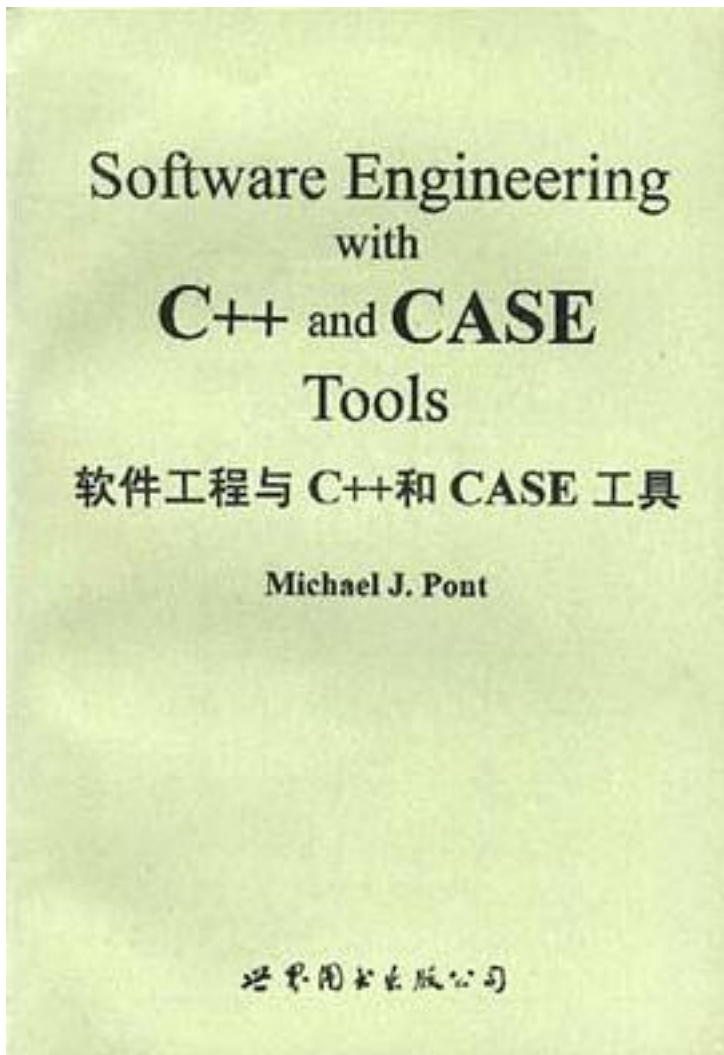


software engineering with c++and CASE Tools



[software engineering with c++and CASE Tools_ 下载链接1](#)

著者:M.J.Pont

出版者:世界图书出版公司

出版时间:1998-03

装帧:平装

isbn:9787506236287

作者介绍:

目录: Contents

Introduction

1.Beginning at the eod

1.1 What is software engineering?

1.2 Engineering quality software

1.3 But hang on a minute

1.4 A thought experiment

1.5 Quality software in the real world

1.6 An offer you can't refuse

1.7 Why use C++?

1.8 SADIE and quality software

1.9 SADIE and CASE tools

1.10 Beginning at the end

1.11 Conclusions

2.Lelcxster Softwre Engineering

2.1 Introduction

2.2 LSE company structure

2.3 A typical scenario

2.4 A more realistic software devlopment model

2.5 Threc different system types

2.6 LSE and SADIE

2.7 One mcthodology, three different methods

2.8 Conclusions

Excrises

3.Goodbye, cruel world!

3.1 Introduction

3.2 TheoriginsofC++

3.3 A first C++ program

3.4 Variable types and declarations

3.5 Manipulating data: some basic operators

3.6 Input and output

3.7 More on the declaration of variables

3.8 Conclusions

Exerciscs

4.Taking control

4.1 Introduction

4.2 Some important operators

4.3 Iteration

4.4 Selection statements

4.5 Advanced operators

4.6 Conclusions

Exercises

1 Process-oriented software deyelopment

5. Case study: Harry Hacker (Programmlng) Ltd

5.1 Introduction

5.2 The scenario

5.3 You need to use functions

5.4 Conclusions

6. Writing C++ functions

6.1 Introduction

6.2 Another look at main

6.3 A first simple function: Sadie

6.4 Declaration vs definition revisited

6.5 Calling a simple function

6.6 Call by value

6.7 Local and global scope

6.8 Call by reference

6.9 ASIOE: Scope rules and storage class

6.10 Detecting and correcting errors

6.11 Using the const modifier

6.12 Conclusions

Exercises

7. Pointers and arrays

7.1 Introduction

7.2 Memory organization

7.3 Assigning values to variables

7.4 Pointers

7.5 Arrays

7.6 Pointers and arrays

7.7 Strings

7.8 More advanced pointer topics

7.9 Conclusions

Exercises

8. Closer look at functions

8.1 Introduction

8.2 Call by address

8.3 More on the const modifier

8.4 Arrays as function parameters

8.5 Returning values from functions

8.6 Example: On strings, constants and functions

8.7 Macros and inline functions

8.8 Recursion

8.9 Flexible functions

8.10 Conclusions

Exercises

9. The standard libraries

9.1 Introduction

9.2 Pragmatics

9.3 Character and string handling

9.4 Mathematical functions

9.5 Graphics functions

9.6 Sorting and searching

9.7 Time and date functions

9.8 Abandoning the sinking ship

9.9 Interrupts

9.10 Conclusions

Exercises

10. The physical process model

10.1 Introduction

10.2 Structure charts

10.3 Process specifications

10.4 An alternative structure chart notation

10.5 Conclusions

Exercises

11.Implementation of the physkal process model

11.1 Introduction

11.2 Therecipe

11.3 The starting point

11.4 Building the system code framework

11.5 Creating robust functions

11.6 Fundamentals of testing

11.7 Using driver programs

11.8 Fieldtrials

11.9 Conclusions

Exercises

12.Process-oriented analysis

12.1 Introduction

12.2 Requirements analysis

12.3 The dataflow diagram

12.4 The DfD hierarehy

12.5 The process specification

12.6 The data dictionary

12.7 Balancing the system

12.8 Documentation

12.9 Conclusions

Exercises

13.Case study: Loughborough Bell Foundry

13.1 Introduction

13.2 Scenario

13.3 Overview of the analysis process

13.4 The business case

13.5 The Context diagram

13.6 The User interface

13.7 The process list

13.8 The walkthrough

13.9 The Level 1 dataflow diagram

13.10 The dataflow diagram hierarchy

13.11 The process specifications

13.12 The data dictionary

13.13 The balanced logical model

13.14 Conclusions

14.Process-oriented analysis of interactive systems

14.1 Introduction

14.2 State-transition diagrams

14.3 Control processes

14.4 The impact of the user interface

14.5 Relating STDs to C++ code

14.6 Some new guidelines

14.7 Conclusions

Exercises

15.Process-oriented design

15.1 Introduction

15.2 Review and refinement of the logical model

15.3 The software and hardware architecture

15.4 The physical process model

15.5 Review and refinement of the physical model

15.6 Reconciling the physical and logical models

15.7 The test strategy

15.8 Conclusions

Exercises

16. Case study: Birstall Bank

16.1 Introduction

16.2 Scenario

16.3 Analysis

16.4 Design

16.5 Implementation

16.6 Conclusions

Part 11 Data-oriented software development

17. Implementing isolated data in C++

17.1 Introduction

17.2 Creating new data types with structs

17.3 Creating struct variables

17.4 Accessing elements of struct variables

17.5 unions

17.6 Creating enumerated types

17.7 Example: A simple payroll program

17.8 A rose by any other name ...

17.9 Conclusions

Exercises

18. Dynamic memory allocation

18.1 Introduction

18.2 Simple dynamic memory allocation

18.3 Dynamic arrays

18.4 More complex data structures

18.5 Conclusions

Exercises

19. Saving data for posterity

19.1 Introduction

19.2 File access fundamentals

19.3 Opening and closing files

19.4 Sequential file access

19.5 Direct file access

19.6 Conclusions

Exercises

20. The relational database

20.1 Introduction

20.2 Fundamentals of database systems

20.3 Why use a relational data model?

20.4 Data normalization issues

20.5 Relationships between tables

20.6 Implementation of RDBs

20.7 Other types of database

20.8 Conclusions

Exercises

21. Entity-relationship diagrams

21.1 Introduction

21.2 Modelling general associations

21.3 Using attribute diagrams

21.4 How to create ERDs

21.5 Implementing from ERDs

- 21.6 Aggregation relationships
- 21.7 Conclusions
- Exercises
- 22.Data-oriented analysis and design
 - 22.1 Introduction
 - 22.2 The analysis phase
 - 22.3 Integration of process and data models
 - 22.4 Decision support systems
 - 22.5 The design phase
 - 22.6 Data normalization and speed of data access
 - 22.7 The implementation phase
 - 22.8 Conclusions
- Exercises
- 23.Case study: Laughing House
 - 23.1 Introduction
 - 23.2 Scenario
 - 23.3 Analysis
 - 23.4 Press cutting
 - 23.5 Design
 - 23.6 Implementation
 - 23.7 Conclusions
- Part 11 Object-oriented software development
- 24.Why we need objects
 - 24.1 Introduction
 - 24.2 A thought experiment revisited
 - 24.3 Encapsulation
 - 24.4 Polymorphism
 - 24.5 Inheritance
 - 24.6 Classes for courses
 - 24.7 Conclusions
- Exercises
- 25.Encapsulation
 - 25.1 Introduction
 - 25.2 Constructor functions
 - 25.3 Being destructive
 - 25.4 Functions, objects and copies
 - 25.5 Copy constructors
 - 25.6 Example: A new array type
 - 25.7 Dealing with errors
 - 25.8 Guidelines
 - 25.9 Conclusions
- Exercises
- 26.A first look at class relationships
 - 26.1 Introduction
 - 26.2 Aggregations (APO relationships)
 - 26.3 Associations (NTKA relationships)
 - 26.4 Relationships between objects
 - 26.5 Conclusions
- Exercises
- 27.Polymorphism
 - 27.1 Introduction
 - 27.2 The this pointer
 - 27.3 Overloading binary operators
 - 27.4 Copy constructors vs assignment

- 27.5 Overloading unary operators
- 27.6 Using friend functions for operator overloading
- 27.7 Guidelines
- 27.8 Conclusions
- Exercises
- 28.Generic programming
 - 28.1 Introduction
 - 28.2 Generic functions
 - 28.3 Generic classes
 - 28.4 Guidelines
 - 28.5 Conclusions
 - Exercises
- 29.Inheritance
 - 29.1 Introduction
 - 29.2 Single inheritance
 - 29.3 Multiple inheritance
 - 29.4 Virtual functions, pointers and polymorphism
 - 29.5 Pure virtual functions
 - 29.6 Guidelines
 - 29.7 Conclusions
 - Exercises
- 30.Class-relationship diagrams
 - 30.1 Introduction
 - 30.2 Entities vs classes
 - 30.3 Class-relationship diagrams
 - 30.4 Identifying appropriate classes
 - 30.5 Conclusions
 - Exercises
- 31.Object-oriented SADIE
 - 31.1 Introduction
 - 31.2 Why CRDs are not enough
 - 31.3 Building the object model
 - 31.4 The components of the object model
 - 31.5 Implementing the object model
 - 31.6 Testing object-oriented systems
 - 31.7 A recipe for success
 - 31.8 Conclusions
 - 31.9 Appendix: Source code for mailing list
 - Exercises
- 32.Case study: BirstaU Bank revisited
 - 32.1 Introduction
 - 32.2 Scenario
 - 32.3 Developing the object model
 - 32.4 Implementation of the class model
 - 32.5 Conclusions
- Conclusions
- 33.EHding at the beginning
 - 33.1 Introduction
 - 33.2 Which method should you use?
 - 33.3 Varying and mixing the recipes
 - 33.4 Conclusions
- 34.Bibliography
- Appendices
 - A CASE tool: Installation

- B CASE tool: Managing a project
- C CASE tool: Installing the sample projects
- D CASE tutorial: General introduction
- E CASE tutorial: Dataflow diagrams
- F CASE tutorial: Data dictionary
- G CASE tutorial: Process specifications
- H CASE tutorial: State transition diagrams
- I CASE tutorial: Structure charts
- J CASE tutorial: Entity-relationship diagrams and class-relationship diagrams
- Index
- • • • • (收起)

[software engineering with c++and CASE Tools_ 下载链接1](#)

标签

评论

[software engineering with c++and CASE Tools_ 下载链接1](#)

书评

[software engineering with c++and CASE Tools_ 下载链接1](#)