Program Comprehension: A Method of Generating Visualized UML Class Diagram

GuHui
College of Computer Science and Technology
Zhejiang University of Technology, ZJUT
Hangzhou, China
gh@zjut.edu.cn

WangHui
College of Computer Science and Technology
Zhejiang University of Technology, ZJUT
Hangzhou, China
wanghuinxpy@163.com

***Abstract*—Program comprehension is an important research content of software engineering. This paper presents a Program comprehension visualization method of using UML class diagram. This method has two parts, first, The program source code is abstracted into database table, the form used to summarize and express program structure and key information, Then, the table was transformed into UML class diagrams, The class diagram can be effective to help understanding program structure and other key information.**

Keywords- program comprehension; table abstract; UML class diagram; visualization

I. 引言

IEEE2004 中解释程序理解是一个从计算机程序中获取知识信息的过程。这些知识信息可以用于程序修改、程序重用和文档整理等方面的工作上。成功的软件理解方法是编译器结构的低层代码分析，逆向工程领域的高层分析和软件可视化技术三者之间的协同结合[1]。

程序理解在软件维护和逆向工程方面有广泛的应用[2]。程序理解活动包含三个部分：阅读有关的文档、阅读源代码和运行程序[3]。第一部分是浏览不同来源的文档，譬如现存的文档需求规范说明书、概要设计说明书或者详细设计说明书，建立对系统整体的分析理解。第二部分是通过略读程序代码，对系统进行整体和部分的分析，可以获取一些数据结构和相应算法的描述，对系统的结构和思想全面理解。第三部分是通过运行程序，动态的分析一些难以理解的代码段。

统一建模语言（Unified Modeling Language，UML）用于对面向对象开发系统的产品进行说明、可视化和文档编制的一种标准语言[4]，可以有效描述和表示程序理解的结果。根据不同的理解层次与需求，UML 便于软件理解在不同的抽象层次上建立目标程序的概念模型，建立从问题领域到程序设计领域的映射集。我们所作的工作包含两个部分：第一部分是根据最基本的源代码视图，对其进行抽取和分析，转化为数据库表格形式。第二部分是将表格抽象为可视化的 UML 图，方便直观的理解程序。

作者简介：
古辉(1956-)，男，山西人，教授，主要从事软件工程、程序理解和多媒体应用技术等方面的研究；
王慧(1985-)，女，宁夏人，硕士研究生，主要研究方向为程序理解。

II. 源代码到数据库表格的抽象

UML 类图的可视化分两个步骤：首先从计算机程序中抽取知识信息并分类放入数据库的表格中去；然后再根据数据库表格的内容计算画出相应的类图（可视化）。总体上，类包含三个部分：类名，成员和该类的方法。UML 类图的可视化过程，就是从程序源代码中把这三个部分提取并抽象出来。

源代码到数据库表格的抽象的规则：

(1)将每个类的成员定义为一个表格。根据代码中的信息，我们把类中简单的成员，如整型、字符型等，将这些简单的成员直接映射为表格中的字段。如果类中的某些成员是复杂数据类型的，如结构体等，则映射为表格中的多个相应字段。对于其他一些可以由其他成员生成的派生属性，我们不需要映射到表格中去，因为派生属性可以由表格的查询操作得到。表格中，类的成员与表格的字段间的一一对应。成员包含其名称，类型和可见性。

(2)将每个类的方法也定义为一个表格，提取方法的名称，返回值类型和可见性，完善类的信息。

每个类都建立上述两种表，使得每个类的信息就比较完整，可以很方便地查询类的信息。按照这样的规则映射后，可以将图 1 中 Person 类的代码转换为表 1 和表 2。

```
class Person
{
public:
   void   work();
   char   sex;
   int    IDcard;
   string name;
};
```

```
class teacher:public Person
{
public:
   int   teacherno;
   void teach();
};
```

```
class student:public Person
{
public:
   int   studentno;
   void study();
};
```

图 1 三个类

表 1 类 Person 的成员

type	char	int	string
name	sex	IDcard	name
visibility	public	public	public
Classname	Person	Person	Person

978-1-4244- 7618-3 /10/$26.00 ©2010 IEEE

表 2 类 Person 的方法

rtype	void
name	work()
visibility	public
Classname	Person

对于类的继承关系，父表是由父类的成员映射成的表，子表是由子类自身的成员映射成的表，这样可以减少不必要的冗余。然后再建立一张继承的表，用于表示类与类之间的继承关系，显示每一个类的子类和父类。根据图 1 中代码的继承关系，可以建立表 3。

表 3 继承关系

Classname	Person	teacher	student
subclass	teacher,student		
superclass		Person	Person

把程序代码中类的部分按照上述规则映射到表格后，可以很清楚地得出类的各种属性、方法以及类与类之间的关系，根据这些属性和关系再建立 UML 类就方便很多了。

III. 数据库表格转化为 UML 类图

类图显示出类、接口以及它们之间的静态结构和关系，它用于描述系统的结构化设计。类图最基本的元素是类或者接口。类的属性和操作之前附加一个可见性修饰符。加号(+)表示具有公共(public)，减号(-)表示私有(private)，#号表示受保护的(protected)。

为了将数据库表格中的程序信息转换为 UML 类图，首先给出几个定义：

定义 1：一个类 C=(Classname, Member, Function)，其中 Classname 表示类的名称，Member 表示类的属性，Function 表示类的方法。

定义 2：一个类的成员 M=(type, name, visibility)，其中 type 表示成员的数据类型，name 表示成员的名称，visibility 表示成员的可见性，visibility ∈ {“protected”, “public”, “private”}。

定义 3：一个类的方法 F=(rtype, name, visibility)，其中 rtype 表示方法的返回类型，name 表示方法的名称，visibility 表示方法的可视性，与定义 2 中的相同。

表格 1～3 能够满足生成 UML。根据表 1、表 2 和定义 1～3 可以获取 Person 类的信息，Classname=Person，Member={sex, IDcard, name}, Function={work}. 其中类中成员 sex=(char, sex, public), IDcard=(int, IDcard, public), name={string, name, public}, 类中方法 work={void, work, public}.根据生成 UML 类图的方法，可以得出类 Person 的类图。同样的，也可以得出类 teacher 和类 student 的类图，如图 2 所示：

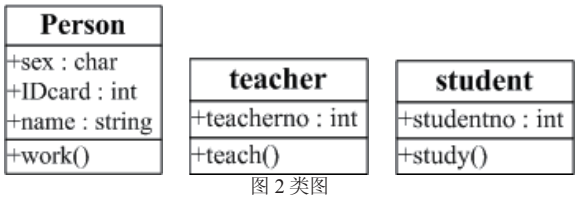

图 2 类图

最后还要确定类与类之间的关系，根据继承关系表得出相应的继承关系。依照继承关系（表 3），可以完善图 2 的类图表示，得到完整的 UML 类图，泛化采用空心的箭头直线来表示，如图 3 所示。

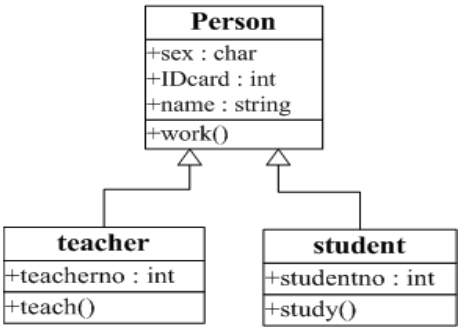

图 3 UML 类图

V. 结束语

本文主要研究了如何利用 C++代码生成 UML 类图。首先使用映射规则把代码抽象为数据库表格，然后从数据库表格中获取类的相关信息生成 UML 类图。本方法具有普遍的应用性，在实际中可以使得程序结构一目了然，便于程序代码被理解。此外，对表格的设计、表格的冗余也进行了一定的研究，提高了表格的使用效率。不足之处是没有生成相应的文档，在今后的工作中应当对文档的生成进行深入地研究。

致谢

浙江省自然科学基金项目：基于最小信息集的程序理解信息抽取模型研究（93531068）

REFERENCES

[1] W. L¨owe, M. Ericsson, J. Lundberg, and T. Panas, “Software comprehension - integrating program analysis and software visualization,”in 2nd Conf. on Software Engineering Research and Practise in Sweden, 2002.

[2] Christian Pich, Lev Nachmanson, George G. Robertson. “Visual analysis of importance and grouping in software dependency graphs,” ACM. Software Visualization.2008, Sep: 29-32.

[3] Grubb P，Takang A A．”Software Maintenance : Concepts and Practice,” Publishing House of Electronics Industry,2004.

[4] X.G.Wang, “UML Unified Modeling Practical Guide,” Beijing Tsinghua University Press, 2009.(In Chinese)