OOPSE——一种基于C++/Java的程序分析系统

OOPSE—a Program Analysis System for C++/Java

刘小东　李必信　杨朝晖　李宣东　郑国梁

（南京大学计算机软件新技术国家重点实验室　南京210093）

（南京大学计算机科学与技术系　南京210093）

Abstract　With the development of object-oriented technology, more and more tools and environments are designed for supporting object-oriented programming. They facilitate software development. We develop a program analysis system for C++/Java—OOPSE, which is used for program slicing, fluctuation analysis, software matrics and type analysis. In this paper, we introduce it in detail.

Keywords　Program analysis, Type analysis, Program slicing, Software matrics, Fluctuation analysis

1　引言

面向对象技术自八十年代以来，在软件的各个领域得到了广泛应用，基于对象技术的面向对象软件开发方法也随之兴起，并被广泛采用。面向对象软件开发方法强调采用面向对象的观点认识客观世界，采用面向对象的方法模拟客观世界，使得软件结构和问题结构相一致，相对降低了软件的复杂性，方便了软件的设计、编程、维护和使用，从而特别适合于大型的、复杂的软件系统的开发。尽管面向对象的软件开发方法具有很多优点，但也存在着一些问题。例如，语言的灵活性、描述能力和程序的可靠性、功能之间难以统一，多态性和动态定连使得运行时很难确定在给定的程序点对象的动态类型，封装与继承的矛盾，以及多继承机制复杂的语义等使得人工管理面向对象的软件开发过程变得困难，同时，这也会导致软件开发效率的降低。为此，人们纷纷研究各种类型的支持面向对象软件开发过程的工具和环境，其中最具代表性的有：由北京大学以杨芙清院士为首开发的"青鸟工程"工具，由贝尔实验室的Prem Devanbu和Laura Eaves共同开发的Gen++，由Danny B. Lange和Yuichi Nakamura共同开发的环境Program Explorer等等。这些工具或环境的研制给面向对象的程序设计带来了极大的便利。为了探索新的软件分析支撑工具和环境设计和开发的新思路、新方法，围绕面向对象程序设计工具和环境，我们重点研究开发了一种基于C++/Java的程序分析系统—OOPSE，本文详细介绍了OOPSE的设计思想、总体框架和实现技术。

2　设计思想

本项目的研究目标是围绕面向对象程序设计支撑工具和环境，开发基于C++/Java的实用性程序分析工具，探索软件工具和环境设计和开发的新思路、新方法。

本项目的研究工作首先是对C++/Java程序进行代码分析，主要利用程序切片技术、度量和统计技术以及波动分析技术分别对C++/Java源代码进行粗粒度切片和分层切片、度量分析、统计分析以及波动效果分析等，从而达到对C++/Java程序进行测试、调试、分析、理解和维护的目的，然后在源代码分析的基础上，抽象出可见方法类层次图(VM-CHG)，并进一步计算继承集和改写集，确定改写边界，从而解决虚函数调用问题；另一方面，利用类层次图(CHG)和等价路径进行子对象识别以及可见方法和主导方法确定。再在类层次图(CHG)、程序调用图(PCG)、类型表(TT)以及过程间控制流图(ICFG)的基础上进行适当的类型分析。最后，利用可视化技术，对分析的各种结果分别进行显示或模型化。

3　总体框架

3.1　系统组成与结构

OOPSE主要由扫描器、知识库、分析器、统计工具、波动检测器、度量工具和切片工具等部分组成。系统的总体结构如图1。

3.2　系统功能

该环境提供普通的文字编辑功能，有多个文本编辑窗口。但作为面向对象程序设计支撑环境，其主要功能是：源程序浏览扫描、特征信息收集、分析综合、波动性分析、监测修改变动对程序的影响、程序切片、静态信息综合显示、产生类继承图(CHG)和虚函数调用图(VFCG)等。下面分别介绍各组成部分的功能：1)扫描

器：主要进行简单词法分析和语法分析，生成其他工具所需的中间数据，并将这些数据存放在知识库中。2)知识库：知识库是其他工具进行工作的基础，为其他工具提供数据。3)分析工具：从知识库中提取数据进行综合分析，运用相关的算法、抽象出继承关系和方法调用关系，并用图直观地表示出来。4)波动监测工具：监测程序中相关兴趣点的修改对整个程序的影响，必要时重新调整两个关系图。5)统计和度量工具：对程序的各种静态参数进行统计，并用表图形式直观地显示出来。6)切片工具：利用知识库提供的知识，并结合相关的程序切片标准来计算 Java 程序的各种静态切片。7)可视化工具集：该工具集提供了各种支持可视化的工具，例如类层次图(CHG)、虚函数调用图(VFCG)、坐标图以及直方图等的设计、绘画和显示等都是由该工具集来完成的。

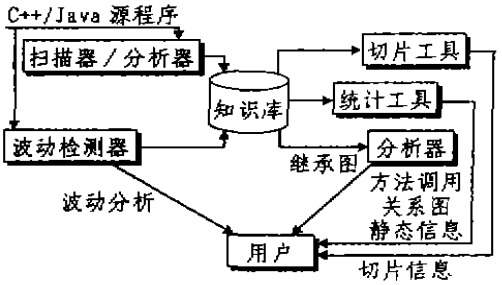

图1

3.3 系统特点

该工具具有下列设计和实现上的特点。1)显示直观：为了让用户能更好地理解程序结构及其相互关系，我们用各种图表将其直观地显示出来。2)界面友好：采用流行的多文本编辑窗口和各种菜单命令，接口简洁明了，用户操作方便。3)交互浏览：用户不仅能从图表上直观地获取程序信息，而且可以在程序和图表间自由切换，实现了交互式浏览。

4 实现技术

首先，作为一个集成的程序设计开发环境，我们沿袭了目前通用的多文档多窗口编辑环境。为了能有效地减少用户的语法错误，以及提供漂亮清楚的显示，我们也将 C++中的关键字用彩色字符显示。这些功能在强大的 VC 5.0的帮助下，它们的实现变得非常的容易，这里我们就不详细叙述了。下面我们主要讨论工具中几个主要组成部分的实现技术。

4.1 知识库的组织及主要数据结构

知识库中的知识是以类为单位组织的、关于一个类的所有知识被组织在一个 CClassInfo 类的实例中，源程序中的每一个类对应于知识库中的一个 CClassInfo 对象，所有的 CClassInfo 对象就构成了 m_pArray 所指向的对象数组。

CClassInfo 的具体结构如下：

```
class CClassInfo:public CObject //m_pArray 中的每一个
    Object 都是它的对象
{
public
    CString m_ClassName;
    CString m_FileName;
    CString m_Path;
    CString m_Fathers; //多个父类时,中间用","隔开
    int m_nDefBegLine;
    int m_nDefEndLine;
    int m_nCountFather; //父类的个数
    int m_nNo; //类编号,供画类继承图时用
    CObArray m_aVars; //记录该 class 中的变量信息
    CObArray m_aMethods; //记录该 class 中的方法信息
    CObArray m_aFriends; //记录该 class 中的友元信息,
                         包括友元类和友元方法
};
```

CClassInfo 中 m_aVars，m_aMethods，m_aFriends 均是对象数组，分别存储该类中的变量、方法、友员的信息，其具体结构分别如下：

```
class CVarInfo:public CObject
{
public:
    CString m_Name;
    CString m_Type; //例如,enum,struct,union
    CString m_TypeName; //例如:结构名、枚举名……
    BOOL m_bSimple; //false—复合类型;true—简单类型
    BOOL m_bStatic;
    int m_nDefLine;
    int m_property; //1—私有;2—保护;3—公有
};

class CMethodInfo:public CObject
{
public:
    int m_nProperty; //1—私有;2—保护;3—公有
    CString m_FileName; //指.cpp 文件的文件名
    CString m_Path; //指.cpp 文件的路径
    CString m_Name;
    CString m_Paras; //只包括参数类型而不包括参数名,
                     用","隔开
    CString m_RetType;
    int m_nDefBegLine;
    int m_nDefEndLine; //在何处定义
    int m_nDeclareLine; //在何处申明
    BOOL m_bRedefined; //0—redefined;1—self-defined
    BOOL m_bVirtual;
    BOOL m_bStatic;
    BOOL m_bInline;
    BOOL m_bOperator; //1—操作符重载
    enum_style
    CONSTRUCTION,
    DECONSTRUCTION,
    NONE  m_style;
};

class CFriendInfo:public CObject
{
public:
    CString m_ClassName;
    CString m_Name;
    CString m_Paras;
    CString m_RetType;
    int m_nDefBegLine;
    int m_nDefEndLine; //在何处定义
    int m_nDeclareLine; //在何处申明
```

BOOL m_bOperator,
BOOL m_bClass; // true--类;false--方法.

4.2 扫描分析

由于扫描器的处理对象是多个源程序,需要对程序进行扫描,但是我们的目的不是编译,所以我们必须自己编写合适的词法语法分析程序。采用的方法仍然是传统的自顶向下的递归子程序法。实现时,每次读入源程序的一个单元(单元是指诸如标识符、保留字、分隔符、隔行符这样的字符串/字符),以此为单位,依照标准C++/Java的语法规范,对程序进行识别。生成的中间数据将填入上面所示的数据结构中。在分析过程中,我们一般假定用户提供的源程序是语法正确的。所以,这里的语法分析将不是一般意义上的含义。我们主要是为了将程序中的有关面向对象机制的信息提取出来,同时为可能的一些异常情况加以提示。

4.3 程序切片

程序切片是一种分析和理解软件的方法,主要是根据数据流分析和控制流分析,把程序中与指定数据项或数据结构有关的程序部分抽取出来,滤掉与其无关的语句,并运用符号演算和逻辑推理等技术手段,帮助人们详细分析程序对指定数据项或数据结构的作用,找出执行特定路径的前置条件等。现在,程序切片技术已经运用于程序调试、代码理解、软件测试、逆向工程和度量分析等方面。

本系统中的切片子系统主要用来计算C++/Java程序的切片,采用的切片算法是基于系统依赖图的两步图形可达性算法[3,4]、粗粒度切片算法和分层切片算法[1,2]。

4.4 波动分析

程序的各个语句之间并不是孤立的,由于程序语句间存在数据依赖关系和控制依赖关系,当一个程序的某个部分被修改时,可能潜在地影响到程序的其它部分。这就是程序中存在的波动效应。

波动监测将记录下修改地点,判断是否是我们的兴趣点,是则判断这次修改将对那些类和对象产生影响,同时对知识库进行修改,以对继承图和调用图进行调整。具体的方法是保存修改之前的类信息作为另一个副本,以便将修改前后的类定义加以对照。不同类型的修改操作,在类信息中所表现出来的变化是不同的,因此在比较类信息的两个副本时要依据不同的标准加以判别。

4.5 度量分析

软件度量是指按照一定的度量理论和准则,对源程序的大小、结构复杂度和科学复杂度等进行度量,以辅助预测软件中最容易产生错误的地方,从而对测试计划和分配测试资源提供支持。软件度量主要包括:程序复杂性度量、模块性度量、可修改性度量、可移植性度量、可扩充性度量、可靠性度量、可维护性度量等等。

本系统主要采用的度量方法有基本度量(basic matrics)、CK度量(Chidamber and Kemerer metrics)和Aoki度量(Aoki metrics)。

4.5.1 基本度量(basic metrics) 基本上是以类为单位的、基于传统方法的度量,用于统计程序中的最基本的信息,有七个度量标准:1)LOC:每个类的代码行数。2)NOA:每个类的属性的个数。3)NIV:每个类的实例变量的个数。4)NCV:每个类的类变量的个数。5)NOM:每个类的成员方法的个数。6)NIM:每个类的实例方法的个数。7)NCM:每个类的类方法的个数。

4.5.2 CK度量 主要是从总体上刻画面向对象系统的结构特征,本工具中包括一张表和六个图,表中统计了六个度量标准:1)WMC类方法的复杂度的带权和。2)DIT类在继承树中的最大深度。3)NOC继承树中一个类的直接子孙的个数。4)CBO与该类耦合的类的数目。5)RFC若RS是该类的响应集合,则RFC=|RS|。在此工具中算法如下:$RS=M_i\cup R_i$,其中M_i为该类中的成员方法集合,R_i指被方法M_i调用的方法集合。6)LCOM类内聚缺乏度。每个度量标准都有一个直观的图型。

4.5.3 Aoki度量 主要是针对继承树的子树的。继承子树是指继承树中的连通子图,其中的类和此树外的任何类不存在任何继承关系。本工具中包括一张表及四个图,其中表中统计了四个度量标准:1)HF表示该类在继承子树中的相对深度。2)RF表示该类在继承子树中的相对引用程度。3)PF表示该类中的成员方法和继承子树其他类的重名程度。4)CP表示该类被继承子树中的类引用的绝对次数。每个度量标准也都有一个直观的图型。

4.6 统计分析

一般而言,在一个稍大一点的工程里,将不可避免地包含有非常多的类、对象及其相应的方法。将这些成分的基本信息统计出来,对软件的维护是很有帮助的。我们认为最清晰的表示方法还是直观显示。对那些互相没有关系的,仅仅是数据罗列的信息,可以用表的形式,如软件有多少个文件,多少类,多少行,类的耦合数,同名属性的个数,继承深度和高度等等,而类的特性可用图直观显示。将每个类作为一个结点,它的下一层将是它封装的数据和方法,不同的结点用不同的图形表示。图和源程序定义点可互相切换,用户双击希望了解的结点,将回到源程序定义点。

结束语 OOPSE是我们开发的一种基于C++/Java语言的程序分析系统。该系统采用程序切片技

(下转第52页)

次迁移活动，再生成目的迁移控制任务 dmig。

(3)dmig 首先建立一 TCP socket 端口，通过自己的信号处理函数将该端口号在适当的(与 smig 同步的)时刻发向 smig(由 pvmd 转发)，smig 获取 dmig 的 TCP 端口后，与 dmig 建立 TCP 连接。然后通过 TCP 连接将取出的 oldtid 进程的图像发送到 dmig，并由其安装，smig 和 dmig 完成任务后相继退出。

(4)mmig 由反馈消息得知迁移工作基本完毕后，向全体相关 pvmd 发消息，通知相关任务继续运行；源 pvmd 强行终止 oldtid 任务；目的 pvmd 向 newtid 进程发出任务重启触发信号；newtid 重启后，从信号处理程序(restart)返回迁移断点继续运行。

4 PVM 任务迁移的实现

(1)在 ddpro.c 中增加 dm_migxxx 类任务迁移消息协议处理函数。在 netswitch 的入口表中增加消息 tag 为 DM_MIGXXX 的 dm_migxxx 函数入口，以处理 pvmd 之间的有关任务迁移的控制消息。

(2)在 tdpro.c 中增加 tm_migxxx 类任务迁移消息协议处理函数。在 loclswitch 的入口表中增加消息 tag 为 TM_MIGXXX 的 tm_migxxx 函数入口，以处理控制迁移任务发来的有关任务迁移的控制消息。

(3)在 lpvm.c 中增加迁移信号处理函数的 stub migtidstart，该函数主要负责相关任务消息处理及迁移活动的同步。在 pvmbeatadk()函数中的消息处理函数初始化部分增加迁移活动映射表 migtaskslist 的初始化，并安装 migtidstart 函数到 mhandle 表中(restart 安装到 lpvm.c 中)，用于进程图像迁移后的任务重启。

(4)修改 spawn 函数，并增加相应的 pvmd 协议处理函数，使 spawn 一个新任务时，主 pvmd 一致性地登记该相关任务组的 family[]邻接表项。在主 pvmd 的 pvmd.c 的 main()函数中初始化 family。

(5)相关任务组中有任务处于 spawn 过程，则该相关任务组不能进行任务迁移活动。负载均衡任务在发出 DM_MIGREQ(oldtid, newtid)之前应作相应的检测。

5 需进一步研究的问题

我们在四台不同配置的 PC 机组成的 PVM 系统上对该方法做了实验，成功地进行了 PVM 任务迁移，经过一段时间的运行实践，证明本文方法是可行的，且达到了预期目的。但本文只研究了在给定条件下一般的用户 PVM 任务的迁移问题，进一步开展的工作有：

(1)组机构和 mailbox 在有关任务迁移时的相应处理技术。

(2)用户任务一般的 fork 子进程(尚未 PVM 任务化)在父进程迁移时的处理。

(3)在任务迁移过程中磁盘文件 I/O 的一般解决方法。

(4)协议的优化，以进一步提高迁移效率和实时性。

参 考 文 献

1 Sunderam V S. PVM: A framework for parallel distributed computing. Concurreney, Practice and Experience, 1990, 2(4): 315～339

2 Geist A, et al. PVM: Parallel Virtual Machine-A Users' Guide and Tutorial for Networked Parallel Computing. Massachusetts: MIT Press, 1994

3 鞠九滨，魏晓辉，徐高潮，尹玉. DPVM：支持任务迁移和排队的 PVM. 计算机学报，1997，20(10)：872～877

4 鞠九滨，王勇. 调度 PVM 任务. 计算机学报，1997，20(5)：470～474

5 张博，王纪成，周兴社. SPMD 模式 PVM 任务均衡分配研究与实现. 计算机研究与发展，1997，34(8)：588～593

6 Dikken L. DynamicPVM: Task migration in PVM. [Technical Report]. Shell Research ICS/155. 1. Nov. 1993

7 Dikken L, et al. DynamicPVM: Dynamic load balancing on Parallel systems. High-Performance Computing and Networking, 1994, 797: 273～277

8 鞠九滨，魏晓辉，郭雷. 利用检查点机制在 PVM 中实现进程迁移. 软件学报，1996，7(3)：175～179

9 Konuru R, et al. A user-level process package for PVM In. 1994 Scalable High-Performance Computing Conference. IEEE Computer Society Press, May 1994. 48～55

10 Casas J, et al. MPVM: A migration transparent version of PVM: [Technical Report, CSE-95-002]. Dept. of Computer Science and Engineering, Oregon Graduate Institute of Science & Technology, February 1995

(上接第20页)

术、度量和统计技术和波动分析技术等，对 C++/Java 语言程序进行分析和理解，显示直观，界面友好，用户可以在程序和图表间自由切换，实现了交互式浏览，达到了方便软件人员进行面向对象程序设计的目的，也是我们在探索软件工具和环境设计和开发的新思路、新方法上的新的尝试。

参 考 文 献

1 李必信，郑国梁，等. 软件理解研究与进展. 计算机研究与发展，1999，36(8)：897～906

2 李必信，郑国梁，等. 一种分析和理解程序的方法—程序切片. 计算机研究与发展，2000，37(3)

3 Weiser M. Program Slicing: Formal, Psychological and Practical Investigations of an Automatic Program Abstraction Method: [Ph. D thesis]. The University of Michigan, Ann Arbor, Michigan, 1979

4 Krishnaswamy A. Program Slicing: An Application of Object-oriented Program Dependency Graph: [Technical Report TR94-108]. Department of Computer Science, Clemson University, 1994