Ali Bahrami

object oriented systems development

using the unified modeling language

OBJECT ORIENTED SYSTEMS DEVELOPMENT

Ali Bahrami

Boeing Applied Research & Technology

Irwin McGraw-Hill

Boston Burr Ridge, IL Dubuque, IA Madison, WI New York San Francisco St. Louis
Bangkok Bogotá Caracas Lisbon London Madrid Mexico City
Milan New Delhi Seoul Singapore Sydney Taipei Toronto

Irwin/McGraw-Hill

A Division of The ***McGraw·Hill*** *Companies*

Vice president/Editor-in-chief: *Michael W. Junior*
Senior sponsoring editor: *Rick Williamson*
Developmental editor: *Christine Wright*
Marketing manager: *Jodi McPherson*
Project managers: *Christine Parker/Alisa Watson*
Production supervisor: *Michael R. McCormick*
Cover designer: *Steven Vena/SrV Unlimited Design*
Supplemental coordinator: *Carol Loreth*
Compositor: *York Graphic Services, Inc.*
Typeface: *10.5/12 Times Roman*
Printer: *R. R. Donnelley & Sons Company*

OBJECT ORIENTED SYSTEMS DEVELOPMENT

Copyright © 1999 by The McGraw-Hill Companies, Inc. All rights reserved. Printed in the United States of America. Except as permitted under the United States Copyright Act of 1976, no part of this publication may be reproduced or distributed in any form or by any means, or stored in a data base or retrieval system, without the prior written permission of the publisher.

This book is printed on acid-free paper.

1 2 3 4 5 6 7 8 9 0 DOC/DOC 9 3 2 1 0 9 8

ISBN 0-256-25348-X

Library of Congress Cataloging-in-Publication Data

Bahrami, Ali.
Object oriented systems development / Ali Bahrami.
p. cm.
Includes index.
ISBN 0-256-25348-X
1. System design. 2. Object-oriented programming (Computer science) I. Title.
QA76.9.S88B33 1999
005.1'2—dc21 98-43126

http://www.mhhe.com

PREFACE

Over the last 20 years, software has become increasingly complex. Today's applications are much more sophisticated and developed to more demanding requirements than in the past. Software development techniques, tools, and technologies are changing rapidly. The software development methods we will use in the next millennium will differ significantly from current methods, and we have no crystal ball to tell us what methods or approaches will be used 20 years (or even 10 years) from now. However, it is apparent that object-oriented development and its core concepts are here to stay. Many schools have recognized this and made the object-oriented systems development course an essential part of the computer information systems or computer science programs. This book is intended for an introductory course in object-oriented systems development at the junior, senior, or first-year graduate level. The main goal of this book is to provide a clear description of the concepts underlying object-oriented systems development.

This book is not centered around any particular programming language or CASE tools. Instead, it discusses fundamental concepts that are applicable to a variety of systems. However, the approach used in this book is based on the best practices that have proven successful in system development and more specifically the work done by Booch, Rumbaugh, and Jacobson. Furthermore, the book uses the Object Management Group's unified modeling language (UML) for modeling, describing, analyzing, and designing an application.

This book has a number of unique features:

- Use of the unified modeling language.
- A comprehensive treatment of the entire system life cycle using object-oriented techniques (with the exception of implementation).
- Inclusion of the Popkin System Architect CASE tool (as a software packaging option).

- Coverage of introductory and essential topics as well as advanced subjects in object-oriented systems development.
- Use of a use-case-driven approach.
- Use of a running case study for applying the lessons learned.
- Appendix providing a documentation template.

STRUCTURE OF THE BOOK

The book contains 14 chapters with a running case study for applying the concepts learned. Each chapter concludes with a summary, a list of the key terms discussed in the chapter, review questions, and problems that require students to apply their knowledge based on the chapter material. The chapters are grouped into five parts.

Part One

The first part provides an overview of object-oriented systems development and discusses why we should study it. In this part, we also look at object basics and the systems development life cycle. Part One consists of Chapter 1, "Overview of Object-Oriented Systems Development"; Chapter 2, "Object Basics"; and Chapter 3, "Object-Oriented Systems Development Life Cycle."

Part Two

The second part introduces various object-oriented methodologies, including the unified approach, which will be used in this text, and an introduction to unified modeling language (UML). This part consists of Chapter 4, "Object-Oriented Methodologies," and Chapter 5, "Unified Modeling Language."

Part Three

The third part introduces object-oriented analysis, which is the process of extracting the needs of a system and what the system must do to satisfy the users' requirements. The goal of object-oriented analysis first is to understand the domain of the problem and the system's responsibilities by understanding how the users use or will use the system. Next, the classes that make up the system must be identified, as well as their behaviors, the relationships among them, and their structure. This part consists of Chapter 6, "Object-Oriented Analysis Process: Identifying Use Cases"; Chapter 7, "Object Analysis: Classification"; and Chapter 8, "Identifying Object Relationships, Attributes, and Methods."

Part Four

The fourth part covers object-oriented design. In this part, we will learn that the classes identified during analysis provide us a framework for the design phase. This part consists of Chapter 9, "The Object-Oriented Design Process and Design Axioms"; Chapter 10, "Designing Classes"; Chapter 11, "Access Layer: Object Storage and Object Interoperability"; and Chapter 12, "View Layer: Designing Interface Objects."

此为试读，需要完整PDF请访问：www.ertongbook.com

Part Five

In this part, different dimensions of software quality and testing are discussed. Testing may be conducted for different reasons. *Quality assurance* testing looks for potential problems in a proposed design. *Usability testing*, on the other hand, tests how well the interface fits user needs and expectations. To ensure *user satisfaction*, we must measure user satisfaction along the way as the design takes form. Part Five consists of Chapter 13, "Software Quality Assurance," and Chapter 14, "System Usability and Measuring User Satisfaction."

Appendices

The book includes two appendices. Appendix A provides a template for documenting a system requirement. The template can be to create an effective system document. Finally, Appendix B provides an overview of Windows and graphical user interface (GUI) basics.

INSTRUCTIONAL SUPPORT MATERIAL

The text is accompanied by an Instructor's CD-ROM. The CD-ROM contains files for an Instructor's Manual consisting of a lecture outline, teaching suggestions, comprehensive Power Point classroom presentation files, the user satisfaction test spreadsheet (see Chapter 14), answers to selected problems and questions, and test bank and computerized testing software with multiple-choice and short-answer questions.

SOFTWARE PACKAGING OPTIONS

Popkin's System Architect CASE tool is available as a packaging option with this text.

ACKNOWLEDGMENTS

No textbook can be published without the involvement of many people, and I would like to acknowledge those who have helped bring this book to fruition. I am grateful, first, to my wife Sue and my older daughter Ava, who have put up with me for the past four years. The publication of this book would not have been possible without the vision and foresight of my editor, Rick Williamson, and the expertise of developmental editor Christine Wright. I am grateful for their support in this project. My special thanks go to Christine Parker and Alisa Watson, the project managers who kept everything on time.

This book has taken four years to complete. It was reviewed by a number of reviewers at different stages of its development and rewritten three times based on the reviewers' comments and suggestions. I thank the reviewers for their constructive comments and encouragement. Their comments have materially enhanced the final copy of this book. These reviewers include

Joseph H. Austin, Jr., Ambassador University
Chris Basso, Signex Corporation
John Carson, George Washington University
Kevin C. Dittman, Purdue University
Jane Fedorowicz, Bentley College
Bill C. Hardgrave, University of Arkansas–Fayetteville
Christopher G. Jones, Utah Valley State College
Jinwoo Kim, Yonsei University, Korea
Michael V. Mannino, University of Colorado–Denver
Stevan Mrdalj, Eastern Michigan University
Richard G. Ramirez, Iowa State University
Elmer G. Swartzmeyer, Georgia State University
Jos Von Hillegersberg, Erasmus University, the Netherlands
Janet Wesson, University of Port Elizabeth, South Africa
David F. Wood, Robert Morris College

This book has been tested for three semesters in my object-oriented software development course at Rhode Island College. I would like to thank all the students who provided me with excellent feedback. Friends and colleagues, who have given me support, ideas, and comments, were invaluable. The friends who have materially contributed to this project include Dr. Crist Costa and Professor Jules Cohen.

Although my father, who encouraged me to write this book, could not see its finish, his spirit was with me throughout the project and kept me going. Last but not least, my youngest daughter was born during the final stages of the book. Her unconditional love energized me to finish this book.

Ali Bahrami
Boeing Applied Research & Technology

CONTENTS

Preface xv

PART ONE

Introduction

1. AN OVERVIEW OF OBJECT-ORIENTED SYSTEMS DEVELOPMENT 3
- **1.1 Introduction** 3
- **1.2 Two Orthogonal Views of the Software** 4
- **1.3 Object-Oriented Systems Development Methodology** 4
- **1.4 Why an Object Orientation?** 5
- **1.5 Overview of the Unified Approach** 6
- **1.6 Organization of This Book** 6
- **1.7 Summary** 11

2. OBJECT BASICS 13
- **2.1 Introduction** 13
- **2.2 An Object-Oriented Philosophy** 14
- **2.3 Objects** 15
- **2.4 Objects Are Grouped in Classes** 16
- **2.5 Attributes: Object State and Properties** 17
- **2.6 Object Behavior and Methods** 18
- **2.7 Objects Respond to Messages** 18
- **2.8 Encapsulation and Information Hiding** 20
- **2.9 Class Hierarchy** 21
  - 2.9.1 Inheritance 23
  - 2.9.2 Multiple Inheritance 25
- **2.10 Polymorphism** 25
- **2.11 Object Relationships and Associations** 26
  - 2.11.1 Consumer-Producer Association 26
- **2.12 Aggregations and Object Containment** 27
- **2.13 Case Study: A Payroll Program** 28
  - 2.13.1 Structured Approach 28
  - 2.13.2 The Object-Oriented Approach 30

2.14 Advanced Topics 32
2.14.1 Object and Identity 32
2.14.2 Static and Dynamic Binding 34
2.14.3 Object Persistence 34
2.14.4 Meta-Classes 34
2.15 Summary 35
3. OBJECT-ORIENTED SYSTEMS DEVELOPMENT LIFE CYCLE 39
3.1 Introduction 39
3.2 The Software Development Process 40
3.3 Building High-Quality Software 42
3.4 Object-Oriented Systems Development: A Use-Case Driven Approach 44
3.4.1 Object-Oriented Analysis—Use-Case Driven 45
3.4.2 Object-Oriented Design 47
3.4.3 Prototyping 47
3.4.4 Implementation: Component-Based Development 49
3.4.5 Incremental Testing 53
3.5 Reusability 53
3.6 Summary 54

PART TWO

Methodology, Modeling, and Unified Modeling Language

4. OBJECT-ORIENTED METHODOLOGIES 61
4.1 Introduction: Toward Unification—Too Many Methodologies 61
4.2 Survey of Some of the Object-Oriented Methodologies 62
4.3 Rumbaugh et al.'s Object Modeling Technique 63
4.3.1 The Object Model 63
4.3.2 The OMT Dynamic Model 63
4.3.3 The OMT Functional Model 64
4.4 The Booch Methodology 65
4.4.1 The Macro Development Process 66
4.4.2 The Micro Development Process 67
4.5 The Jacobson et al. Methodologies 68
4.5.1 Use Cases 68
4.5.2 Object-Oriented Software Engineering: Objectory 70
4.5.3 Object-Oriented Business Engineering 71
4.6 Patterns 71
4.6.1 Generative and Nongenerative Patterns 73
4.6.2 Patterns Template 74
4.6.3 Antipatterns 76
4.6.4 Capturing Patterns 76
4.7 Frameworks 77
4.8 The Unified Approach 78
4.8.1 Object-Oriented Analysis 79
4.8.2 Object-Oriented Design 80
4.8.3 Iterative Development and Continuous Testing 80
4.8.4 Modeling Based on the Unified Modeling Language 80
4.8.5 The UA Proposed Repository 81
4.8.6 The Layered Approach to Software Development 82
4.8.6.1 The Business Layer 83
4.8.6.2 The User Interface (View) Layer 84
4.8.6.3 The Access Layer 84
4.9 Summary 84

5. UNIFIED MODELING LANGUAGE 89

5.1 Introduction 89

5.2 Static and Dynamic Models 90

5.2.1 Static Model 90

5.2.2 Dynamic Model 91

5.3 Why Modeling? 91

5.4 Introduction to the Unified Modeling Language 92

5.5 UML Diagrams 93

5.6 UML Class Diagram 94

5.6.1 Class Notation: Static Structure 94

5.6.2 Object Diagram 94

5.6.3 Class Interface Notation 95

5.6.4 Binary Association Notation 95

5.6.5 Association Role 95

5.6.6 Qualifier 96

5.6.7 Multiplicity 97

5.6.8 OR Association 97

5.6.9 Association Class 97

5.6.10 N-Ary Association 98

5.6.11 Aggregation and Composition 99

5.6.12 Generalization 99

5.7 Use-Case Diagram 101

5.8 UML Dynamic Modeling 103

5.8.1 UML Interaction Diagrams 104

5.8.1.1 UML Sequence Diagram 104

5.8.1.2 UML Collaboration Diagram 105

5.8.2 UML Statechart Diagram 106

5.8.3 UML Activity Diagram 109

5.8.4 Implementation Diagrams 111

5.8.4.1 Component Diagram 112

5.8.4.2 Deployment Diagram 112

5.9 Model Management: Packages and Model Organization 114

5.10 UML Extensibility 115

5.10.1 Model Constraints and Comments 116

5.10.2 Note 117

5.10.3 Stereotype 117

5.11 UML Meta-Model 117

5.12 Summary 118

PART THREE

Object-Oriented Analysis: Use-Case Driven

6. OBJECT-ORIENTED ANALYSIS PROCESS: IDENTIFYING USE CASES 125

6.1 Introduction 125

6.2 Why Analysis Is a Difficult Activity 126

6.3 Business Object Analysis: Understanding the Business Layer 127

6.4 Use-Case Driven Object-Oriented Analysis: The Unified Approach 128

6.5 Business Process Modeling 129

6.6 Use-Case Model 129

6.6.1 Use Cases under the Microscope 131

6.6.2 Uses and Extends Associations 133

6.6.3 Identifying the Actors 134

6.6.4 Guidelines for Finding Use Cases 136

6.6.5 How Detailed Must a Use Case Be? When to Stop Decomposing and When to Continue 136

6.6.6 Dividing Use Cases into Packages 137

6.6.7 Naming a Use Case 137

6.7 Developing Effective Documentation 138

6.7.1 Organizing Conventions for Documentation 139

6.7.2 Guidelines for Developing Effective Documentation 139

6.8 Case Study: Analyzing the ViaNet Bank ATM—The Use-Case Driven Process 140

6.8.1 Background 140

6.8.2 Identifying Actors and Use Cases for the ViaNet Bank ATM System 141

6.8.3 The ViaNet Bank ATM Systems' Packages 146

6.9 Summary 146

7. OBJECT ANALYSIS: CLASSIFICATION 151

7.1 Introduction 151

7.2 Classifications Theory 152

7.3 Approaches for Identifying Classes 154

7.4 Noun Phrase Approach 154

7.4.1 Identifying Tentative Classes 154

7.4.2 Selecting Classes from the Relevant and Fuzzy Categories 155

7.4.3 The ViaNet Bank ATM System: Identifying Classes by Using Noun Phrase Approach 156

7.4.4 Initial List of Noun Phrases: Candidate Classes 156

7.4.5 Reviewing the Redundant Classes and Building a Common Vocabulary 158

7.4.6 Reviewing the Classes Containing Adjectives 159

7.4.7 Reviewing the Possible Attributes 160

7.4.8 Reviewing the Class Purpose 161

7.5 Common Class Patterns Approach 162

7.5.1 The ViaNet Bank ATM System: Identifying Classes by Using Common Class Patterns 163

7.6 Use-Case Driven Approach: Identifying Classes and Their Behaviors through Sequence/Collaboration Modeling 164

7.6.1 Implementation of Scenarios 164

7.6.2 The ViaNet Bank ATM System: Decomposing a Use-Case Scenario with a Sequence Diagram: Object Behavior Analysis 165

7.7 Classes, Responsibilities, and Collaborators 169

7.7.1 Classes, Responsibilities, and Collaborators Process 170

7.7.2 The ViaNet Bank ATM System: Identifying Classes by Using Classes, Responsibilities, and Collaborators 171

7.8 Naming Classes 172

7.9 Summary 174

8. IDENTIFYING OBJECT RELATIONSHIPS, ATTRIBUTES, AND METHODS 177

8.1 Introduction 177

8.2 Associations 178

8.2.1 Identifying Associations 179

8.2.2 Guidelines for Identifying Associations 179

8.2.3 Common Association Patterns 179

此为试读，需要完整PDF请访问：www.ertongbook.com

8.2.4 Eliminate Unnecessary Associations 180

8.3 Super-Sub Class Relationships 181

8.3.1 Guidelines for Identifying Super-Sub Relationship, a Generalization 181

8.4 A-Part-of Relationships—Aggregation 182

8.4.1 A-Part-of Relationship Patterns 183

8.5 Case Study: Relationship Analysis for the ViaNet Bank ATM System 184

8.5.1 Identifying Classes' Relationships 184

8.5.2 Developing a UML Class Diagram Based on the Use-Case Analysis 184

8.5.3 Defining Association Relationships 185

8.5.4 Defining Super-Sub Relationships 186

8.5.5 Identifying the Aggregation/a-Part-of Relationship 187

8.6 Class Responsibility: identifying Attributes and Methods 188

8.7 Class Responsibility: Defining Attributes by Analyzing Use Cases and Other UML Diagrams 189

8.7.1 Guidelines for Defining Attributes 189

8.8 Defining Attributes for ViaNet Bank Objects 190

8.8.1 Defining Attributes for the BankClient Class 190

8.8.2 Defining Attributes for the Account Class 190

8.8.3 Defining Attributes for the Transaction Class 191

8.8.4 Defining Attributes for the ATMMachine Class 191

8.9 Object Responsibility: Methods and Messages 191

8.9.1 Defining Methods by Analyzing UML Diagrams and Use Cases 192

8.10 Defining Methods for ViaNet Bank Objects 192

8.10.1 Defining Account Class Operations 192

8.10.2 Defining BankClient Class Operations 193

8.10.3 Defining CheckingAccount Class Operations 193

8.11 Summary 194

PART FOUR

Object-Oriented Design

9. THE OBJECT-ORIENTED DESIGN PROCESS AND DESIGN AXIOMS 199

9.1 Introduction 199

9.2 The Object-Oriented Design Process 200

9.3 Object-Oriented Design Axioms 202

9.4 Corollaries 203

9.4.1 Corollary 1. Uncoupled Design with Less Information Content 204

9.4.1.1 Coupling 204

9.4.1.2 Cohesion 206

9.4.2 Corollary 2. Single Purpose 206

9.4.3 Corollary 3. Large Number of Simpler Classes, Reusability 206

9.4.4 Corollary 4. Strong Mapping 207

9.4.5 Corollary 5. Standardization 208

9.4.6 Corollary 6. Designing with Inheritance 208

9.4.6.1 Achieving Multiple Inheritance in a Single Inheritance System 211

9.4.6.2 Avoiding Inheriting Inappropriate Behaviors 211

9.5 Design Patterns 212

9.6 Summary 214

10. DESIGNING CLASSES 217

10.1 Introduction 217

10.2 The Object-Oriented Design Philosophy 217

10.3 UML Object Constraint Language 218

10.4 Designing Classes: The Process 219

10.5 Class Visibility: Designing Well-Defined Public, Private, and Protected Protocols 219

10.5.1 Private and Protected Protocol Layers: Internal 221

10.5.2 Public Protocol Layer: External 221

10.6 Designing Classes: Refining Attributes 221

10.6.1 Attribute Types 222

10.6.2 UML Attribute Presentation 222

10.7 Refining Attributes for the ViaNet Bank Objects 223

10.7.1 Refining Attributes for the BankClient Class 223

10.7.2 Refining Attributes for the Account Class 223

10.7.3 Refining Attributes for the Transaction Class 224

Problem 10.1 224

10.7.4 Refining Attributes for the ATMMachine Class 224

10.7.5 Refining Attributes for the CheckingAccount Class 224

10.7.6 Refining Attributes for the SavingsAccount Class 224

10.8 Designing Methods and Protocols 225

10.8.1 Design Issues: Avoiding Design Pitfalls 226

10.8.2 UML Operation Presentation 227

10.9 Designing Methods for the ViaNet Bank Objects 227

10.9.1 BankClient Class VerifyPassword Method 228

10.9.2 Account Class Deposit Method 228

10.9.3 Account Class Withdraw Method 229

10.9.4 Account Class CreateTransaction Method 229

10.9.5 Checking Account Class Withdraw Method 230

10.9.6 ATMMachine Class Operations 230

10.10 Packages and Managing Classes 230

10.11 Summary 232

11. ACCESS LAYER: OBJECT STORAGE AND OBJECT INTEROPERABILITY 237

11.1 Introduction 237

11.2 Object Store and Persistence: An Overview 238

11.3 Database Management Systems 239
11.3.1 Database Views 240
11.3.2 Database Models 240
11.3.2.1 Hierarchical Model 240
11.3.2.2 Network Model 241
11.3.2.3 Relational Model 241
11.3.3 Database Interface 242
11.3.3.1 Database Schema and Data Definition Language 242
11.3.3.2 Data Manipulation Language and Query Capabilities 242
11.4 Logical and Physical Database Organization and Access Control 243
11.4.1 Shareability and Transactions 243
11.4.2 Concurrency Policy 244
11.5 Distributed Databases and Client-Server Computing 245
11.5.1 What Is Client-Server Computing? 245
11.5.2 Distributed and Cooperative Processing 248
11.6 Distributed Objects Computing: The Next Generation of Client-Server Computing 250
11.6.1 Common Object Request Broker Architecture 251
11.6.2 Microsoft's ActiveX/DCOM 252
11.7 Object-Oriented Database Management Systems: The Pure World 252
11.7.1 Object-Oriented Databases versus Traditional Databases 254
11.8 Object-Relational Systems: The Practical World 255
11.8.1 Object-Relation Mapping 256
11.8.2 Table-Class Mapping 257
11.8.3 Table-Multiple Classes Mapping 258
11.8.4 Table-Inherited Classes Mapping 258
11.8.5 Tables-Inherited Classes Mapping 258
11.8.6 Keys for Instance Navigation 259
11.9 Multidatabase Systems 260
11.9.1 Open Database Connectivity: Multidatabase Application Programming Interfaces 262
11.10 Designing Access Layer Classes 264
11.10.1 The Process 265
11.11 Case Study: Designing the Access Layer for the ViaNet Bank ATM 269
11.11.1 Creating an Access Class for the BankClient Class 269
11.12 Summary 275
12. VIEW LAYER: DESIGNING INTERFACE OBJECTS 281
12.1 Introduction 281
12.2 User Interface Design as a Creative Process 281
12.3 Designing View Layer Classes 284
12.4 Macro-Level Process: Identifying View Classes by Analyzing Use Cases 285
12.5 Micro-Level Process 287
12.5.1 UI Design Rule 1. Making the Interface Simple 286

12.5.2 UI Design Rule 2. Making the Interface Transparent and Natural 290

12.5.3 UI Design Rule 3. Allowing Users to Be in Control of the Software 290

12.5.3.1 Make the Interface Forgiving 291

12.5.3.2 Make the Interface Visual 291

12.5.3.3 Provide Immediate Feedback 291

12.5.3.4 Avoid Modes 292

12.5.3.5 Make the Interface Consistent 292

12.6 The Purpose of a View Layer Interface 292

12.6.1 Guidelines for Designing Forms and Data Entry Windows 293

12.6.2 Guidelines for Designing Dialog Boxes and Error Messages 296

12.6.3 Guidelines for the Command Buttons Layout 298

12.6.4 Guidelines for Designing Application Windows 299

12.6.5 Guidelines for Using Colors 300

12.6.6 Guidelines for Using Fonts 302

12.7 Prototyping the User Interface 302

12.8 Case Study: Designing User Interface for the ViaNet Bank ATM 304

12.8.1 The View Layer Macro Process 305

12.8.2 The View Layer Micro Process 308

12.8.3 The BankClientAccessUI Interface Object 309

12.8.4 The MainUI Object Interface 309

12.8.5 The AccountTransactionUI Interface Object 309

12.8.6 The CheckingAccountUI and SavingsAccountUI Interface Objects 311

12.8.7 Defining the Interface Behavior 311

12.8.7.1 Identifying Events and Actions for the BankClientAccessUI Interface Object 313

12.8.7.2 Identifying Events and Actions for the MainUI Interface Object 313

12.8.7.3 Identifying Events and Actions for the Savings-AccountUI Interface Object 314

12.8.7.4 Identifying Events and Actions for the Account-TransactionUI Interface Object 315

12.9 Summary 317

PART FIVE

Software Quality

13. SOFTWARE QUALITY ASSURANCE 325

13.1 Introduction 325

13.2 Quality Assurance Tests 326

13.3 Testing Strategies 328

13.3.1 Black Box Testing 328

13.3.2 White Box Testing 329

13.3.3 Top-Down Testing 329

13.3.4 Bottom-Up Testing 330

13.4 Impact of Object Orientation on Testing 330

13.4.1 Impact of Inheritance in Testing 331

13.4.2 Reusability of Tests 331

13.5 Test Cases 331

13.5.1 Guidelines for Developing Quality Assurance Test Cases 332

13.6 Test Plan 333

13.6.1 Guidelines for Developing Test Plans 334

13.7 Continuous Testing 335

13.8 Myers's Debugging Principles 337

13.9 Case Study: Developing Test Cases for the ViaNet Bank ATM System 337

13.10 Summary 338

14. SYSTEM USABILITY AND MEASURING USER SATISFACTION 341

14.1 Introduction 341

14.2 Usability Testing 343

14.2.1 Guidelines for Developing Usability Testing 344

14.2.2 Recording the Usability Test 345

14.3 User Satisfaction Test 345

14.3.1 Guidelines for Developing a User Satisfaction Test 346

14.4 A Tool For Analyzing User Satisfaction: The User Satisfaction Test Template 347

14.5 Case Study: Developing Usability Test Plans and Test Cases for the ViaNet Bank ATM System 350

14.5.1 Develop Test Objectives 350

14.5.2 Develop Test Cases 350

14.5.3 Analyze the Tests 351

14.6 Summary 352

Appendices

Appendix A Document Template 355

Appendix B Introduction to Graphical User Interface 381

Glossary 391

Index 399

INTRODUCTION

The objective of Part I is to provide an overview of object-oriented systems development and why we should study it. In this part, we also look at object basics and the systems development life cycle. Part I consists of Chapters 1, 2, and 3.

此为试读，需要完整PDF请访问：www.ertongbook.com