<<软件体系结构>>

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

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 andimplementation of algorithm were the crucial process of software designing then.However, more and more tasks in low level, such as memory management andnetwork communication, have been automatized or at least can be reused with littleeffort and cost.Programmers and designers, with the help of high level programminglanguages and wieldy development tools, can pay more attention to problems, ratherthan bury themselves into the machine code manuals. However, the side effect ofthese 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

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 BooleanExpression.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 TerminalExpression （such as VariableExpression and Constant）.This type of role implements the Evaluation operation in the BooleanExpression that arerelated to terminals， every terminal in the Boolean Expression needs an objectinstance of this class. The third role is NonterminalExpression（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 NonterminalExpression 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 interpreter system. In this and the following part， we will introduce the implementation methods ofBoolean expression evaluation system. When encountering the real implementationof Boolean expression， we have many details to deal with， and the process qualityof 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 areconstructing 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， Evaluateoperation 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 andcode generation， etc. A proper way is to use a visitor， so as to avoid defining thisoperation 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 theirparent nodes. So， the inner state and outer state in the terminal node are explicitlydifferent. We can implement those using Flyweight design patterns. In the implementation of Boolean expression evaluation system， we define twooperations in the Boolean expression. The first operation is Evaluate， which evaluatethe value of the specified Boolean expression in the context， and this context mustprovide 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 evaluationof Boolean expression， but also do the grammar 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 definedin the BooleanExpression， the Evaluate operation is the basic operation in theprocess of computing Boolean Expression. It interprets a Boolean expression andreturns a simple result. But in the above system， we do not only have the Evaluateoperation， the replacement and copy can also be treated as interpreter， and the onlydifference is the interpretation for the sentence.　　……

版权说明