Real-Time Software Design for Embedded Systems

This textbook takes the reader from use cases to complete software architectures for real-time embedded systems using SysML, UML, and MARTE and shows how to apply the COMET/RTE design method to real-world problems. The author covers key topics such as use cases for real-time systems, state machines for real-time control, architectural patterns for distributed and hierarchical real-time control and for real-time component-based software architectures, performance analysis of real-time designs using real-time scheduling, and timing analysis on single- and multiple-processor systems.

Five complete case studies illustrating design issues include a light rail control system, a railroad crossing control system, a microwave oven control system, and an automated highway toll system.

Organized as an introduction followed by several self-contained chapters, the book is perfect for experienced software engineers wanting a quick reference at each stage of the analysis, design, and development of large-scale real-time embedded systems, as well as for advanced undergraduate or graduate courses in computer science, software engineering, systems engineering, and computer engineering programs.

Hassan Gomaa is Professor and former chair of the Department of Computer Science at George Mason University. Gomaa has more than thirty years of experience in software engineering, in both industry and academia. He has taught short in-depth industrial courses on real-time software design in North America, Europe, Japan, and South Korea. He has published more than 200 technical papers and is the author of four other textbooks on software design, including Software Modeling and Design and Designing Software Product Lines with UML.
REAL-TIME SOFTWARE
DESIGN FOR
EMBEDDED SYSTEMS

Hassan Gomaa
George Mason University
To Gill, William and Neela, Alex and Nicole,
Amanda and Neil, and Edward
Contents

Preface page xv
Annotated Table of Contents xix
Acknowledgments xxv

Part I Overview

1 Introduction 3
1.1 The Challenge 3
1.2 Real-Time Embedded Systems and Applications 3
1.3 Characteristics of Real-Time Embedded Systems 5
1.4 Distributed Real-Time Embedded Systems 7
1.5 Cyber-Physical Systems 9
1.6 Requirements for Real-Time Software Design Method for Embedded Systems 10
1.7 COMET/RTE: A Real-Time Software Design Method for Embedded Systems 10
1.8 Visual Modeling Languages: UML, SysML, and MARTE 11
1.9 Summary 11

2 Overview of UML, SysML, and MARTE 12
2.1 Model-Driven Architecture with SysML and UML 12
2.2 Use Case Diagrams 14
2.3 Classes and Objects 14
2.4 Class Diagrams 15
2.5 Interaction Diagrams 17
2.6 State Machine Diagrams 19
2.7 Package Diagrams 20
2.8 Concurrent Sequence and Communication Diagrams 20
2.9 Deployment Diagrams 23
2.10 Composite Structure Diagrams 24
Contents

2.11 UML Extension Mechanisms and Profiles 26
2.12 SysML 27
2.13 MARTE Profile 28
2.14 Timing Diagrams 29
2.15 Tool Support for UML, SysML, and MARTE 30
2.16 Summary 31

3 Real-Time Software Design and Architecture Concepts 32
3.1 Object-Oriented Concepts 32
3.2 Information Hiding 34
3.3 Inheritance 36
3.4 Active and Passive Objects 37
3.5 Concurrent Processing 37
3.6 Cooperation between Concurrent Tasks 39
3.7 Information Hiding Applied to Access Synchronization 42
3.8 Runtime Support for Real-Time Concurrent Processing 43
3.9 Task Scheduling 45
3.10 Software Architecture and Components 47
3.11 Summary 48

Part II Real-Time Software Design Method

4 Overview of Real-Time Software Design Method for Embedded Systems 51
4.1 COMET/RTE System and Software Life Cycle model 51
4.2 Phases in COMET/RTE Life Cycle model 52
4.3 Comparison of the COMET/RTE Life Cycle with Other Software Processes 56
4.4 Survey of Design Methods for Real-Time Embedded Systems 57
4.5 Multiple Views of System and Software Architecture 59
4.6 Summary 60

5 Structural Modeling for Real-Time Embedded Systems with SysML and UML 61
5.1 Static Modeling Concepts 62
5.2 Categorization of Blocks and Classes using Stereotypes 66
5.3 Structural Modeling of the Problem Domain with SysML 66
5.4 Structural Modeling of the System Context 69
5.5 Hardware/Software Boundary Modeling 72
5.6 Structural Modeling of the Software System Context 72
5.7 Defining Hardware/Software Interfaces 76
5.8 System Deployment Modeling 77
5.9 Summary 78
# Contents

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td><strong>Use Case Modeling for Real-Time Embedded Systems</strong></td>
<td>79</td>
</tr>
<tr>
<td>6.1</td>
<td>Use Cases</td>
<td>79</td>
</tr>
<tr>
<td>6.2</td>
<td>Actors</td>
<td>80</td>
</tr>
<tr>
<td>6.3</td>
<td>Identifying Use Cases</td>
<td>85</td>
</tr>
<tr>
<td>6.4</td>
<td>Documenting Use Cases in the Use Case Model</td>
<td>87</td>
</tr>
<tr>
<td>6.5</td>
<td>Specifying Nonfunctional Requirements</td>
<td>88</td>
</tr>
<tr>
<td>6.6</td>
<td>Examples of Use Case Descriptions</td>
<td>88</td>
</tr>
<tr>
<td>6.7</td>
<td>Use Case Relationships</td>
<td>92</td>
</tr>
<tr>
<td>6.8</td>
<td>The <em>Include</em> Use Case Relationship</td>
<td>92</td>
</tr>
<tr>
<td>6.9</td>
<td>The <em>Extend</em> Use Case Relationship</td>
<td>94</td>
</tr>
<tr>
<td>6.10</td>
<td>Use Case Packages</td>
<td>98</td>
</tr>
<tr>
<td>6.11</td>
<td>Summary</td>
<td>99</td>
</tr>
<tr>
<td>7</td>
<td><strong>State Machines for Real-Time Embedded Systems</strong></td>
<td>100</td>
</tr>
<tr>
<td>7.1</td>
<td>State Machines</td>
<td>101</td>
</tr>
<tr>
<td>7.2</td>
<td>Examples of State Machine</td>
<td>103</td>
</tr>
<tr>
<td>7.3</td>
<td>Events and Guard Conditions</td>
<td>103</td>
</tr>
<tr>
<td>7.4</td>
<td>Actions</td>
<td>105</td>
</tr>
<tr>
<td>7.5</td>
<td>Hierarchical State Machines</td>
<td>113</td>
</tr>
<tr>
<td>7.6</td>
<td>Cooperating State Machines</td>
<td>118</td>
</tr>
<tr>
<td>7.7</td>
<td>Inherited State Machines</td>
<td>119</td>
</tr>
<tr>
<td>7.8</td>
<td>Developing State Machines from Use Cases</td>
<td>121</td>
</tr>
<tr>
<td>7.9</td>
<td>Example of Developing a State Machine from a Use Case</td>
<td>122</td>
</tr>
<tr>
<td>7.10</td>
<td>Summary</td>
<td>125</td>
</tr>
<tr>
<td>8</td>
<td><strong>Object and Class Structuring for Real-Time Embedded Software</strong></td>
<td>126</td>
</tr>
<tr>
<td>8.1</td>
<td>Object and Class Structuring Criteria</td>
<td>126</td>
</tr>
<tr>
<td>8.2</td>
<td>Object and Class Structuring Categories</td>
<td>127</td>
</tr>
<tr>
<td>8.3</td>
<td>Object Behavior and Patterns</td>
<td>128</td>
</tr>
<tr>
<td>8.4</td>
<td>Boundary Classes and Objects</td>
<td>129</td>
</tr>
<tr>
<td>8.5</td>
<td>Entity Classes and Objects</td>
<td>136</td>
</tr>
<tr>
<td>8.6</td>
<td>Control Classes and Objects</td>
<td>137</td>
</tr>
<tr>
<td>8.7</td>
<td>Application Logic Classes and Objects</td>
<td>139</td>
</tr>
<tr>
<td>8.8</td>
<td>Summary</td>
<td>141</td>
</tr>
<tr>
<td>9</td>
<td><strong>Dynamic Interaction Modeling for Real-Time Embedded Software</strong></td>
<td>143</td>
</tr>
<tr>
<td>9.1</td>
<td>Object Interaction Modeling</td>
<td>144</td>
</tr>
<tr>
<td>9.2</td>
<td>Message Sequence Description</td>
<td>145</td>
</tr>
<tr>
<td>9.3</td>
<td>Approach for Dynamic Interaction Modeling</td>
<td>145</td>
</tr>
<tr>
<td>9.4</td>
<td>Stateless Dynamic Interaction Modeling</td>
<td>146</td>
</tr>
<tr>
<td>9.5</td>
<td>Examples of Stateless Dynamic Interaction Modeling</td>
<td>147</td>
</tr>
<tr>
<td>9.6</td>
<td>State Dependent Dynamic Interaction Modeling</td>
<td>150</td>
</tr>
<tr>
<td>9.7</td>
<td>Example of State Dependent Dynamic Interaction Modeling: Microwave Oven System</td>
<td>154</td>
</tr>
<tr>
<td>9.8</td>
<td>Summary</td>
<td>162</td>
</tr>
</tbody>
</table>
10 Software Architectures for Real-Time Embedded Systems

10.1 Overview of Software Architectures 164
10.2 Multiple Views of a Software Architecture 166
10.3 Transition from Analysis to Design 170
10.4 Separation of Concerns in Subsystem Design 172
10.5 Subsystem Structuring Criteria 175
10.6 Decisions about Message Communication between Subsystems 181
10.7 Summary 183

11 Software Architectural Patterns for Real-Time Embedded Systems

11.1 Software Design Patterns 184
11.2 Layered Software Architectural Patterns 186
11.3 Control Patterns for Real-Time Software Architectures 190
11.4 Client/Service Software Architectural Patterns 194
11.5 Basic Software Architectural Communication Patterns 197
11.6 Software Architectural Broker Patterns 203
11.7 Group Message Communication Patterns 206
11.8 Documenting Software Architectural Patterns 209
11.9 Applying Software Architectural Patterns 209
11.10 Summary 210

12 Component-Based Software Architectures for Real-Time Embedded Systems

12.1 Concepts for Component-Based Software Architectures 212
12.2 Designing Distributed Component-Based Software Architectures 212
12.3 Component Interface Design 213
12.4 Designing Composite Components 217
12.5 Examples of Component-Based Software Architecture 218
12.6 Component Structuring Criteria 221
12.7 Design of Service Components 223
12.8 Distribution of Data 227
12.9 Software Deployment 228
12.10 Design of Software Connectors 229
12.11 Summary 232

13 Concurrent Real-Time Software Task Design

13.1 Concurrent Task Structuring Issues 234
13.2 Categorizing Concurrent Tasks 234
13.3 I/O Task Structuring Criteria 235
13.4 Internal Task Structuring Criteria 242
13.5 Task Priority Criteria 248
Contents

13.6 Task Clustering Criteria 249
13.7 Design Restructuring by Using Task Inversion 256
13.8 Developing the Task Architecture 257
13.9 Task Communication and Synchronization 258
13.10 Task Interface and Task Behavior Specifications 264
13.11 Summary 265

14 Detailed Real-Time Software Design 266
14.1 Design of Composite Tasks 266
14.2 Synchronization of Access to Classes 274
14.3 Designing Monitors 278
14.4 Designing Connectors for Inter-Task Communication 284
14.5 Task Event Sequencing Logic 291
14.6 Detailed Real-Time Software Design in Robot and Vision Systems 293
14.7 Implementing Concurrent Tasks in Java 295
14.8 Summary 296

15 Designing Real-Time Software Product Line Architectures 297
15.1 Software Product Line Engineering 298
15.2 Problem Description of Microwave Oven SPL 299
15.3 Requirements Modeling for Software Product Lines 299
15.4 Analysis Modeling for Software Product Lines 303
15.5 Design Modeling for Software Product Lines 308
15.6 Summary 310

Part III Analysis of Real-Time Software Designs

16 System and Software Quality Attributes for Real-Time Embedded Systems 313
16.1 Scalability 313
16.2 Performance 315
16.3 Availability 315
16.4 Safety 316
16.5 Security 317
16.6 Maintainability 318
16.7 Modifiability 319
16.8 Testability 320
16.9 Traceability 321
16.10 Reusability 322
16.11 Summary 323

17 Performance Analysis of Real-Time Software Designs 324
17.1 Real-Time Scheduling Theory 325
17.2 Real-Time Scheduling for Aperiodic Tasks and Task Synchronization 330
17.3 Generalized Real-Time Scheduling Theory 331
17.4 Performance Analysis Using Event Sequence Analysis 336
17.5 Performance Analysis Using Real-Time Scheduling Theory and Event Sequence Analysis 338
17.6 Advanced Real-Time Scheduling Algorithms 339
17.7 Performance Analysis of Multiprocessor Systems 340
17.8 Estimation and Measurement of Performance Parameters 343
17.9 Summary 345

18 Applying Performance Analysis to Real-Time Software Designs 346
18.1 Example of Performance Analysis Using Event Sequence Analysis 346
18.2 Example of Performance Analysis Using Real-Time Scheduling Theory 351
18.3 Example of Performance Analysis Using Real-Time Scheduling Theory and Event Sequence Analysis 354
18.4 Design Restructuring 367
18.5 Summary 368

Part IV Real-Time Software Design Case Studies for Embedded Systems

19 Microwave Oven Control System Case Study 371
19.1 Problem Description 371
19.2 Structural Modeling 372
19.3 Use Case Modeling 373
19.4 Object and Class Structuring 377
19.5 Dynamic State Machine Modeling 379
19.6 Dynamic Interaction Modeling 383
19.7 Design Modeling 395
19.8 Performance Analysis of Real-Time Software Design 403
19.9 Component-Based Software Architecture 406
19.10 Detailed Software Design 413
19.11 System Configuration and Deployment 415

20 Railroad Crossing Control System Case Study 417
20.1 Problem Description 417
20.2 Structural Modeling 418
20.3 Use Case Modeling 422
20.4 Dynamic State Machine Modeling 426
20.5 Object and Class Structuring 429
20.6 Dynamic Interaction Modeling 429
20.7 Design Modeling 435
20.8 Performance Analysis of Real-Time Software Design 441
20.9 Component-Based Software Architecture 443
20.10 System Configuration and Deployment 450

21 Light Rail Control System Case Study 451
21.1 Problem Description 451
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>21.2 Structural Modeling</td>
<td>452</td>
</tr>
<tr>
<td>21.3 Use Case Modeling</td>
<td>455</td>
</tr>
<tr>
<td>21.4 Dynamic State Machine Modeling</td>
<td>464</td>
</tr>
<tr>
<td>21.5 Subsystem Structuring</td>
<td>471</td>
</tr>
<tr>
<td>21.6 Object and Class Structuring</td>
<td>471</td>
</tr>
<tr>
<td>21.7 Dynamic Interaction Modeling</td>
<td>474</td>
</tr>
<tr>
<td>21.8 Design Modeling</td>
<td>486</td>
</tr>
<tr>
<td>21.9 Subsystem Integrated Communication Diagrams</td>
<td>486</td>
</tr>
<tr>
<td>21.10 Design of Distributed Light Rail System</td>
<td>487</td>
</tr>
<tr>
<td>21.11 Component-Based Software Architecture</td>
<td>495</td>
</tr>
<tr>
<td>21.12 System Configuration and Deployment</td>
<td>499</td>
</tr>
<tr>
<td>22 Pump Control System Case Study</td>
<td>500</td>
</tr>
<tr>
<td>22.1 Problem Description</td>
<td>500</td>
</tr>
<tr>
<td>22.2 Structural Modeling</td>
<td>501</td>
</tr>
<tr>
<td>22.3 Use Case Modeling</td>
<td>501</td>
</tr>
<tr>
<td>22.4 Object and Class Structuring</td>
<td>504</td>
</tr>
<tr>
<td>22.5 Dynamic State Machine Modeling</td>
<td>504</td>
</tr>
<tr>
<td>22.6 Dynamic Interaction Modeling</td>
<td>506</td>
</tr>
<tr>
<td>22.7 Design Modeling</td>
<td>507</td>
</tr>
<tr>
<td>23 Highway Toll Control System Case Study</td>
<td>513</td>
</tr>
<tr>
<td>23.1 Problem Description</td>
<td>513</td>
</tr>
<tr>
<td>23.2 Use Case Modeling</td>
<td>514</td>
</tr>
<tr>
<td>23.3 Software System Context Modeling</td>
<td>516</td>
</tr>
<tr>
<td>23.4 Object and Class Structuring</td>
<td>516</td>
</tr>
<tr>
<td>23.5 Dynamic State Machine Modeling</td>
<td>517</td>
</tr>
<tr>
<td>23.6 Dynamic Interaction Modeling</td>
<td>517</td>
</tr>
<tr>
<td>23.7 Design Modeling</td>
<td>519</td>
</tr>
<tr>
<td>Appendix A: Conventions Used in This Textbook</td>
<td>525</td>
</tr>
<tr>
<td>Appendix B: Catalog of Software Architectural Patterns</td>
<td>530</td>
</tr>
<tr>
<td>Appendix C: Pseudocode Templates for Concurrent Tasks</td>
<td>551</td>
</tr>
<tr>
<td>Appendix D: Teaching Considerations</td>
<td>557</td>
</tr>
<tr>
<td>Glossary</td>
<td>559</td>
</tr>
<tr>
<td>Bibliography</td>
<td>573</td>
</tr>
<tr>
<td>Index</td>
<td>581</td>
</tr>
</tbody>
</table>
Preface

OVERVIEW

This book describes a comprehensive concurrent object-oriented and component-based method for the real-time software design of distributed embedded systems and the cyber components of cyber-physical systems.

The book starts with a discussion of the characteristics of real-time embedded systems and a description of the important concepts in the design of these systems. It then describes a detailed object-oriented and component-based method for developing architectural and detailed designs of real-time embedded software. The design method and the impact of design decisions are further illustrated through the use of detailed case studies covering a range of real-time embedded systems. All examples and case studies are documented using the industry standard UML, SysML, and MARTE visual modeling languages and notations.

The book is aimed at both the professional market and the academic market, particularly at the graduate level. It assumes a basic background in UML and object-oriented principles, although a brief overview is given of each.

WHAT THIS BOOK PROVIDES

There are various textbooks on the market describing general object-oriented analysis and design concepts and methods. However, real-time and embedded systems have special needs, which are only treated superficially in these books. Other books describe real-time systems in general or provide a survey-based approach. The focus of this book is on real-time software design for embedded systems. Because real-time systems are usually embedded, the method described in the book takes a systems-engineering perspective addressing system-wide issues involving both hardware and software.

This book provides a comprehensive treatment of the application of object-oriented and component-based concepts to the analysis and design of complex real-time and embedded software. The distinguishing features of this book are that it:

1. Describes fundamental concepts in the software design of object-oriented real-time and embedded systems. This includes concurrent tasks; the
Preface

object-oriented concepts of information hiding, classes, and inheritance; distributed component technology; software architectures; finite state machines; and performance analysis of real-time software designs using real-time scheduling.

2. Describes in considerable detail a concurrent object-oriented analysis and design method for real-time and embedded software that is suitable for use in large and complex industrial software development efforts.

3. Seamlessly and systematically integrates several important design concepts for real-time software design, including concurrency, objects, components, services, architectural design patterns, software product lines, and real-time scheduling.

4. Presents several detailed case studies, illustrating different characteristics of real-time and embedded software systems, providing a step-by-step description of how to proceed from real-time systems requirements analysis to detailed software design. All case studies are documented using the SysML, UML 2, and MARTE visual modeling languages and notations.

5. Provides appendixes on a catalog of architectural design patterns and pseudocode templates for detailed task design and includes a glossary and a bibliography, as well as teaching considerations on how to teach industrial and academic courses based on it.

INTENDED AUDIENCE

This book is intended for both professional and academic audiences. The professional audience includes systems engineers, software engineers, computer engineers, analysts, architects, designers, programmers, project leaders, technical managers, and quality assurance specialists, who are involved in the design and development of large-scale real-time and embedded software systems in industry and government. The academic audience includes senior undergraduate and graduate-level students in computer science, software engineering, systems engineering, and computer engineering, as well as researchers in the field.

WAYS TO READ THIS BOOK

This book may be read in various ways. It can be read in the order in which it is presented, in which case Chapters 1 through 3 provide introductory concepts; Chapter 4 provides an overview of the COMET/RTE real-time software design method for embedded systems; Chapters 5 through 18 provide an in-depth treatment of real-time software design; and Chapters 19 through 23 provide detailed case studies.

Alternatively, some readers may wish to skip some chapters, depending on their level of familiarity with the topics discussed. Chapters 1 through 3 are introductory and may be skipped by experienced readers. Readers familiar with software design concepts may skip Chapter 3. Readers particularly interested in real-time software design can proceed directly to the description of COMET/RTE, starting in Chapter 4. Readers who are not familiar with UML, SysML, or MARTE can read Chapter 2 in conjunction with Chapters 4 through 18.

Experienced software designers may also use this book as a reference, referring to various chapters as their projects reach a particular stage of the requirements,
Preface

analysis, or design process. Each chapter is relatively self-contained. For example, at
different times one might refer to Chapter 5 for a discussion of structural modeling
using SysML and UML, Chapter 6 for a description of use cases, and to Chapter 7 for
a description of state machines. Chapter 10 can be referenced for an overview of real-
time software architectures; Chapter 11 and Appendix B for software architectural
patterns; Chapter 12 for component-based software architectures; and Chapter 13
for concurrent real-time task design with MARTE. Chapter 15 can be consulted for
software product line design; Chapter 16 for system and software quality attributes;
and Chapters 17 and 18 for performance analysis of real-time software designs. One
can also improve one’s understanding of how to use the COMET/RTE method by
reading the case studies in Chapters 19–23, because each case study explains the deci-
sions made at each step of requirements, analysis, and design.

Hassan Gomaa
George Mason University
November 2015
Email: hgomaa@gmu.edu
www: http://mason.gmu.edu/~hgomaa
Annotated Table of Contents

PART I: OVERVIEW

Chapter 1. Introduction
This chapter provides an overview of real-time embedded systems and applications and then describes the major characteristics of real-time embedded systems, both centralized and distributed. This chapter also provides an overview of the emerging field of cyber-physical systems, for which real-time software is a critical component. This chapter then introduces COMET/RTE, the design method for real-time embedded systems described and applied in the book.

Chapter 2. Overview of UML, SysML, and MARTE
This chapter describes the main features of the UML, SysML, and MARTE visual modeling languages and notations that are particularly suited for real-time design using the COMET/RTE method. The purpose of this chapter is not to be a full exposition of UML, SysML, and MARTE, because several detailed books exist on these topics, but rather to provide a brief overview of each, in particular those parts that are used by COMET/RTE.

Chapter 3. Real-Time Software Design and Architecture Concepts
This chapter describes key concepts in the software design of concurrent object-oriented real-time embedded systems as well as important concepts for developing the architecture of these systems. The concurrent processing concept is introduced and the issues of communication and synchronization between concurrent tasks are described. Some general design concepts are also discussed from the perspective of their applicability to real-time design, including object-oriented design concepts of information hiding and inheritance, software architecture, and software components. This chapter also briefly discusses technology issues related to real-time software design, including real-time operating systems and task scheduling.
PART II: REAL-TIME SOFTWARE DESIGN METHOD

Chapter 4. Overview of Real-Time Software Design Method for Embedded Systems

This chapter provides an overview of the software design method for real-time embedded systems called COMET/RTE (Concurrent Object Modeling and Architectural Design Method for Real-Time Embedded systems), which uses the SysML, UML, and MARTE visual modeling languages and notations. This chapter also describes the iterative system and software life cycle of COMET/RTE and how it compares to other life cycles. It then describes the main steps in using COMET/RTE.

Chapter 5. Structural Modeling for Real-Time Embedded Systems with SysML and UML

This chapter describes how structural modeling can be used as an integrated approach for system and software modeling of embedded systems consisting of both hardware and software components, using SysML and UML. This chapter describes structural modeling of the problem domain, structural modeling of the hardware/software system context, hardware/software boundary modeling, structural modeling of the software system context, defining hardware/software interfaces, and system deployment modeling.

Chapter 6. Use Case Modeling for Real-Time Embedded Systems

This chapter describes how use case modeling can be applied to real-time embedded systems from both systems engineering and software engineering perspectives. After an overview of the basic principles of use cases, it provides a more in-depth focus on capturing the functional and nonfunctional requirements for real-time and embedded systems. It also explains the difference between system and software use cases and actors.

Chapter 7. State Machines for Real-Time Embedded Systems

This chapter describes state machine modeling concepts, which are particularly important for reactive real-time systems. This chapter covers events, states, conditions, actions and activities, entry and exit actions, composite states, and hierarchical state machines with sequential and orthogonal substates. The issues of developing cooperating state machines, inheritance in state machines, and deriving state machines from use cases are also addressed.

Chapter 8. Object and Class Structuring for Real-Time Embedded Software

This chapter describes the identification and categorization of software classes and objects, in particular the role the class plays in the real-time software, including boundary, control, and entity classes. It also describes the corresponding behavior pattern for each category of object.

This chapter describes dynamic interaction modeling concepts. Interaction diagrams are developed for each use case, including the main scenario and alternative scenarios. Specific discussions on state dependent real-time embedded systems cover dynamic interaction modeling for state dependent object interactions. This chapter describes how state machines and interaction diagrams relate to each other and how to make them consistent with each other.

Chapter 10. Software Architectures for Real-Time Embedded Systems

This chapter introduces software architectural concepts for distributed real-time embedded systems. Issues in Software Architectural Design are described. The benefits of developing multiple views of a software architecture are explained. This chapter also provides an introduction to software components and component-based software architectures. The transition from requirements and analysis to architectural design is carefully explained. Separation of concerns in subsystem design and subsystem structuring criteria are also described. This is followed by designing subsystem message communication interfaces.

Chapter 11. Software Architectural Patterns for Real-Time Embedded Systems

The role of architectural design patterns in developing the real-time software architecture is described. An overview of software architectural patterns is presented, including architectural structure and communication patterns. Architectural patterns for real-time systems are described, including layered patterns, real-time control patterns, client/service patterns, brokering patterns, and event-based subscription/notification patterns.

Chapter 12. Component-Based Software Architectures for Real-Time Embedded Systems

This chapter describes how a distributed real-time architecture is designed as a component-based software architecture, which can be deployed to multiple nodes in a distributed environment. Component design issues are described, including composite and simple components, component interface design with provided and required interfaces, ports, and connectors. The design of service components and distributed software connectors are also described. Component configuration and deployment issues are explained.

Chapter 13. Concurrent Real-Time Software Task Design

This chapter describes the design of concurrent tasks using the MARTE real-time modeling notation. Concurrent task structuring is described, including event-driven tasks, periodic tasks, and demand driven tasks. Task clustering of objects is also described. Design of task interfaces is described, including synchronous and
asynchronous message communication, event synchronization, and communication through passive objects. The implications of different types of message communication on the concurrent behavior of the software architecture are described.

Chapter 14. Detailed Real-Time Software Design
This chapter describes the detailed design of concurrent tasks. The design of composite tasks with nested passive classes is described. Task synchronization of access to passive classes is described using mutual exclusion, multiple readers and writers, and monitors. The design of connectors for inter-task communication is explained. The implementation of concurrent tasks as Java threads is briefly described.

Chapter 15. Designing Real-Time Software Product Line Architectures
This chapter describes the characteristics of real-time software product lines. The important concepts of feature modeling, and modeling commonality and variability, are explained. How to model variability in use cases, static and dynamic models, and software architectures is explained. The chapter goes on to describe how to model common and variable components in software product line architectures. The engineering of software applications from product line artifacts is explained.

PART III: ANALYSIS OF REAL-TIME SOFTWARE DESIGNS

Chapter 16. System and Software Quality Attributes for Real-Time Embedded Systems
This chapter describes system and software quality attributes and how they are used to evaluate the quality of the real-time embedded system and software architecture. System quality attributes include scalability, performance, availability, safety, and security. Software quality attributes include maintainability, modifiability, testability, traceability, and reusability. This chapter also discusses how the COMET/RTE real-time design method supports the system and software quality attributes.

Chapter 17. Performance Analysis of Real-Time Software Designs
This chapter presents methods for analyzing the performance of real-time embedded software designs. It describes two approaches for analyzing the performance of a design, real-time scheduling theory and event sequence analysis, which are then combined to analyze a concurrent multitasking design. Advanced real-time scheduling algorithms, including deadline monotonic scheduling, dynamic priority scheduling, and multiprocessor scheduling, are described. Practical approaches for analyzing the performance of multiprocessor systems including multicore systems are also described. Estimation and measurement of performance parameters are discussed.

Chapter 18. Applying Performance Analysis to Real-Time Software Designs
This chapter applies the real-time performance analysis concepts and theory described in Chapter 17 to the real-time design of a Light Rail Control System.
Real-time scheduling theory and event sequence analysis are both applied to analyze the performance of the concurrent multitasking design. The performance of the design executing on single-processor and multiprocessor systems is also analyzed and compared.

PART IV: REAL-TIME SOFTWARE DESIGN CASE STUDIES FOR EMBEDDED SYSTEMS

Chapter 19. Microwave Oven Control System Case Study
This chapter describes how the COMET-RTE design method is applied to the design of the embedded real-time software for a consumer product – a microwave oven control system.

Chapter 20. Railroad Crossing Control System Case Study
This chapter describes how the COMET-RTE design method is applied to the design of the embedded real-time software for a safety critical railroad crossing control system.

Chapter 21. Light Rail Control System Case Study
This chapter describes how the COMET-RTE design method is applied to the design of an embedded light rail control system, in which the automated control of driverless trains must be done safely and in a timely manner.

Chapter 22. Pump Control System Case Study
This chapter describes a concise case study of how the COMET-RTE design method is applied to the design of the embedded real-time software for a pump control system.

Chapter 23. Highway Toll Control System Case Study
This chapter describes a concise case study of how the COMET-RTE design method is applied to the design of the distributed embedded real-time software for a highway toll control system.

APPENDIX A. CONVENTIONS USED IN THIS TEXTBOOK
The conventions for naming requirements, analysis, and design artifacts are described. The conventions used for message sequence numbering on interaction diagrams are described.

APPENDIX B. CATALOG OF SOFTWARE ARCHITECTURAL PATTERNS
Each architectural structure and communication pattern is described using a standard design pattern template.
xxiv Annotated Table of Contents

APPENDIX C. PSEUDOCODE TEMPLATES FOR CONCURRENT TASKS
The pseudocode for several different kinds of concurrent tasks is provided.

APPENDIX D. TEACHING CONSIDERATIONS
An outline is given for teaching academic (both graduate and senior undergraduate) courses and industrial courses.

GLOSSARY
BIBLIOGRAPHY
INDEX
Acknowledgments

I gratefully acknowledge the reviewers of earlier drafts of the manuscript for their constructive comments. Hakan Aydin very carefully reviewed Chapter 17 on performance analysis and made several valuable and insightful comments. Kevin Mills and Rob Pettit provided very thorough and constructive reviews of several chapters. The anonymous reviewers provided many helpful comments. I am very grateful to the students in my software modeling and design, real-time software analysis and design, and reusable software architecture courses at George Mason University for their enthusiasm, dedication, and valuable feedback. Many thanks are due to Aparna Keshavanurthy, Ehsan Kouroshfar, Carolyn Koerner, Nan Li, and Upsorn Praphamontripong, for their hard work and careful attention producing the figures. I am also very grateful to the Cambridge University Press editorial and production staff, including Lauren Cowles, and the production staff at Aptara.

I gratefully acknowledge the Software Engineering Institute (SEI) for the material provided on real-time scheduling, on which parts of Chapter 17 are based. I also gratefully acknowledge the permission given to me by Pearson Education, Inc., to use material from my earlier textbooks, *Designing Concurrent, Distributed, and Real-Time Applications with UML*, © 2000 Hassan Gomaa, Reproduced by permission of Pearson Education, Inc., and *Designing Software Product Lines with UML*, © 2005 Hassan Gomaa, Reproduced by permission of Pearson Education, Inc.

Last, but not least, I would like to thank my wife, Gill, for her encouragement, understanding, and support.