基于类之间的依赖关系确定类的规模

胡顺仁[1,2] 欧 阳[1]

(重庆工学院网络中心 重庆400050)[1] (重庆大学光电工程学院 重庆400044)[2]

摘 要 类之间的依赖关系,对于面向对象系统分析、设计和测试都有重要的意义。本文首先对类之间的依赖关系进行了定义和说明,并细分其为数据依赖和方法依赖,在此基础上,对类之间的依赖关系进行了度量,提出依赖度和被依赖度两种度量方法,并以此确定类地规模大小。

关键词 OOS,依赖关系,数据依赖,方法依赖,依赖度,被依赖度,分解,合并

Affirming Class Size Based on Dependency Relations among Classes

HU Shun-Ren[1,2] OU Yang[1]

(College of Electronic and Automation Chongqing Institute of Technology, Chongqing 400050, China)[1]

(College of Optoelectronic Engineering, Chongqing University, 400044, China)[2]

Abstract The dependency relations among classes are important for object-oriented analysis, design and test. In this paper, the definition of class dependency is first introduced, and fractionated it into data dependency and method dependency, then, this paper describes the two methods: depending metrics and depended metrics. Finally, two methods are introduced to affirm class size based on them.

Keywords OOS, Dependency relation, Data dependency, Method dependency, Depending metrics, Depended metrics, Decomposing, Merging

在面向对象系统分析和设计(OOA&D)过程中,类设计是一个重要而不可或缺的环节,有些类可以进一步细分,而有些类又可以合并再一起。确定类的规模大小没有一个严格的标准,实际设计过程中往往根据设计者本人的水平和经验来确定,人为因素造成类设计的质量良莠不齐,也影响面向对象设计的整个过程。

类之间的依赖关系是类之间联系中最重要、最普遍的一种联系,这种关系广泛的存在面向对象系统中。对类之间的依赖关系进行严格的语义定义和度量,并以此来确定类的规模大小[1~4]。

1 类之间的依赖关系的定义和度量

1.1 依赖关系的定义

在面向对象系统OOS中,S为OOS中的所有类的集合,记为:

$S=\{C_1,C_2,\cdots,C_n\}$,

其中,n为集合S中类的个数。我们定义类之间的依赖关系如下:

定义1 依赖关系D为定义在集合S上的二元关系,设存在两个任意的类C_i和C_j,且$C_i\in S, C_j\in S$,如果类C_i被修改、删除或移动后,则类C_j也会出现相应的变化,称C_j依赖于类C_i,记为:$D=\{\langle C_j,C_i\rangle \mid C_i\in S, C_j\in S\}$。

上述定义基本上描述了依赖关系的特点,但是这种描述是粗略和不完善的,没有能揭示出类具体是如何被修改、删除或移动的,所以我们需要进一步重新定义依赖关系。

由于每个类都有两个基本特征:属性和操作,可以将类看作为属性集合和操作集合的集合,类C_i和C_j记为:$C_i=\{A_i, M_i\}$, $C_j=\{A_j, M_j\}$。其中,$A_i=\{a_{i1},a_{i2}\cdots,a_{is}\}$, $M_i=\{m_{i1},m_{i2}\cdots,m_{iu}\}$。

A_i为类C_i中所有属性的集合,M_i为类C_i中所有操作的集合。s,u为类C_i属性和操作的数目。

$A_j=\{a_{j1},a_{j2}\cdots,a_{jt}\}$, $M_j=\{m_{j1},m_{j2}\cdots,m_{jv}\}$

A_j为类C_j中所有属性的集合,M_j为类C_j中所有操作的集合。t,v为类C_j属性和方法的数目。

那么,类C_i和C_j之间所有的各种关系都是$((A_i\cup M_i)\times(A_j\cup M_j))\cup((A_j\cup M_j)\times(A_i\cup M_i))$的子集(×为笛卡儿乘积),本文研究的类之间的依赖关系D也是其中的一个子集,定义如下:

定义2 设类C_i和C_j之间的依赖关系D为定义在集合$(A_i\cup M_i),(A_j\cup M_j)$上的二元关系,记为:

$$D=\{\langle x,y\rangle \mid (x\in(A_i\cup M_i)\wedge y\in(A_j\cup M_j))\vee(x\in(A_j\cup M_j)\wedge y\in(A_i\cup M_i))\}$$

其中,序偶$\langle x,y\rangle$表示元素x依赖于元素y,即元素y的变化会引起元素x的变化。

1.2 依赖关系的两种基本细分类型

如前所述,类具有属性和操作这两个基本的特征,类之间的依赖关系也是这两个基本性质外部表现。类之间依赖性的产生,正是类的属性和操作之间的相互依赖所表现。我们将为属性所引起的依赖称为数据依赖,记为R_{DD};另一个为操作所引起的依赖称为方法依赖,记为R_{MD}。这两种依赖类型实际上是类之间依赖关系的细分,其定义和描述如下:

1.2.1 数据依赖[5]

定义3 设存在两个属性a_{ir}和a_{js},且$a_{ir}\in A_i, a_{js}\in A_j$,若在类$C_j$中存在属性$a_{js}$对类$C_i$的属性$a_{ir}$的引用或类$C_i$中属

胡顺仁 讲师,硕士,主要研究方向为网络安全、软件工程,光电技术。欧 阳 副教授,硕士,主要研究方向为网络安全、软件工程和系统集成。

性 a_{ie} 通过参数形式传递给类 C_j 的属性值 a_{jf} 的情况，则称属性 a_{jf} 数据依赖于数据 a_{ie}，记为 $a_{jf}\ R_{DD}\ a_{ie}$。

定义4 设存在一个属性 $a_{ie}, a_{ie}\in A_i$，一个操作 $m_{jv}\in M_j$，若在类 C_j 中存在操作 m_{jv} 对类 C_i 中的属性 a_{ie} 调用的情况，则称操作 m_{jv} 数据依赖于属性 a_{ie}，记为 $m_{jv}\ R_{DD}\ a_{ie}$。

综上定义3，4可得：

定义5 设存在属性 $a_{ie}, a_{ie}\in A_i$，若类 C_i 中的属性 a_{ie} 的修改、删除、移动会引起类 C_j 中属性和操作的相应变化，则称类 C_j 数据依赖于类 C_i，记为 $C_j\ R_{DD}\ C_i$。

1.2.2 *方法依赖* 面向对象系统中存在各种各样的操作，由于类的继承性、多态性、动态联编、消息调用等因素，使得方法依赖变得要比数据依赖复杂。所以，研究方法依赖只能局限在静态上描述。

定义6 设两操作 m_{iu}、m_{jv}，且 $m_{iu}\in M_i, m_{jv}\in M_j$，如果存在操作 m_{jv} 调用操作 m_{iu} 或操作 m_{jv} 接受操作 m_{iu} 的消息激励的情况，则称 m_{jv} 方法依赖于 m_{iu}，记为 $m_{jv}\ R_{MD}\ m_{iu}$。

定义7 设两操作 m_{iw}、m_{jv}，且 $m_{iw}\in M_i, m_{jv}\in M_j$，若类 C_i 是类 C_j 的子类，并且存在操作 m_{jv} 为类 C_j 中的虚拟函数，而操作 m_{iw} 为操作 m_{jv} 的同名函数，则称 m_{iw} 方法依赖于 m_{jv}，记为 $m_{iw}\ R_{MD}\ m_{jv}$。

综上定义6，7可得：

定义8 设两操作 m_{iw}、m_{jv}，且 $m_{iw}\in M_i, m_{jv}\in M_j$，若类 C_i 中操作 m_{jv} 的修改、删除、移动会引起类 C_j 中操作的相应变化，则称操作 C_j 方法依赖于操作 C_i，记为 $C_j\ R_{MD}\ C_i$。

如图1所示的两个 C++ 类说明了上述两种依赖类型。带实心箭头的实线表示依赖关系，箭头指向为依赖方向。

```
Class A                    Class B
{ public:                  { public:
    int i;                     int j;
    void oal();                void obl()
}                              {
                                 A va;
                                 j=va.i+1;
                                 va.oal()
                               }
                           };
```

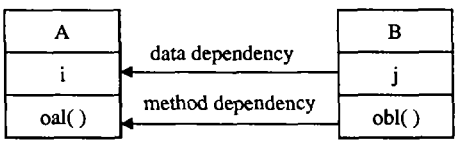

图1 C++程序和它的数据依赖，方法依赖图

2 依赖关系度量

度量类之间的依赖关系之前，本文首先给出依赖度和被依赖度[6~8]的定义：

定义9 面向对象系统中，依赖度为一个类对另一个类依赖的程度，被依赖度为一个类被另一个类依赖的程度。

定义10 类 C_i 对类 C_j 的相对依赖度 rd_{C_i} 为类 C_i 指向类 C_j 的数据依赖和方法依赖关系数目与类 C_i 和 C_j 之间总关系数目的比值。

定义11 类 C_i 对类 C_j 的相对被依赖度 rb_{C_j} 为类 C_j 指向类 C_i 的数据依赖和方法依赖关系数目与类 C_i 和 C_j 之间总关系数目的比值。

根据上述定义，我们可得到如下性质：

定理1 类 C_i 对类 C_j 的相对依赖度 rd_{C_i}，记为：

$$rd_{C_i}=\frac{|DI((A_i\cup M_i)\times(A_j\cup M_j))|}{|(A_i\cup M_i)\times(A_j\cup M_j)|}$$

其有效取值范围为：$0\leqslant rd_{C_i}\leqslant 1$

定理2 类 C_i 对类 C_j 的相对被依赖度 rb_{C_i}，记为：

$$rb_{C_i}=\frac{|DI((A_j\cup M_j)\times(A_i\cup M_i))|}{|(A_i\cup M_i)\times(A_j\cup M_j)|}$$

根据上述两个定义，很显然我们得到如下结论：

定理3 $rd_{C_j}=rb_{C_i}, rb_{C_j}=rd_{C_i}$

其有效取值范围为：$0\leqslant rb_{C_i}\leqslant 1$

性能分析：当相对依赖度 $rd_{C_i}=0$ 时，表示类 C_i 不依赖于类 C_j，这种关系为零关系，即类 C_i 零依赖类 C_j；当 $rd_{C_i}+rb_{C_i}=1$ 时，表示类 C_i 的所有属性、方法和类 C_j 的所有属性、方法都存在依赖关系，这时的关系为满依赖；当 $rd_{C_i}=0$ 且 $rb_{C_i}=0$ 时，表明类 C_i 和类 C_j 之间不存在依赖关系，这时关系为独立关系。

相对度量应该说是比较科学、客观地反映了两个类之间相互依赖的程度，它不仅考虑依赖关系的数目，而且还用笛卡儿乘积来度量两个类之间的总关系，包含了两个类之间关系的规模大小。

3 依赖性度量的细分

在前面依赖性分类中，类之间的依赖关系被细分为数据依赖和方法依赖。这两种类型的依赖可以单独存在，也可以同时存在两个类或同一个类之间。两个类或同一个类之间也可以存在一个或多个数据依赖、方法依赖。我们在前述基础上，可以对两个类之间的依赖关系度量进行细分，有利于在 OOD 中更全面、详细的把握类之间的依赖特性。

定义12（数据依赖度） 描述一个类 C_i 对另一个类 C_j 数据依赖的程度，即类 C_i 指向类 C_j 的数据依赖关系数目与类 C_i 和 C_j 之间总关系数目的比值，记为 dd_{C_i}，则

$$dd_{C_i}=\frac{|DI(A_i\times(A_j\cup M_j))|}{|(A_i\cup M_i)\times(A_j\cup M_j)|}$$

其有效取值范围为：$0\leqslant dd_{C_i}\leqslant 1$

定义13（方法依赖度） 描述一个类 C_i 对另一个类 C_j 方法依赖的程度，即类 C_i 指向类 C_j 的方法依赖关系数目与类 C_i 和 C_j 之间总关系数目的比值记为 md_{C_i}，则

$$md_{C_i}=\frac{|DI(M_i\times(A_j\cup M_j))|}{|(A_i\cup M_i)\times(A_j\cup M_j)|}$$

其有效取值范围为：$0\leqslant md_{C_i}\leqslant 1$

数据依赖度、方法依赖度是对类之间的依赖程度的进一步细化描述，它比依赖度、被依赖度对类依赖程度的描述更具体。很显然，我们从其定义上可以得到如下性质：

定理4 $dd_{C_i}+md_{C_i}=d_{C_i}$

性能分析：数据依赖度 $dd_{C_i}=0$ 时，表示类 C_i 的属性不依赖类 C_j 的属性和方法；同样，方法依赖度 $md_{C_i}=0$，表示类 C_i 的方法不依赖类 C_j 的属性和方法；当 $dd_{C_i}=0$ 且 $md_{C_i}=0$ 时，得 $d_{C_i}=0$，这时的关系为零关系；当 $dd_{C_i}+md_{C_i}=1$ 时，得 $d_{C_i}=1$，这时的关系为满关系。

4 确定类的规模大小

在面向对象软件开发过程（如 UML 支持的 Rational 公司的统一软件开发过程）中，设计者在需求分析模型的基础上从用况实例中抽象生成出最初的类，这些类只是具有一定的逻辑意义，但是对类的规模大小基本没考虑。这种情况对于较小的问题领域空间影响不大，但是 UML 应用的领域空间较

（下转第194页）

她通知 VFM，VoIP 服务必须交换进来，以提供服务。

服务提供接口支持在服务提供商和 VFM 之间进行数据交换和数据请求。

网关接口支持在 VFM 和 Admin Bundle 之间进行数据交换和数据请求。

结束语 利用 OSGI 的应用动态装载能力，虚拟 OSGI 框架不仅能够动态按需装载应用程序，而且解决了嵌入式设备上内存对 OSGI 框架的限制，还初步解决了普适计算模式中的应用框架问题。本文提出了虚拟 OSGI 框架模型，但是没有将它具体实现，只是给出了虚拟 OSGI 框架结构模型。在 OSGI 标准下，实现虚拟 OSGI 框架，构建普适计算应用的平台，将是作者下一步迫切需要进行研究的工作。

参 考 文 献

1 王诚. 计算机组成原理. 北京：清华大学出版社，2001

2 Szymanski J W. Embedded internet technology in process control devices，IEEE 2000

3 Chen K. Programming Open Service Gateways With Java Embedded Server. US：Addison Wesley pub co (sd). 2001

（上接第191页）

大，类的数目过多会影响设计的成本和延长软件开发周期。下面介绍通过类的分解和合并来确定类的合适的规模大小。

4.1 分解

所谓的分解就是将规模较大、类自我的依赖度低的类逻辑分解成若干个子类，分解后的每个子类都应该具有一定的逻辑意义，并且每个子类内部依赖度较高，子类间的依赖度较低。具体分解算法如下：

```
1)计算待分解类 B 自我的相对依赖度 rd
2)IF 类 B 自我的相对依赖度 rd 较小
    按照类内部的逻辑关系将类 B 初步分解为子类 B1，B2
  ELSE
    类 B 不能分解，跳到 5
  ENDIF
3)B=B1，转到1
4)B=B2，转到1
5)结束
```

如图2所示，类 B 规模较大，自我的依赖度较低，通过分解算法将类 B 分解成子类 B1，B2，…，Bn。

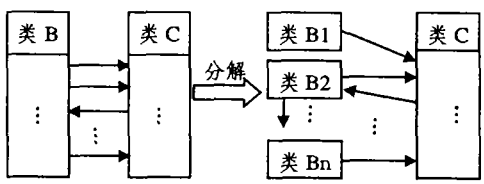

图2 将类 B 分解并降低其依赖

4.2 合并

所谓的合并就是将类之间的依赖度高、规模较小的两个类合并成一个类，使得原先错综复杂的类之间的依赖关系被隐藏。但是，需要注意一点：合并后的类的规模不应该过大。显然，类的合并过程是类的分解过程的逆过程。具体合并算法如下：

```
1)计算类 B 和类 C 的相对依赖度 rdB，rdC
2)IF rdB 且 rdC 较大
    合并类 B 和类 B 为类 BC
ENDIF
```

如图3所示为类 B 和类 C 的合并过程。

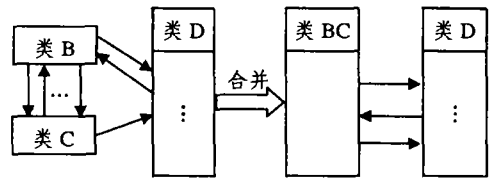

图3

结束语 依据类的依赖关系来分解或合并类的方法，只是为我们在类的分析和设计阶段提供指导方法，我们不可能也没有必要给出一个类具体的依赖度为多少才合适，或者给出依赖度合理的范围。设计者可以在不同的情况，针对具体不同的类，综合利用这两种方法，确定一个规模大小、依赖度都比较合适的类。

参 考 文 献

1 Priestley M. Practical Object-Oriented Design With UML. McGraw-Hill Companies，Inc，2000. 111～152

2 Booch G，Rumbaugh J，Jacobson I. Unified Modeling Language User Guide(in Chinese). Beijing：China Machine press，2001

3 Rumbaugh J，Jacobson I，Booch G. Unified Modeling Language Reference Manual(in Chinese). Beijing：China Machine press，2001

4 周之英. 现代软件工程(中). 科学出版社，2000. 358～428

5 方菲，等. 面向对象软件回归测试技术研究. 软件学报，2001，12(3)：72～75

6 Jones C. Applied Software Measurement，Mcgraw-hill，1986. 36～126

7 Fenton N E，Software Metrics，A Rigorous Approach，New York：Chapman & Hall，1991. 45～78

8 Chidamber R，Kemerer F. A metrics suite for object-oriented design. IEEE Trans on Software Engineering，1994，20：45～66