EMBEDDED TECHNOLOGY™ SERIES

ePrimaryTurnLight->GEN(evGOTORed);
eSecondaryStraightLight->GEN(evGOTORed);
eSecondaryTurnLight->GEN(evGOTORed);
ePrimaryStraightLight->setIsFlashing(TRUE);
ePrimaryTurnLight->setIsFlashing(TRUE);

Real-Time UML Workshop for Embedded Systems

CD-ROM INCLUDED CONTAINING
•
Demo version of the Rhapsody UML tool
•
Models of the solutions

Bruce Powel Douglass

Newnes

Real-Time UML Workshop for Embedded Systems

Real-Time UML Workshop for Embedded Systems

by Bruce Powel Douglass, Ph.D.

ELSEVIER

AMSTERDAM • BOSTON • HEIDELBERG • LONDON
NEW YORK • OXFORD • PARIS • SAN DIEGO
SAN FRANCISCO • SINGAPORE • SYDNEY • TOKYO

Newnes is an imprint of Elsevier

Newnes

Newnes is an imprint of Elsevier
30 Corporate Drive, Suite 400, Burlington, MA 01803, USA
Linacre House, Jordan Hill, Oxford OX2 8DP, UK

Library of Congress Cataloging-in-Publication Data

Application submitted

British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library.

ISBN-13: 978-0-7506-7906-0
ISBN-10: 0-7506-7906-9

For information on all Newnes publications
visit our website at www.books.elsevier.com

07 08 09 10 10 9 8 7 6 5 4 3 2 1

Printed in the United States of America

This book is dedicated to my family—
two boys who make me proud,
a wife any man would envy,
and a stepdaughter who is so bright, she's scary.
It's good to be me—thanks, guys!

This Page Intentionally Left Blank

Contents

Preface . . . . **xiii**

Audience . . . . *xiv*

Goals . . . . *xv*

Where to Go After the Book . . . . *xv*

Evaluate UML on ARM . . . . *xv*

Acknowledgments . . . . **xvii**

About the Author . . . . **xix**

What's on the CD-ROM? . . . . **xxi**

Chapter 1: Introduction **1**

Basic Modeling Concepts of the UML *1*

Structural Elements and Diagrams *5*

Small Things: Objects, Classes, and Interfaces 5

Relations 10

Big Things: Subsystems, Components, and Packages 17

Behavioral Elements and Diagrams *19*

Actions and Activities 19

Operations and Methods 20

Activity Diagrams 20

Statecharts 22

Interactions 27

Use Case and Requirements Models *31*

Summary *33*

Check Out the CD-ROM *33*

Chapter 2: The Harmony Process **35**

Introduction *35*

The Harmony Development Process *36*

Why Process? 36

Harmony Process Overview 41

The Systems Engineering Harmony Workflows in Detail 43
The Incremental (Spiral) Development Workflows in Detail 47
Increment Review (Party!) Workflow 48
Design with the Harmony Process 55
Implementation 60
Test 61
Summary 63

Chapter 3: Specifying Requirements 65

Overview 65
Problem 3.1 Identifying Kinds of Requirements for Roadrunner Traffic Light Control System 67
Problem 3.2 Identifying Use Cases for the Roadrunner Traffic Light Control System 69
Problem 3.3 Mapping Requirements to Use Cases 69
Problem 3.4 Identifying Use Cases for the Coyote UAV System 70
Problem 3.5 Identifying Parametric Requirements 70
Problem 3.6 Capturing Quality of Service Requirements in Use Cases . . 71
Problem 3.7 Operational View: Identifying Traffic Light Scenarios 71
Problem 3.8 Operational View: CUAVS Optical Surveillance Scenarios 73
Problem 3.9 Specification View: Use-Case Description 73
Specification View: State Machines for Requirements Capture 74
Problem 3.10 Specification View: Capturing Complex Requirements . . . 76
Problem 3.11 Operational to Specification View: Capturing Operational Contracts 77
References 82

Chapter 4: Systems Architecture 83

Overview 83
Problem 4.1 Organizing the Systems Model 85
Problem 4.2 Subsystem Identification 90
Problem 4.3 Mapping Operational Contracts into Subsystem Architecture 92
Problem 4.4 Identifying Subsystem Use Cases 101
Looking Ahead 107

Chapter 5: Object Analysis 109
Overview 109
Key Strategies for Object Identification 111
Underline the Noun Strategy. 113
Identify the Causal Agents. 113
Identify Services (Passive Contributors or Server Objects) 114
Identify Messages and Information Flows 114
Identify Real-World Items 114
Identify Physical Devices. 114
Identify Key Concepts. 115
Identify Transactions 115
Identify Persistent Information 115
Identify Visual Elements 115
Identify Control Elements. 116
Apply Scenarios. 116
Problem 5.1 Apply Nouns and Causal Agents Strategies. 116
Problem 5.2 Apply Services and Messages Strategies. 125
Problem 5.3 Apply Real-World Items and Physical Devices Strategies . . 127
Problem 5.4 Apply Key Concepts and Transaction Strategies 128
Problem 5.5 Apply Identify Visual Elements and Scenarios Strategies . . 128
Problem 5.6 Merge Models from the Various Strategies 137
Looking Ahead 139

Chapter 6: Architectural Design 141
Overview 141
Problem 6.1 Concurrency and Resource Architecture. 147
Problem 6.2 Distribution Architecture. 158
Problem 6.3 Safety and Reliability Architecture 163
Looking Ahead 177

Chapter 7: Mechanistic and Detailed Design 179
Overview 179
Mechanistic Design 180
Delegation Pattern Strategy. 183
Interface Abstraction Pattern Strategy 185
Detailed Design. 187
Problem 7.1 Applying Mechanistic Design Patterns—Part 1. 192

Problem 7.2 Applying Mechanistic Design Patterns—Part 2. 196
Problem 7.3 Applying Detailed-Design State Behavior Patterns 201
Problem 7.4 Applying Detailed Design Idioms 206
Summary 214

Chapter 8: Specifying Requirements: Answers 215
Answer 3.1 Identifying Kinds of Requirements 215
Answer 3.2 Identifying Use Cases for Roadrunner Traffic Light Control System 216
Answer 3.3 Mapping Requirements to Use Cases. 219
Answer 3.4 Identifying Use Cases for Coyote UAV System 220
Answer 3.5 Identifying Parametric Requirements 222
Answer 3.6 Capturing Quality of Service Requirements. 223
Answer 3.7 Operational View: Identifying Traffic Light Scenarios 224
Answer 3.8 Operational View: CUAVS Optical Surveillance Scenarios. 228
Answer 3.9 Specification View: Use-Case Descriptions. 231
Answer 3.10 Specification View: Capturing Complex Requirements. . . 232
Answer 3.11 Operational to Specification View: Capturing Operational Contracts 238
References 242

Chapter 9: Systems Architecture: Answers 243
Answer 4.1 Organizing the Systems Model 243
Answer 4.2 Subsystem Identification 250
Answer 4.3 Mapping Operational Contracts into the Subsystem Architecture. 256
Answer 4.4 Identifying Subsystem Use Cases. 267

Chapter 10: Object Analysis: Answers 273
Answer 5.1 Apply Nouns and Causal Agents Strategies 273
Answer 5.2 Apply Services and Messages Strategies 291
Answer 5.3 Applying the Real-World Items and Physical Devices Strategies. 297
Answer 5.4 Apply Key Concepts and Transaction Strategies 299
Answer 5.5 Identify Visual Elements and Scenarios Strategies 303
Answer 5.6 Merge Models from the Various Strategies 315

Chapter 11: Architectural Design: Answers. 317
Answer 6.1 Concurrency and Resource Architecture. 317
Answer 6.2 Distribution Architecture. 319
Answer 6.3 Safety and Reliability Architecture 323

Chapter 12: Mechanistic and Detailed Design: Answers . . . 339
Answer 7.1 Applying Mechanistic Design Patterns—Part 1 339
Answer 7.2 Applying Mechanistic Design Patterns—Part 2 341
Answer 7.3 Applying Detailed-Design State Behavior Patterns. 346
Answer 7.4 Applying Detailed-Design Idioms. 351

Appendix A: The Roadrunner Intersection Controller System Specification. 357
Overview 357
The Intersection Controller (IC) 357
Configuration Parameters 358
Intersection Modes 361
The Vehicle Detector 365
Vehicular Traffic Light. 366
Pedestrian Light and Sensor 367
Front Panel Display. 368
Remote Communications 369
Power 370

Appendix B: The Coyote Unmanned Air Vehicle System (CUAVS) Specification 371
Overview 371
Primary CUAV System Components 371
The Unmanned Air Vehicle (UAV) 371
The Coyote Mission Planning and Control System (CMPCS) 372
Coyote Payloads 372
The Coyote Datalink Subsystem (CDS) 373
Detailed Requirements. 373
The Unmanned Air Vehicle (UAV) 373
Flight Modes. 373
Mission Modes 374
The Coyote Mission Planning and Control System (CMPCS). 374

The Coyote Reconnaissance Sensor Suite Payload (CSSP) 375
The Coyote Hellfire Attack Payload (CHAP). 376
The Coyote Datalink Subsystem (CDS). 377

Appendix C: UML Notational Summary. 379

Index. 401

Preface

Most books on UML or real-time systems, even those that I have written, are what I call "lecture books." They might have exercises at the end of the chapters, but they are basically organized around "Here, let me tell you about that…" This book is fundamentally different. It is meant to be a workbook, a set of guided exercises that teach by practice and example, not by exposition. As such, we won't discuss in detail various aspects of the UML, or MDA, or even process. We will mention such things in passing, but the expectation is that the reader already has a reasonable understanding of the UML and wants practice and experience in the application of the UML to embedded and real-time systems development. This book is meant as a companion to another of my books, *Real-Time UML, Third Edition: Advances in the UML for Real-Time Systems.*[1] That book *is* a lecture book and explains, in some detail, the structural, behavioral, and functional aspects of the UML, particularly how it applies to the development of real-time and embedded systems.

The present book is organized primarily around two example problems, whose detailed problem statements can be found in the appendices. To give some breadth of experience in the exercises, both a small-scale and large-scale application are provided. The small-scale example is an intersection traffic-light control system. This is a small-scale system but is still complex enough to be a rich source of work for the reader.[2] The larger-scale example is an Unmanned Air Vehicle (UAV), although it is more properly called an Unmanned Combat Air Vehicle (UCAV). This system is highly complex and offers good experience in doing requirements analysis and architectural design for large-scale systems, something the traffic-light control system

[1] Douglass, Bruce Powel, *Real-Time UML, Third Edition: Advances in the UML for Real-Time Systems,* Addison-Wesley, 2004.

[2] One of the Laws of Douglass is "Anything is simple if you don't have to do it!" This certainly applies in spades to the development of real-time systems, since even "simple" systems can be very complex.

cannot. The point of these examples is to provide a context for the reader to explore the application of UML to the specification, analysis, and design of such systems. It is most assuredly *not* to provide complete designs of such systems. Such systems are created by teams (or, as in the case of the UCAV, teams of teams). If the purpose of the book was to provide complete designs of such systems, the book would be between 20 and 100 times its present size.

This book can be thought of as being composed of several parts. First is a brief overview of the basics—UML and process—provided in Chapters 1 and 2. The next section is a series of chapters that walk through the phases of an incremental spiral process—requirements analysis, object analysis, architectural design, mechanistic design, and detailed design—posing problems that occur in the context of the examples. The next section provides solutions for the problems posed in the previous section. Lastly, there is some end-matter—the appendices that provide the problem statements and a brief notational guide to the UML.

All the UML diagrams in this book are created in the Rhapsody™ tool from Telelogic, and a demo copy of Rhapsody is provided in this book. The full version offers validation, model execution, and production-quality code generation facilities, as well as interfaces to other tools, such as requirements management, XMI import/export, etc. While these features are enormously helpful in actual development, they are not relevant to the purpose of the book. If you prefer to use another tool for the exercises, or even (gasp) pen and paper, you can do that as well.

The solutions are, of course, the "pointy end" of the stick. They are meant, not so much as the absolute truth or measure of goodness in UML modeling, but instead as examples of solutions to the problem. If your solution differs from the one provided and still meets the solution criteria, then that's perfectly fine. There are always many good solutions to any problem.[3] If your solution differs from the provided solution, I recommend that you study the differences between the two and determine under which conditions one or the other would be preferable.

Audience

The book is oriented towards the practicing professional software developer and the computer science major, in the junior or senior year. It focuses on practical experience by solving posed problems from the context of the examples. The book assumes a reasonable proficiency in UML and at least one programming language (C++ is used

[3] There are always even more *bad* solutions to any problem as well!

in this book, but C, Java, Ada, or any equivalent 3GL can be used). This book is meant as a companion to the author's *Real-Time UML, Third Edition: Advances in the UML for Real-Time Systems,* but that book is not required to use this workbook. Other books that may help the reader by providing more depth in various topic areas include the author's *Doing Hard Time: Developing Systems with UML, Objects, Frameworks and Patterns,* Addison-Wesley, 1999, and *Real-Time Design Patterns: Robust Scalable Architecture for Real-Time Systems,* Addison-Wesley, 2002.

Goals

The goal of this book is simply to give the reader practical experience in modeling different aspects of real-time and embedded systems. While there is some introduction early in the first two chapters of the book and at the beginning of the problem chapters, the goal is to pose problems for the reader to model. The book includes a demo version of the powerful Rhapsody™ tool, but the use of that tool is not required to go through the exercises.

By the time you've reached the end of the book, you will hopefully have the expertise to tackle the real modeling issues that you face in your life as a professional developer of real-time and embedded systems.

Where to Go After the Book

If you're interested in tools, training, or consulting, see *www.telelogic.com* or *www.ilogix.com*. The author teaches classes and consults worldwide on UML, MDA, DoDAF, architectural design, design patterns, requirements modeling, use cases, safety critical development with UML, behavioral modeling, the development process improvement, project management and scheduling, and quite a bit more. You can contact him for training or consulting services at *Bruce.Douglass@telelogic.com*. He also runs a (free) yahoo group forum at *http://groups.yahoo.com/group/RT-UML*—come on down! The I-Logix website also has many white papers available for downloading on different topics that may be of interest.

Evaluate UML on ARM

On the CD-ROM with this book, you will find the Rhapsody UML tool from Telelogic (for further information, see *What's on the CD-ROM?*). Do you want to evaluate the UML on an ARM microcontroller using C? Willert Software Tools delivers an evaluation version for the UML for generating ARM software. You can download this at *http://www.willert.de/rxf_eval*.

The evaluation software contains the following:

- Telelogic Rhapsody demo version
- Keil MicroVision/ARM compiler
  A good ARM compiler with a very usable IDE. Compiler, simulator and debugger integrated in a nice package.
- Willert WST52lt Bridge
  The Willert Framework RXF (modified version of the Rhapsody framework) optimized for small microcontrollers, code size under 4k ROM and 200 bytes of RAM, where speed and code size are critical.

The evaluation versions are limited but fully functional for the evaluation models. The generated models run in the Keil simulator. If you want to see the models run on real hardware you can order an evaluation kit for an attractive price. This kit contains a Keil MCB2130 Board with the Philips ARM7 chip and a uLink USB/JTAG debugger interface.

Bruce Powel Douglass, Ph.D.
Summer, 2006

Acknowledgments

I want to thank my editor, Tiffany Gasbarrini, for nagging me when I clearly needed it and being supportive when I need that too. My reviewers—Ian Macafee, Markus Rauber, Mark Richardson, Bart Jenkins, Frank Braun, and David McKean—were also helpful in keeping me honest. I have no doubt that errors remain, and I claim full responsibility for those. Most of all, I want to thank my family for keeping me sane, or at least trying to, while I burned the midnight electrons, creating this book.

About the Author

Bruce was raised by wolves in the Oregon wilderness. He taught himself to read at age 3 and calculus before age 12. He dropped out of school when he was 14 and traveled around the U.S. for a few years before entering the University of Oregon as a mathematics major. He eventually received his M.S. in exercise physiology from the University of Oregon and his Ph.D. in neurophysiology from the University of South Dakota Medical School, where he developed a branch of mathematics called autocorrelative factor analysis for studying information processing in multicellular biological neural systems.

Bruce has worked as a software developer in real-time systems for over 25 years and is a well-known speaker, author, and consultant in the area of real-time embedded systems. He is on the Advisory Board of the *Embedded Systems* and *UML World* conferences, where he has taught courses in software estimation and scheduling, project management, object-oriented analysis and design, communications protocols, finite state machines, design patterns, and safety-critical systems design. He develops and teaches courses and consults in real-time object-oriented analysis and design and project management and has done so for many years. He has authored articles for many journals and periodicals, especially in the real-time domain.

He is the Chief Evangelist[1] for Telelogic (formerly I-Logix), a leading producer of tools for real-time systems development. Bruce worked with Rational and the other UML partners on the specification of the UML. He is one of the co-chairs of the Object Management Group's Real-Time Analysis and Design Working Group. He is the author of several other books on software, including *Doing Hard Time: Developing Real-Time Systems with UML, Objects, Frameworks and Patterns* (Addison-Wesley, 1999), *Real-Time Design Patterns: Robust Scalable Architecture for Real-Time*

[1] Being a Chief Evangelist is much like being a Chief Scientist, except for the burning bushes and stone tablets.

Systems (Addison-Wesley, 2002), *Real-Time UML, Third Edition: Advances in the UML for Real-Time Systems* (Addison-Wesley, 2004), and several others, including a short textbook on table tennis.

Bruce enjoys classical music and has played classical guitar professionally. He has competed in several sports, including table tennis, bicycle racing, running, and full-contact Tae Kwon Do, although he currently only fights inanimate objects that don't hit back.

Bruce does extensive consulting and training throughout the world. If you're interested, contact him at *Bruce.Douglass@telelogic.com*.

What's on the CD-ROM?

- Demo version of the Rhapsody UML tool from Telelogic and various models of the solutions.
- README.TXT file in the root directory of the CD-ROM contains detailed instructions for how to install the software and obtain a temporary license to use the tool while working through the exercises in the book.