Preface

> Mankind, under the grace of God, hungers for spiritual peace, esthetic achievements, family security, justice, and liberty, none directly satisfied by industrial productivity. But productivity allows the sharing of the plentiful rather than fighting over scarcity; it provides time for spiritual, esthetic, and family matters. It allows society to delegate special skills to institutions of religion, justice, and the preservation of liberty.
>
> HARLAN MILLS
> *DPMA and Human Productivity*

As computer professionals, we strive to build systems that work and are useful; as software engineers, we are faced with the task of creating complex systems in the presence of constrained computing and human resources. Object-oriented (OO) technology has evolved as a means of managing the complexity inherent in many different kinds of systems. The object model has proven to be a very powerful and unifying concept.

Changes to the Second Edition

Since the publication of the second edition of *Object-Oriented Analysis and Design with Applications*, we have seen major technological advances. This list includes some highlights, among many others.

- High-bandwidth, wireless connectivity to the Internet is widely available.
- Nanotechnology has emerged and has started to provide valuable products.

- Our robots are cruising the surface of Mars.
- Computer-generated special effects have enabled the film industry to recreate any world imaginable with complete realism.
- Personal hovercraft are available.
- Mobile phones have become pervasive to the point of being disposable.
- We have mapped the human genome.
- Object-oriented technology has become well established in the mainstream of industrial-strength software development.

We have encountered the use of the object-oriented paradigm throughout the world. However, we still encounter many people who have not yet adopted the object paradigm of development. For both of these groups, this revision of this book holds much value.

For the person new to object-oriented analysis and design (OOAD), this book gives the following information:

- The conceptual underpinnings of and evolutionary perspective on object orientation
- Examples of how OOAD can be applied across the system development lifecycle
- An introduction to the standard notation used in system and software development, the Unified Modeling Language (UML 2.0)

For the experienced OOAD practitioner, the content herein provides value from a different perspective.

- UML 2.0 is still new to even experienced practitioners. Here you will see the key changes in the notation.
- More focus on modeling is provided, per feedback received about the previous edition.
- You can gain a great appreciation for "why things are the way they are" in the object-oriented world, from the Concepts section of the book. Many people may never have been exposed to this information on the evolution of the OO concepts themselves. Even if you have been, you may not have grasped its significance when you were first learning the OO paradigm.

There are four major differences between this edition and the previous publication.

1. UML 2.0 has been officially approved. Chapter 5, Notation, will introduce UML 2.0. To enhance the reader's understanding of this notation, we explicitly distinguish between its fundamental and advanced elements.

2. This edition introduces some new domains and contexts in the Applications chapters. For example, the application domains range broadly across various levels of abstraction from high-level systems architecture to the design details of a Web-based system.
3. When the previous edition was published, C++ was relatively new, as was the very concept of OO programming. Readers tell us that this emphasis is no longer a primary concern. There is an abundance of OO programming and technique books and training available, not to mention additional programming languages designed for OO development. Therefore, most of the coding discussions have been removed.
4. Finally, in response to requests received from readers, this edition focuses much more on the modeling aspects of OOAD. The Applications section will show you how to use the UML, with each chapter emphasizing one phase of the overall development lifecycle.

Goals

This book provides practical guidance on the analysis and design of object-oriented systems. Its specific goals are the following:

- To provide a sound understanding of the fundamental concepts and historical evolution of the object model
- To facilitate a mastery of the notation and process of object-oriented analysis and design
- To teach the realistic application of object-oriented analysis and design within a variety of problem domains

The concepts presented all stand on a solid theoretical foundation, but this is primarily a pragmatic book that addresses the practical needs and concerns of software engineering practitioners, from the architect to the software developer.

Audience

This book is written for the computer professional as well as for the student.

- For the practicing systems and software developer, we show you how to effectively use object-oriented technology to solve real problems.
- In your role as an analyst or architect, we offer you a path from requirements to implementation, using object-oriented analysis and design. We

develop your ability to distinguish "good" object-oriented architectures from "bad" ones and to trade off alternate designs when the perversity of the real world intrudes. Perhaps most important, we offer you fresh approaches to reasoning about complex systems.

- For the program manager, we provide insight on topics such as allocation of resources of a team of developers, software quality, metrics, and management of the risks associated with complex software systems.
- For the student, we provide the instruction necessary for you to begin acquiring several important skills in the science and art of developing complex systems.

This book is also suitable for use in undergraduate and graduate courses as well as in professional seminars and individual study. Because it deals primarily with a method of software development, it is most appropriate for courses in software engineering and as a supplement to courses involving specific object-oriented programming languages.

Structure

The book is divided into three major sections—Concepts, Method, and Applications—with considerable supplemental material woven throughout.

Concepts

Section I examines the inherent complexity of software and the ways in which complexity manifests itself. We present the object model as a means of helping us manage this complexity. In detail, we examine the fundamental elements of the object model such as: abstraction, encapsulation, modularity, and hierarchy. We address basic questions such as "What is a class?" and "What is an object?" Because the identification of meaningful classes and objects is the key task in object-oriented development, we spend considerable time studying the nature of classification. In particular, we examine approaches to classification in other disciplines, such as biology, linguistics, and psychology, and then apply these lessons to the problem of discovering classes and objects in software systems.

Method

Section II presents a method for the development of complex systems based on the object model. We first present a graphic notation (i.e., the UML) for object-

oriented analysis and design, followed by a generic process framework. We also examine the pragmatics of object-oriented development—in particular, its place in the software development lifecycle and its implications for project management.

Applications

Section III offers a collection of five nontrivial examples encompassing a diverse selection of problem domains: system architecture, control systems, cryptanalysis, data acquisition, and Web development. We have chosen these particular problem domains because they are representative of the kinds of complex problems faced by the practicing software engineer. It is easy to show how certain principles apply to simple problems, but because our focus is on building useful systems for the real world, we are more interested in showing how the object model scales up to complex applications. The development of software systems is rarely amenable to cookbook approaches; therefore, we emphasize the incremental development of applications, guided by a number of sound principles and well-formed models.

Supplemental Material

A considerable amount of supplemental material is woven throughout the book. Most chapters have sidebars that provide information on related topics. We include an appendix on object-oriented programming languages that summarizes the features of a few common languages. We also provide a glossary of common terms and an extensive classified bibliography that lists references to source material on the object model.

A Note about Tools

Readers always ask about the tools used to create the diagrams in the book. Primarily, we have used two fine tools for the diagrams: IBM Rational Software Architect and Sparx Systems Enterprise Architect. Why not use just one? The reality of the marketplace is that no tool does everything. The longer you do OOAD, you will eventually find some atypical “corner case” that no tool supports. (In that case, you may have to resort to basic drawing tools to show what you want.) But don’t let those rare instances stop you from using robust OOAD tools such as those we mentioned.

Using This Book

This book may be read from cover to cover or it may be used in less structured ways. If you are seeking a deep understanding of the underlying concepts of the object model or the motivation for the principles of object-oriented development, you should start with Chapter 1 and continue forward in order. If you are primarily interested in learning the details of the notation and process of object-oriented analysis and design, start with Chapters 5 and 6; Chapter 7 is especially useful to managers of projects using this method. If you are most interested in the practical application of object-oriented technology to specific problems, select any or all of Chapters 8 through 12.