软件开发 OPEN描述 软件过程 (17)
CASE

计算机科学 1994 Vol. 21 №. 1

71-74

软件过程的 OPEN 描述与面向过程的 CASE 环境

TP311.52

王 晖 张 然
(复旦大学计算机科学系 上海 200433)

摘要 This paper introduces an object-oriented and Petri-net-based software process description OPEN. Petri-net-based description, which is able to express the task of every phase of software development and their relationships more clearly, is helpful for the management and execution of the project. Object-oriented approach can raise the development efficiency and quality of software process model. This paper also introduces one sort of development model of software process and architecture of process-oriented CASE.

1. 引言

软件是一种无形的逻辑产品，是软件开发人员的创造性思维的产物。尽管我们已有了一些软件质量的度量方法，但在软件开发过程结束以前，我们还无法对其质量作出精确可靠的预测。到目前为止，我们只能说："合适的开发方法和良好的组织管理是成功的一半。"

过去对软件开发过程的描述是以不很严格的文档形式(如开发计划)存在的，甚至只存在于项目管理人员的头脑中，其执行过程具有很大的随意性。各类开发人员对开发过程常常有不同的理解，影响了相互间的交流与合作，妨碍了开发工作的顺利开展。另一方面，在一个软件开发完成以后，管理人员难以对开发工作作出精确的分析总结。以往的开发经验难以在以后的工作中有效地继承，每次开发活动的规划几乎都要从零开始。

软件过程就是为解决这些问题而提出的。过程是开发某种产品，实现某种任务的系统化的手段[5]。软件过程试图通过对软件开发过程加以系统严格地描述，为开发人员提供一个标准的、无歧义的软件开发规划，使他们相互之间能更有效地交流。如果软件过程是用形式化语言描述的，就可以像软件一样进行设计、修改，并在计算机上运行，使软件开发过程的管理更加高效。同时，软件过程描述还可容易地被以后的软件开发活动加以复用，有利于总结开发经验，少走弯路。软件工程领域中的许多成果都能够在软件过程的框架内统一起来，形成一个整体。软件过程成熟度(软件开发组织和个人对软件过程的认识与实践水平)已成为衡量一个国家软件产业水平的重要标志[2]。

以软件过程概念为核心的 CASE 就是面向过程的 CASE，它可以帮助用户描述出软件过程，并在项目管理人员的控制下运行软件过程，使软件的开发有章可循，提高软件产品的质量和开发效率。

2. 软件过程模型的描述

软件过程模型可用多种形式加以描述，如基于过程型程序设计的 APPL/A[6]、基于 JSD 的 JMSOP[7]等。本文主要考虑以 Petri 网为基础的描述方法。

2.1 基于 Petri 网的过程模型描述

基于 Petri 网的过程模型描述方法有多种变体(如[1][3][4])，其中一类([1][4])将软件过程描述主要分为活动类型集合和对象类型集合两部分。

活动类型与 Petri 网中的 T-结点相对应，

它是软件开发过程中各项活动的抽象。其定义一般都包含前置条件、后继条件、活动语义等。前置条件是活动输入对象属性的组合，后继条件是输出对象属性的组合。活动的语义是一段可执行的软件片断，它定义了活动是如何进行的。

对象类型与 Petri 网中的 S-结点相对应，是各开发活动产生的文档和数据的抽象。

图 1 是一个简单的过程模型片断的图形表示。

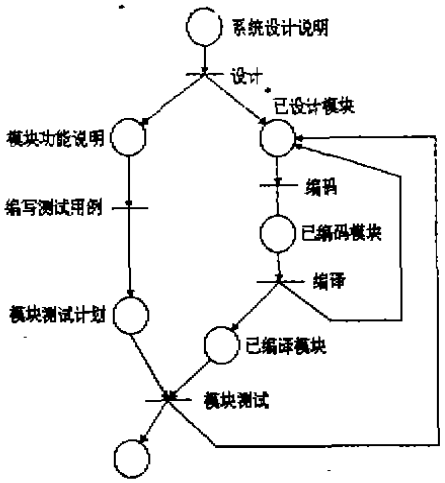

图 1 Petri 网表示的过程模型

过程模型的 Petri 网描述基于对软件开发活动的功能分解，由于 Petri 网具有较强的表达能力，因而能够比较准确地反映实际开发过程。

2.2 面向对象的 Petri 网过程模型描述

OPEN 描述试图通过将面向对象程序设计融入软件过程的描述中，使过程模型的开发、运行和维护能够以面向对象的方式进行，提高过程描述本身的质量。OPEN 描述包含对象类定义和活动类定义两部分。

2.2.1 对象类定义

下图是对象类 Module 的定义。

```
Object subclass Module is
Data
    attr;          Attribute;
    design;        Text;
    code;          Text;
    obj;           ObjectFile;
    err;           Text;
Method
    ATTR return Attribute is
        Begin
            return attr;
        End;
    SET_ATTR(attribute;Attribute) is
        Begin
            attr;=attribute;
        End;
    GET_DESIGN return Text is
        Begin
            return design;
        End;
    GET_ERR return Text is
        Begin
            return err;
        End;
    GET_CODE return Text is
        Begin
            return code;
        End;
    SET_CODE(file;Text) is
        Begin
            code;=file;
        End;
End Module;
```

图 2 对象类 Module 的定义

对象类的定义包含对象的数据结构和对象允许的操作两部分。对象的数据结构在对象外是不可以直接存取的，它被封装在对象内部，只有通过在方法中定义的操作才可以读取和修改对象内部的数据。

对象类定义了一类具有相同结构和界面的对象。对象类并不一定与 Petri 网中某一 S-结点对应。与 S-结点对应的是具有一定属性的同类对象的集合，与 Petri 网中 token 相对应的是对象实例。图 1 中“已设计模块”和“已编码模块”都是 Module 的实例集合。

对象类之间可以相互继承，这样可以提高过程模型的可复用性，提高设计效率。

2.2.2 活动类定义

下图是活动类 EditCode 的定义。

```
Activity subclass EditCode is
Parameter
    mod;in out Module;
Method
    PRECONDITION return Bool is
        Begin
            return(mod. ATTR=designed);
        End;
    POSTCONDITION return Bool is
        Begin
            return(mod. ATTR=coded);
        End;
    SEMANTICS is
        Var
            file;Text;
        Begin
```

```
            display(mod.GET_DESIGN);
            /* 阅读模块设计 */
            file:=mod.GET_ERR;
            if (file( )nil)
                display(file);
            /* 阅读编译错误信息 */
            endif;
            file:=mod.GET_CODE;
            edit(file);
            /* 编码 */
            mod.SET_CODE(file);
            mod.SET_ATTR(coded);
    End;
End EditCode;
```

图 3　活动类 EditCode 的定义

活动类的定义包括活动的参数、数据结构和方法。参数定义了活动的输入输出的类型。数据结构定义了活动内部的数据，活动实例外部只有通过方法才能对数据进行操作。方法中有三个(PRECONDITION，POSTCONDITION，SEMANTICS)是所有活动类都具有的。PRECONDITION 是关于活动输入的布尔函数，POSTCONDITION 是关于活动输出的布尔函数，SEMANTICS 是对活动输入输出进行的操作。

活动类同样可以继承。所有活动类都是类 Activity 的子类，每个活动类中的 PRECONDITION、POSTCONDITION 和 SEMANTICS 都是对 Activity 中相应方法的重载。

3. 软件过程的设计和运行

软件过程的设计和运行可以用图 4 的模型表示[4]。

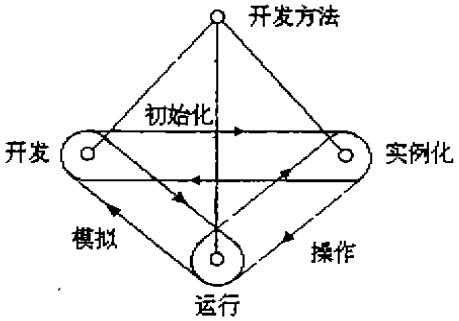

图 4　软件过程开发模型

可以看到，软件过程的开发和运行是一个在一定开发方法指导下往复地、增量地实现的过程。不同的开发方法实际上决定了不同的面向过程 CASE 的工作方式。

3.1 模拟周期

在模拟周期中，过程模型开发人员根据需求说明，选择已有的过程模型加以修改、组合或重新设计，产生一个新的完整的过程模型。对于 OPEN 描述而言，过程模型实际上就是一些相关的对象类和活动类的定义。由于这些描述可以通过面向对象的机制加以复用，使得开发人员可以充分地利用以往的工作成果。

模型产生后，需要对模型进行模拟运行加以测试。根据测试结果对模型加以裁剪或重新设计。最后得到一个合乎需要的过程模型。

3.2 初始化周期

在初始化周期中，要对模拟周期产生的过程模型进行实例化，将各种资源赋给软件过程。由于实际资源往往存在一些特殊的限制，原有模型可能不适用，所以必要时要返回模拟周期进行模型裁剪或重新设计。

3.3 操作周期

在操作周期中，软件过程投入运行，进行软件开发的实际活动。软件过程的执行过程是动态的，不仅过程实例会发生变化，模型也可能根据实际需要改动。

4. 面向过程 CASE 的体系结构

面向过程 CASE 的体系结构如图 5 所示。

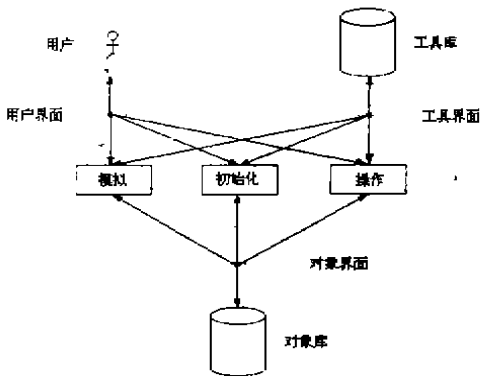

图 5　面向过程 CASE 体系结构

用户包括使用 CASE 的所有人员。工具库包含所有用于过程开发与软件开发的工具。用

户和过程都可以通过工具界面引用这些工具。对象库中包含了过程开发和运行所需的所有信息，对这些信息的存取是通过对象界面进行的。模拟、初始化、操作三个部件分别与过程开发中三个同名周期相对应。三个周期中进行的工作分别通过这三个部件进行。

5. 结论

以上我们着重介绍了面向对象的基于Petri网模型的软件过程描述方法OPEN。

使用Petri网描述软件过程的主要优点在于，类Petri网描述能够显式地指出软件开发过程中产生的各种产品对象和生成这些对象的各项开发活动，能够比较直观地表示对象之间、活动之间以及对象与活动之间的相互依赖的顺序关系或相互独立的并行关系。管理人员可以利用类Petri网描述进行各种静态和动态的分析，找出开发活动中的瓶颈，合理安排进度、分配任务、调整资源，从而为用工程化方法进行项目管理提供有力的支持。

引入面向对象技术的目的是为了将面向对象程序设计的优越性结合到软件过程模型自身的设计过程中，使开发的过程模型具有良好的数据封装性和继承性，便于复用和维护，提高过程模型的质量与开发效率。

OPEN描述尚存在一些不够完善的地方。软件开发活动中一些重要的内容，如版本管理等，还不能被OPEN在较高层次上直接地加以表示。今后的工作将在这些方面展开。

参考文献

[1]W. Deiters, and V. Gruhn, Managing Software Processes in the Environment MELMAC, Software Eng. Notes, vol. 15, no, 6, pp. 193-205, 1990

[2]W. S. Humphrey, D. H. Kitson, and J. Gale, A Comparison of U. S. and Japanese Software Process Maturity, in Proc. 13th Int. Conf. on Software Eng.

[3]L. Liu, and E. Horowitz, A Formal Model for Software Project Management, IEEE Trans. Software Eng., vol. 15, no. 10, pp. 1280-1293, 1989

[4]N. H. Madhavji, and W. Schafer, Prism-Methodology and Process-Oriented Environment, IEEE Trans. Software Eng., vol. 17, no. 12, pp. 1270-1283, December 1991

[5]L. Osterweil, Software processes are software too, in Proc. 9th Int. Conf. on Software Eng. Los Alamitos, CA; IEEE Computer Soc. Press, pp. 2-13, 1987

[6]S. M. Sutton, O. Heimbigner, and L. J. Osterweil, Language Constructs for Managing Change in Process-Centered Environments, Software Eng. Notes, vol. 15, no. 6, pp. 206-217, 1990

[7]R. Zhang, Y. Zhang, and Y. Ye, The software Process Model JMSOP, Proc. Beijing Int. CASE Symposium' 91, 1991

（接第54页）

5. 应用实例

例 取一个模糊函数：

$f=(a\vee b)\wedge(c\vee d\vee e)$ （a,b,c,d,e均取0，或1）准备用一个四层神经网络实现之。

事先用(5-5-5-1)网络，学习过程中取：

$\varepsilon_1=2.65\times10^{-4}$ $\varepsilon_2=1.12\times10^{-4}$

$\eta=1.0$ $\alpha=0.9$ $\theta_1=0.9$ $\theta_2=0.001$

采用自构形学习方法。表1给出了r_{ij}和θ_i。

由表1可看出，隐层1中，1.4.5合并，2.3合并。

隐层2中，1.2.3.4四个隐节点合并，5与阈值合并。

通过自构形，原来的(5-5-5-1)⇒(5-2-1-1)网络。