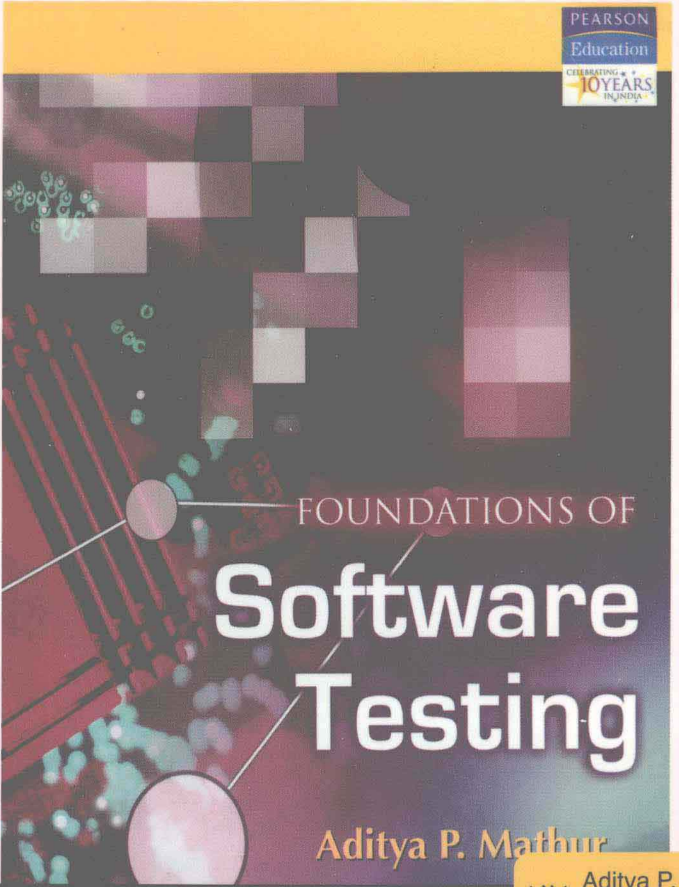
HZ BOOKS
华章教育

经 典 原 版 书 库

PEARSON
Education

软件测试基础教程

（英文版）



PEARSON
Education
CELEBRATING 10 YEARS IN INDIA

FOUNDATIONS OF
Software
Testing

Aditya P. Mathur

(美) Aditya P. Mathur 著
普度大学

机械工业出版社
China Machine Press

经典原版书库

软件测试基础教程

（英文版）

Foundations of Software Testing

（美） Aditya P. Mathur 著
普度大学

机械工业出版社
China Machine Press

本书版权登记号：图字：01-2008-3271

图书在版编目（CIP）数据

软件测试基础教程（英文版）/（美）马瑟（Mathur, A. P.）著. —北京：机械工业出版社，2008.8

(经典原版书库)

书名原文：Foundations of Software Testing

ISBN 978-7-111-24732-6

Ⅰ.软…　Ⅱ.马…　Ⅲ.软件－测试－教材－英文　Ⅳ.TP311.5

中国版本图书馆CIP数据核字（2008）第109992号

机械工业出版社（北京市西城区百万庄大街22号　邮政编码　100037）
责任编辑：迟振春
北京京北制版厂印刷・新华书店北京发行所发行
2008年8月第1版第1次印刷
145mm×210mm・22.125印张
标准书号：ISBN 978-7-111-24732-6
定价：49.00元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换
本社购书热线：(010) 68326294

出版者的话

文艺复兴以降，源远流长的科学精神和逐步形成的学术规范，使西方国家在自然科学的各个领域取得了垄断性的优势；也正是这样的传统，使美国在信息技术发展的六十多年间名家辈出、独领风骚。在商业化的进程中，美国的产业界与教育界越来越紧密地结合，计算机学科中的许多泰山北斗同时身处科研和教学的最前线，由此而产生的经典科学著作，不仅擘划了研究的范畴，还揭示了学术的源变，既遵循学术规范，又自有学者个性，其价值并不会因年月的流逝而减退。

近年，在全球信息化大潮的推动下，我国的计算机产业发展迅猛，对专业人才的需求日益迫切。这对计算机教育界和出版界都既是机遇，也是挑战；而专业教材的建设在教育战略上显得举足轻重。在我国信息技术发展时间较短的现状下，美国等发达国家在其计算机科学发展的几十年间积淀和发展的经典教材仍有许多值得借鉴之处。因此，引进一批国外优秀计算机教材将对我国计算机教育事业的发展起到积极的推动作用，也是与世界接轨、建设真正的世界一流大学的必由之路。

机械工业出版社华章分社较早意识到“出版要为教育服务”。自1998年开始，华章分社就将工作重点放在了遴选、移译国外优秀教材上。经过多年的不懈努力，我们与Pearson，McGraw-Hill，Elsevier，MIT，John Wiley & Sons，Cengage等世界著名出版公司建立了良好的合作关系，从他们现有的数百种教材中甄选出Andrew S. Tanenbaum，Bjarne Stroustrup，Brain W. Kernighan，Dennis Ritchie，Jim Gray，Afred V. Aho，John E. Hopcroft，Jeffrey D. Ullman，Abraham Silberschatz，William Stallings，Donald E. Knuth，John L. Hennessy，Larry L. Peterson等大师名家的一批经典作品，以“计算机科学丛书”为总称出版，供读者学习、研究及珍藏。大理石纹理的封面，也正体现了这套丛书的品位和格调。

“计算机科学丛书”的出版工作得到了国内外学者的鼎力襄助，国内的专家不仅提供了中肯的选题指导，还不辞劳苦地担任了翻译和审校的工作；而原书的作者也相当关注其作品在中国的传播，有的还专程为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近两百个品种，这些书籍在读者中树立了良好的口碑，并被许多高校采用为正式教材和参考书籍。 其影印版“经典原版书库”作为姊妹篇也被越来越多实施双语教学的学校所采用。

权威的作者、经典的教材、一流的译者、严格的审校、精细的编辑，这些因素使我们的图书有了质量的保证。随着计算机科学与技术专业学科建设的不断完善和教材改革的逐渐深化，教育界对国外计算机教材的需求和应用都将步入一个新的阶段，我们的目标是尽善尽美，而反馈的意见正是我们达到这一终极目标的重要帮助。华章分社欢迎老师和读者对我们的工作提出建议或给予指正，我们的联系方法如下：

华章网站：www.hzbook.com
电子邮件：hzedu@hzbook.com
联系电话：(010) 68995264
联系地址：北京市西城区百万庄南街1号
邮政编码：100037

HZ BOOKS
华章教育
华章科技图书出版中心

PREFACE

Welcome to Foundations of Software Testing! This book intends to offer exactly what its title implies. It is important that students planning a career in information technology take a course in software testing. It is also important that such a course offer students an opportunity to acquire material that will remain useful throughout their careers in a variety of software applications, products, and changing environments. This book is an introduction to exactly such material and hence an appropriate text for a course in software testing. It distills knowledge developed by hundreds of testing researchers and practitioners from all over the world and brings it to its readers in an easy-to-understand form.

Test generation, selection, prioritization, and assessment lie at the foundation of all technical activities involved in software testing. Appropriate deployment of the elements of this strong foundation enables the testing of different types of software applications as well as testing for various properties. Applications include Object Oriented systems, Web services, graphical user interfaces, embedded systems, and others, and properties relate to security, timing, performance, reliability, and others.

The importance of software testing increases as software pervades more and more into our daily lives. Unfortunately, few universities offer full-fledged courses in software testing and those that do often struggle to identify a suitable text. I hope that this book will allow academic institutions to create courses in software testing, and those that already offer such courses will not need to hunt for a textbook or rely solely on research publications.

Conversations with testers and managers in commercial software development environments have led me to believe that though software testing is considered an important activity, software testers often complain of not receiving treatment at par with system designers and developers. I believe that raising the level of sophistication in the material covered in courses in software testing will lead to superior testing practices, high-quality software, and thus translate into positive impact on the career of software testers. I hope that exposure to even one-half of the material in this book will establish a student's respect for software testing as a discipline in its own right and at the same level of maturity as subjects such as compilers, databases, algorithms, and networks.

Target audience: It is natural to ask: What is the target level of this book? My experience, and that of some instructors who have used earlier drafts, indicates that this book is best suited for use at senior undergraduate and early graduate levels. While the presentation in this book is aimed at students in a college or university classroom, I believe that both practitioners and researchers will find it useful. Practitioners, with patience, may find this book as a rich source of techniques they could learn and adapt in their development and test environment. Researchers are likely to find it to be a rich source of reference material.

Nature of material covered: Software testing covers a wide spectrum of activities. At a higher level, such activities appear to be similar whereas at a lower level they might differ significantly. For example, most software development environments engage in test execution. However, test execution for an operating system is carried out quite differently than that for a pacemaker; while one is an open system, the other is embedded and hence the need for different ways to execute tests.

The simultaneous existence of similarities and differences in each software testing activity leads to a dilemma for an author as well as an instructor. Should a book and a course focus on specific software development environments, and how they carry out various testing activities? Or should they focus on specific testing activities without any detailed recourse to specific environments? Either strategy is subject to criticism and leaves the students in a vacuum regarding the applications of testing activities or about their formal foundations.

I have resolved this dilemma through careful selection and organization of the material. Parts I, II, and III of this book focus primarily on the foundations of various testing activities. Part I illustrate through examples the differences in software test processes as applied in various software development organizations. Techniques for generating tests from models of expected program behavior are covered in Part II, while the measurement of the adequacy of the tests so generated, and their enhancement, is considered in Part III.

Organization: This book is organized into three parts. Part I covers terminology and preliminary concepts related to software testing. Chapter 1, the only chapter in this part, introduces a variety of terms and basic concepts that pervade the field of software testing. Some adopters of earlier drafts of this book have covered the introductory material in this chapter during the first two or three weeks of an undergraduate course.

Part II covers various test-generation techniques. Chapter 2 introduces the most basic of all test-generation techniques widely applicable in almost any software application one can imagine. These include equivalence partitioning, boundary-value analysis, cause–effect graphing, and predicate testing. Chapter 3 introduces powerful and fundamental techniques for automatically generating tests from finite state models. Three techniques have been selected for presentation in this chapter: W-, Wp-, and Unique Input-Output methods. Finite state models are used in a variety of applications such as in OO testing, security testing, and GUI testing. Generation of combinatorial designs and tests is the topic of Chapter 4. Regression testing forms an integral part of all software development environments where software evolves into newer versions and thus undergoes extensive maintenance. Chapter 5 introduces some fundamental techniques for test selection, prioritization, and minimization of use during regression testing.

Part III is an extensive coverage of an important and widely applicable topic in software testing: test enhancement through measurement of test adequacy. Chapter 6 introduces a variety of control-flow- and data-flow-based code coverage criteria and explains how these could be used in practice. The most powerful of test adequacy criteria based on program mutation are introduced in Chapter 7. While some form of test adequacy assessment is used in almost every software development organization, material covered in these chapters promises to take adequacy assessment and test enhancement to a new level, thereby making a significant positive impact on software reliability.

Practitioners often complain, and are mostly right, that many white-box adequacy criteria are impractical to use during integration and system testing. I have included a discussion on how some of the most powerful adequacy assessment criteria can be, and should be, used even beyond unit testing. Certainly, my suggestions to do so assume the availability of commercial-strength tools for adequacy assessment.

Each chapter ends with a detailed bibliography. I have tried to be as comprehensive as possible in citing works related to the contents of each chapter. I hope that instructors and students will find, the Bibliographic Notes sections rich and helpful in enhancing their knowledge beyond this book. Citations are also a testimony to the rich literature in the field of software testing.

What does this book not cover?: Software testing consists of a large number of related and intertwined activities. Some of these are technical, some administrative, and some merely routine. Technical activities include test case and oracle design at the unit, subsystem, integration, system, and regression levels. Administrative activities include manpower planning, budgeting, and reporting. Planning activities include test planning, quality assessment and control, and manpower allocation. While some planning activities are best classified as administrative, for example manpower allocation, others such as test planning are intertwined with technical activities like test case design.

Several test-related activities are product specific. For example, testing of a device driver often includes tasks such as writing a device simulator. Simulators include heart simulator in testing cardiac pacemakers, a USB port simulator useful in testing I/O drivers, and an airborne drone simulator used in testing control software for airborne drones. While such activities are extremely important for effective testing and test automation, they often require a significant development effort. For example, writing a device simulator and testing it is both a development and a testing activity. Test-generation and assessment techniques described in this book are applicable to each of the product-specific test activity. However, product-specific test activities are illustrated in this book only through examples and not described in any detail. My experience has been that it is best for students to learn about such activities through industry-sponsored term projects.

Suggestions to instructors: There is a wide variation in the coverage of topics in courses in software testing I have tried to cover most, if not all, of the important topics in this area. Tables 1 and 2 provide suggested outline of undergraduate and graduate courses, respectively, that could be based entirely on this book.

Sample undergraduate course in software testing: We assume a semester-long undergraduate course worth 3-credits that meets twice a week, each meeting lasts 50 min and devotes a total of 17 weeks to lectures, examinations, and project presentations. The course has a 2-h per week informal laboratory and requires students to work in small teams of three or four to complete a term project. The term project results in a final report and possibly a prototype testing tool. Once every 2 weeks, students are given one laboratory exercise that takes about 4–6 h to complete.

Table 3 contains a suggested evaluation plan. Carefully designed laboratory exercises form an essential component of this course. Each exercise offers the students an opportunity to use a testing tool to accomplish a task. For example, the objective of a laboratory exercise could be to familiarize the students with JUnit as test runner or JMeter as a tool for the performance

Table 1 A sample undergraduate course in software testing

Week	Topic	Chapter
1	Course objectives and goals, project assignment, testing terminology, and concepts	1
2	Test process and management	1
3	Errors, faults, and failures	1
4	Boundary-value analysis, equivalence partitioning, decision tables	2
5, 6	Test generation from predicates	2
7	Interim project presentations	
	Review, midterm examination	
8	Test adequacy: control flow	6
9	Test adequacy: data flow	6
10, 11	Test adequacy: program mutation	7
12, 13, 14	Special topics, e.g. OO testing and, security testing	Separate volume
15, 16	Review, final project presentations	
17	Final examination	

Table 2 A sample graduate course in software testing

Week	Topic	Chapter
1	Course objectives and goals, testing terminology and concepts	1
2	Test process and management	Separate volume
	Errors, faults, and failures	Separate volume
3	Boundary-value analysis, equivalence partitioning, decision tables	2
4	Test generation from predicates	2
5, 6	Test generation from finite-state models	3
7, 8	Combinatorial designs	4
	Review, midterm examination	
9	Test adequacy: control flow	6
10	Test adequacy: data flow	6
11, 12	Test adequacy: program mutation	7
13, 14	Special topics, e.g. real-time testing and security testing	Separate volume
15, 16	Review, research presentations	
17	Final examination	

Table 3 Suggested evaluation components of the undergraduate and graduate courses in software testing

Level	**Component**	**Weight**	**Duration**
Undergraduate	Midterm examination	15 points	90 min
	Final examination	25 points	120 min
	Quizzes	10 points	Short duration
	Laboratory assignments	10 points	10 assignments
	Term project	40 points	Semester
Graduate	Midterm examination	20 points	90 min
	Final examination	30 points	120 min
	Laboratory assignments	10 points	5 assignments
	Research/Term project	40 points	Semester

Table 4 A sample set of tools to select from for use in undergraduate and graduate courses in software testing

Purpose	**Tool**	**Source**
Combinatorial designs	AETG	
Code coverage measurement	TestManager™	JUnit CodeTest Suds
Defect tracking Bugzilla FogBugz	GUI testing WebCoder	JfcUnit Mutation testing muJava
Proteum		
Performance testing	Performance Tester	JMeter Regression testing Eggplant Suds Test management ClearQuest™ TestManager
Telcordia Technologies IBM Rational Freeware Freescale Semiconductor	Telcordia Technologies Freeware Fog Creek Software Crimson Solutions Freeware	Professor Jeff Offut offutt@ise.gmu.edu
Professor Jose Maldonado jcmaldon@icmc.usp.br	IBM Rational™ Apache, for Java Redstone Software Telcordia Technologies	IBM Rational™ BM Rational™

measurement of Web services. Instructors should be able to design laboratory exercises based on topics covered during the previous weeks. A large number of commercial and open-source-testing tools are available for use in a software-testing laboratory.

Sample graduate course in software testing: We assume a semester-long course worth 3-credits. The students entering this course have not had any prior course in software testing, such as the undergraduate course described above. In addition to the examinations, students will be required to read and present recent research material. Students are exposed to testing tools via unscheduled laboratory exercises.

Testing tools: There is a large set of testing tools available in the commercial, freeware, and open-source domains. A small sample of such tools is listed in Table 4.

Evolutionary book: I expect this book to evolve over time. Advances in topics covered in this book, and any new topics that develop, will be included in subsequent editions. Any errors found by me and/or reported by the readers will be corrected. Readers are encouraged to visit the following site for latest information about the book.

www.pearsoned.co.in/adityapmathur

While this book covers significant material in software testing, several advanced topics could not be included to limit its size. I am planning a separate volume of the book to take care of the advanced topics on the subject and can be used by students who would like to know much more about software testing as well as professionals in the industry.

Cash awards: In the past, I have given cash rewards to students who carefully read the material and reported any kind of error. I plan to retain the cash-reward approach as a means for continuous quality improvement.

Aditya P. Mathur

ACKNOWLEDGMENTS

A number of people have been of significant assistance in the writing of this book. I apologize to those whose names do not appear here but should have; the omission is purely accidental.

I thank Rich DeMillo for introducing me to the subject of software testing and supporting my early research. Rich pointed me to the literature that helped with the acquisition of knowledge and of appreciation for software testing. I am grateful to Bob Horgan who influenced and supported my thinking of the relationship between testing and reliability and the importance of code coverage. Bob freely shared the early and later versions of the χSuds toolset. I continue to believe that χSuds is so far the best toolset available for test-adequacy assessment and enhancement. My sincere thanks are also to Ronnie Martin for spending endless hours meticulously reviewing many of my technical reports.

I thank Donald Knuth warmly for sharing details of the errors of TEX and to his staff for sharing with me the earlier versions of TEX. Thanks to Hiralal Agrawal for patiently answering questions related to dynamic slicing. Thanks to Farokh Bastani, Fevzi Belli, Jim Berger, Kai-Yuan Cai, Ram Chillarege, Sid Dalal, Raymond DeCarlo, Marcio Delamaro, Phyllis Frankl, Arif Ghafoor, Amrit Goel, Dick Hamlet, Mats Heimdahl, Michael A. Hennell, Bill Howden, Ashish Jain, Pankaj Jalote, Rick Karcick, Bogdan Korel, Richard Lipton, Yashwant Malaiya, Josè Maldonado, Simanta Mitra, John Musa, Jeff Offutt, Tom Ostrand, Amit Paradkar, Alberto Pasquini, Ray Paul, C. V. Ramamoorthy, Vernon Rego, Nozer Singpurwalla, Mary-Lou Soffa, Rajesh Subramanian, Kishor Trivedi, Jefferey Voas, Mladen Vouk, Elaine Weyuker, Lee White, and Martin Woodward for discussions and offering constructive criticisms that led to the thrashing of many of my incorrect, often silly, notions about software testing and reliability (and often about life too).

I thank Jim Mapel, Marc Loos, and several other engineers at Boston Scientific Inc. (formerly Guidant Corporation) for introducing me to the sophisticated test processes for ensuring highly reliable cardiac devices. Thanks to Klaus Diaconu, Mario Garzia, Abdelsalam Heddaya, Jawad Khaki, Nar Ganapathy, Adam Shapiro, Peter Shier, Robin Smith, Amitabh Srivastava, and the many software engineers at Microsoft in the Windows Reliability and Device Driver teams for helping me understand the complex and sophisticated test processes and tools used at Microsoft to assure the delivery of a highly reliable operating system to millions of people around the globe.

I acknowledge the efforts of anonymous reviewers for reviewing and providing useful comments. I thank Muhammad Naeem Ayyaz, Abdeslam En-Nouaary, Joao Cangussu, and Eric Wong for editing earlier drafts of this book, and some of the associated presentations, used in undergraduate and graduate courses. Feedback from instructors and students in their classes proved invaluable.

Many thanks to Emine Gokce Aydal, Christine Ayers, Jordan Fleming, Nwokedi Idika, K. Jayaram, Yuanlu Jiang, Ashish Kundu, Yu Lei, Jung-Chi Lin, Shuo Lu, Ammar Masood, Kevin McCarthy, Roman Joel Pacheco, Tu Peng, Van Phan, James Roberts, Chetak Sirsat, Kevin Smith, Travis Steel, Yunlin Xu, Il-Chul Yoon, Hiroshi Yamauchi, and Brandon Wuest for reporting and correcting errors after careful reading of various chapters of earlier drafts.

The cooperation of David Boardman, João Cangussu, Mei-Hwa Chen, Byoungju Choi, Praerit Garg, Sudipto Ghosh, Neelam Gupta, Vivek Khandelwal, Edward Krauser, Saileshwar Krishnamurthy, Tsanchi Li, Pietro Michielan, Scott Miller, Manuela Schiona, Baskar Sridharan, and Brandon Wuest for working with me patiently in software testing research cannot be overlooked.

I thank Professor T. S. K. V. Iyer for constantly asking me whether the book is complete and thus being a major driving force behind the completion of this book. My sincere thanks to Raymond Miller, Nancy Griffith, Bill Griffith, and Pranas Zunde for their years of support and the welcome I received when I arrived in a new country. Thanks to John Rice and Elias Houstis for their support in terms of facilities and equipment essential for experimentation and for being wonderful colleagues. I thank Susanne Hambrusch and Ahmed Sameh for supporting me in the development of software engineering and software testing courses at Purdue. I thank members of the facilities staff of the Computer Science Department for assistance with setting up laboratory equipment and software used by students in graduate and undergraduate course offerings at Purdue University. Thanks to Patricia Minniear for working hard to make timely copies of drafts of this book for the use of students.

My sincere thanks to Professors S. Venkateswaran and L. K. Maheshwari and the faculty and staff of the Computer Science Department at BITS, Pilani, for offering me a friendly work environment during my sabbatical year. I thank my dear friend Mohan Lal and his family for their support over many years, and especially during my visit to Pilani where I wrote some portions of this book. I thank all the employees of the BITS Guest House (VFAST) whose generosity and love must have had some positive effect on the quality of this book.

The contribution of Hanna Lena Kovenock who spent countless hours iterating over the cover cartoon of early drafts of this book depicting the "chair development" team in the animal kingdom. Hanna is a great artist and I was fortunate to get her assistance.

I am indebted to my friends Ranjit Gulrajani and Pundi Narasimhan and their families for emotional support over many years. I would like to thank my parents, Amma, and my other family members whose unflinching love and support was essential for the completion of this project. I must thank my wonderful children Gitanjali and Ravishankar, and son-in-law Abhishek Gangwal, for asking "When will the book be in print?" I thank my collies Raja and Shaan, from whom I took away precious playtime. And last but not the least, my heartiest thanks to my wife Jyoti Iyer Mathur for her constant love and support.

Aditya P. Mathur

Contents