ISSN 1000-9825, CODEN RUXUEW
*Journal of Software*, Vol.17, No.8, August 2006, pp.1661−1668
DOI: 10.1360/jos17161

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

# 一种基于功能需求层次凝聚的程序聚类方法*

赵 伟+, 张 路, 梅 宏, 孙家骕

(北京大学 信息科学技术学院 软件所,北京 100871)

# A Functional Requirement Based Hierarchical Agglomerative Approach to Program Clustering

ZHAO Wei+, ZHANG Lu, MEI Hong, SUN Jia-Su

(Software Institute, School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China)

+ Corresponding author: Phn: +86-10-62751794, Fax: +86-10-62751792, E-mail: zhaow@sei.pku.edu.cn, http://www.pku.edu.cn

**Zhao W, Zhang L, Mei H, Sun JS. A functional requirement based hierarchical agglomerative approach to program clustering. *Journal of Software*, 2006,17(8):1661−1668.** http://www.jos.org.cn/1000-9825/17/1661.htm

**Abstract**: Program clustering for large and complex systems improves the effectiveness and efficiency of software maintenance and is a basis for acquiring reusable components. In this paper, a functional requirement based hierarchical agglomerative approach to this problem is proposed. In this approach, the semantic information existing in the descriptions of functional requirements is employed to acquire a high-level logic view of the given system. Furthermore, the corresponding source code artifacts for each requirement are acquired through the dynamic analysis. The requirements hierarchy and the requirement-artifact relationships are then used to recover the hierarchical organization of the source code. The clustering results of this approach have the mapping to the application domain. In addition, due to the dynamic analysis, the granularity of this approach is flexible.

**Key words**: program clustering; component acquisition; architecture acquisition; HAC (hierarchical agglomerative clustering); hierarchical agglomeration of functional requirement

摘 要: 对大型复杂系统进行聚类分析能够改善软件维护的效率和效果,同时也是获取可复用构件的基础.提出一种基于需求层次凝聚的程序聚类方法来解决这个问题.该方法利用存在于需求描述中的语义信息获取问题域的高层逻辑,结合对源代码的动态分析,最终获取对源代码的分解划分.使用该方法获取的划分结果具有到问题域的映射;并且由于采用了动态的分析策略,该方法具有灵活的划分粒度.

关键词: 程序聚类;构件提取;构架提取;层次凝聚的聚类;功能需求层次凝聚
中图法分类号: TP311 文献标识码: A

软件产业经过多年的发展,目前已存在大量正在运转的软件系统.对于这些系统,通常需要不断地维护,修正其中的错误、增加新功能或进行结构优化.这些系统中的可复用成分也需要提取出来,包装成构件,用于构造

* Supported by the National Natural Science Foundation of China under Grant No.60403015 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant No.2005AA113030 (国家高技术研究发展计划(863)); the National Grand Fundamental Research 973 Program of China under Grant No.2002CB312003 (国家重点基础研究发展规划(973))

Received 2005-08-26; Accepted 2006-01-09

新系统.为了完成这些任务,往往需要将整个系统划分为一些相对较小且易于控制的子系统.在文献中,这种分解系统的活动通常被称为程序聚类.一般而言,分解出来的子系统(或称为聚类)由一些密切协作的代码实体组成,它们实现系统的一组密切相关的功能或对系统的其他部分提供明确的服务[1].

目前,研究人员对程序聚类进行了大量研究,所提出的方法大致可分为两类:一类方法假设在语法结构上高度关联的代码实体在应用逻辑上也高度关联,即程序员严格遵循高内聚和低耦合的开发原则.这类方法仅仅利用程序代码中的形式化信息(如代码实体间的关系)进行聚类分析,其代表是文献[2–9];另一类方法还考虑了程序或软件中其他制品的非形式化信息(如代码中的注释、标识符的命名、文件名、文件的所有者以及路径信息等),其代表有文献[10,11].然而,由于第 1 类方法所依赖的假设在实际中往往并不成立[10],许多软件系统难以采用此类方法.另外,第 2 类方法目前所考虑的非形式化信息比较有限,其有效性在实际中也经常难以保证(如标识符名和文件名可能使用非标准的缩写),而且许多信息只与粒度较大的代码实体关联(如文件名),因此,这类方法的聚类结果目前还较为粗糙.

本文提出一种利用功能需求信息辅助程序聚类的方法,属于上述第 2 类方法.该方法根据目标系统功能需求语义上的关联,利用文本聚类技术恢复功能的层次结构,并以此为指导,对系统的代码进行动态聚类分析.该方法不要求第一类方法所依赖的假设,与现有第 2 类方法相比,该方法使用的非形式化信息更为全面、准确,并具有灵活的聚类粒度.

本文第 1 节介绍方法所应用的层次凝聚的文本聚类技术.第 2 节详细描述基于功能需求层次凝聚的程序聚类方法.第 3 节给出实验结果.第 4 节讨论相关工作.第 5 节总结全文.

## 1 层次凝聚的聚类(HAC)

文本聚类最初是为改善信息检索的有效性而提出的,它将被检索的文档按照语义上的相似程度划分为不同的类别,使得查询能够在相应的划分中更快、更准地进行.我们的方法采用了层次凝聚的文本聚类方法(hierarchical agglomerative clustering,简称 HAC)[12]获取功能的层次结构.HAC 自底向上分析目标文档,每个文档起初被看作一个最小的聚类,两个相似度最大的聚类继而被合并为一个更大的聚类,这个过程一直延续到所有文档合并为一个聚类.现已提出了多种衡量聚类相似度的指标,最常用的衡量指标是两个聚类$\Gamma$和$\Delta$的并集$\Phi$的自相似度(self-similarity).文档集合$\Phi$的自相似度定义见公式(1).

$$s(\Phi)=\frac{1}{\binom{|\Phi|}{2}}\sum_{d_1,d_2\in\Phi}s(d_1,d_2)=\frac{2}{|\Phi|(|\Phi|-1)}\sum_{d_1,d_2\in\Phi}s(d_1,d_2) \tag{1}$$

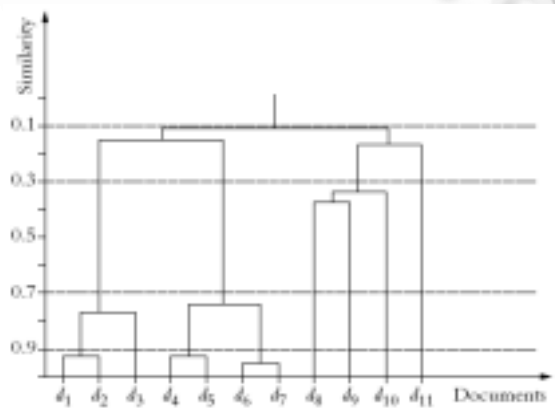


Fig.1 An example of dendrogram

图 1 系统树图的例子

该定义表明,集合的自相似度为这个集合中两两文档间相似度的平均值,其中文档 $d_1$ 与 $d_2$ 的相似度 $s(d_1,d_2)$可以通过基于词频表示的向量间的余弦值来计算(具体见第 2.1.1 节).

上述过程最终产生一个称为系统树图(dendrogram)的结构.图 1 给出了一个以特定方式绘制的系统树图[12].应用 HAC 需要基于聚类质量的度量从系统树图中选取合适的聚类结果.因此,在 HAC 方法的每一轮聚类过程中,除了要计算每两个聚类的相似度以决定哪两个聚类需要合并之外,同时还需要计算出一个衡量当前聚类质量的指标.该指标的计算不仅仅考虑当前每个聚类内的自相似度,还考虑了当前不同聚类间的相异度(与相似度计算一致,相似度越大,相异度越小).相似度与相异度越是同时都高,聚类质量因数

也就越高.利用这一指标可以判断是否已经获得期望的聚类结果.

## 2 基于功能需求层次凝聚的程序聚类方法

本文提出的程序聚类方法是一种非形式化信息辅助的程序聚类方法.该方法的基本思想是,以功能需求描述中的语义信息为辅助,结合对功能与代码实体间映射关系的计算,以此恢复出代码实体的聚类.图 2 给出了该方法的 3 个主要步骤.

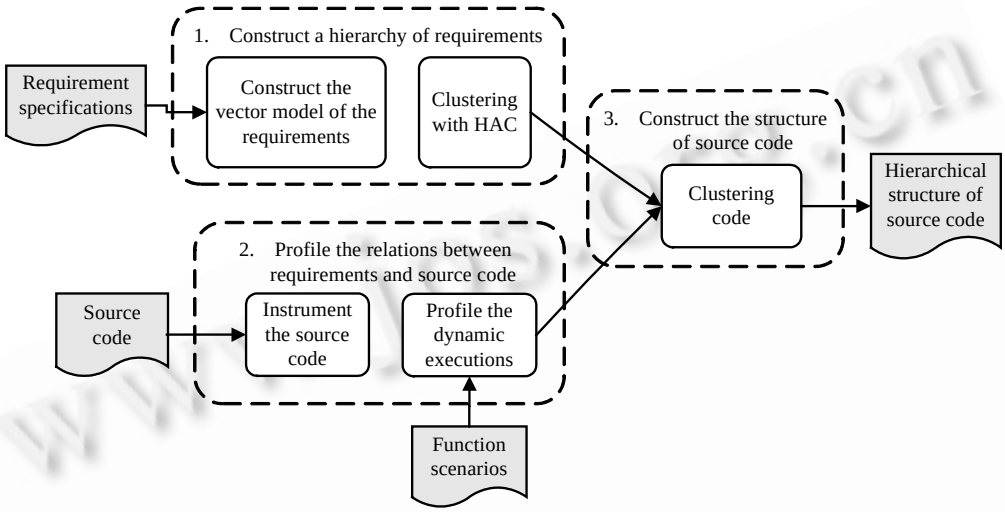


Fig.2 Functional requirement based hierarchical agglomerative approach to program clustering

图 2 基于功能需求层次凝聚的程序聚类方法

### 2.1 构造功能的层次结构

实际系统的所有原子功能(不可以再分为两个或两个以上的功能)可以划分为不同的功能组.例如:一个计算器应用系统的“加”、“减”等功能可被划分为一个功能组,完成计算功能;“打印数字”和“打印字符串”等将被划分为另一个功能组,实现打印功能.通常,根据具体系统的复杂程度以及对功能相似性约束的强弱,这种对系统功能的分组是多层次的.基于功能需求层次凝聚的程序聚类方法的第 1 步就在于恢复出这个层次结构.我们应用 HAC 方法对系统功能需求的描述进行处理,获取所有功能的系统树图.根据聚类质量因数,从系统树图中挑选出满足要求的聚类层次,从而建立目标系统的功能层次结构.

#### 2.1.1 构造功能描述向量

应用 HAC 方法需要把功能描述转换为向量表示.每个功能描述被看作是文本聚类场景下的一个文档,应用信息检索技术中的文本预处理的基本法则(去掉冠词、介词、代词等那些对于表达句子的含义贡献不大的词汇;还原动词为原形;还原名词为单数形式等)处理这些文档,提取出整个文档集合的索引词汇.基于这些索引词汇,每个功能表示为向量 $\vec{d}=(w_1,w_2,\ldots,w_t)$.其中:$t$ 是整个文档集合中索引词汇的个数;$w_i(i=1,2,\ldots,t)$是第 $i$ 个索引词汇在该功能向量 $\vec{d}$ 中的权重.我们使用 TF (term frequency)权重方法计算每个索引词汇在相应向量中的权重.TF 权重方法利用索引词汇在文档中出现的频率作为其在相应向量中的权重(见公式(2)).

$$w_i=\frac{freq_i}{\max_l freq_l} \tag{2}$$

#### 2.1.2 使用 HAC 构造功能层次结构

在得到每个功能的向量表示之后,我们应用 HAC 方法分析这些功能向量获取其层次结构.应用 HAC 方法分为两个步骤:首先进行自底向上的聚类分析,获取系统树图;继而从生成的系统树图中挑选出符合要求的聚类层次构造功能的层次结构.

如第 1 节所述,HAC 分析是一个迭代过程.起初,每个功能被看作是一个最小的聚类,在第 1 轮聚类计算中,HAC 根据聚类间的相似度选择两个最相似的功能,将其合并为一个新的聚类,该聚类与其他聚类将作为下一轮聚类计算的输入,同时考虑聚类间的相异度为当前一轮的聚类结果计算出一个聚类质量因数.假设被分析的目标系统有 $N$ 个功能,经过第 1 轮的聚类计算之后,获得 $N$–1 个聚类.经过 $N$–1 轮聚类计算后,所有系统功能被相继地凝聚为一个聚类集合,HAC 计算终止,并获得了针对整个聚类过程的系统树图.

生成的系统树图包含了每一轮的聚类结果以及每一轮聚类结果的聚类质量因数.我们利用这个聚类质量因数挑选出符合要求的聚类层次,构造最终的功能层次结构.首先,聚类质量因数最高的那一轮聚类结果被挑选出来作为一个聚类层次.然而,与每个功能都是一个聚类的最低层以及所有功能为一个聚类的最高层一起,挑选质量因数最高的一轮聚类结果只能将系统功能划分为 3 层,这在实际中往往是不够的.因此,我们也挑选那些聚类质量因数与最大值的差值在某个阈值之内的聚类层次作为最终的功能聚类层次.

### 2.2 获取功能和代码的映射

为了动态地获取功能和代码之间的映射关系,首先为每个功能设计使用场景(与测试用例类似,用以触发该功能表现出来的输入序列);然后对源代码进行插装并重新编译插装后的源代码,得到可执行目标码;最后执行每一个功能场景,被插装的代码记录触发每个功能所依赖的代码实体,从而获得功能和代码间的映射关系.

### 2.3 获取代码实体的层次结构

在得到功能的层次结构和功能与代码的映射关系后,就可以利用这些信息来恢复代码实体的层次组织结构.恢复的方法基于以下策略:对于两个相似的功能,如果每个功能对应一组代码实体,特定于每个功能的代码和两个功能共享的代码实体对于这两个功能的实现起着不同的作用,从逻辑上就能将这些代码划分到不同的聚类集合.

如上文所述,方法第 1 步所获取的功能层次结构是树形结构.图 3 给出了一个部分的功能层次结构的例子.其中省略了一些代表具体功能的节点.图中的功能层次是一个 5 层的树结构,该层次结构的底层(function level)是第 1 层聚类结果(clustering level 1)所对应的具体功能(一部分被省略),层次结构自底向上的第 2 层是从聚类出的系统树图中挑选出的第 1 层功能聚类结果,后继的聚类结束于该层次结构的根节点.

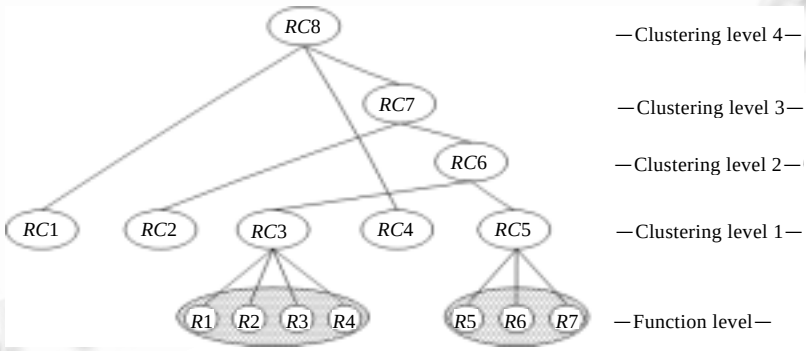


Fig.3 An example partial hierarchy of functional requirements

图 3 部分的功能层次结构的例子

代码实体层次结构的恢复从处于功能层次结构第 1 层的节点开始,这些节点都各自包含一组属于它们的功能,在图 3 中由阴影所示的节点表示,我们称这些节点代表的聚类为特定于这组功能的聚类.而那些从根节点到该节点的路径上的所有节点是这组功能的支撑聚类.经过这一步的计算之后,功能层次结构中的每一个节点映射为由一些代码实体组成的聚类集合.

对功能层次结构最低层的那些功能集合代表的特定于这些功能的构件的计算,是通过对所有这些功能相关的代码实体进行异或(⊕)运算得到的.假定与图 3 中功能 $R1$,$R2$,$R3$ 和 $R4$ 相关的代码实体的集合分别为 $U_{R1}$,$U_{R2}$,$U_{R3}$ 和 $U_{R4}$,特定于功能 $R1$,$R2$,$R3$ 和 $R4$ 的构件(即图 3 中下方用阴影标识出的节点)包含的代码实体为

$U_{R1}\oplus U_{R2}\oplus U_{R3}\oplus U_{R4}$.

计算所有支撑构件所包含的代码实体是一个迭代的传播过程,开始于最低层的支撑构件.最低层的支撑构件所包含的代码实体的初始值是与其相关的所有功能的相关代码实体的交集(∩),得到了所有最低层的支撑构件所包含的代码实体的初始值以后,上一层支撑构件的初始值同样可以通过其子节点的交集运算得到.同时,其子节点所包含的代码实体的最终值可以通过从其初始值中减掉其父节点的初始值得到.这一计算过程一直传播到整个层次结构的根节点结束,层次结构的根节点对应的支撑构件是所有功能的支撑构件.在图 3 中,节点 *RC*3 所包含的代码实体的初始值为 $U_{R1}\cap U_{R2}\cap U_{R3}\cap U_{R4}$,*RC*3 的父节点 *RC*6 的初始值可以通过计算其子节点 *RC*3 和 *RC*5 的初始值的交集获得.同时,*RC*3 和 *RC*5 所包含的代码实体的最终值则可由从其初始值中减掉其父节点 *RC*6 的初始值得到.

## 3 实验研究

### 3.1 实验方法

我们应用该方法对 GNU 上的 DC 系统[13]进行了实验验证.DC 系统是一个支持无限精度的桌面算术计算器.它由 2.7KLOC 的 ANSI C 语言程序组成,并带有完整的功能需求说明,包含 74 个函数、49 个功能.为了实现我们的方法,实验中还使用了两个开源的工具 MC[14]和 TANAGRA[15],用以完成第 1 步对功能的聚类.MC 是 GNU 的一个开源软件,完成从文本到向量模型的转换;TANAGRA 是法国里昂大学的研究人员专门为学术研究开发的一个数据挖掘软件,其中实现了 HAC 方法.

我们利用 MC 将 DC 系统的功能描述转化为向量模型,并将其作为 TANAGRA 的输入进行 HAC 分析,生成这些功能的系统树图.根据前面所述的筛选策略挑选出合格的聚类级别构造出 DC 系统的功能层次结构,筛选过程中我们使用的阈值为 0.1.DC 系统的功能和代码实体间的映射通过使用设计良好的功能场景运行插装后的代码获取.最后,我们根据第 2.3 节中给出的策略来恢复代码间的层次结构.

由于我们已经非常熟悉 DC 系统,在实验中将一个我们手工划分的代码聚类结果作为对照组,与我们的方法所获取的结果进行了对比.由于构件提取和构架恢复是程序聚类的两个主要目标,我们主要针对这两个方面把我们的方法的结果与对照组的结果进行对比分析.在考察构件提取的效果时,我们采用查全率和查准率作为评价准则.对于构架的恢复能力,我们从构架的质量和语义映射能力两个方面对我们的方法获取的构架给出了定性的分析,这可以反映我们的方法恢复构架的能力以及恢复出来的构架的可用性.

### 3.2 实验数据分析

#### 3.2.1 构件提取的效果

从获取的构件个数上来看,DC 系统的 74 个函数被我们的方法划分为 16 个构件(即聚类),手工分析得到的结果为 19 个.我们的方法获取的 16 个构件中有 13 个能够在手工分析的结果中找到对应.我们应用查准率和查全率来评价本文中的方法对于获取构件个数上的效果.这里的查准率是指获取的真正的构件个数与获取的所有构件的个数的比值;查全率是指获取的真正的构件个数与所有的真实构件个数的比值.该方法在获取的构件个数上的查准率为 81.25%,查全率为 68.42%.

获取的构件在查准率上比查全率有较好的表现.也就是说,我们的方法获取的构件具有较高的可信度,而查全率较低是一个弱点.6 个构件没有被恢复出来的原因主要有 3 个:首先,那些完成一些基本程序功能(如分配和释放内存)的低级别构件无法通过系统功能描述信息的辅助被识别出来,同时,它们在代码中的出现也与功能层次结构没有关联,这样的构件在 DC 系统中有两个;查全率较低的另一个原因在于一些本应有的功能组合层次没有恢复出来,这是因为恢复这些层次所依赖的信息并不存在于功能描述信息中,我们的方法因此漏掉了 DC 系统中的一个构件;最后,依赖索引词的相似性进行聚类分析本身的不精确性也导致了某些恢复出来的功能聚类是不正确的,从而使得后续的计算得到的构件也是错误的,同时也丢失了一些正确的构件.

如前所述,我们的方法恢复出的构件有 13 个能够在真实的构件中找到对应.接下来对这 13 个构件作进一

步的评价.这里的查全率表示我们的方法恢复出的正确构件中包含的真实的代码实体的个数与该构件所应具有的真实的代码实体的个数的比值;查准率表示我们的方法恢复出的正确构件中包含的真实的代码实体的个数与该构件中包含的所有代码实体的个数的比值.表 1 列出了这 13 个构件所包含的代码实体在查全率和查准率上的情况以及平均值.

**Table 1** Precision and recall of code artifacts of correct components acquired by our approach
表 **1** 我们的方法所获取的正确构件中包含代码实体的查准率和查全率

| No. | Precision (%) | Recall (%) |
|---|---|---|
| 1 | 100 | 100 |
| 2 | 90 | 75 |
| 3 | 100 | 100 |
| 4 | 100 | 42.86 |
| 5 | 100 | 100 |
| 6 | 69.23 | 100 |
| 7 | 100 | 66.67 |
| 8 | 100 | 100 |
| 9 | 100 | 100 |
| 10 | 50 | 100 |
| 11 | 100 | 100 |
| 12 | 100 | 100 |
| 13 | 100 | 100 |
| Avg. | 93.02 | 91.12 |

从表 1 可以看出,这些构件无论其查全率还是查准率都是非常高的,在最好的情况下,所得到的构件与真实的构件是完全一致的.然而,异常情况仍然存在,有些构件要么查全率不高,要么查准率较低.而且,通常当查准率不够高时,查全率相对会高一些,反之亦然.无论是查全率还是查准率,其不够理想的原因都在于对功能进行聚类这一步骤的不精确性.查全率较低是因为一些功能聚类中漏掉了本应属于这个聚类的功能,而错把不应属于某个聚类的功能划分到了该聚类就会导致较低的查准率.

3.2.2 获取构架的能力及其可用性

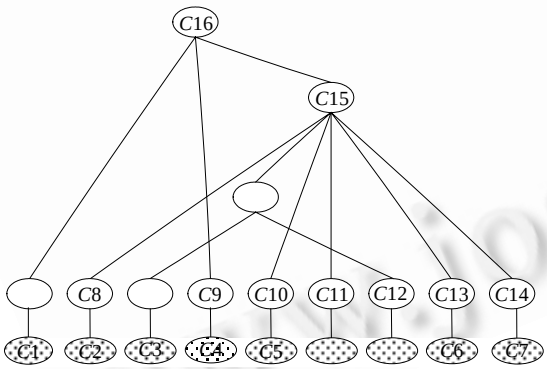


Fig.4 The hierarchical architecture of DC
图 4 DC 系统的层次结构

我们的方法获取的构架是一种层次结构,两个相继的层次间的关系是特定-支撑关系.也就是说,对于该构架中给定的一个构件,属于该构件的代码实体是该节点所包含的所有功能的特定代码成分,而该构件节点的父节点所代表的构件则是这些功能的支撑代码部分,它们同时支撑其他功能.很明显,这样的一个层次结构不仅为代码建立了一个逻辑上的划分,同时也提供了每个构件与其有所贡献的系统功能的语义映射.我们的方法为 DC 系统恢复的层次结构为 5 层,如图 4 所示.

在图 4 中,没有标注的节点不对应到最终的构件,其余有标注的节点均对应到最终提取出的构件.其中,构件 *C*4 是完成一组与计算相关的功能的特定构件,构件 *C*9 为其支撑构件,完成数据预处理和准备的功能,构件 *C*16 为所有功能的支撑构件,进行环境的初始化以及读取输入数据.

## 4 相关工作

将复杂系统划分为相对独立且易于理解的子系统并将其看作候选构件能够有效地辅助软件维护和软件演化.因此,对于该问题的探索一直是学术研究中一个非常活跃的领域.如前所述,目前针对该问题的研究主要分为两大类:一类方法仅仅利用程序代码中的形式化信息进行系统的划分,本文称这类方法为依赖代码中形式化

信息的程序聚类方法;另一类方法不仅利用了程序中的形式化信息,还考虑了程序或软件中其他制品的非形式化信息,我们称这类方法为非形式化信息辅助的程序聚类方法.

### 4.1 依赖代码中形式化信息的程序聚类方法

划分那些过程式编程语言实现的遗产系统常常也被称为模块化问题.其典型的解决方案是,首先在代码中辨识出过程和全局变量,然后根据这些实体是否具有类似的特点将其归类.这里所说的特点是指程序结构或语义上的特性(如使用了相同的全局变量、拥有相同类型的输入参数、返回了相同类型的返回值,等等[2]).那些划分到同一组里的过程和全局变量可以被当作接下来的再工程任务中的候选对象类.Lindig 和 Snelting[3]以及 Deursen 和 Kuipers[4]应用形式化的概念分析技术来分析代码实体所具有的特性上的相似性,从而对实体进行聚类,以得到候选的对象类.可见,针对过程式编程语言实现的遗产系统的聚类分析主要是考察代码实体所具有的静态结构上的相似特性.

利用程序中的形式化信息的方法中还有一个分支关注对程序结构的抽象表示的分析[5−9].这些抽象表示通常是树或图的结构,其中的节点代表代码中的实体(如过程式语言中的过程、面向对象语言中的类,等等),节点间的边表示实体间的关系(如过程式语言中的调用关系、面向对象语言中的继承关系,等等).基于此,程序聚类问题就是在这些图或树结构中寻找最优划分.衡量划分优劣的度量尺度对于这一类方法是必要的.通常,软件工程中高内聚低耦合的原则是这些度量尺度的基础.另外,寻找最优划分的复杂度,会随着图或树结构中节点个数的增加呈指数级增长.因此,很多方法利用一些启发式原则转而去寻找次优解.同时,不同的方法也提出了许多不同的度量尺度来衡量高内聚和低耦合.Chiricota 等人根据高内聚和低耦合的度量计算程序抽象表示中边的权值并删掉那些权值在某个阈值之下的边,从而将表示整个系统的图划分为多个独立的子图以达到系统划分的目的[8].Mancoridis 等人提出了 MQ 度量尺度并使用爬山算法和基因算法来解决复杂度问题[6,7].文献[9]考虑了边的方向和权值来对系统进行划分,划分的过程是一个迭代的过程.Muller 等人的工作[5]与前面的工作不同,它是一种交互式的方法,根据软件工程师对高内聚低耦合的判断筛选出代码中的聚类.

依赖代码中形式化信息的程序聚类方法假设在语法结构上高度关联的代码实体在应用逻辑上也高度关联,本文提出的方法对此并无要求.此外,本文提出的方法动态分析程序,分析粒度与静态方法相比更为灵活.

### 4.2 非形式化信息辅助的程序聚类方法

除了代码中的形式化信息,代码和软件的其他制品中的非形式化信息对划分系统也提供了一定的辅助.Anquetil 和 Lethbridge 根据代码源文件名中的问题域概念为代码源文件提供了一个概念上的浏览器[11].该方法需要一个种子文件作查询,然后检索出与该文件概念上相关的其他文件,并将其视为一个聚类.Andritsos 等人提出的方法是在程序聚类方面新近的研究成果[10].该方法为代码中的形式化信息和代码中以及其他制品中的非形式化信息提供了一个统一的模型,基于此模型进行聚类计算.Andritsos 等人考察的非形式化信息包括有代码源文件的路径结构、代码源文件的所有者信息、代码行数以及上次修改的时间.

非形式化信息辅助的程序聚类方法与仅依赖代码中形式化信息的方法相比,考虑了更多的有用信息.但目前所考虑的非形式化信息仍较有限,其有效性在实际中也经常难以保证,而且许多信息只与粒度较大的代码实体关联.因此,这类方法的聚类结果目前还比较粗糙.本文提出的方法利用了现有方法并没有考虑到的功能需求描述中的信息,它们更为丰富和准确.

## 5 结束语

本文提出了一种程序聚类方法,该方法利用存在于功能描述中的语义信息恢复出系统功能的层次结构,并利用这个功能层次结构,最终获得代码间的高层组织结构.与现有方法相比,我们的方法具有以下 3 个方面的优势:首先,它并不要求所针对的程序满足高内聚和低耦合等开发原则这一假设,从而克服了依赖代码中形式化信息的聚类方法适用面窄的不足;其次,我们的方法采用动态分析获取功能与代码间的映射,使得聚类的粒度非常灵活;最后,在所恢复出的聚类中,我们的方法能够提供获取的聚类和相应的功能间语义上的映射,为进一步的

理解以及将其包装成构件提供更多帮助.

**References**:

[1] Mitchell BS, Mancoridis S. Comparing the decompositions produced by software clustering algorithms using similarity measurements. In: Werner B, ed. Proc. of the 17th IEEE Int'l Conf. on Software Maintenance. Los Alamitos: IEEE Computer Society, 2001. 744−753.

[2] Schwanke R. An intelligent tool for re-engineering software modularity. In: Proc. of the 13th IEEE Int'l Conf. on Software Engineering. Los Alamitos: IEEE Computer Society, 1991. 83−92.

[3] Lindig C, Snelting G. Assessing modular structure of legacy code based on mathematical concept analysis. In: Proc. of the 19th IEEE Int'l Conf. on Software Engineering. Los Alamitos: IEEE Computer Society, 1997. 349−359.

[4] Deursen AV, Kuipers T. Identifying objects using cluster and concept analysis. In: Proc. of the 21st IEEE Int'l Conf. on Software Engineering. Los Alamitos: IEEE Computer Society, 1999. 246−255.

[5] Muller H, Orgun M, Tilley S, Uhl J. A reverse engineering approach to subsystem structure identification. Journal of Software Maintenance: Research and Practice, 1993,5(4):181−204.

[6] Mancoridis S, Mitchell BS, Rorres C, Chen Y, Gansner ER. Using automatic clustering to produce high-level system organizations of source code. In: Kelly K, ed. Proc. of the 6th IEEE Int'l Workshop on Program Comprehension. Los Alamitos: IEEE Computer Society, 1998. 45−52.

[7] Mancoridis S, Mitchell B, Chen Y, Gansner ER. Bunch: A clustering tool for the recovery and maintenance of software system structures. In: Palagi L, ed. Proc. of the 15th IEEE Int'l Conf. on Software Maintenance. Los Alamitos: IEEE Computer Society, 1999. 50−62.

[8] Chiricota Y, Jourdan F, Melancon G. Software components capture using graph clustering. In: Kawada S, ed. Proc. of the 11th IEEE Int'l Workshop on Program Comprehension. Los Alamitos: IEEE Computer Society, 2003. 217−226.

[9] Luo J, Zhao W, Qin T, Jiang R, Zhang L, Sun J. A decomposition method for object-oriented systems based on iterative analysis of the directed weighted graph. Journal of Software, 2004,15(9):1292−1300 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/15/1292.htm

[10] Andritsos P, Tzerpos V. Information-Theoretic software clustering. IEEE Trans. on Software Engineering, 2005,31(2):150−165.

[11] Anquetil N, Lethbridge T. Extracting concepts from file names: A new file clustering criterion. In: Werner B, ed. Proc. of the 20th IEEE Int'l Conf. on Software Engineering. Los Alamitos: IEEE Computer Society, 1998. 84−93.

[12] Chakrabarti S. Mining the Web Discovering Knowledge from Hypertext Data. San Francisco: Morgan Kaufmann Publishers, 2003.

[13] GNU. DC: An arbitrary precision calculator. 2001. http://www.gnu.org/directory/GNU/bc.html

[14] GNU. MC: Converts text documents into a vector space model. 2004. http://www.cs.utexas.edu/users/jfan/dm/

[15] Rakotomalala R. TANAGRA: A free software for research and academic purposes. In: Proc. of the EGC 2005, RNTI-E-3, Vol.2. 2005. 697−702.

附中文参考文献:

[9] 罗景,赵伟,秦涛,姜人宽,张路,孙家骕.一种基于有向带权图迭代的面向对象系统分解方法.软件学报,2004,15(9):1292−1300. http://www.jos.org.cn/1000-9825/15/1292.htm

赵伟(1977－),女,陕西西安人,博士,主要研究领域为软件工程,软件复用,软件维护,程序理解.

梅宏(1963－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为软件工程.

张路(1973－)男,副教授,CCF 高级会员,主要研究领域为软件工程,软件复用,程序理解,配置管理.

孙家骕(1946－),男,教授,博士生导师,CCF 高级会员,主要研究领域为计算机语言及编译技术,软件工程,软件逆向工程.