UML 2 Certification Guide
Morgan Kaufmann OMG Press

Morgan Kaufmann Publishers and the Object Management Group™ (OMG) have joined forces to publish a line of books addressing business and technical topics related to OMG’s large suite of software standards.

OMG is an international, open membership, not-for-profit computer industry consortium that was founded in 1989. The OMG creates standards for software used in government and corporate environments to enable interoperability and to forge common development environments that encourage the adoption and evolution of new technology. OMG members and its board of directors consist of representatives from a majority of the organizations that shape enterprise and Internet computing today.

OMG’s modeling standards, including the Unified Modeling Language™ (UML®) and Model Driven Architecture® (MDA), enable powerful visual design, execution and maintenance of software, and other processes—for example, IT Systems Modeling and Business Process Management. The middleware standards and profiles of the Object Management Group are based on the Common Object Request Broker Architecture® (CORBA) and support a wide variety of industries.

More information about OMG can be found at http://www.omg.org/.

Forthcoming Morgan Kaufmann OMG Press Titles

UML 2 Certification Guide: Fundamental and Intermediate Exams
Tim Weilkiens and Bernd Oestereich

Real-Life MDA: Solving Business Problems with Model Driven Architecture
Michael Guttmann and John Parodi

Architecture Driven Modernization: A Series of Industry Case Studies
Bill Ulrich
UML 2 Certification Guide

Fundamental and Intermediate Exams

Tim Weilkiens
Bernd Oestereich
For Ben
CONTENTS

LIST OF METAMODELS xi

FOREWORD Richard Mark Soley xv

FOREWORD Ivar Jacobson xvii

PREFACE xix

I INTRODUCTION 3

1.1 What Is UML? 3
  1.1.1 The Three Amigos 3
  1.1.2 The Object Management Group 4
  1.1.3 The History of UML 4
  1.1.4 UML Compliance Levels 6
  1.1.5 UML Subspecifications 6
  1.1.6 The Metamodel of UML 2.0 6

1.2 The UML Certification Program 7
  1.2.1 Fundamental Level 9
  1.2.2 Intermediate Level 9
  1.2.3 Advanced Level 9
  1.2.4 Prerequisites and Registration 10
  1.2.5 Examination Procedure 10

1.3 Exam Preparation 11

2 OCUP FUNDAMENTAL 15

2.1 General Basics 15
  2.1.1 Examination Topics 15
  2.1.2 Datatypes 15
## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1.3 Overview of Diagrams</td>
<td>17</td>
</tr>
<tr>
<td>2.1.4 Stereotypes</td>
<td>20</td>
</tr>
<tr>
<td><strong>2.2 Class Diagrams</strong></td>
<td></td>
</tr>
<tr>
<td>2.2.1 Examination Topics</td>
<td>22</td>
</tr>
<tr>
<td>2.2.2 Basic Concepts</td>
<td>22</td>
</tr>
<tr>
<td>2.2.3 Namespaces</td>
<td>25</td>
</tr>
<tr>
<td>2.2.4 Typed Elements</td>
<td>27</td>
</tr>
<tr>
<td>2.2.5 Multiplicities</td>
<td>29</td>
</tr>
<tr>
<td>2.2.6 Value Specification</td>
<td>31</td>
</tr>
<tr>
<td>2.2.7 Constraints</td>
<td>33</td>
</tr>
<tr>
<td>2.2.8 Instance Specification</td>
<td>34</td>
</tr>
<tr>
<td>2.2.9 Classifier</td>
<td>35</td>
</tr>
<tr>
<td>2.2.10 Features</td>
<td>38</td>
</tr>
<tr>
<td>2.2.11 Operations</td>
<td>40</td>
</tr>
<tr>
<td>2.2.12 Properties</td>
<td>42</td>
</tr>
<tr>
<td>2.2.13 Associations</td>
<td>46</td>
</tr>
<tr>
<td>2.2.14 Classes</td>
<td>53</td>
</tr>
<tr>
<td>2.2.15 Generalization</td>
<td>55</td>
</tr>
<tr>
<td>2.2.16 Packages</td>
<td>58</td>
</tr>
<tr>
<td>2.2.17 Dependencies</td>
<td>63</td>
</tr>
<tr>
<td>2.2.18 Interfaces</td>
<td>67</td>
</tr>
<tr>
<td><strong>2.3 Behavior Basics</strong></td>
<td>71</td>
</tr>
<tr>
<td>2.3.1 Examination Topics</td>
<td>72</td>
</tr>
<tr>
<td>2.3.2 Introduction</td>
<td>72</td>
</tr>
<tr>
<td>2.3.3 The Call Model</td>
<td>74</td>
</tr>
<tr>
<td>2.3.4 Behavior Parameters</td>
<td>75</td>
</tr>
<tr>
<td><strong>2.4 Activity Diagrams</strong></td>
<td>77</td>
</tr>
<tr>
<td>2.4.1 Examination Topics</td>
<td>77</td>
</tr>
<tr>
<td>2.4.2 Token Flow</td>
<td>83</td>
</tr>
<tr>
<td>2.4.3 Control Nodes</td>
<td>86</td>
</tr>
<tr>
<td>2.4.4 Object Nodes</td>
<td>88</td>
</tr>
<tr>
<td><strong>2.5 Interaction Diagrams</strong></td>
<td>92</td>
</tr>
<tr>
<td>2.5.1 Examination Topics</td>
<td>92</td>
</tr>
<tr>
<td>2.5.2 Interactions</td>
<td>93</td>
</tr>
<tr>
<td>2.5.3 Communication, Timing, and Interaction</td>
<td></td>
</tr>
<tr>
<td>Overview Diagrams</td>
<td>99</td>
</tr>
<tr>
<td><strong>2.6 Use Cases</strong></td>
<td>104</td>
</tr>
<tr>
<td>2.6.1 Examination Topics</td>
<td>104</td>
</tr>
<tr>
<td>2.6.2 Use Cases and Actors</td>
<td>104</td>
</tr>
<tr>
<td>2.6.3 Use Case Relationships</td>
<td>108</td>
</tr>
</tbody>
</table>
3 OCUP INTERMEDIATE

3.1 Composite Structure Diagrams
   3.1.1 Examination Topics 117
   3.1.2 Structured Classifiers 118
   3.1.3 Connectable Elements 120
   3.1.4 Ports and Classes 124
   3.1.5 Invocation Actions, Triggers, and Variables 128
   3.1.6 Collaboration 130

3.2 Component Diagrams
   3.2.1 Examination Topics 133
   3.2.2 Components 134
   3.2.3 Connectors 139

3.3 Behavior Basics
   3.3.1 Examination Topics 142
   3.3.2 Communications 143
   3.3.3 The SimpleTime Model 147

3.4 Action Models
   3.4.1 Examination Topics 150
   3.4.2 Actions 150
   3.4.3 Invocation Actions 152
   3.4.4 Opaque Actions 159
   3.4.5 Object Actions 159
   3.4.6 Structural Feature Actions 164
   3.4.7 Link Actions 170
   3.4.8 Variable Actions 176
   3.4.9 Other Actions 182

3.5 Activity Diagrams
   3.5.1 Examination Topics 184
   3.5.2 Object Nodes 184
   3.5.3 Control Nodes 186
   3.5.4 Activity Partitions 190
   3.5.5 Structured Activity Nodes 194
   3.5.6 Conditional Nodes, Loop Nodes, and Sequence Nodes 196
   3.5.7 Exception Handling 201

3.6 Interaction Diagrams
   3.6.1 Examination Topics 203
   3.6.2 Interaction References 203
   3.6.3 Interaction Operations 207
   3.6.4 Connection Points (Gate) 218
   3.6.5 Communication Diagrams 220
3.6.6 Timing Diagrams 221
3.6.7 Interaction Overview Diagrams 221
3.7 State Diagrams 223
3.7.1 Examination Topics 223
3.7.2 State Machines 223
3.8 Deployment Diagrams 240
3.8.1 Examination Topics 241
3.8.2 Artifacts 241
3.8.3 Nodes 244
3.9 Profiles 247
3.9.1 Examination Topics 249
3.9.2 Stereotypes 249
3.9.3 Profiles 254
3.10 Standard Stereotypes 256
3.10.1 Examination Topics 257

APPENDIX 259
GLOSSARY 269
REFERENCES 283
INDEX 285
LIST OF METAMODELS

Figure 2.2   The metamodel of data types. 17
Figure 2.12  The basic metamodel concepts. 24
Figure 2.15  The metamodel for NamedElement. 27
Figure 2.16  The metamodel for namespaces. 28
Figure 2.18  The metamodel for typed elements. 29
Figure 2.20  The multiplicity metamodel. 30
Figure 2.23  The metamodel for value specifications. 32
Figure 2.25  The metamodel for constraints. 34
Figure 2.27  The metamodel for instance specifications. 35
Figure 2.32  The classifier metamodel. 37
Figure 2.33  The metamodel for features. 39
Figure 2.34  The metamodel for parameters. 39
Figure 2.35  The metamodel for operations. 41
Figure 2.39  The metamodel for properties. 45
Figure 2.51  The metamodel for associations. 53
Figure 2.53  The metamodel for Class. 55
Figure 2.56  The metamodel for generalizations. 57
Figure 2.57  The metamodel for redefinable elements. 57
Figure 2.64  The metamodel for packages. 63
Figure 2.72  The metamodel for dependencies. 67
Figure 2.79  The metamodel for interfaces. 71
Figure 2.81  The metamodel for basic behavior. 74
Figure 2.88  The metamodel for activities. 81
Figure 2.89  The metamodel for activity nodes. 82
Figure 2.93  The metamodel for activity edges. 85
Figure 2.96  The metamodel for control nodes. 88
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.103</td>
<td>The metamodel for pins.</td>
<td>91</td>
</tr>
<tr>
<td>2.113</td>
<td>The metamodel for interactions.</td>
<td>102</td>
</tr>
<tr>
<td>2.114</td>
<td>The metamodel for lifelines.</td>
<td>102</td>
</tr>
<tr>
<td>2.115</td>
<td>The metamodel for messages.</td>
<td>103</td>
</tr>
<tr>
<td>2.116</td>
<td>The metamodel for event occurrences.</td>
<td>103</td>
</tr>
<tr>
<td>2.122</td>
<td>The metamodel for use cases and actors.</td>
<td>108</td>
</tr>
<tr>
<td>2.126</td>
<td>The metamodel for use case relationships.</td>
<td>112</td>
</tr>
<tr>
<td>3.6</td>
<td>The metamodel for structured classifiers.</td>
<td>120</td>
</tr>
<tr>
<td>3.11</td>
<td>The metamodel for connectors.</td>
<td>124</td>
</tr>
<tr>
<td>3.18</td>
<td>The metamodel for ports.</td>
<td>127</td>
</tr>
<tr>
<td>3.19</td>
<td>The metamodel for class.</td>
<td>127</td>
</tr>
<tr>
<td>3.23</td>
<td>The metamodel for invocation actions and triggers.</td>
<td>129</td>
</tr>
<tr>
<td>3.24</td>
<td>The metamodel for variables.</td>
<td>129</td>
</tr>
<tr>
<td>3.28</td>
<td>The metamodel for collaborations.</td>
<td>132</td>
</tr>
<tr>
<td>3.29</td>
<td>The metamodel for collaboration uses.</td>
<td>132</td>
</tr>
<tr>
<td>3.39</td>
<td>The metamodel for components.</td>
<td>139</td>
</tr>
<tr>
<td>3.43</td>
<td>The metamodel for assembly connectors.</td>
<td>141</td>
</tr>
<tr>
<td>3.48</td>
<td>The metamodel for receptions.</td>
<td>145</td>
</tr>
<tr>
<td>3.50</td>
<td>The metamodel for events.</td>
<td>146</td>
</tr>
<tr>
<td>3.51</td>
<td>The metamodel for triggers.</td>
<td>146</td>
</tr>
<tr>
<td>3.53</td>
<td>The metamodel for SimpleTime.</td>
<td>148</td>
</tr>
<tr>
<td>3.57</td>
<td>The metamodel for actions.</td>
<td>151</td>
</tr>
<tr>
<td>3.59</td>
<td>The metamodel for call actions.</td>
<td>153</td>
</tr>
<tr>
<td>3.60</td>
<td>The metamodel for call behavior actions.</td>
<td>153</td>
</tr>
<tr>
<td>3.63</td>
<td>The metamodel for call operation actions.</td>
<td>155</td>
</tr>
<tr>
<td>3.65</td>
<td>The metamodel for send signal actions.</td>
<td>156</td>
</tr>
<tr>
<td>3.67</td>
<td>The metamodel for broadcast signal actions.</td>
<td>156</td>
</tr>
<tr>
<td>3.68</td>
<td>The metamodel for send object actions.</td>
<td>157</td>
</tr>
<tr>
<td>3.69</td>
<td>The metamodel for invocation actions from the BasicActions package.</td>
<td>157</td>
</tr>
<tr>
<td>3.70</td>
<td>The metamodel for invocation actions from the IntermediateActions package.</td>
<td>158</td>
</tr>
<tr>
<td>3.71</td>
<td>The metamodel for opaque actions.</td>
<td>159</td>
</tr>
<tr>
<td>3.73</td>
<td>The metamodel for create object actions.</td>
<td>161</td>
</tr>
<tr>
<td>3.74</td>
<td>The metamodel for destroy object actions.</td>
<td>161</td>
</tr>
<tr>
<td>3.75</td>
<td>The metamodel for test identity actions.</td>
<td>162</td>
</tr>
<tr>
<td>3.76</td>
<td>The metamodel for read self actions.</td>
<td>162</td>
</tr>
</tbody>
</table>
LIST OF METAMODELS

Figure 3.77 The metamodel for object actions. 163
Figure 3.79 The metamodel for structural feature actions. 165
Figure 3.80 The metamodel for read structural feature actions. 166
Figure 3.81 The metamodel for structural features. 166
Figure 3.82 The metamodel for clear structural feature actions. 167
Figure 3.83 The metamodel for write structural feature actions. 167
Figure 3.84 The metamodel for add structural feature value actions. 168
Figure 3.85 The metamodel for remove structural feature value actions. 168
Figure 3.86 The metamodel for structural feature actions. 169
Figure 3.89 The metamodel for link actions with LinkEndData elements. 172
Figure 3.90 The metamodel for read link actions. 172
Figure 3.91 The metamodel for write link actions. 172
Figure 3.92 The metamodel for CreateLinkAction and LinkEndCreationData. 173
Figure 3.93 The metamodel for destroy link actions. 174
Figure 3.94 The metamodel for clear association actions. 175
Figure 3.95 The metamodel for link identifications. 175
Figure 3.96 The metamodel for link actions and clear association actions. 176
Figure 3.98 The metamodel for variable actions. 178
Figure 3.99 The metamodel for read variable actions. 178
Figure 3.100 The metamodel for write variable actions. 179
Figure 3.101 The metamodel for add variable value actions. 179
Figure 3.102 The metamodel for remove variable value actions. 180
Figure 3.103 The metamodel for clear variable actions. 180
Figure 3.104 The metamodel for variable actions. 181
Figure 3.106 The metamodel for miscellaneous actions. 182
Figure 3.110 The metamodel for central buffer nodes. 185
Figure 3.117 The metamodel for control nodes. 190
Figure 3.122 The metamodel for activity groups. 193
Figure 3.123 The metamodel for activity partitions. 194
Figure 3.126 The metamodel for structured activity nodes. 196
Figure 3.130 The metamodel for conditional nodes. 199
Figure 3.131 The metamodel for loop nodes. 200
Figure 3.132 The metamodel for sequence nodes. 200
Figure 3.135 The metamodel for exception handlers. 202
<table>
<thead>
<tr>
<th>Figure</th>
<th>Metamodel Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.141</td>
<td>The metamodel for iteration references (InteractionUse).</td>
<td>207</td>
</tr>
<tr>
<td>3.159</td>
<td>The metamodel for combined fragments.</td>
<td>217</td>
</tr>
<tr>
<td>3.162</td>
<td>The metamodel for connection points (gates).</td>
<td>219</td>
</tr>
<tr>
<td>3.186</td>
<td>The metamodel for state machines.</td>
<td>238</td>
</tr>
<tr>
<td>3.187</td>
<td>The metamodel for redefining state machines.</td>
<td>239</td>
</tr>
<tr>
<td>3.188</td>
<td>The metamodel for time events.</td>
<td>240</td>
</tr>
<tr>
<td>3.193</td>
<td>The metamodel for artifacts.</td>
<td>244</td>
</tr>
<tr>
<td>3.196</td>
<td>The metamodel for nodes.</td>
<td>247</td>
</tr>
<tr>
<td>3.204</td>
<td>The metamodel for stereotypes.</td>
<td>253</td>
</tr>
<tr>
<td>3.208</td>
<td>The metamodel for profiles.</td>
<td>256</td>
</tr>
</tbody>
</table>
The Unified Modeling Language (UML) is one of the biggest success stories in the information technology (IT) industry. Once used solely as a way to sketch the possible requirements or operations of an IT system, UML is now used in a variety of ways by people with very different backgrounds; for example, by

- business planners, as a language to specify the planned operation of a business process, perhaps in concert with a business process language such as the Business Process Modeling Notation (BPMN).
- consumer device engineers, as a way to outline the requirements for an embedded device and the way it is to be used by an end user.
- software architects, as an overall design for a major stand-alone software product.
- IT professionals, as an agreed-on set of models to integrate existing applications.
- database professionals, to manage the integration of databases into a data warehouse, perhaps in concert with a data warehousing language such as the Common Warehouse Metamodel (CWM).
- software developers, as a way to develop systems that are flexible in the face of changing business requirements and implementation infrastructure.

These are just a few examples of how UML is used; there are many more. Since the creation of the UML standard in 1997, the language has even been extended into a full systems engineering language (SysML) to solve the integrated development problems of systems development professionals.

Clearly, there’s some utility and expressibility in a language that has many dozens of implementations, both open-source and closed, and that has found its way into every major (and most not-so-major) integrated development environment. That’s not enough for a language to be successful, however; an entire ecosystem is needed to make a standard like UML successful. One needs to be able
to find, train, and evaluate modeling professionals to know that one’s projects will actually be carried out on time, on budget, and within other constraints. This means that change must be instilled in the IT organization.

It’s wonderful that the UML standard and related standards (e.g., the Meta Object Facility, MOF, and XML Metadata Interchange, XMI) allow tools to share models and diagrams; this means that developers can choose and easily implement their own tool chains, integrating the right reverse-engineering tool with the right model display tool with the right code generator. But all this technical infrastructure is pointless unless the human dimension is covered too. Development teams that rely on UML, from the business analysts to the architects to the systems analysts to the programmers, must learn to speak the lingo.

The OMG’s standards specifications (found at http://www.omg.org/) define the norms, but more is needed to ensure team builders that they are hiring, building, and delivering quality to their internal and external customers. In 2003, the Object Management Group teamed up with the UML Technology Institute to deal with this issue; they developed the OMG-Certified UML Professional (OCUP) program. The OCUP program defines UML expertise at three professional levels (Fundamental, Intermediate, and Advanced) and tests ability against those measures in a fair, unbiased, worldwide testing program. The OCUP UML Intermediate test taken in Bangalore is equivalent to the OCUP UML Intermediate test taken in Paris; passing that test clearly marks the test-taker as a leader in the field—someone who understands the need for expertise in modeling and who has taken the time and effort to improve his or her development expertise.

This ground-breaking book is focused on preparing you to pass the OCUP Fundamental and Intermediate tests. This allows you to show yourself, your peers, and your employer that you understand the value of modeling to the creation of quality systems that are delivered faster, better, and cheaper. The authors of this tome have proved their mettle not only by passing both tests themselves but also by teaching UML expertise in classroom and conference settings that has led to remarkably high OCUP test-passing rates.

You have taken the right first step to prepare yourself to show what you know; and this book will help you pass the tests and earn the certification!

Richard Mark Soley, Ph.D.
Chairman and Chief Executive Officer
Object Management Group, Inc.
FOREWORD

The Unified Modeling Language (UML) can be used like any other natural language. It is spoken in several ways.

Some people use UML slang to sketch a model to communicate with colleagues or to store some notes. Others speak a UML dialect. They use some vocabulary that is not formally defined, but which is easy to use, pragmatic, and a lot of people understand the right thing.

Most software developers speak pragmatic UML. Not perfect—some special words are rarely or never used. Sometimes the blur of their models leads to misunderstandings. However, generally speaking, it works well.

Some people are real UML virtuosos. So to speak the Goethe’s, Schiller’s, and Lessing’s of the UML. They know every detail of the Unified Modeling Language and know how to use it.

Today it is a must for a software developer to communicate with UML. Surely it is more important to know how to use UML instead of knowing each vocabulary or every grammar rule of the UML. However, just like in natural languages, you can battle your way with basic knowledge of language formalism. Although the content is correct and good, a missing knowledge of language formalism can lead to misunderstandings, wrong or laborious models.

If you would like to use UML professionally, you should know the language well. The three UML certification levels define the level of language knowledge. The Fundamental level makes sense for all UML users. This book prepares you for it and the Intermediate level. That second level is important for every developer who is more than a simple UML user, such as architects, instructors, coaches, tool vendors, senior developers, MDA developers, and so on.

For these people, UML diagrams are not simple pictures for communication and documentation purposes. A UML diagram is a view of a formally defined model. That is the content of the Intermediate certification. It is only this formal foundation that turns the Unified Modeling Language into something powerful. In contrast to natural languages like Swedish or German, UML is understandable.
for computers. And in contrast to programming languages, such as C++, Java, C#, and Co, UML is easy to read by humans: A picture is much more expressive than a thousand words.

Developers who know this level can clearly, concisely, and unambiguously communicate sophisticated and complex facts. Their models are less mistakable and have less margin for interpretation. Finally, they can work more effectively and efficiently.

This book focuses on preparing you to pass the OCUP Fundamental and Intermediate examinations and shows you, your peers, and your employer that you understand the value of modeling for the creation of quality software that can be delivered better, faster, and cheaper.

The authors have proved their mettle not only by passing the tests themselves, but by teaching UML expertise in classroom settings that have led to remarkably high test-passing rates, just weeks after the OCUP Program became available worldwide. You have taken the right first step to prepare yourself to show what you know—and this book is the right tool to get you there!

_Ivar Jacobson_
PREFACE

WHY THIS BOOK?

NOT AN INTRODUCTION TO UML

It is a lot easier to work your way through this book than through the UML specifications. This book is not meant to give you an introduction to UML. Rather, working through this book requires you to have fundamental UML knowledge. There are other books to familiarize yourself with the UML basics (see References in the appendix).

This book systematically covers the topics relevant to the test. It is designed to prepare candidates well for the Fundamental and Intermediate UML certification exams. It prepares you exactly but exclusively for these tests. The Advanced level will be addressed in the second edition of the German book. (This translation is based on the first edition.)

COVERAGE MAP

The OMG publishes a coverage map (see http://www.omg.org/uml-certification/UML_Exams_Coverage_Map.pdf), which indicates the topics and areas the test covers.

We, the authors of this book, have experience with the test. In fact, Tim has served as a beta tester in the OMG test program. In Germany, oose GmbH was the first company to offer preparatory courses. Within one month of starting the Fundamental test program in Germany, 60 candidates took the test, and only one failed. Before the test, we interviewed the candidates to get an idea of their experience and knowledge. After the test, we asked them what they had found particularly difficult and how they would improve their preparation.

All these findings, and more, have found their way into this book. Note, however, that we don’t know all the questions that can come up in the test. And
even if we knew them, we wouldn’t pass them on because it would make the test
and the certificate worthless.

This is one of the reasons why it won’t be sufficient to merely consume
this book, and learning everything by heart would be just as wrong. This book,
however, will help you to better understand the topics that the test addresses so
that you will know the material and will succeed on the test regardless of the way
the questions are asked.

PREREQUISITES

This book is intended to specifically prepare candidates to take the UML certifi-
cation exams at the Fundamental and Intermediate levels. As such, it requires the
knowledge of certain general basics of UML and object orientation.

Just as the Intermediate section of the test builds on the Fundamental section,
the Intermediate section of this book assumes that you are familiar with the
material from the Fundamental section.

WHAT MOTIVATED THIS BOOK?

Since the launch of the UML certification program in early 2004, we have gained
extensive experience with the tests and proper preparations; we have helped several
hundred candidates pass the test. Henceforth, the demand for our test preparations
for the Fundamental and Intermediate exams has continually increased.

Our candidates’ success and the ever-increasing demand encouraged us to write
this book based on our sound training documentation and the feedback from
people who have used our materials to prepare for the UML certification tests.
This book includes the topics recommended by OMG for the Fundamental and
Intermediate levels and additionally considers important practical experience.

It is our desire to make test preparation as easy as possible for you. Who, after
all, likes to cram for tests? Exactly. So we left out lengthy explanations of material
that would not be required or useful in preparing yourself for the tests. Finding
out, compiling, and representing the relevant material in an easy-to-understand
way were our goals.

HOW TO USE THIS BOOK

We present the material in this book in an arrangement that mimics the OCUP
examination. We address the topics and subtopics that appear on the tests and
do so in a certain scheme: Definition, Notation and Semantics, Metamodel, and
Checklist. The checklist at the end of each topic presents hands-on test-type
questions for you to answer, to help you make sure that you have understood the section’s material.

New or important terms introduced in the text are highlighted in italics. In the margins are numbered references that match the number of the questions in the checklists; if you find any question difficult, you can use those numbers to find the section that you need to review. In the text you will see small round symbols with a combined letter-number name—for example, M1. These symbols refer to the metamodel in the figure that’s being discussed.

Chapter 2, OCUP Fundamental, represents the preparatory material for the Fundamental test and Chapter 3, OCUP Intermediate, discusses and explains the test material for the Intermediate exam. Each one contains many figures and examples.

Although Chapters 2 and 3 represent self-contained preparatory material for each of the two test levels covered in this book, Chapter 3 builds on the Fundamental material dealt with in Chapter 2. Where material in Chapter 3 refers to the Fundamental test, you will find marginal comments in emphasizing this fact.

Since every discipline has its particular lingo, we have prepared a glossary of terms specific to UML, which you will find in the appendix. These terms and definitions are divided into a Fundamental and an Intermediate section and correspond to the original definitions from the UML specification. The appendix also contains a list of typical Fundamental exam questions (not the real ones, though), and the correct answers at the end of the section. Finally, the appendix includes a list of references for further reading.

ACKNOWLEDGMENTS

We would like to acknowledge the people who have made this book possible. In particular, they include Christa Preisendanz, Tim Cox, Marilyn Rash, our colleagues from oose Innovative Informatik GmbH, and our customers who did the UML certification.

We thank Bran Selic, Conrad Bock, and all other members of the UML RTF for their great work for the UML.

We also wish to thank Richard M. Soley and Ivar Jacobson for their forewords.
UML 2 Certification Guide