

SOFTWARE DEVELOPMENT USING EIFFEL

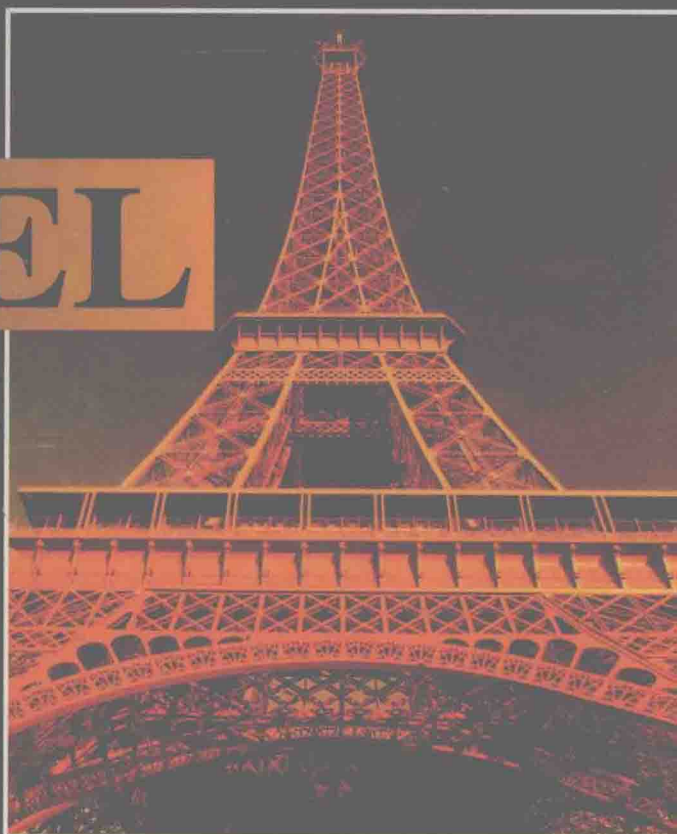
THERE CAN BE LIFE
OTHER THAN C++

Presents the
latest version
of Eiffel

Shows
Eiffel in
action

Demonstrates
Booch '94
method and
notation

Compares
Eiffel
with C++



RICHARD WIENER

THE
OBJECT-ORIENTED
SERIES

SOFTWARE DEVELOPMENT USING EIFFEL: THERE CAN BE LIFE OTHER THAN C++

RICHARD WIENER



Prentice Hall PTR, Englewood Cliffs, New Jersey 07632

Library of Congress Cataloging-in-Publication Data

Wiener, Richard, 1941–

Software development using Eiffel : there can be life other than

C++ / Richard Wiener.

p. cm. — (Prentice Hall object-oriented series)

Includes index.

ISBN 0-13-100686-X (hardcover)

1. Computer software—Development. 2. Eiffel (Computer program language) I. Title. II. Series.

QA76.76.D47W52 1995

94-37783

005.13'3—dc20

CIP

Editorial/Production Management: *Digital Communications Services*

Cover Design: *Miguel Ortiz*

Manufacturing: *Alexis R. Heydt*

Acquisitions Editor: *Paul W. Becker*



© 1995 Prentice Hall PTR

Prentice-Hall, Inc.

A Simon & Schuster Company

Englewood Cliffs, New Jersey 07632

The publisher offers discounts on this book when ordered in bulk quantities. For more information, contact Corporate Sales Department, Prentice Hall PTR, 113 Sylvan Avenue, Englewood Cliffs, NJ 07632. Phone: 800-382-3419; Fax: 201-592-2249; e-mail: dan_rush@prenhall.com

All rights reserved. No part of this book may be reproduced, in any form or by any means, without permission in writing from the publisher.

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN 0-13-100686-X

Prentice-Hall International (UK) Limited, *London*

Prentice-Hall of Australia Pty. Limited, *Sydney*

Prentice-Hall Canada Inc., *Toronto*

Prentice-Hall Hispanoamericana, S.A., *Mexico*

Prentice-Hall of India Private Limited, *New Delhi*

Prentice-Hall of Japan, Inc., *Tokyo*

Simon & Schuster Asia Pte. Ltd., *Singapore*

Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*

SOFTWARE DEVELOPMENT USING EIFFEL:

THERE CAN BE LIFE OTHER THAN C++

PRENTICE HALL
OBJECT-ORIENTED
SERIES

D. COLEMAN, P. ARNOLD, S. BODOFF,
C. DOLLIN, H. GILCHRIST, F. HAYES
AND P. JEREMAES

*Object-Oriented Development:
The Fusion Method*

S. COOK AND J. DANIELS
Designing Object Systems

B. HENDERSON-SELLERS
A Book of Object-Oriented Knowledge

B. HENDERSON-SELLERS
Book Two of Object-Oriented Knowledge

H. KILOV AND J. ROSS
Information Modelling

P. KRIEF
Prototyping with Objects

K. LANO AND H. HAUGHTON
Object-Oriented Specification Case Studies

J. LINDSKOV KNUDSEN, M. LÖFGREN,
O. LEHRMANN MADSEN AND B. MAGNUSSEN
*Object-Oriented Environments:
The Mjølner Approach*

M. LORENZ
*Object-Oriented Software Development:
A Practical Guide*

M. LORENZ
Object-Oriented Software Metrics

D. MANDRIOLI AND B. MEYER (eds)
*Advances in Object-Oriented
Software Engineering*

B. MEYER
An Object-Oriented Environment

B. MEYER
Eiffel: The Language

B. MEYER
Reusable Software

B. MEYER AND J. M. NERSON
Object-Oriented Applications

POMBERGER AND BALSCHKE
*An Object-Oriented Approach in
Software Engineering*

P. J. ROBINSON
Hierarchical Object-Oriented Design

R. SWITZER
Eiffel: An Introduction

K. WALDEN AND J. M. NERSON
Seamless Object-Oriented Software Architecture

R. WIENER
Software Development Using Eiffel

*This book is dedicated
to my best friend, my wife
and my love, Hanne.*

Preface

THE BOOK

*T*his book is aimed at computer scientists, software development professionals, educators and students, programmers, and in particular C++ programmers, who wish to examine a compelling alternative to the C++ programming language for implementing object-oriented software systems. This alternative is Eiffel. This book is also about object-oriented analysis and design using the Booch '94 method with Eiffel implementation. The reader is assumed to have prior experience with object-oriented programming, most likely using C++ or Smalltalk, and at least basic familiarity with the concepts of object-oriented programming.

Part 1 of the book, Chapters 1 through 7, presents the major features of the Eiffel language from a programmer's perspective. These chapters contain many small but complete examples of Eiffel coding. These are used to illustrate the typical use of various language features. Part 2 of the book, Chapters 8 through 11, shows Eiffel in action. These chapters present case studies that, in addition to Eiffel implementation, show the use of the Booch '94 method of object analysis and design.

The standard reference book for the Eiffel programming language is *Eiffel: The Language*, written by Bertrand Meyer (Prentice Hall, 1992), the creator of Eiffel. This present book is meant to complement this reference, not compete with it. It is recommended that the reader also acquire this standard reference about Eiffel since there is no attempt in this book to duplicate the rigorous and fine-grained details concerning some of the "legal" issues related to the Eiffel language. Such issues are typically of greater concern to compiler writers but not programmers.

ITS CONTENT

Chapter 1, “The Flavor of Eiffel,” presents a broad overview of the language, showing its major features. No attempt is made to provide detailed explanations of the many language features that are previewed in this chapter. A small class hierarchy associated with writing instruments is used to sample the flavor of Eiffel.

Chapter 2, “Classes and Objects,” introduces and illustrates the various components of an Eiffel class. An Eiffel class serves as an implementation of an abstract data type, as well as a software module. The main issues discussed in this chapter include the creation of objects, features and their visibility, object assignment and equality, copying and cloning objects, polymorphism and dynamic-binding, generic classes and expanded types.

Chapter 3, “Correct Programs,” introduces the powerful assertion-handling facilities of Eiffel. Pre and postconditions and class invariants are explained and illustrated. The notions of programming by contract and subcontracting are explored. Finally, Eiffel’s exception-handling mechanism is presented.

Chapter 4, “Generic Container Classes,” introduces the concept of constrained and unconstrained “genericity” and the construction of reusable software components. Several container classes, including a list, ordered list, stack, queue, search tree and sortable array, are presented to illustrate the basic ideas.

Chapter 5, “Inheritance in Eiffel,” discusses the role of inheritance in object-oriented programming. The overall structure of an Eiffel inheritance clause and its many possible subclauses are presented and illustrated. These include redefinition, renaming, export scope and “undefining.” Repeated inheritance structures are examined. The construction and role of abstract classes using deferred routines are explored. The issue of “mix-ins” or code inheritance is discussed.

Chapter 6, “Polymorphism and Conformance,” focuses on the rules associated with late-binding in Eiffel. An application of polymorphism involving the concept of the “generality of numbers” is presented as a non-trivial application of polymorphism.

Chapter 7, “C++ && Eiffel or is it C++ & Eiffel or is it C++ and Eiffel? : A Clash of Cultures,” provides a direct technical comparison of C++ and Eiffel.

Chapter 8, “Object-Oriented Analysis and Design,” introduces the Booch ’94 method and its notation. A supermarket checkout line simulation is used to illustrate the basic ideas.

Chapter 9, “An Ecological Simulation,” presents a case study involving the construction of several dozen classes that illustrates the Booch analysis and design process and the use of the Eiffel language. After completing the initial design, some simple maintenance is performed to illustrate the importance of building a robust software structure.

Chapter 10, “A Game of Strategies and Investment,” presents another case study that again involves the construction of several dozen classes. This example focuses on the extensive use of late-binding as a central design principle. After

the initial design is completed, it is critically examined and replaced with an improved design.

Chapter 11, “Simulated Annealing,” shows the application of object-oriented analysis, design and programming in a less traditional area — heuristic algorithm design. An abstract class that encapsulates the basic mechanism of simulated annealing is constructed. A concrete class that solves the classic “Travelling Salesperson Problem” is constructed as a subclass of the abstract simulated annealing class to demonstrate the use and power of inheritance in this non-traditional setting.

STYLE AND FORMAT

An area of potential controversy in this book is the deviation from the “standard” Eiffel style recommended in Bertrand Meyer’s reference *Eiffel: The Language* (ETL).

The arguments in favor of using a “standard” style are understood and appreciated by this author. It is this author’s deep conviction that programming is as much an art as a science. It is an aesthetic and expressive activity that involves the personality as well as the intellect of the programmer. This author believes that every programmer, Eiffel or otherwise, should develop a style that is personal, comfortable and most importantly consistent. Of course, consistent is different from uniform. The programs in this book hopefully illustrate this principle.

The “standard” Eiffel style recommends that multiple word identifiers use an underscore to separate individual words. For example, *average_force* to represent the idea of average force. This book sometimes adheres to this recommendation, but not always. A more Smalltalk influenced style is often evident. This Smalltalk style would use the identifier *averageForce* to represent the idea of average force.

The “standard” Eiffel style involves a myriad of details related to the layout of various language artifacts, formation of identifier names, as well as the use of boldface fonts for all keywords. This recommended style is presented in Appendix A of the ETL. Since there are about 15 pages of details presented there, no attempt will be made here to summarize the “standard” style recommendations. The reader is urged to read this Appendix in the ETL and then decide for himself or herself.

A sample of “standard” Eiffel style is presented in the Appendix of this book so that the reader can either see what they are missing or be further encouraged to develop their own personal style.

ACKNOWLEDGMENTS

Several people contributed to this book through discussions of many of the later examples. In particular, thanks go to two computer scientists very close to me: Erik, my son, and Hanne, my wife.

Jim McKim, of the Hartford Graduate Center, did more than review the first draft manuscript. He poured over every line of code in every example, making constructive suggestions, and contributed significantly to improvements in the book. Thank you so much Jim for your extremely constructive and useful effort.

The case studies in Chapters 8, 9 and 10 were class tested in a graduate course at the University of Colorado at Colorado Springs. I wish to thank Wes Munsil, a student in the class, for his constructive feedback throughout the course.

Other useful reviews that led to the improvement of the final manuscript were provided by Dr. Stuart Greenfield of Marist College and Paul Blanco of Object Center in Boston. I wish to thank Paul for his enthusiastic encouragement and help at every stage of this project.

A special thanks goes to Bertrand Meyer, not only for creating the fine language Eiffel, but for his encouragement, friendship and support throughout this project. His classic work, *Object-Oriented Software Construction*, published by Prentice Hall in 1988 has served as an inspiration to me and many other workers in this field. His work associated with Eiffel and his writings in general have made significant contributions to our field.

I wish to thank Interactive Software Engineering (ISE) in Goleta, California, for providing me with the latest versions of their compiler and tools as Eiffel has evolved over the years. In particular, I wish to thank Annie Meyer and Fredrik Deramat for their help.

I thank Tower Technology in Austin, Texas, for also providing me with the latest versions of their compiler and tools. Their president, Robert Howard, and marketing director, Madison Cloutier, have provided tremendous help and encouragement throughout this project. Many of the programs and examples in this book were developed and tested using the TowerEiffel system.

I am most grateful to Paul Becker, Publisher at Prentice Hall, for his continuing support.

Finally, on a personal note, I am so grateful for the patience, encouragement, love and support of my wife Hanne. It is to you that this book is dedicated.

The Language

Contents

Preface	xiii
----------------	-------------

Part 1 The Language

Chapter 1: The Flavor of Eiffel	1
1.1 Introduction	1
1.2 The Flavor of Eiffel — Some Writing Instruments	4
1.2.1 A Quick Specification	4
1.2.2 Some Analysis	4
1.2.3 Naming Conventions	6
1.2.4 The Eiffel Implementation	6
1.2.5 Implementation in C++ — A Big Problem With No Easy Solution	16
1.3 A Final Warm-up Example	17
1.3.1 Discussion of Class FINANCIAL	20
1.3.2 Discussion of Class APPLICATION	25
1.4 Some General Observations About Eiffel	26
 Chapter 2: Classes and Objects	 29
2.1 Classes and Objects	29
2.2 Eiffel Classes	30
2.3 Features	32
2.3.1 Visibility	34
2.3.2 Routines	35
2.3.3 Attributes	38

2.4	The Creation of Objects	40
2.5	Some Simple Classes	41
2.5.1	A Simple Class POINT	41
2.5.2	A Simple Class LINE_SEGMENT	43
2.5.3	Some Additions to Classes POINT and LINE_SEGMENT	44
2.6	Semantics of Object Assignment and Equality	50
2.6.1	Deep and Shallow Copies	51
2.6.2	Deep and Shallow Clones	54
2.6.3	Deep and Shallow Equality Testing	54
2.6.4	Assignment Operation	55
2.7	Polymorphism and Dynamic-binding — Learning to Ski, Walk and Run	55
2.8	Generic Classes	57
2.9	Constant Objects	61
2.10	Types	64
2.10.1	Anchored Types	64
2.10.2	Expanded Types	67
2.11	Summary	72
Chapter 3:	Correct Programs	75
3.1	Quality Software	75
3.2	Software Components and the Notion of a Software Contract	76
3.3	Programming By Contract	77
3.3.1	Contract Enforcement	78
3.3.2	Location of Exception	79
3.3.3	Exception Handler	79
3.4	Preconditions	79
3.5	Postconditions	82
3.6	Class Invariants and Consistency	84
3.7	Subcontracting — Propagation of Assertions in Subclasses	85
3.8	Loop Invariants	86
3.9	The Check Assertion	88
3.10	Enabling Assertions	88
3.11	Summary	88
Chapter 4:	Generic Container Classes	91
4.1	An Introduction to “Genericity”	91
4.2	Unconstrained and Constrained Generic Container Classes	97
4.2.1	List	97

4.2.2	Ordered List	102
4.2.3	Stack	105
4.2.4	Queue	108
4.2.5	Search Tree	109
4.2.6	"Sortable" Arrays Re-visited	115
4.3	Summary	119
Chapter 5:	Inheritance in Eiffel	121
5.1	The Role of Inheritance in Object-Oriented Programming	121
5.2	Overall Structure of Inheritance Clause	123
5.3	Redefinition of Features	124
5.3.1	Formal Rules for Redefinition	129
5.3.2	Redefining a Function as an Attribute	130
5.4	Deferred Classes and Key Methods	132
5.5	Renaming and Selection of Features	136
5.5.1	Accessing a Parent Routine from a Redefined Routine in a Child Class	138
5.5.2	Using Renaming to Resolve Name Clashes	139
5.6	Modifying the Export Status of Inherited Features	141
5.7	How to "Undefine" a Parent Feature in a Subclass — The Join Mechanism	142
5.8	Repeated Inheritance	147
5.8.1	An Example — Vehicle Classes	147
5.8.2	Rules for Repeated Inheritance, Expanded Classes and a Little Language Law	162
5.9	"Mix-ins" or Code Inheritance and the Inheritance Relationship	164
5.9.1	Iterators — An Application of Inheritance	167
5.10	Conclusions	171
Chapter 6:	Polymorphism and Conformance	175
6.1	Polymorphism and Object-Oriented Programming	175
6.2	Conformance	179
6.2.1	Conformance of Signatures	179
6.2.2	Conformance to a Non-generic Type	180
6.2.3	Conformance to a Generic Type	180
6.2.4	Arithmetic Operators and the "Balancing Rule"	180
6.3	An Application of Polymorphism —The Generality of Numbers	180
6.3.1	Background	180

6.3.2	The Application	181
6.3.3	Maintenance	189
6.3.4	Summary	190
6.4	Conclusions	190

Chapter 7: C++ & Eiffel or is it C++ & Eiffel or is it C++ and Eiffel? A Clash of Cultures **191**

7.1	The Language Wars?	191
7.1.1	Religion, Politics and, Oh Yes, Programming Languages	192
7.1.2	The Cultures	192
7.1.3	The Readability of C or C++	192
7.1.4	C++ Is a Hybrid Language	193
7.1.5	Complexity of C++	193
7.1.6	Complexity of Eiffel	193
7.1.7	Learning C++ and Eiffel	194
7.2	What You Get May Not Be What You See	194
7.2.1	Bringing Objects to Life — On the Stack and On the Heap	194
7.3	What You See May Not Be What You Get	202
7.3.1	Overloading is a Blessing When Exact Matching to Signatures Occurs	203
7.3.2	Exact Matches Occur Only in Heaven!	203
7.3.3	Some Conclusions	204
7.4	Generic Functions and Templates	204
7.4.1	Eiffel Offers Constrained Genericity	204
7.4.2	C++ is More Permissive	205
7.4.3	A Serious C++ Problem: The Need for Specialized Template Functions	205
7.4.4	Some Conclusions	206
7.5	Default Values for Parameters	207
7.6	Assertions	207
7.7	Virtual Functions and Late-Binding	208
7.7.1	The Clairvoyance Principle	209
7.7.2	Some Conclusions	210
7.8	Inheritance and the Clairvoyance Principle	210
7.8.1	Inheritance Provides the Basis for Software Reuse	211
7.8.2	Must a Subclass Designer Have Access to the Source Code of its Ancestors?	211
7.8.3	How Inheritance in C++ Supports Software Reuse	211
7.8.4	An Example that Compares Inheritance in C++ and Eiffel	212
7.8.5	Some Conclusions	216
7.8.6	Some Maintenance — Specifications for a New Subclass	216
7.8.7	More Conclusions	220

7.9 Multiple Inheritance and the Clairvoyance Principle	220
7.9.1 Virtual Base Classes in C++ and the Clairvoyance Problem	221
7.9.2 Conclusions	225
7.10 Readability of C++ and Eiffel	226
7.11 More on the Complexity of C++ and Eiffel	231
7.11.1 Power, Complexity and Usefulness	231
7.11.2 The “++” Part of C++	232
7.12 Why the C++ Bandwagon Effect—The Politics of Languages	235
7.13 Concluding Remarks	236

Part 2 Case Studies

Chapter 8: Object-Oriented Analysis and Design	239
8.1 Object-Oriented Modeling	239
8.2 Basic Steps in Object-Oriented Analysis	241
8.3 A Modeling Example — Supermarket Checkout Lines — The Analysis	243
8.4 Basic Steps in Object-Oriented Design	252
8.5 Supermarket Checkout Line Simulation Continued — The Design	253
8.6 Implementation of Checkout Line Simulation	262
8.7 Simulation Output	274
Chapter 9: An Ecological Simulation	277
9.1 Introduction	277
9.2 Informal Requirements	277
9.2.1 Characteristics Shared By All Sea Creatures	278
9.2.2 Specific Rules of Movement For Each Sea Creature	278
9.2.3 Overall Flow of Simulation	280
9.2.4 Simulation Output	280
9.3 Analysis	280
9.3.1 Similarities and Differences Among The Various Kinds of Cells	282
9.4 The Design	288
9.4.1 Explanation of Main Event Loop Object-Scenario Diagram	289
9.4.2 Explanation of move_empty Mechanism in Class Reproducible	289
9.4.3 Explanation of reproduce Mechanism in Class Reproducible	291
9.4.4 Explanation of Key Mechanism move in Class Predator	292
9.4.5 Explanation of Key Mechanism move in Class Scavenger	293
9.4.6 Explanation of starve Mechanism for Class Tuna	293
9.4.7 Explanation of Key Mechanism starve for Class Shark	293

9.5 Eiffel Implementation	299
9.6 Initial Conclusions	322
9.7 Maintenance on the Ecological Simulation	322
9.7.1 New Requirement	322
9.7.2 More Analysis	322
9.7.3 More Design	323
9.7.4 Implementation Changes	324
9.8 Final Conclusions	326
Chapter 10: A Game of Strategies and Investment	327
10.1 Introduction	327
10.2 Informal Requirements	327
10.3 Analysis	332
10.4 Design	334
10.5 Eiffel Implementation	344
10.6 Critical Examination of the Design of Version 1 — The Motivation for Version 2	370
10.7 Version 2 of the Game of Strategies and Investment	373
10.8 Concluding Remarks	380
Chapter 11: Simulated Annealing	383
11.1 Motivation	383
11.2 Combinatorial Optimization and Simulated Annealing	384
11.2.1 Stochastic Search	385
11.2.2 Annealing of Metals	385
11.2.3 Annealing Applied to Combinatorial Optimization — Simulated Annealing	386
11.3 The Components of The Algorithm	388
11.3.1 The Overall Flow of the Algorithm	388
11.3.2 Determining Thermal Equilibrium	389
11.4 Abstract Class ANNEALING and Supporting Class STATE	389
11.5 Solution of Travelling Salesperson Problem	400
11.6 Performance Comparisons	409
11.6.1 Eiffel Solution	410
11.6.2 C Solution	410
11.6.3 Conclusions From This Example	411
11.7 Final Words	411
Appendix	413
Index	417