Software Engineering with C++ and CASE Tools

软件工程与 C++和 CASE 工具

Michael J. Pont

世界图书出版公司

Software engineering with C++ and CASE tools

江苏工业学院图书馆
藏书章

Michael J. Pont
University of Leicester

世界图书出版公司
北京・广州・上海・西安

书　　名：Software Engineering with C++ and CASE Tools
作　　者：M.J.Pont
中 译 名：软件工程与 C++ 和 CASE 工具
出 版 者：世界图书出版公司北京公司
印 刷 者：北京中西印刷厂
发　　行：世界图书出版公司北京公司(北京朝内大街 137 号　100010)
开　　本：大 32　　印张：31
版　　次：1998 年 3 月第 1 版　　1998 年 3 月第 1 次印刷
书　　号：7-5062-3628-1
版权登记：图字 01-97-2024
定　　价：148.00 元

世界图书出版公司北京公司已获得 Addison-Wesley Longman Limited 授权在中国境内独家重印发行

Preface

This book focuses on the engineering of substantial bespoke systems: that is, on the development of specialized software products for a single or limited number of clients. For all but the most trivial of projects, this development will be carried out jointly by a team of engineers.

My specific objectives in writing are as follows:

- to provide a pragmatic introduction to the field of software engineering;
- to compare and contrast the three major software development strategies in widespread use at the present time: process-oriented ('structured') development, data-oriented development and object-oriented development;
- to introduce the C++ programming language, and to illustrate, through a series of substantial case studies, the engineering of C++ programs for a broad spectrum of software projects, from decision support systems to real-time embedded systems;
- to demonstrate the practical benefits of computer-aided software engineering (CASE) by including a high-quality CASE tool with the book, and making use of this tool for all examples in the text.

Readers are assumed to be professionally interested in some aspect of software engineering. They may, for example, be practising software engineers trained in COBOL or FORTRAN and wishing to update their skills. They may be software engineering, computer science or electronic engineering students, or lecturers on such programmes. Readers are expected to have had some previous high-level programming experience, but no specific experience with C or C++ is assumed. Within a university or college environment, the material may be found most suitable for intermediate-level (second year in the UK) students, perhaps following a short introductory programming course.

The material in the text has been thoroughly 'field tested', having formed the basis for a number of commercial training courses which I have presented over several years. It has also been used in my university undergraduate and postgraduate teaching, first in the Department of Computer Science at the University of Sheffield, and more recently in the Department of Engineering at the University of Leicester.

Instructor's Guide

To make it easier for you to use *Software Engineering with C++ and CASE Tools* in a university or college environment, an Instructor's Guide, written by the author, is available from Addison-Wesley. The guide provides solutions to many of the exercises, plus some further examples. It also contains suggestions for conducting various types of software engineering courses, and for associated practical work.

The CASE tool included with this book is produced by SELECT Software Tools. If you wish to use the SELECT CASE tool for university or college teaching, then you may be interested to know that SELECT Software Tools offer a very substantial discount on academic site licences for a full network version of the single-user CASE tool included in this package. You can contact the company as follows:

SELECT Software Tools
Westmoreland House
80–86 Bath Road
Cheltenham
Gloucestershire GL53 7JT
UK

or,

SELECT Software Tools
Suite #84 Brookhollow Office Park
1526 Brookhollow Drive
Santa Ana
California 92705
USA

Compact disk

The code for the SELECT CASE tool itself (executable files) is included on the compact disk (CD) attached inside the back cover. Details of how to install the CASE tool are given in Appendix A. With the exception of the first study (which is concerned solely with programming), the case studies employ this tool. All of the associated diagrams and files are included on the CD. Details of how to install the case study files are given in Appendix C.

Copies of all the C++ programs given in the text (source files) are also included on the CD. Except in a small number of cases (explicitly noted in the text) the programs are written in 'standard' C++ (Stroustrup, 1991). The samples have been tested with a number of compilers, including the Microsoft Visual C++ compiler (v1.0), the Borland C++ compiler (v3.5), and the Watcom C++ compiler (v10.0). Details of how to install the code samples are given on the CD in the file 'README.WRI' in the directory '\BOOK\SOURCE'.

此为试读，需要完整PDF请访问：www.ertongbook.com

Video

A video introduction to object-oriented programming based on this book will be available from the author in June 1996. Full details of the video are available on the World Wide Web (http://www.engg.le.ac.uk/Staff/Mike.Pont/index.html), by electronic mail (soft_eng_video@sun.engg.le.ac.uk) or by post from:

Dr Michael J. Pont
Department of Engineering
University of Leicester
University Road
Leicester LE1 7RH
UK

Acknowledgements

In a project such as this, one name appears on the cover, but the book only exists because of the efforts of others behind the scenes. It is a pleasure, therefore, to be able to publicly thank those who have been involved in bringing this work to fruition.

I thank all my friends and colleagues at Sheffield University and Leicester University for many helpful discussions: Fernando Schlindwein and Derek Andrews both deserve special mention. I thank John Fothergill both for his encouragement, and also for enlightening me about the roots of the engineering profession. The book wouldn't have appeared at all without the active support of Barrie Jones, then Head of the Department of Engineering in Leicester. Particular thanks are also due to Andrew Norman, without whom the C++ examples in this book would include many more bugs than they do. Pop Sharma and Andy Willby each deserve a medal for surviving many debugging sessions.

This book has arisen, in part, from my experiences with software development on a number of research projects over the past ten years, mainly involving the computer simulation of parts of the human auditory nervous system. This work would not have been possible without the support of a great many people, among them Bob Damper, Phil Green and John Frisby. Particular thanks are also due to my long-suffering postgraduate students – David Sewell, Chen Pang Wong, Kien Seng Wong and Eric Worrall – for putting up with absence of their supervisor (in mind when not in body) during the gestation of this text in the past 12 months. Thanks are also due to David for his help with several of the code examples, to Kien Seng for the screen shot in Chapter 14, and to Chen Pang and Eric for their work on the appendices.

I thank everyone – university students and those from industry and elsewhere – who have endured my courses in programming and software engineering over recent years, for asking tough questions and teaching me a great deal in the process. Thanks are particularly due to Daniel Grimwade, Vasanthi Sundaramoorthy, Graham Cottle, Noel John Bernatt, Y.K. Ng, Lee Pasifull, Stuart Urban, Tony Vickers and Daniel Yeoh for useful comments on the evolving manuscript.

I thank Emanuela Moreale for taking the trouble to provide a detailed review of the whole of an early draft of the manuscript, and Gordon Pont and Andrew Pont for useful comments on an even earlier draft. I thank the (anonymous) reviewers of the evolving manuscript for many helpful suggestions.

I thank Simon Plumtree at Addison-Wesley for believing in this large, complicated project and getting it off the ground. Also at Addison-Wesley, Sheila Chatten spent many weeks dealing patiently with my frequent email queries, and generally keeping the book on the rails. I thank the staff at Select Software Tools: in particular, the book could not have appeared in this form without the efforts of Virginia Bray.

I thank Bruce Springsteen for writing and recording 'Greetings from Asbury Park NJ'.

I thank all the folk at Leicester Software Engineering – Jane, Stephanie and Harry – without whom none of this would have been possible.

Last, and most of all, I thank Sarah, my partner and my best friend. After a year which we have both shared with 'The Book', any sanity I have left is entirely due to her.

Michael J. Pont
Leicester, March 1996

Contents

Preface vii

Introduction **1**

1 **Beginning at the end** **3**
- 1.1 What is software engineering? 3
- 1.2 Engineering quality software 4
- 1.3 But hang on a minute … 5
- 1.4 A thought experiment 6
- 1.5 Quality software in the real world 8
- 1.6 An offer you can't refuse 9
- 1.7 Why use C++? 10
- 1.8 SADIE and quality software 10
- 1.9 SADIE and CASE tools 11
- 1.10 Beginning at the end 12
- 1.11 Conclusions 14

2 **Leicester Software Engineering** **15**
- 2.1 Introduction 15
- 2.2 LSE company structure 15
- 2.3 A typical scenario 16
- 2.4 A more realistic software development model 18
- 2.5 Three different system types 19
- 2.6 LSE and SADIE 21
- 2.7 One methodology, three different methods 22
- 2.8 Conclusions 25
- Exercises 26

3 **Goodbye, cruel world!** **27**
- 3.1 Introduction 27
- 3.2 The origins of C++ 27
- 3.3 A first C++ program 30
- 3.4 Variable types and declarations 36

	3.5	Manipulating data: some basic operators	40
	3.6	Input and output	41
	3.7	More on the declaration of variables	46
	3.8	Conclusions	50
		Exercises	50
4	**Taking control**		**51**
	4.1	Introduction	51
	4.2	Some important operators	51
	4.3	Iteration	59
	4.4	Selection statements	73
	4.5	Advanced operators	87
	4.6	Conclusions	89
	Exercises		89
Part I	**Process-oriented software development**		**91**
5	**Case study: Harry Hacker (Programming) Ltd**		**93**
	5.1	Introduction	93
	5.2	The scenario	93
	5.3	You need to use functions	97
	5.4	Conclusions	100
6	**Writing C++ functions**		**101**
	6.1	Introduction	101
	6.2	Another look at `main()`	101
	6.3	A first simple function: `Sadie()`	103
	6.4	Declaration vs definition revisited	103
	6.5	Calling a simple function	106
	6.6	Call by value	107
	6.7	Local and global scope	111
	6.8	Call by reference	117
	6.9	ASIDE: Scope rules and storage class	120
	6.10	Detecting and correcting errors	127
	6.11	Using the `const` modifier	129
	6.12	Conclusions	132
	Exercises		133
7	**Pointers and arrays**		**135**
	7.1	Introduction	135
	7.2	Memory organization	135
	7.3	Assigning values to variables	141
	7.4	Pointers	143
	7.5	Arrays	146
	7.6	Pointers and arrays	150
	7.7	Strings	157
	7.8	More advanced pointer topics	167
	7.9	Conclusions	170
	Exercises		171

8 A closer look at functions **174**
8.1 Introduction 174
8.2 Call by address 174
8.3 More on the `const` modifier 177
8.4 Arrays as function parameters 177
8.5 Returning values from functions 184
8.6 Example: On strings, constants and functions 186
8.7 Macros and inline functions 191
8.8 Recursion 194
8.9 Flexible functions 194
8.10 Conclusions 197
Exercises 199

9 The standard libraries **201**
9.1 Introduction 201
9.2 Pragmatics 201
9.3 Character and string handling 204
9.4 Mathematical functions 220
9.5 Graphics functions 222
9.6 Sorting and searching 224
9.7 Time and date functions 227
9.8 Abandoning the sinking ship 228
9.9 Interrupts 228
9.10 Conclusions 229
Exercises 229

10 The physical process model **231**
10.1 Introduction 231
10.2 Structure charts 233
10.3 Process specifications 235
10.4 An alternative structure chart notation 240
10.5 Conclusions 241
Exercises 241

11 Implementation of the physical process model **243**
11.1 Introduction 243
11.2 The recipe 243
11.3 The starting point 244
11.4 Building the system code framework 246
11.5 Creating robust functions 249
11.6 Fundamentals of testing 256
11.7 Using driver programs 262
11.8 Field trials 265
11.9 Conclusions 266
Exercises 267

12 Process-oriented analysis **270**
12.1 Introduction 270
12.2 Requirements analysis 271

	12.3	The dataflow diagram	280
	12.4	The DfD hierarchy	284
	12.5	The process specification	285
	12.6	The data dictionary	289
	12.7	Balancing the system	291
	12.8	Documentation	292
	12.9	Conclusions	293
	Exercises		294
13	**Case study: Loughborough Bell Foundry**		**295**
	13.1	Introduction	295
	13.2	Scenario	295
	13.3	Overview of the analysis process	295
	13.4	The business case	296
	13.5	The Context diagram	297
	13.6	The user interface	298
	13.7	The process list	298
	13.8	The walkthrough	299
	13.9	The Level 1 dataflow diagram	300
	13.10	The dataflow diagram hierarchy	300
	13.11	The process specifications	301
	13.12	The data dictionary	302
	13.13	The balanced logical model	302
	13.14	Conclusions	303
14	**Process-oriented analysis of interactive systems**		**304**
	14.1	Introduction	304
	14.2	State-transition diagrams	306
	14.3	Control processes	311
	14.4	The impact of the user interface	317
	14.5	Relating STDs to C++ code	320
	14.6	Some new guidelines	322
	14.7	Conclusions	324
	Exercises		324
15	**Process-oriented design**		**327**
	15.1	Introduction	327
	15.2	Review and refinement of the logical model	328
	15.3	The software and hardware architecture	337
	15.4	The physical process model	344
	15.5	Review and refinement of the physical model	350
	15.6	Reconciling the physical and logical models	354
	15.7	The test strategy	356
	15.8	Conclusions	358
	Exercises		358
16	**Case study: Birstall Bank**		**360**
	16.1	Introduction	360
	16.2	Scenario	360

此为试读，需要完整PDF请访问：www.ertongbook.com

16.3 Analysis 360
16.4 Design 381
16.5 Implementation 385
16.6 Conclusions 423

Part II Data-oriented software development 425

17 Implementing isolated data in C++ 427
17.1 Introduction 427
17.2 Creating new data types with `structs` 429
17.3 Creating `struct` variables 430
17.4 Accessing elements of `struct` variables 431
17.5 `unions` 433
17.6 Creating enumerated types 434
17.7 Example: A simple payroll program 437
17.8 A rose by any other name … 441
17.9 Conclusions 441
Exercises 442

18 Dynamic memory allocation 443
18.1 Introduction 443
18.2 Simple dynamic memory allocation 444
18.3 Dynamic arrays 447
18.4 More complex data structures 448
18.5 Conclusions 456
Exercises 456

19 Saving data for posterity 459
19.1 Introduction 459
19.2 File access fundamentals 459
19.3 Opening and closing files 462
19.4 Sequential file access 464
19.5 Direct file access 468
19.6 Conclusions 474
Exercises 474

20 The relational database 476
20.1 Introduction 476
20.2 Fundamentals of database systems 477
20.3 Why use a relational data model? 477
20.4 Data normalization issues 480
20.5 Relationships between tables 487
20.6 Implementation of RDBs 491
20.7 Other types of database 496
20.8 Conclusions 497
Exercises 497

21 Entity-relationship diagrams 500
21.1 Introduction 500

	21.2	Modelling general associations	500
	21.3	Using attribute diagrams	504
	21.4	How to create ERDs	505
	21.5	Implementating from ERDs	506
	21.6	Aggregation relationships	512
	21.7	Conclusions	515
	Exercises		515
22	**Data-oriented analysis and design**		**518**
	22.1	Introduction	518
	22.2	The analysis phase	519
	22.3	Integration of process and data models	521
	22.4	Decision support systems	523
	22.5	The design phase	524
	22.6	Data normalization and speed of data access	525
	22.7	The implementation phase	526
	22.8	Conclusions	527
	Exercises		527
23	**Case study: Laughing House**		**531**
	23.1	Introduction	531
	23.2	Scenario	531
	23.3	Analysis	532
	23.4	Press cutting	544
	23.5	Design	545
	23.6	Implementation	550
	23.7	Conclusions	586
Part III	**Object-oriented software development**		**587**
24	**Why we need objects**		**589**
	24.1	Introduction	589
	24.2	A thought experiment revisited	590
	24.3	Encapsulation	592
	24.4	Polymorphism	600
	24.5	Inheritance	605
	24.6	Classes for courses	608
	24.7	Conclusions	609
	Exercises		610
25	**Encapsulation**		**611**
	25.1	Introduction	611
	25.2	Constructor functions	611
	25.3	Being destructive	617
	25.4	Functions, objects and copies	619
	25.5	Copy constructors	624
	25.6	Example: A new array type	628
	25.7	Dealing with errors	638

25.8 Guidelines 645
25.9 Conclusions 645
Exercises 645

26 A first look at class relationships 648
26.1 Introduction 648
26.2 Aggregations (APO relationships) 648
26.3 Associations (NTKA relationships) 652
26.4 Relationships between objects 657
26.5 Conclusions 660
Exercises 661

27 Polymorphism 662
27.1 Introduction 662
27.2 The `this` pointer 663
27.3 Overloading binary operators 665
27.4 Copy constructors vs assignment 674
27.5 Overloading unary operators 676
27.6 Using `friend` functions for operator overloading 681
27.7 Guidelines 686
27.8 Conclusions 687
Exercises 687

28 Generic programming 688
28.1 Introduction 688
28.2 Generic functions 688
28.3 Generic classes 693
28.4 Guidelines 700
28.5 Conclusions 700
Exercises 700

29 Inheritance 701
29.1 Introduction 701
29.2 Single inheritance 701
29.3 Multiple inheritance 713
29.4 Virtual functions, pointers and polymorphism 718
29.5 Pure virtual functions 724
29.6 Guidelines 726
29.7 Conclusions 727
Exercises 728

30 Class-relationship diagrams 729
30.1 Introduction 729
30.2 Entities vs classes 729
30.3 Class-relationship diagrams 730
30.4 Identifying appropriate classes 734
30.5 Conclusions 737
Exercises 737

31 Object-oriented SADIE 738
31.1 Introduction 738
31.2 Why CRDs are not enough 738
31.3 Building the object model 739
31.4 The components of the object model 740
31.5 Implementing the object model 746
31.6 Testing object-oriented systems 753
31.7 A recipe for success 754
31.8 Conclusions 755
31.9 Appendix: Source code for mailing list 756
Exercises 765

32 Case study: Birstall Bank revisited 767
32.1 Introduction 767
32.2 Scenario 767
32.3 Developing the object model 768
32.4 Implementation of the class model 781
32.5 Conclusions 829

Conclusions 831

33 Ending at the beginning 833
33.1 Introduction 833
33.2 Which method should you use? 833
33.3 Varying and mixing the recipes 835
33.4 Conclusions 835

34 Bibliography 836

Appendices 841
A CASE tool: Installation 843
B CASE tool: Managing a project 849
C CASE tool: Installing the sample projects 864
D CASE tutorial: General introduction 866
E CASE tutorial: Dataflow diagrams 903
F CASE tutorial: Data dictionary 920
G CASE tutorial: Process specifications 931
H CASE tutorial: State transition diagrams 934
I CASE tutorial: Structure charts 947
J CASE tutorial: Entity-relationship diagrams and class-relationship diagrams 952

Index 959

Introduction

此为试读,需要完整PDF请访问: www.ertongbook.com