



#### 图书基本信息

- 书名: <<软件体系结构>>
- 13位ISBN编号:9787308054539
- 10位ISBN编号:7308054535
- 出版时间:2008-12
- 出版时间:浙江大学出版社
- 作者:邢剑宽,郑翔覃征
- 页数:337

版权说明:本站所提供下载的PDF图书仅提供预览和简介,请支持正版图书。

更多资源请访问:http://www.tushu007.com



#### 内容概要

Building software nowadays is far more difficult than it can be done several decadesago. At that time, software engineers focused on how to manipulate the computer towork and then solve problems correctly. The organization of data and implementation of algorithm were the crucial process of software designing then. However, more and more tasks in low level, such as memory management and network communication, have been automatized or at least can be reused with littleeffort and cost. Programmers and designers, with the help of high level programming anguages and wieldy development tools, can pay more attention to problems, rather than bury themselves into the machine code manuals. However, the side effect of these utilities is that more complicated problems are given according to therequirements from military, enterprise and so on, in which the complexity growsrapidly day by day. We believe that software architecture is a key to deal with it.



#### 书籍目录

1 Introduction to Software Architecture 1.1 A Brief History of Software Development 1.1.1 The Evolution of-Programming Language——-'Abstract Level1.1.2 The Evolution of Software Development- Concerns1.1.3 The Origin and Growth of Software Architecture1.2 Introduction to Software Architecture1.2.1 Basic Terminologies1.2.2 Understanding IEEE 1471-20001.2.3 Views Used in Software Architecture1.2.4 Why We Need Software Architecture 1.2.5 Where Is Software Architecture in Software Life Cycle 1.3 SummaryReferences2 Architectural Styles and Patterns2.1 Fundamentals of Architectural Styles and Patterns2.2 Pipes Filters2.2.1 Style Description 2.2.2 Study Case 2.3 Object-oriented 2.3.1 Style Description 2.3.2 Study Case 2.4 Event-driven 2.4.1 Style Description 2.4.2 Study Case 2.5 Hierarchical Layer 2.5.1 Style Description 2.5.2 Study Case 2.6 Data Sharing 2.6.1 Style Description 2.6.2 Study Case 2.7 Virtual Machine 2.7.1 Style Description 2.7.2 Study Case 2.8 Feedback Loop2.8.1 Style Description2.8.2 Study Case2.9 Comparison among Styles2.10 Integration of Heterogeneous Styles2.11 SummaryReferences3 Application and Analysis of Architectural Styles3.1 Introduction to SMCSP3.1.1 Program Background 3.1.2 Technical Routes 3.1.3 Function Design 3.2 System Realization 3.2.1 The Pattern Choice3.2.2 Interaction Mechanism3.2.3 Realization of Mobile Collaboration3.2.4 Knowledge-based Design3.3 SummaryReferences4 Software Architecture Description4.1 Formal Description of Software Architecture4.1.1 Problems in Informal Description4.1.2 Why Are Formal Methods Necessary4.2 Architectural Description Language4.2.1 Introduction to ADL4.2.2 Comparing among Typical ADLs4.2.3 Describing Architectural Behaviors4.3 Study Case: WRIGHT System4.3.1 Description of Component and Connector4.3.2 Description of Configuration 4.3.3 Description of Style 4.3. CSP-Semantic Basis of Formal Behavior Description 4.4 FEAL : An Infrastructure to Construct ADLs4.4.1 Design Purpose4.4.2 F EC4.4.3 FEAL Structure4.4.4 FEAL Mapper4.4.5 Examples of FEAL Application 4.5 Summary......5 Design Strategies in Architecture Level 6 Software Architecture IDE7 Evaluating Software Architecture8 Flexible Software Architecture9 A Vision on Software ArchitectureIndex

## <<软件体系结构>>

#### 章节摘录

(3) It is easy to implement grammar. It is similar to define the class of every node in the abstract grammar tree, and these classes are easy to write directly. Generally, they can be automatically generated by compiler or grammar analysis generat or. In this part, we will describe the roles in the Boolean expression system. Generally speaking, there are five roles in this type of system. The first one is Boolean Expression. This role declares an abstract Evaluate operation, this interface is shared by all the nodes of the Boolean expression abstract gammar tree. The second role is Terminal Expression (such as Variable Expression and Constant). This type of role implements the Evaluation operation in the Boolean Expression that arerelated to terminals, every terminal in the Boolean Expression needs an objectinstance of this class. The third role is Nonterminal Expression (such asAndExpression, Or Expression and NotExpression). Every rule in the Booleanexpression grammar needs an object instance of NonterminalExpression, and we must maintain object instance of Boolean Expression for every symbolin every rulein the Boolean expression grammar. We also need to implement the Evaluate operation for every NonterminalExpression in the grammar. In the NonterminalExpressionEvaluate operation, we must call the Evaluate operation for every symbol in the grammar. The fourth role is context ( this is "the inner state of interpreter engine") It includes the global information besides the interpreter. The fifth role is client. Client will constructs a special Boolean expression's abstract grammar tree in the Boolean expression's definition, and this abstract grammar tree is composed by theinstance objects of TerminalExpression and NonterminalExpression. The client willalso call the Evaluate operation. The collaboration relationship between these five roles can be simply describedas follows : At first, the client constructs a Boolean Expression, which is an abstractgrammar tree which is composed by instances of TerminalExpression andNonterminalExpression. Then the client initiates the context and calls interpret operation. Then every Nonterminal Expression defines the Evaluate operation of theaccording expression, and all the Evaluate operation of the expression forms thebasis of-recursive evaluation.At last, the Evaluate operation of every node uses the context to store and access the states of In this and the following part, we will introduce the implementation methods of Boolean interpreter system. expression evaluation system. When encountering the real implementation of Boolean expression, we have many details to deal with , and the process quality of these details directly influences the whole system's performance. Theseproblems are mainly incarnated in the following aspects : The first problem is to construct the abstract grammar tree. The interpreterstyle does not specify how to construct an abstract grammar tree in detail, that is tosay, the interpreter style does not involve syntax analysis. But when we are constructing an abstract grammar tree, we need to use a table-driven grammaranalysis program to finish this task; we can also use the recursive decline grammaranalysis program to construct the abstract grammar tree. The second problem is how to define the Evaluate operation. In fact, Evaluate operation does not need to be defined and implemented in the expression's classes. If we need to construct a new interpreter frequently, we can use the Visitor style indesign pattern theory, put the Evaluate operation in an independent "Visitor" object, this method may be better. For instance, a program design language has manyoperations on abstract gammar tree, such as type check, code optimization and code generation, etc. A proper way is to use a visitor, so as to avoid defining this operation in every class. The third problem is the shared terminals. In some grammars, many terminalsmay occur in the same sentences (such as true and false in-Boolean expressionevaluation system). In this case, it is better to share the copy of that symbol. Theterminal nodes usually do not store their positions in the grammar tree, in theprocess of evaluation, any context information they required is transferred by their parent nodes. So, the inner state and outer state in the terminal node are explicitly different. We can implement those using Flyweight In the implementation of Boolean expression evaluation system, we define two operations in design patterns. the Boolean expression. The first operation is Evaluate, which evaluate the value of the specified Boolean expression in the context, and this context must provide 6'true" or "false" for every variable. The second operation is Replace, which replaces a variable with an expression so as to generate new Booleanexpression. The Replace operation makes the system not only finish the evaluation of Boolean expression, but also do the grammar

### 第一图书网, tushu007.com



analysis of the Boolean expression. Because of the manuscript length constraint, we will not describe theimplementation details of each subclass. The interpreter style has an important characteristic: we can use manyoperations to "interpret" the same sentence. Among the three operations we defined in the Boolean Expression

, the Evaluate operation is the basic operation in the process of computing Boolean Expression. It interprets a Boolean expression and returns a simple result. But in the above system , we do not only have the Evaluate operation , the replacement and copy can also be treated as interpreter , and the only difference is the interpretation for the sentence.

# 第一图书网, tushu007.com



#### 版权说明

本站所提供下载的PDF图书仅提供预览和简介,请支持正版图书。

更多资源请访问:http://www.tushu007.com