a selection of techniques suitable for near-term application, with sufficient technical background to understand their domain of applicability and to consider variations to suit technical and organizational constraints. Techniques of only historical interest and techniques that are unlikely to be practical in the near future are omitted.

- It promotes a vision of software testing and analysis as integral to modern software engineering practice, equally as important and technically demanding as other aspects of development. This vision is generally consistent with current thinking on the subject, and is approached by some leading organizations, but is not universal.

- It treats software testing and static analysis techniques together in a coherent framework, as complementary approaches for achieving adequate quality at acceptable cost.

Why This Book?

One cannot "test quality into" a badly constructed software product, but neither can one build quality into a product without test and analysis. The goal of acceptable quality at acceptable cost is both a technical and a managerial challenge, and meeting the goal requires a grasp of both the technical issues and their context in software development.

It is widely acknowledged today that software quality assurance should not be a phase between development and deployment, but rather a set of ongoing activities interwoven with every task from initial requirements gathering through evolution of the deployed product. Realization of this vision in practice is often only partial. It requires careful choices and combinations of techniques fit to the organization, products, and processes, but few people are familiar with the full range of techniques, from inspection to testing to automated analyses. Those best positioned to shape the organization and its processes are seldom familiar with the technical issues, and vice versa. Moreover, there still persists in many organizations a perception that quality assurance requires less skill or background than other aspects of development.

This book provides students with a coherent view of the state of the art and practice, and provides developers and managers with technical and organizational approaches to push the state of practice toward the state of the art.

Who Is This Book For?

Students who read portions of this book will gain a basic understanding of principles and issues in software test and analysis, including an introduction to process and organizational issues. *Developers*, including quality assurance professionals, will find a variety of techniques with sufficient discussion of technical and process issues to support adaptation to the particular demands of their organization and application domain. *Technical managers* will find a coherent approach to weaving software quality assurance into the overall software process. All readers should obtain a clearer view of the interplay among technical and nontechnical issues in crafting an approach to software quality.

Students, developers, and technical managers with a basic background in computer science and software engineering will find the material in this book accessible without additional preparation. Some of the material is technically demanding, but readers may skim it on a first reading to get the big picture, and return to it at need.

A basic premise of this book is that effective quality assurance is best achieved by selection and combination of techniques that are carefully woven into (not grafted onto) a software development process for a particular organization. A software quality engineer seeking technical advice will find here encouragement to consider a wider context and participate in shaping the development process. A manager whose faith lies entirely in process, to the exclusion of technical knowledge and judgment, will find here many connections between technical and process issues, and a rationale for a more comprehensive view.

How to Read This Book

This book is designed to permit selective reading. Most readers should begin with Part I, which presents fundamental principles in a coherent framework and lays the groundwork for understanding the strengths and weaknesses of individual techniques and their application in an effective software process. Part II brings together basic technical background for many testing and analysis methods. Those interested in particular methods may proceed directly to the relevant chapters in Part III of the book. Where there are dependencies, the *Required Background* section at the beginning of a chapter indicates what should be read in preparation. Part IV discusses how to design a systematic testing and analysis process and incorporates it into an overall development process, and may be read either before or after Part III.

Readers new to the field of software test and analysis can obtain an overview by reading Chapters

1	Software Test and Analysis in a nutshell
2	A Framework for Test and Analysis
4	Test and Analysis Activities within a Software Process
10	Functional Testing
11	Combinatorial Testing
14	Model-Based Testing
15	Testing Object-Oriented Software
17	Test Execution
18	Inspection
19	Program Analysis
20	Planning and Monitoring the Process

Notes for Instructors

This book can be used in an introductory course in software test and analysis or as a supplementary text in an undergraduate software engineering course.

An introductory graduate-level or an undergraduate level course in software test and analysis can cover most of the book. In particular, it should include

- All of Part I (Fundamentals of Test and Analysis), which provides a complete overview.
- Most of Part II (Basic Techniques), which provides fundamental background, possibly omitting the latter parts of Chapters 6 (Dependence and Data Flow Models) and 7 (Symbolic Execution and Proof of Properties). These chapters are particularly suited for students who focus on theoretical foundations and those who plan to study analysis and testing more deeply.
- A selection of materials from Parts III (Problems and Methods) and IV (Process).

For a course with more emphasis on techniques than process, we recommend

- Chapter 10 (Functional Testing), to understand how to approach black-box testing.
- The overview section and at least one other section of Chapter 11 (Combinatorial Testing) to grasp some combinatorial techniques.

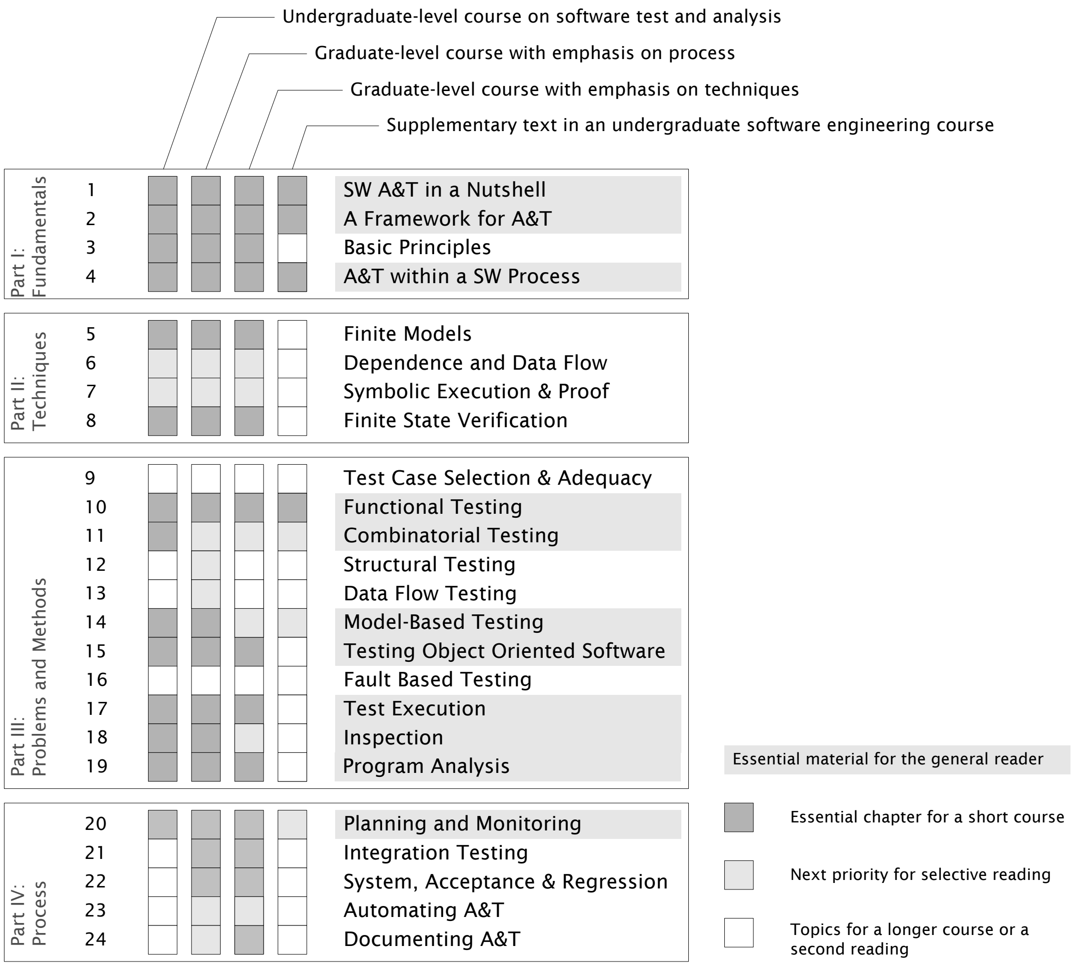
- Chapter 12 (Structural Testing), through Section 12.3, to introduce the basic coverage criteria.
- Chapter 13 (Data Flow Testing), through Section 13.3, to see an important application of data flow analysis to software testing.
- The overview section and at least one other section of Chapter 14 (Model-based Testing) to grasp the interplay between models and testing.
- Chapter 15 (Testing Object-Oriented Software) to appreciate implications of the object-oriented paradigm on analysis and testing.
- Chapter 17 (Test Execution), to manage an easily overlooked set of problems and costs.
- Chapter 18 (Inspection) to grasp the essential features of inspection and appreciate the complementarity of analysis and test.
- Chapter 19 (Program Analysis) to understand the role of automated program analyses and their relation to testing and inspection techniques.
- Chapters 20 (Planning and Monitoring the Process), 21 (Integration and Component-based Software Testing), and 22 (System, Acceptance, and Regression Testing) to widen the picture of the analysis and testing process.

For a stronger focus on software process and organizational issues, we recommend

- Chapter 10 (Functional Testing), a selection from Chapters 11 and 14 (Combinatorial Testing and Model-Based Testing), and Chapters 15 (Testing Object-Oriented Software), 17 (Test Execution), 18 (Inspection), and 19 (Program Analysis) to provide a basic overview of techniques.
- Part IV, possibly omitting Chapter 23 (Automating Analysis and Test), for a comprehensive view of the quality process.

When used as a supplementary text in an undergraduate software engineering course, Chapters 1 (Software Test and Analysis in a Nutshell), and 2 (A Framework for Test and Analysis) can provide a brief overview of the field. We recommend completing these two essential chapters along with either Chapter 4, or a selection of chapters from Part III, or both, depending on the course schedule. Chapter 4 (Test and Analysis Activities within a Software Process) can be used to understand the essential aspects of a quality process. The following chapters from Part III will help students grasp essential techniques:

- Chapter 10 (Functional Testing) and a selection of techniques from Chapters 11 (Combinatorial Testing) and 14 (Model-Based Testing), to grasp basic black-box testing techniques.
- Chapter 12 (Structural Testing), through Section 12.3, to introduce basic coverage criteria.



		Undergraduate-level course on software test and analysis	Graduate-level course with emphasis on process	Graduate-level course with emphasis on techniques	Supplementary text in an undergraduate software engineering course	
Part I: Fundamentals	1	▓	▓	▓	▓	▒ SW A&T in a Nutshell
	2	▓	▓	▓	▓	▒ A Framework for A&T
	3	▓	▓	▓	□	Basic Principles
	4	▓	▓	▓	▓	▒ A&T within a SW Process
Part II: Techniques	5	▓	▓	▓	□	Finite Models
	6	░	░	░	□	Dependence and Data Flow
	7	░	░	░	□	Symbolic Execution & Proof
	8	▓	▓	▓	□	Finite State Verification
Part III: Problems and Methods	9	□	□	□	□	Test Case Selection & Adequacy
	10	▓	▓	▓	▓	▒ Functional Testing
	11	▓	░	░	░	▒ Combinatorial Testing
	12	□	░	□	□	Structural Testing
	13	□	░	□	□	Data Flow Testing
	14	▓	▓	░	░	▒ Model-Based Testing
	15	▓	▓	▓	□	▒ Testing Object Oriented Software
	16	□	□	□	□	Fault Based Testing
	17	▓	▓	▓	□	▒ Test Execution
	18	▓	▓	░	□	▒ Inspection
	19	▓	▓	▓	□	▒ Program Analysis
Part IV: Process	20	▓	▓	▓	░	▒ Planning and Monitoring
	21	□	▓	▓	□	Integration Testing
	22	□	▓	▓	□	System, Acceptance & Regression
	23	□	░	░	□	Automating A&T
	24	□	░	▓	□	Documenting A&T

▒	Essential material for the general reader
▓	Essential chapter for a short course
░	Next priority for selective reading
□	Topics for a longer course or a second reading

Figure 1: Selecting core material by need

- Chapter 15 (Testing Object-Oriented Software), through Section 15.3, to appreciate implications of the object oriented paradigm on analysis and testing.
- Chapter 17 (Test Execution), to manage an easily overlooked set of problems and costs.
- Chapter 18 (Inspection), to grasp the essential features of inspection.

In addition, Chapter 20 (Planning and Monitoring the Process) is useful to gain a deeper appreciation of the interplay between software quality activities and other aspects of a software process.

If the computer science graduate curriculum does not include a course devoted to analysis and testing, we recommend that a graduate software engineering course also cover Chapters 5 (Finite Models), 8 (Finite State Verification), and 19 (Program Analysis) to provide essential technical background.

Supplementary material and a discussion forum are available on the book Web site, http://www.wiley.com/college/pezze