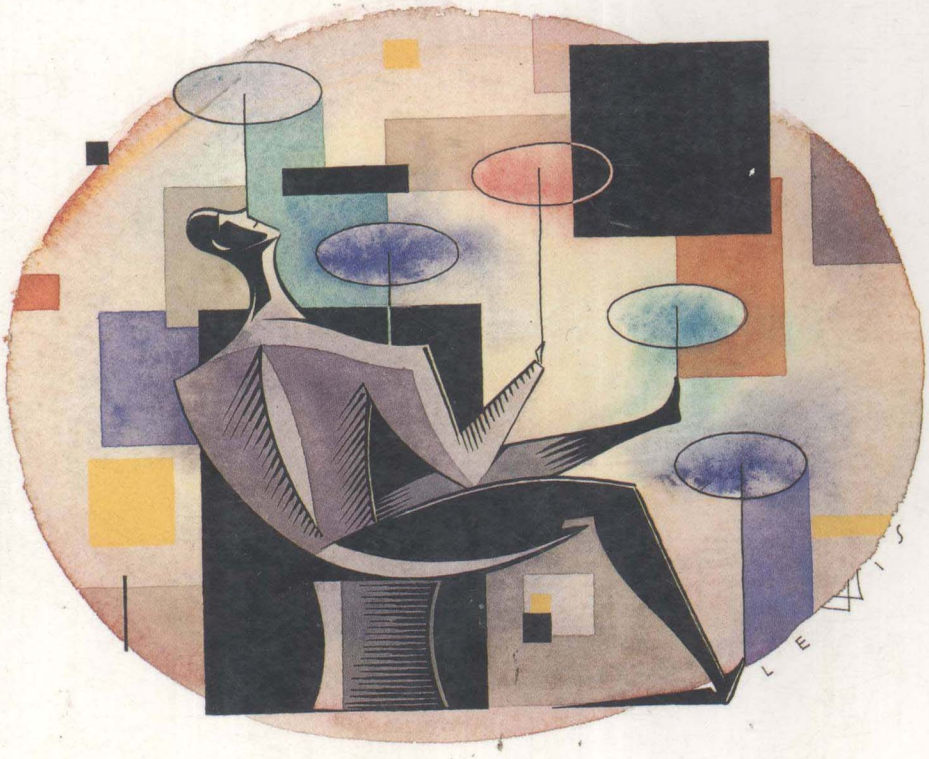
OpenStep™ for Enterprises



Object-Oriented Development for Windows and UNIX

Objectory and Booch Methods

Distributed Objects for Client/Server Computing

Nancy Craighill

Includes 3.5" Disk

OpenStep™ for Enterprises

Nancy Craighill

Wiley Computer Publishing

John Wiley & Sons, Inc.
New York • Chichester • Brisbane • Toronto • Singapore • Weinheim

Publisher: Katherine Schowalter
Editor: Marjorie Spencer
Managing Editor: Frank Grazioli
Electronic Products Associate Editor: Mike Green
Text Design & Composition: Benchmark Productions, Inc.

All diagrams were created with DIAGRAM!2 by Lighthouse Design, Ltd.

Library of Congress Cataloging-in-Publication Data:
Craighill, Nancy, 1962-
OpenStep for enterprises / Nancy Craighill
p.cm
Includes bibliographical references (p.).
ISBN 0-471-30859-5 (paper : alk. paper)
1. Applicaton software--Development. 2. OpenStep. I. Title
QA 76.76.A65C73 1996
005.2--dc20 96-17988
CIP

ISBN 0-471-30859-5
Printed in the United States of America
10 9 8 7 6 5 4 3 2 1

About the Author

Nancy Craighill is currently an independent software consultant and sometimes technical writer. At SRI International she used Objective-C and other Stepstone products to develop command and control systems for the U.S. Army. At the Stepstone Corporation she developed a 2D graphics class library, and authored the Objective-C column for the Journal of Object-Oriented Programming (JOOP). At Sony Electronics, Inc., she prototyped two high-end video editing systems. The first system was implemented in C++, X Windows, and Motif, and the second was implemented in NEXTSTEP. Recently she freelanced as a technical writer for NeXT Software, and wrote portions of NeXT's *OpenStep Reference Manual*. She also contributed chapters to other books: *Computer Graphics through Object-Oriented Programming* (published by John Wiley & Sons), and *Applications in Object-Oriented Programming* (published by Addison-Wesley).

To Sean and Lina

Acknowledgments

Thanks to NeXT Software for creating OpenStep, something worth writing about. NeXT staff answered many questions, especially Blaine Garst and the whole Technical Publications Department. Ron Hayden provided the opportunity for me to work with his talented staff and supported this book project in ways too numerous to mention. However, special thanks goes to Greg Wilson who lent me his equipment, schlepped my computer around, and even jumped my car in the pouring rain.

Thanks to Ivar Jacobson for creating the Jacobson Method (OOSE) and those indispensable interaction diagrams. Other Rational Software staff made individual contributions. Sten Jacobson and Haakan Dyrhage reviewed the analysis and design chapters, and Doug Earl provided expertise on the Booch Notation.

A special mention goes to my many reviewers. Vicki de Mey, Martin Fong, Bill Hunt, Randy Knolle, and Hemang Shukla provided thoughtful reviews and criticisms of the first draft. Dave Ciemiewicz, Henry McGilton, and Robert Nielson made excellent suggestions on the final draft. Henry McGilton also contributed immensely to the book design. Brad Cox and Kent Beck contributed to the philosophical direction of the book.

Thanks to John Wiley & Sons for their patience as circumstances beyond our control stretched this project from one to three years long. At first the topic was NEXTSTEP; then later it became OpenStep. Since the manuscript depended highly on the software, it became impossible to predict when the book might be published. Ironically, the release date of OpenStep for Windows coincided with the birth of my second child.

I couldn't have completed this project without the support from my family, especially my husband Earl for his never ending faith and encouragement. And last but certainly not least, my father Ernst Knolle and other family members for providing many months of day care.

Figures

Tables

Preface

Goals

OpenStep is the premier object-oriented (OO)† development environment now available on Microsoft Windows, Sun Solaris and NeXT Mach platforms. OpenStep is not just a product, as in *OpenStep for Windows* sold by NeXT, but a specification of reusable classes called *frameworks*. Since it is an open standard, any software manufacture can provide an implementation of OpenStep. OpenStep is not new; it is proven technology that evolved from NeXT's original NEXTSTEP programming environment for Mach which has shipped since 1989.

For software developers OpenStep provides:

- *Objective-C*, a powerful hybrid OO language,
- *Interface Builder* for building user interfaces,
- *Project Builder* for organizing files, compiling and debugging code,
- *Foundation Kit* and *Application Kit*, several frameworks for quickly constructing those custom applications.

Support for *Distributed Objects*, the ability to send messages between processes, is implemented as an extension to the Objective-C language and through Foundation Kit classes. NeXT has ported its Distributed Objects to other platforms, such as HP-UX, and now supports interoperability with Microsoft OLE/COM objects and OMG CORBA implementations. Thus, your OpenStep objects can communicate with objects across heterogeneous platforms. In addition, an add-on product from NeXT, called *Enterprise Objects Framework (EOF)*, provides object persistency using traditional databases. All of these tools are the key to developing exciting new applications, such as hypermedia, groupware, and authoring tools.

† To save a couple of trees, "object-oriented" is abbreviated as "OO" throughout this book.

However, simply learning the OpenStep development tools does not ensure success with OO technology. Yes, you can quickly build custom applications, but without understanding the process of OO development and using proven methods of approach, the nifty user interface you create may just be a nice wrapper around the same old "spaghetti" code with the same old maintenance problems.

Perhaps the first application you develop using OpenStep is successful, but now that it is in use by real customers you are overwhelmed with problem reports and requests for enhancements and wondering if starting from scratch would be easier. Or, your first application was received well within the company, and now your organization wants to adopt the technology enterprise-wide, but you're having difficulty scaling up your custom application to meet these new demands.

At this point, you might blame the environment and technology (it hasn't lived up to your expectations) without realizing that the environment is only one ingredient for successful OO development. Using OpenStep for serious software development on a large scale requires more understanding of the OO development process.

The goal of this book is to help you succeed in using OpenStep, not just for one application, but enterprise-wide, by building suites of interoperable applications. OO development on a large scale requires not only new tools, but a new philosophy about software development, new management style, and an underlying architecture that supports interoperability. Specifically, the goals of this book are:

- Introduce the OO Software Development Life Cycle and different management styles.
- Teach OO analysis and design methods, notations, and techniques.
- Teach OpenStep development tools and frameworks.
- Provide deeper understanding and appreciation of Objective-C.
- Demonstrate the power of Distributed Objects by focusing on problems faced when designing client-server applications.

This book contains real design and code examples by developing a theme application, called *Cards*, throughout. The application is an analysis tool, a computerization of CRC or modelling cards, that has aspects of both hypermedia and groupware. The design and implementation of Cards is non-trivial, making it ideal for teaching advanced features of OpenStep. Since Cards is an analysis tool, you can also use it when

developing your own applications. The complete source code is provided on the enclosed disk so you can extend it to meet your needs.

By covering the entire OO development process and advanced features of OpenStep, this book gives you all the necessary skills to succeed with reuse in your organization and build interoperable systems suitable for the new information age.

Audience

This book is suitable for computer professionals, program managers, and students. Specific sections of the book address the process of OO development and new management techniques. It contains in-depth design and programming examples using OpenStep. Chapter 6—*Implementation*, written in tutorial style, and the source code on the enclosed disk could be used in a software engineering course.

Organization

Chapter 1—*Introduction* provides "The Big Picture" of what is taking place in the software industry today and where it is headed. It also explains why OO technology in general, and OpenStep in particular, is the fastest vehicle to developing next generation software.

Chapters immediately following Chapter 1 explain the process of OO development including the OO life cycle and iteration strategies, teach the Jacobson OO analysis and design method, and apply the Booch notation. Chapter 5—*Design* introduces OpenStep classes and reusable designs called mechanisms. Chapter 6—*Implementation* teaches OpenStep development tools, frameworks, and more mechanisms. You can implement your own version of the example application while reading this chapter.

The final chapter, Chapter 7—*Distributed Objects*, ties it all together. It's not only a goal for large enterprises to use Distributed Objects. Distributed applications, including client-server applications, require more attention to design—you can't just hack a distributed application together and expect it to interoperate. OO methods and notations can

really help to understand the complexity of distributed applications, and applying these techniques produces better results.

The appendices also contain important information:

- Appendix A—*A Laboratory For Teaching OO Thinking* contains a reprint of the Beck and Cunningham paper that is used as the requirements specification for the Cards application.
- Appendix B—*Objective-C* contains a brief introduction to the language and syntax while also explaining how messaging in Objective-C works.
- Appendix C—*Booch Lite* contains a description of the Booch Notation with adaptations for Objective-C and Distributed Objects.
- Appendix D—*Class Specifications* contains specifications for principle Cards classes. Use this appendix as a reference when examining the source code and extending the application.
- Appendix E—*Further Reading* contains references to additional information.

Using the Book

Chapter 1 provides the background and rationale for why this technology is so important. Chapter 2 may be skipped if you are already versed in OO programming concepts. Chapter 3, Chapter 4, and Chapter 5, are best read in succession and provide the foundation for the examples in the rest of the book. Chapter 6 covers the single user, single process version of Cards. Chapter 7 adds support for groupware, and assumes familiarity with the Cards design, Objective-C and Booch notation, and therefore should be read last. Read Appendix B if you are unfamiliar with the language or want to understand how it works. Be sure to read Appendix C if you are unfamiliar with the Booch Notation before reading Chapter 5.

Conventions

This book is after all a programming book and contains many references to programming "things," often OpenStep specific. Therefore it is worth mentioning some conventions used in this book.

Bold denotes words or characters that are to be taken literally. Specifically, method names, instance variables, other local and global variables, and types will appear in **bold**. For example, Objective-C method names such as **setTitle:**, and types such as **id** and **int** appear in **bold**. On the other hand, classes, protocols, categories, notifications and exceptions will not be emphasized but always appear capitalized, as is the OpenStep convention.

If unspecified, a method name is always assumed to be an instance method, otherwise the statement will be qualified as in "the **init** class method."

To improve the legibility of this book, words are sometimes coined from class and method names. For example, the term *views* refers to instances of NSView and the phrase "an object is released" implies that an object was sent the **release** message.

Contents