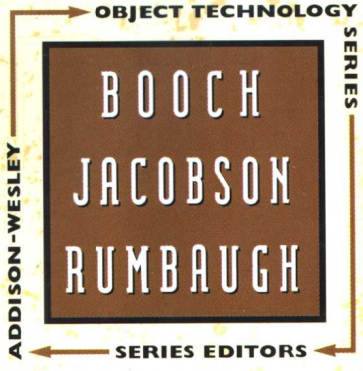
UML与面向对象设计影印丛书

对象软件测试全程策略

A PRACTICAL GUIDE TO TESTING OBJECT-ORIENTED SOFTWARE

〔美〕 JOHN D. McGREGOR
DAVID A. SYKES 编著



科学出版社
www.sciencep.com

UML与面向对象设计影印丛书

对象软件测试全程策略

〔美〕 John D. McGregor
David A. Sykes 编著

科学出版社
北京

图字：01-2003-4434

内 容 简 介

软件测试是确保软件产品质量的必需过程和手段。在开发面向对象软件和基于组件的软件时，同样必须设计并实施有效的测试。本书明确指出了面向对象软件和面向过程软件测试任务的不同之处，论述了面向对象软件测试所面临的机遇和挑战。书中介绍了如何将测试工作整合进软件开发的各个阶段，以及各阶段的具体测试内容是什么。本书注重结合实例来讲解对象软件测试的过程和技巧，因而具有很强的实践性和指导意义。

本书内容完整，结构合理，适合软件开发测试领域的技术人员和项目管理人员阅读，也可作为软件学院的教材。

ISBN 0-201-32564-0

图书在版编目(CIP)数据

对象软件测试全程策略（美）麦格雷戈（McGregor, G. D.）（美）西科斯（Sykes, D. A.）编著.
—影印本. —北京：科学出版社，2003
（UML 与面向对象设计影印丛书）
ISBN 7-03-011914-2
Ⅰ.对… Ⅱ.①麦…②西… Ⅲ. 软件—测试—英文 Ⅳ.TP311.5

中国版本图书馆 CIP 数据核字（2003）第 062353 号

策划编辑：李佩乾/责任编辑：李佩乾
责任印制：吕春珉/封面制作：东方人华平面设计室

科学出版社 出版
北京东黄城根北街16号
邮政编码:100717
http://www.sciencep.com
双青印刷厂 印刷
科学出版社发行 各地新华书店经销
*
2003 年 8 月第 一 版 开本：787×960 1/16
2003 年 8 月第一次印刷 印张：25 1/2
印数：1—2 500 字数：612 000

定价：45.00 元

（如有印装质量问题，我社负责调换〈环伟〉）

影印前言

随着计算机硬件性能的迅速提高和价格的持续下降，其应用范围也在不断扩大。交给计算机解决的问题也越来越难，越来越复杂。这就使得计算机软件变得越来越复杂和庞大。20 世纪 60 年代的软件危机使人们清醒地认识到按照工程化的方法组织软件开发的必要性。于是软件开发方法从 60 年代毫无工程性可言的手工作坊式开发，过渡到 70 年代结构化的分析设计方法、80 年代初的实体关系开发方法，直到面向对象的开发方法。

面向对象的软件开发方法是在结构化开发范型和实体关系开发范型的基础上发展而来的，它运用分类、封装、继承、消息等人类自然的思维机制，允许软件开发者处理更为复杂的问题域和其支持技术，在很大程度上缓解了软件危机。面向对象技术发端于程序设计语言，以后又向软件开发的早期阶段延伸，形成了面向对象的分析和设计。

20 世纪 80 年代末 90 年代初，先后出现了几十种面向对象的分析设计方法。其中，Booch, Coad/Yourdon、OMT 和 Jacobson 等方法得到了面向对象软件开发界的广泛认可。各种方法对许多面向对象的概念的理解不尽相同，即便概念相同，各自技术上的表示法也不同。通过 90 年代不同方法流派之间的争论，人们逐渐认识到不同的方法既有其容易解决的问题，又有其不容易解决的问题，彼此之间需要进行融合和借鉴；并且各种方法的表示也有很大的差异，不利于进一步的交流与协作。在这种情况下，统一建模语言(UML)于 90 年代中期应运而生。

UML 的产生离不开三位面向对象的方法论专家 G. Booch、J. Rumbaugh 和 I. Jacobson 的通力合作。他们从多种方法中吸收了大量有用的建模概念，使 UML 的概念和表示法在规模上超过了以往任何一种方法，并且提供了允许用户对语言做进一步扩展的机制。UML 使不同厂商开发的系统模型能够基于共同的概念，使用相同的表示法，呈现彼此一致的模型风格。1997 年 11 月 UML 被 OMG 组织正式采纳为标准的建模语言，并在随后的几年中迅速地发展为事实上的建模语言国际标准。

UML 在语法和语义的定义方面也做了大量的工作。以往各种关于面向对象方法的著作通常是以比较简单的方式定义其建模概念，而以主要篇幅给出过程指导，论述如何运用这些概念来进行开发。UML 则以一种建模语言的姿态出现，使用语言学中的一些技术来定义。尽管真正从语言学的角度看它还有许多缺陷，但它在这方面所做的努力却是以往的各种建模方法无法比拟的。

从 UML 的早期版本开始，便受到了计算机产业界的重视，OMG 的采纳和大公司的支持把它推上了实际上的工业标准的地位，使它拥有越来越多的用户。它被广泛地用

于应用领域和多种类型的系统建模，如管理信息系统、通信与控制系统、嵌入式实时系统、分布式系统、系统软件等。近几年还被运用于软件再工程、质量管理、过程管理、配置管理等方面。而且它的应用不仅仅限于计算机软件，还可用于非软件系统，例如硬件设计、业务处理流程、企业或事业单位的结构与行为建模，等等。

在 UML 陆续发布的几个版本中，逐步修正了前一个版本中的缺陷和错误。即将发布的 UML2.0 版本将是对 UML 的又一次重大的改进。将来的 UML 将向着语言家族化、可执行化、精确化等理念迈进，为软件产业的工程化提供更有力的支撑。

本丛书收录了与面向对象技术和 UML 有关的 12 本书，反映了面向对象技术最新的发展趋势以及 UML 的新的研究动态。其中涉及对面向对象建模理论研究与实践的有这样几本书：《面向对象系统架构及设计》主要讨论了面向对象的基本概念、静态设计、永久对象、动态设计、设计模式以及体系结构等近几年来面向对象技术领域中的新的理论知识与方法；《用 UML 进行用况对象建模》主要介绍了面向对象的需求阶段、分析阶段、设计阶段中用况模型的建立方法与技术；《高级用况建模》介绍了在建立用况模型中需要注意的高级的问题与技术；《UML 面向对象设计基础》则侧重于经典的面向对象理论知识的阐述。

涉及 UML 在特定领域的运用的有这样几本：《UML 实时系统开发》讨论了进行实时系统开发时需要对 UML 进行扩展的技术；《用 UML 构建 Web 应用程序》讨论了运用 UML 进行 Web 应用建模所应该注意的技术与方法；《面向对象系统测试：模型、视图与工具》介绍了将 UML 应用于面向对象的测试领域所应掌握的方法与工具；《对象、构件、框架与 UML 应用》讨论了如何运用 UML 对面向对象的新技术——构件-框架技术建模的方法策略。《UML 与 Visual Basic 应用程序开发》主要讨论了从 UML 模型到 Visual Basic 程序的建模与映射方法。

介绍面向对象编程技术的有两本书：《COM 高手心经》和《ATL 技术内幕》，深入探讨了面向对象的编程新技术——COM 和 ATL 技术的使用技巧与技术内幕。

还有一本《Executable UML 技术内幕》，这本书介绍了可执行 UML 的理念与其支持技术，使得模型的验证与模拟以及代码的自动生成成为可能，也代表着将来软件开发的一种新的模式。

总之，这套书所涉及的内容包含了对软件生命周期的全过程建模的方法与技术，同时也对近年来的热点领域建模技术、新型编程技术作了深入的介绍，有些内容已经涉及到了前沿领域。可以说，每一本都很经典。

有鉴于此，特向软件领域中不同程度的读者推荐这套书，供大家阅读、学习和研究。

北京大学计算机系 蒋严冰 博士

Preface

Testing software is a very important and challenging activity. This is a book for people who test software during its development. Our focus is on object-oriented and component-based software, but you can apply many of the techniques discussed in this book regardless of the development paradigm. We assume our reader is familiar with testing procedural software—that is, software written in the procedural paradigm using languages such as C, Ada, Fortran, or COBOL. We also assume our reader is familiar and somewhat experienced in developing software using object-oriented and component-based technologies. Our focus is on describing *what* to test in object-oriented development efforts as well as on describing techniques for *how* to test object-oriented software, and how testing software built with these newer technologies differs from testing procedural software.

What is software testing? To us, testing is the evaluation of the work products created during a software development effort. This is more general than just checking part or all of a software system to see if it meets its specifications. Testing software is a difficult process, in general, and sufficient resources are seldom available for testing. From our standpoint, testing is done throughout a development effort and is not just an activity tacked on at the end of a development phase to see how well the developers did. We see testing as part of the process that puts quality into a software system. As a result, we address the testing of all development products (models) even before any code is written.

We do not necessarily believe that you will apply everything we describe in this book. There are seldom enough resources available to a development effort to do all the levels and kinds of testing we would like. We hope you will find a number of approaches and techniques that will prove useful to and affordable for your project.

In this book we describe a set of testing techniques. All of the techniques we describe have been applied in practice. Many of these techniques have been used in a wide variety of industries and on projects of vastly different sizes. In Chapter 3, we will consider the impact of some of these variables on the types of testing that are routinely performed.

To describe these techniques, we rely in many cases on one or more examples to illustrate their application. We hope from these examples and from our explanations that you can apply the same techniques to your project software in a straightforward manner. The complete code for these examples, test code, and other resources can be obtained from *http://cseng.aw.com/book/0.3828.0201325640.00.html*.

In order to make this book as useful as possible, we will provide two major organizational threads. The physical layout of the book will follow the usual sequence of events as they happen on a project. Model testing will be addressed earlier than component or code testing, for example. We will also include a set of questions that a tester might ask when he or she is faced with specific testing tasks on a project. This testing FAQ will be tied into the main body of the text with citations.

We have included alternative techniques and ways of adapting techniques for varying the amount of testing. Testing life-critical or mission-critical software requires more effort than testing an arcade game. The summary sections of each chapter should make these choices clear.

This book is the result of many years of research, teaching, and consulting both in the university and in companies. We would like to thank the sponsors of our research, including COMSOFT, IBM, and AT&T for their support of our academic research. Thanks to the students who assisted in the research and those who sat through many hours of class and provided valuable feedback on early versions of the text. The consultants working for Korson-McGregor, formerly Software Architects, made many suggestions and worked with early versions of the techniques while still satisfying client needs. The employees of numerous consulting clients helped us perfect the techniques by providing real problems to be solved and valuable feedback. A special thanks to Melissa L. Russ (formerly Major) who helped teach several tutorials and made her usual insightful comments to improve the material.

Most of all, we wish to thank our families for enduring our mental and physical absences and for the necessary time to produce this work: Gayle and Mary Frances McGregor; Susan, Aaron, Perry, and Nolan Sykes.

JDM
DAS

Contents

Chapter 2 The Testing Perspective 15

Chapter 3 Planning for Testing 65

Chapter 5 Class Testing Basics 163

Chapter 6 Testing Interactions 213

Introduction

Chapter 1

Testing software well has always been challenging, but the process is fairly well understood. Some combination of unit testing, integration testing, system testing, regression testing, and acceptance testing will help to deliver usable systems.

We wanted to write this book because most people seem to believe that testing *object-oriented* software is not much different from testing procedural software. While many of the general approaches and techniques for testing are the same or can be adapted from traditional testing approaches and techniques, our experience and our research has demonstrated that some things are different and present new challenges. At the same time, well-designed object-oriented software developed as part of an incremental process provides opportunities for improvements over traditional testing processes.

Object-oriented programming language features of inheritance and polymorphism present new technical challenges to testers. We describe solutions for many of these challenges. In this book, we describe processes and techniques for testing object-oriented software effectively during all phases of a development effort. Our approach to testing software is quite comprehensive and one that we believe software development organizations should undertake. At the same time, we realize that resources available for testing are limited and that there are many effective ways to develop software, so we think it is reasonable to pick and choose among the techniques we present in this book.

The adoption of object-oriented technologies brings changes not only in the programming languages we use but in most aspects of software development.

We use incremental development processes, refocus and use new notations for analysis and design, and utilize new programming language features. The changes promise to make software more maintainable, reusable, flexible, and so on. We have written this book because changes in the way we *develop* software produces changes in the way we *test* software, from both managerial and technical perspectives. The following changes provide opportunities for improving the testing process:

- We have an opportunity to change attitudes toward testing. In many environments, managers and developers view testing as a necessary evil. Testing that needs to be done by the developers themselves interrupts code production. Reviews, code inspections, and writing unit test drivers take time and money. Testing processes imposed on the developers for the most part just get in the way of coding. However, if we can make everyone appreciate that testing contributes to developing the right software from the start, and that it can actually be used to measure progress and keep development on track, then we can build even better software.
- We have an opportunity to change where testing fits into a development process. Almost everyone recognizes that the sooner problems are found, the cheaper they are to fix. Unit testing and integration testing uncover problems, but don't usually start until coding has started. System testing is typically done near the end of a development effort or perhaps at certain planned milestones. System testing is treated as a way to see how well the developers did in meeting requirements. Of course, this is a wrong approach. Decisions about how much testing is adequate, when it should be performed, and who should do it should be made only in the context of a well-considered testing strategy that works with the project's software development process. We will show how testing activities can begin early. We will show how testing and development activities can be intertwined and how each can contribute to a successful outcome of the other.
- We have an opportunity to use new technology to do the testing. Just as object-oriented technologies have benefits for production software, they also can realize benefits in test software. We will show how you can test object-oriented analysis and design models, and how you can use object-oriented programming techniques to develop unit test drivers and reduce the coding necessary to test software components.

Who Should Read This Book?

We have written this book for

- *Programmers* who already work in testing software, but want to know more about testing object-oriented software.