ISSN 1673-9418 CODEN JKYTA8
Journal of Frontiers of Computer Science and Technology
1673-9418/2011/05(00)-0000-00
DOI: 10.3778/j.issn.1673-9418.2011.00.000

E-mail: fcst@vip.163.com
http://www.ceaj.org
Tel: +86-10-51616056

使用程序聚类技术的模块重构风险分析方法*

朱天梅，吴毅坚+，彭鑫，赵文耘

复旦大学 计算机科学技术学院软件工程实验室，上海 201203

Measuring the Refactoring Risk of Modules Using Software Clustering

ZHU Tianmei, WU Yijian+, PENG Xin, ZHAO Wenyun

Software Engineering Laboratory, School of Computer Science, Fudan University, Shanghai 201203, China
+ Corresponding author: Phn: +86-21-51355343, Fax: +86-21-51355358, E-mail: wuyijian@fudan.edu.cn

ZHU Tianmei, WU Yijian, PENG Xin, ZHAO Wenyun. Measuring the Refactoring Risk of Modules Using Software Clustering. Journal of Frontiers of Computer Science and Technology, 2011, 5(0): 1-000.

Abstract: As software evolves, its modularity gradually degrades. Software refactoring is an important means for software modularity adjustment, but which modules are most in need of refactoring is difficult to predict. A novel approach for measuring the refactoring risk of modules by software clustering is proposed. Using structural clustering and semantic clustering, two different kinds of implied modularity views can be recovered from the implementation as reference modularity. By comparing the differences between the realistic modular structure and the reference modular structure, modules with high refactoring risk are identified. A comparative experiment conducted on three open source software shows that the predicting result produced by the proposed approach conforms well to the actual refactoring activities, indicating the effectiveness of our approach.

Key words: Refactoring Risk; Software Modularity; Software Clustering; Software Metrics; Software Quality

摘 要: 随着软件系统的演化，其模块化结构会逐渐退化。软件重构是调整系统结构的重要手段，但哪些模块最需要重构却难以预测。本文提出了一种基于程序聚类技术的模块重构风险分析方法，该方法通过对目标系统进行结构聚类和语义聚类获得其参考模块化结构，然后比较现实模块化结构与参考模块化结构之间的差异，对程序模块的设计质量进行评价，识别出系统中重构风险较高的模块。实验以三个开源软件的演化历史作为研究对象，通过与传统的模块化度量方法进行比较，表明采用本文方法获得的预测结果与实际重构活动有较好的吻合度，从而验证了该方法的有效性。

*The Natural Science Foundation of China under Grant No. 60903013 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant No. SS2012AA010102 (国家高技术研究发展计划(863)).
Received 2000-00, Accepted 2000-00.

关键词：重构风险；模块化；程序聚类；软件度量；软件质量
文献标识码：A 中图法分类号： TP311.5

1 引言

模块化是面向对象软件最重要的性质之一。软件系统的模块化结构体现了设计开发人员对软件系统的认识，是辅助程序理解、指导后续开发、简化后期维护的重要资源。然而，软件系统的模块化结构往往会随着软件系统的不断演化而逐渐偏离软件设计者最初的设计结构，并且这种偏离往往会损害软件质量[1]。因此，如何客观地评价软件系统的模块化质量，以及如何发现和改正软件模块化结构中可能存在的问题，已经成为一个非常重要的研究课题。

在评价软件模块化质量的过程中，内聚度和耦合度是最常被提及的两个概念。内聚度是对模块内部元素之间关系紧密程度的描述；耦合度是对不同模块相互依赖程度的描述[2]。普遍认为好的模块应该具有“高内聚、低耦合”的特性。然而，由于模块间的依赖关系种类繁多，重要性各异，因此难以客观地度量依赖关系。

为了获得“高内聚、低耦合”的程序模块结构，就需要分析和整理软件制品之间错综复杂的相互关系。程序聚类是一种常用的软件模型恢复技术，在逆向工程领域应用广泛；其本质是通过分析软件元素之间的相互关系，将相关（或相似）的元素划分到同一个簇中（趋向高内聚），同时将不相关（或互异）的元素分隔到不同的簇中（趋向低耦合）。因此，程序聚类技术天然是获取“高内聚、低耦合”的程序模块化结构的方法。

本文提出了一种基于程序聚类技术的模块重构风险分析方法。该方法基于程序聚类技术获取软件系统的参考模块化结构（聚类结果），然后通过分析比较现实模块化结构与参考模块化结构之间的差异，从软件系统中识别出最可能存在重构风险的软件模块。基于该分析结果，软件维护人员可以更有针对性地选择合适的软件模块进行重构，项目管理人员也可实施更合理的项目资源分配。

本文结构组织如下：第 2 部分介绍了相关基本概念；第 3 部分论述了模块重构风险分析方法的过程；第 4 部分利用几个开源软件对所提出的方法进行了实验分析，给出了实验目标、实验过程和评价方法，并对实验结果进行了分析和讨论。第 5 部分讨论了实验有效性。第 6 部分总结全文，并指出了下一步研究方向。

2 相关概念

2.1 模块重构风险

在软件设计之初，有经验的软件设计人员通常会精心设计系统的模块化结构，尽量保证系统结构的合理性。然而，随着软件系统的不断演化，由于种种原因系统结构会逐渐遭到破坏。为了提高代码的可读性和系统的可维护性，开发维护人员需要对系统进行重构。在软件工程学里，重构代码通常是指不改变代码的外部行为情况下而修改源代码。重构既不修正系统错误，也不增加新的功能。重构只是对程序内部结构进行调整，让代码更加容易理解，更容易维护。重构代码可以是结构层面的重构，也可以是语意层面的重构，不同的重构手段施行时，可能是结构的调整或是语意的转换，但前提是不影响代码在转换前后的行为。当系统包结构不合理时，也需要对包结构进行重构调整（如包的拆分、合并、以及类文件从一个包移动到另一个包）。

在系统演化的过程中，对包进行重构调整的可能性，通常被称作模块的重构风险。如果可以客观地评估软件系统的模块化质量，发现软件模块化结构中可能存在的问题，分析出各个模块的重构风险，软件维护人员就可以更有针对性地对软件模块进行维护，项目管理人员也可以据此实施更合理的项目资源分配。我们的前期工作[3]给出了一种通过持续分析软件演化过程中不同模块化视图之间偏离趋势，对系统模块化质量的变化进行评价和解释的方法。但该方法尚不涉及对单个模块的重构风险的评估。

2.2 模块化度量与程序聚类

内聚度、耦合度是度量软件模块质量最常用的、最重要的两个度量指标。Brito 等人在[4] 将“模块耦合”定义为属于不同模块的类之间的耦合关系，将“模块内聚”定义为模块内部的类之间的耦合关系。根据类之间依赖信息来源的不同（结构依赖，语义相关依赖），分别定义“结构耦合和结构内聚”，以及“语义耦合和语义内聚”。在软件维护过程中，低内聚、高耦合的软件模块，其重构风险

往往更大。

聚类是一种被广泛应用的数据挖掘技术，体现了“物以类聚”的自然哲学。程序聚类是指应用于软件领域的聚类技术，通常被用于辅助理解复杂软件系统[5]、重组优化软件系统体系结构[6]、识别可重用模块[7]等方面。

程序聚类对象可以是任意粒度的软件制品，如代码片段、方法、类、源文件等等。本文探讨的软件聚类方法以源文件为基本聚类对象，其目标是获得合理的包结构。

聚类算法可分为层次化聚类算法和非层次化聚类算法。采用层次化的聚类算法的好处在于不需要事先给定所期望的簇数目，可方便地对不同层次的聚类结果进行分析；其缺点则是很难确定哪层的聚类结果最有价值。在程序的模块化分析中，常用层次化聚类算法对程序进行分析。

层次聚类算法对给定数据对象进行层次上的分解。根据层次分解的顺序，可分为凝聚算法(自下向上)[14]和分裂算法[15](自上向下)，本文实验采用凝聚层次聚类算法（基于 Weka[16]提供的工具包实现）对目标程序模块进行聚类分析。

3 基于程序聚类技术的模块重构风险分析方法

本文提出的基于程序聚类技术的模块重构风险分析方法，基于程序聚类技术获取软件系统的参考模块化结构，然后通过分析比较现实模块化结构与参考模块化结构之间的差异，从软件系统中识别出最可能存在重构风险的软件模块。下面介绍该方法的两个主要步骤：参考模块化结构的获取和模块重构风险的分析。

3.1 获取参考模块化结构

计算软件制品（如程序类、源文件等）间的结构依赖距离，我们首先使用设计结构矩阵（Design Structure Matrix）来表达软件制品之间的结构依赖关系[8]。矩阵中的行、列均表示软件制品，当且仅当第 i 行的软件制品依赖于第 j 列的软件制品时，M_{ij} 被赋值为 *1*，否则赋值为 *0*。这样，每个软件制品都可表示成一个 N 维向量，其中 N 是软件制品的总数目。通过计算不同软件制品在 N 维空间中的欧式距离或者夹角余弦值，便得到了软件制品间的结构依赖距离。

计算软件制品之间的语义相关距离，则先提取出源文件中的关键词，获得“词-文档”矩阵，然后运用 LSI(Latent Semantic Indexing[9])技术将关键词空间压缩成 d 维的概念空间，相应地，每个源文件都是 d 维空间中的一个向量。通过计算不同软件制品在 d 维空间中的欧式距离或者夹角余弦值，便得到了软件制品间的语义相关距离。

通过源码信息提取、程序聚类等一系列处理，便得到了系统的参考模块化结构（聚类结构）。该结构是对软件系统模块化结构的一种合理描述，将在下一步中作为参照物，用来评估现实系统中各个模块的重构风险。

3.2 分析模块重构风险

聚类结果分析通常包括对聚类结果簇的规模合理性的考察、以及对聚类簇的内容合理性的考察。通常认为包含对象少于 5 或者多于 100 的簇是不合理的，即星云和黑洞[6]。除了簇的规模必须要合理之外，其是否表达了完整的业务内容也十分关键，通常通过比较聚类结果与参考分布之间的相似程度判断聚类簇的内容合理性。参考分布的获取往往需要领域专家的参与。本文方法将系统的现实模块化结构当作原始聚类划分，将基于程序聚类技术获得的聚类结果作为参考划分，通过比较原始划分与参考划分之间的相似度，评判原始划分的合理性。

我们采用被广泛应用的算法 MoJoFM（e**F**fectiveness **M**easure based on **MO**ve and **JO**in distance [10]）计算聚类之间的相似程度。MoJoFM 是 MoJo（**MO**ve and **JO**in[11],[12]）算法的归一化版本。MoJo 算法计算从一个划分变换到另一个划分所需的“移动”和“合并”操作的最少步数。给定两个划分 A 和 B，A、B 之间的相似度可由如下公式算得：

其中 mno(A,B) 表示从划分 A 变到划分 B 所需进行的“移动”和“合并”操作的最少总步数，max(mno($\forall A$,B)) 表示离划分 B 最远的划分变换到 B 所需进行的“移动”和“合并”操作的最少总步数。MoJoFM 的取值区间为[0%, 100%]。取值越大，表示 A 分布越像 B 分布。

基于程序聚类的度量方法，通过对软件系统的源代码分别进行语法和语义聚类获得软件系统相对合理的参考模块化结构，然后比较现实模块化结构与参考模块化结构之间的相似程度，根据相似度评价软件系统的模块化质量。

根据依赖关系的不同，我们使用 $R_{St}(S)$，$R_{Se}(S)$ 分别表示通过结构聚类和语义聚类获得的系统 S 的参考模块化结构，A(S)表示系统 S 的现实模块化结构。MoJoFM(A(S),$R_{St}(S)$) 和 MoJoFM(A(S),$R_{Se}(S)$) 分别表示现实模块化结构与结构参考模块化结构和语义参考模块化结构之间的相似程度。另外我们采用分散度衡量原始划分中的某个簇在变换过程中，变化的剧烈程度。

定义 1 分散度。给定原始划分 $A=\{a_1,a_2,\cdots,a_n\}$，参考划分 $B=\{b_1, b_2,\cdots,b_m\}$，其中 a_i 和 b_j 分别表示划分 A 中的第 i 个簇和划分 B 中的第 j 个簇。A_j 表示 A 划分中包含簇 b_j 中元素的簇的集合，即 $A_j=\{a_i \mid a_i$ 包含 b_j 中的一个或多个元素，$a_i \in A\}$，Bi 表示 B 划分中包含簇 a_i 中元素的簇的集合，即 $Bi=\{b_j \mid b_j$ 包含 a_i 中的一个或多个元素，$b_j \in B\}$。B 划分中的簇 b_j 的影响权重可定义为 $\text{Weight}(b_j)=|A_j|$。A 划分中簇 a_i 在划分 B 中的分散度 $\text{Dispersion}(a_i,B)$ 被定义为：，A 划分中所有簇在 B 划分中分散度的集合 Dispersion(A,B)，则被定义为：

系统 $S=\{p_1, p_2,\cdots,p_n\}$ 中的模块 p_i 在结构参考模块化结构和语义参考模块化结构中的分散度可分别表示为 $\text{Dispersion}(p_i,R_{St}(S))$ 和 $\text{Dispersion}(p_i,R_{Se}(S))$。

定义 2 重构风险。重构风险是指软件模块在软件演化的过程中，被重构的可能性。给定软件系统 $S=\{m_1,m_2,\cdots,m_n\}$，$S'=\{m'_1,m'_2,\cdots,m'_m\}$ 表示系统 S 发生演化后的模块化结构，P（R）表示事件 R 发生的概率，则模块 m_i 的重构风险可表示为 $\text{risk}(m_i)= P(\text{Dispersion}(m_i,S')!=1)$。

根据现实模块结构中的模块在参考模块化结构中的分散度，可以评估系统中每个模块的重构风险。分散度越大，重构风险越高。

4 实验

为了分析本文方法的有效性，我们针对多个开源系统进行了对比实验。这一部分将介绍实验设计、实验结果以及我们对实验结果的分析和总结。

4.1 研究问题和评价方法

我们的实验聚焦于对如下两个研究问题的回答：

Q1. 基于程序聚类的质量评估是否有具有参考价值？

Q2. 基于程序聚类的度量方法，是否在某些情况下优于基于内聚度和耦合度的度量方法？

为了回答问题 Q1，我们必须预先分别准备已知的具有不合理的和相对合理的模块化结构的软件系统。然后分别计算这些系统的 MoJoFM(A(S),$R_{St}(S)$)和 MoJoFM(A(S),$R_{Se}(S)$)。如果相对合理的系统，其相应的度量指标取值也较高，那么，就验证了基于程序聚类技术的度量指标的质量评估方法是具有参考价值的。

针对研究问题 Q2，我们通过不同的度量方法预测目标系统中各模块的重构风险，并按风险大小由高到低排序；同时分析该系统实际的演化历史确定实际被重构的模块，并根据重构的剧烈程度由大到小排序。预测结果序列与实际重构历史序列之间的吻合程度称为预测吻合度。预测吻合度通过序列吻合度来表示；预测吻合度越高，说明预测效果越好。

定义 3 序列吻合度。对于序列 A 和序列 B，$\text{ahead}(A,B,a_i)$ 表示 A、B 序列中均处在元素 a_i 之前的元素个数。序列 A、B 之间的序列吻合度定义为：例如，

当 $A=\{a_1,a_2,a_3\}$, $B=\{a_3,a_1,a_2\}$ 时，$\text{ahead}(A,B,a_1)=0$,$\text{ahead}(A,B,a_2)=1$,$\text{ahead}(A,B,a_3)=0$, 则 $\text{conformance}(A,B)=(2\times(0+1+0))/(3\times(3-1))=1/3$;

当 $A=\{a_1,a_2,a_3\}$，$B=\{a_1,a_2,a_3\}$ 时，$\text{ahead}(A,B,a_1)=0$,$\text{ahead}(A,B,a_2)=1$,$\text{ahead}(A,B,a_3)=2$, 则 $\text{conformance}(A,B)=(2\times(0+1+2))/(3\times(3-1))=1$。

4.2 目标系统

本试验以开源软件项目 JFreeChart[1]，JHotDraw[2] 和 JEdit[3] 的各个版本为实验对象。从 SourceForge.net 的版本控制库，我们获得上述开源系统共计 124 个版本的源代码，每个项目的基本数据如表 1 所示。

[1] http://www.jfree.org/jfreechart/

[2] http://www.jhotdraw.org/, http://www.randelshofer.ch/oop/jhotdraw/index.html

[3] http://www.jedit.org/

表 1 目标系统的基本情况

Table1 Properties of the Target Systems

项目	版本数	模块数	源文件数	代码行数（千行）	注释行数（千行）	备注
JFreeChart	50	5-63	86-805	8.0-107.4	5.9-107.6	限 soure 目录下
JHotDraw	14	7-41	128-429	9.3-55.9	4.6-29.2	限 soure 目录下，测试用例和示例代码除外
JEdit	60	11-29	138-503	29.2-105.8	10.4-48.4	限 JEdit Core 部分

4.3 实验过程

4.3.1 基于程序聚类的质量评估是否有具有参考价值？

由于本实验所选择的开源项目，均是公认具有良好设计和维护过程的项目，因此，维护日志中记录的系统重构行为可以认为是对系统的优化。我们通过两种方式，获取具有对比性的对象系统。第一种方式，即通过阅读各项目的维护日志，找出具有系统重构行为的相邻版本。另一种方式，首先去除系统的模块结构信息，然后随机生成其模块结构，记作 Ran(S)。为了验证系统结构的好坏，可以先获取两种参考结构：反例结构和正例结构；然后从如下两个角度分析目标结构的优劣：1.与反例结构尽量不相似，2.与正例结构尽量相似。因为，随机产生的系统结构 Ran(S)是完全随意的结构，因此可以认为 Ran(S)是一个典型的反例结构；而程序聚类结果良好的结构特性决定其可以充当典型的正例结构。通常情况下，与随机获得的结果之间存在巨大的差异是正例结构必须具备的一个特性。并且，越合理的现实模块结构越可能与相应的正例结构更相似。通过仔细阅读系统维护日志，我们选取 JHotDraw 版本 7.3.1 和 7.4 作为对比实验对象，其中维护日志明确记录版本 7.4 是对版本 7.3.1 包结构的优化调整。另外，我们随机形成了版本 7.3.1 的随机模块结构 Ran(*7.3.1*)。软件基本设计元素的剧烈变化将导致无法准确追踪基本设计元素在不同结构中的映射关系；因此，针对不同软件版本的事实结构，我们定制特定的参考结构。即版本 7.4 的正例参考结构为 $R_{Se}(7.4)$，版本 7.3.1 的正例参考结构为 $R_{Se}(7.3.1)$，反例参考结构为 Ran(7.3.1)。显然，这些系统结构的合理程度依次为：A(*7.4*)>A(*7.3.1*)>Ran(*7.3.1*)，因此，如果基于程序聚类度量指标的质量评估具有参考价值，那么应该存在如下不等式关系：MoJoFM(A(7.4),$R_{Se}(7.4)$) > MoJoFM(A(7.3.1),$R_{Se}(7.3.1)$) > MoJoFM(Ran(7.3.1), R_{Se} (7.3.1))。实验结果如表 2 所示，从表中可知实验结果与实际情况相符合。这从一定程度上验证了基于程序聚类度量指标的质量评估是非常具有参考价值的。

表 2 质量评估结果对比

Table2 Comparison Results of the Quality Assessment

系统结构相似度比较	取值
MoJoFM(A(7.3.1),$R_{Se}(7.3.1)$)	63.16
MoJoFM(A(7.4),$R_{Se}(7.4)$)	70
MoJoFM(Ran(7.3.1),R_{Se} (7.3.1))	16.58

4.3.2 基于程序聚类的度量方法，是否在某些情况下优于简单计算内聚度和耦合度的度量方法？

基于三个目标系统 124 个版本的源代码，我们采用多种预测方法分析每一个有后续版本的源代码，并根据其直接后续版本获取现实重构序列，共计获得 121 组数据。图 1 描述了通过不同方法所得到的预测吻合度的分布情况，其中纵轴取值表示预测序列与现实序列之间的吻合度，横轴表示不同的预测方法。CLUSTER_se 表示基于语义聚类的预测方法，CLUSTER_st 表示基于结构聚类的预测方法，COUPLING_st 表示基于结构耦合度的预测方法，COHESION_st 表示基于结构内聚度的预测方法，COUPLING_se 表示基于语义耦合度的预测方法，COHESION_se 表示基于语义内聚度的预测方法。从图 1 中可以看出，基于程序聚类技术的预测方法（CLUSTER_se 和 CLUSTER_st）与简单计算内聚度和耦合度的预测方法相比，预测吻合度均具有较高的均值。其中，基于结构聚类的预测方法（CLUSTER_se）表现最佳。

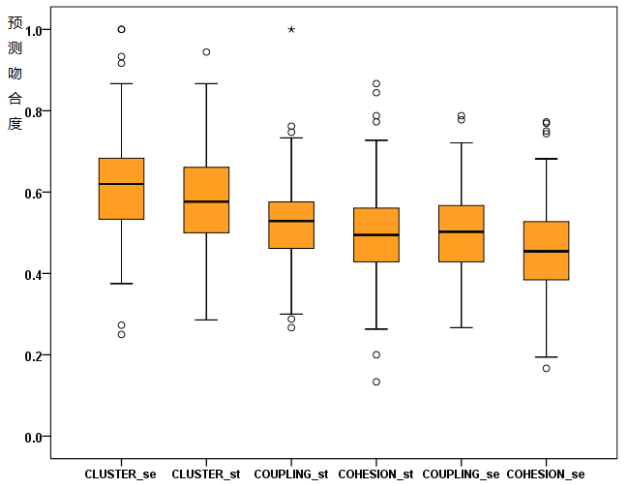

图 1 预测吻合度分布

Fig.1 the Distribution of Prediction Conformance

以 JHotDraw 版本 7.3.1 演化到版本 7.4 为例，根据代码分析获知，在此次演化中，包\jhotdraw\draw、\jhotdraw\app\action、\jhotdraw\gui、\jhotdraw\app、\jhotdraw 的结构被调整，其中\jhotdraw\draw 和\jhotdraw\app\action 的调整最为剧烈。

表 3 给出了 JHotDraw 版本 7.3.1 的包名-编号映射表。表 4 给出了基于各种度量指标的重构风险预测序列，以及系统维护历史中实际的重构调整序列（Dispersion(A(*7.3.1*),A(*7.4*))），并已按风险从高到低排序。很显然，基于程序聚类的度量方法其序列吻合度要高于简单计算内聚度和耦合度的预测方法。因此可以肯定，在某些情况下，基于程序聚类的度量方法要比简单计算内聚度和耦合度的度量方法更具优势。

表 3 程序模块编号对照表

Table3 Labels of Software Modules

模块	编号
\src\main\java\org\jhotdraw\draw	m1
\src\main\java\org\jhotdraw\app\action	m2
\src\main\java\org\jhotdraw\app	m3
\src\main\java\org\jhotdraw\gui	m4
\src\main\java\org\jhotdraw	m5
\src\main\java\org\jhotdraw\gui\datatransfer	m6
\src\main\java\org\jhotdraw\color	m7
\src\main\java\org\jhotdraw\gui\event	m8
\src\main\java\org\jhotdraw\gui\fontchooser	m9
\src\main\java\org\jhotdraw\gui\plaf\palette	m10
\src\main\java\org\jhotdraw\geom	m11
\src\main\java\org\jhotdraw\util	m12
\src\main\java\org\jhotdraw\draw\action	m13
\src\main\java\org\jhotdraw\xml	m14

表 4、重构风险预测结果对比

Table4 Comparative Result of the Refactoring Risk Prediction

分析方法	所得序列（按重构风险由高到低排列）	序列吻合度
Dispersion(A(*7.3.1*),A(*7.3.1*))	(m1,m2,m3,m4,m5,m6,m7,m8,m9,m10,m11,m12,m13,m14)	1
Dispersion(A(*7.3.1*),R_{St}(*7.3.1*))	(m1,m13,m2,m4,m11,m12,m5,m3,m14,m10,m7,m8,m6,m9)	0.58
Dispersion(A(*7.3.1*),R_{Se}(*7.3.1*))	(m1,m13,m3,m2,m4,m5,m11,m7,m6,m8,m9,m10,m12,m14)	0.8
CohesionQ_{Se}(*7.3.1*)	(m9,m12,m6,m8,m11,m4,m14,m5,m3,m10,m13,m7,m2,m1)	0.40
CouplingQ_{Se}(*7.3.1*)	(m1,m4,m9,m12,m7,m11,m14,m5,m8,m3,m6,m10,m13,m2)	0.57
CohesionQ_{St}(*7.3.1*)	(m12,m8,m4,m14,m2,m13,m11,m5,m3,m6,m9,m1,m10,m7)	0.44
CouplingQ_{St}(*7.3.1*)	(m11,m12,m7,m1,m4,m3,m8,m5,m6,m14,m13,m2,m9,m10)	0.56

5 方法的有效性讨论

本文所提出的重构风险预测方法是基于程序分析技术和聚类技术的。因此相关工具和技术的可靠性直接影响参考模块化结构的质量。为了避免相关技术的影响，实验采纳了一种最为广泛使用的算法进行程序聚类。同时，对聚类结果的质量采用权威性（所产生的划分应该与参考划分尽量相似）和分布合理性（应该尽可能避免黑洞簇和星云簇[6]）进行了评估。我们使用 *NED* (non-extreme distribution 非极端分布率[13])对聚类结果的分布合理性做了评估。在我们实验的 124 个系统中，结构聚类结果中有超过四分之三的 *NED* 值在 0.72 以上，语义聚类结果中有超过四分之三的 *NED* 值在 0.85 以上，这样的结果整体上是可接受的。

另外，在实验中所用的三个开源 Java 项目并不能代表所有软件。尽管这些项目是精心挑选的开源项目，并且具有较长的演化历史和较完整维护日志的项目，具有一定的代表性，但是否存在其他项目（开源或非开源的软件）可能不适用于本文的

预测方法，尚不可知。鉴于此，我们不期望通过基于程序聚类的度量方法完美地度量软件系统的模块化质量，但希望能提供一种方式来评估软件维护期间模块发生变化的可能性。

6 结论和展望

在本文中，我们提出了基于程序聚类的软件模块化质量度量方法，通过分析包结构与参考模块化结构之间的差异评估软件的模块化质量。目前，我们考虑两种参考模块化结构，即结构聚类模块化结构和语义聚类模块化结构。基于 3 个开源系统的 124 个版次的源代码，我们进行的经验研究，验证了该方法的有效性。我们发现，包结构与参考模块化结构之间的差异，在很大程度上表明设计质量演化的状态，特别是质量退化的趋势。基于程序聚类的度量指标捕获了内聚度和耦合度所不能捕获的信息；并能提供非常重要参考建议；在某些情况下，表现出了优于传统度量方法的特点。对软件模块化质量的连续监测和评估，为未来的演变提供了有用的反馈信息。

然而，本文方法只对模块的重构风险进行评估和排序，并不能给出具体重构策略和重构建议。对具体代码的重构指导，不在本文的研究范围之内。同时，我们也已经注意到本文方法仍有待在更广泛的范围进一步证实。为了克服信息不足，以及主观因素对演化理解的影响，我们正试图将研究扩展到实验室内部项目和更多的外部开源项目，以增加我们的结论的一般性。在今后的工作中，我们也将提供更多的模块化质量评估意见，为演化决策提供更全面的趋势监测反馈。

References:

[1] G. Bavota, A. De Lucia, A. Marcus, R. Oliveto, “Software Re-Modularization Based on Structural and Semantic Metrics,”[C]. in WCRE 2010: Beverly, MA. 195-204.

[2] W. P. Stevens, G. J. Myers, L. L. Constantine, "Structured design,"[J]. IBM Systems Journal, vol. 13, 1974, 115-139.

[3] Tianmei Zhu, Yijian Wu, Xin Peng, Zhenchang Xing, Wenyun Zhao, “Monitoring Software Quality Evolution by Analyzing Deviation Trends of Modularity Views,”[C]. In Proceedings of the 18th Working Conference on Reverse Engineering (WCRE'11), 2011, 229-238.

[4] Brito E Abreu, F. and M. Goulao. “Coupling and cohesion as modularization drivers: are we being over-persuaded?”[C]. In Proceedings of the Fifth European Conference on Software Maintenance and Reengineering, 2001.

[5] S. Mancoridis, B. S. Mitchell, Y. Chen, and E. R. Gansner, “Bunch: A clustering tool for the recovery and maintenance of software system structures,”[C]. in ICSM ’99: Proceedings of the IEEE International Conference on Software Maintenance. Washington, DC, USA: IEEE Computer Society, 1999, pp. 50-59.

[6] N. Anquetil, C. Fourrier, and T. C. Lethbridge, “Experiments with clustering as a software remodularization method,”[C]. in WCRE ’99: Proceedings of the 6th Working Conference on Reverse Engineering. Washington, DC, USA: IEEE Computer Society, 1999, pp. 235-255.

[7] Y. S. Maarek, D. M. Berry, and G. E. Kaiser, “An information retrieval approach for automatically constructing software libraries,”[C]. IEEE Transactions on Software Engineering, vol. 17, no. 8, pp. 800-813, 1991.

[8] D. V. Steward, “The Design Structure System: A Method for Managing the Design of Complex Systems,”[C] IEEE Transactions on Engineering Management, 1981, 28: 71-74.

[9] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, R. Harshman, “Indexing by Latent Semantic Analysis,”[J]. Journal of the American Society for Information Science, vol. 41, 1990, 391-407.

[10] W. Zhihua, V. Tzerpos, “An effectiveness measure for software clustering algorithms,”[C]. in Proceedings of the 12th IEEE International Workshop on Program Comprehension, 2004, 194-203.

[11] V. Tzerpos, R. C. Holt, “MoJo: a distance metric for software clusterings,”[C]. in WCRE, 1999, 187-193.

[12] W. Zhihua, V. Tzerpos, “An optimal algorithm

for MoJo distance,"[C]. in Proceedings of the 11th IEEE International Workshop on Program Comprehension, 2003, 227-235.

[13] G. Scanniello, M. Risi, G. Tortora, "Architecture Recovery Using Latent Semantic Indexing, K-Means: An Empirical Evaluation,"[C]. in SEFM '10, 2010, 103-112.

[14] A. K. Jain, R. C. Dubes, "Algorithms for Clustering Data," [M]. Prentice Hall, Englewood Cliffs, 1988.

[15] L. Kaufman, P. J. Rousseeuw, "Finding Groups in Data: An Introduction to Cluster Analysis," [M]. Wiley series in probability and mathematical statistics. John Wiley & Sons Inc., New York, 1990.

[16] Weka [EB/OL]. [2011-06]. http://www.cs.waikato.ac.nz/~ml/weka/index.html.

ZHU Tianmei was born in 1985. He is a master student and teaching assistant in Fudan University, China. His research interests include software maintenance and software reengineering.

朱天梅(1985-), 男, 湖南娄底人, 复旦大学 2009 级硕士研究生, 助教, 主要研究领域为软件维护与再工程。

WU Yijian was born in 1979. He received his Ph.D. degree from Fudan University in 2006. He is an assistant professor at Fudan University. His research interests include software maintenance and evolution, software product line, and software architecture.

吴毅坚(1979-), 男, 上海人, 2006 年在复旦大学获得博士学位，现为复旦大学计算机科学技术学院讲师，主要研究领域为软件维护和演化、软件产品线、软件体系结构。在国际会议和国内期刊上发表论文 10 余篇，主持自然科学基金项目 1 项，作为技术骨干参加多项软件开发项目和科研项目。

PENG Xin was born in 1979. He received his Ph.D. degree from Fudan University in 2006. He is a associate professor at Fudan University. His research interests include software maintenance, adaptable software, and software reuse.

彭鑫(1979-), 男, 湖北黄冈人, 2006 年在复旦大学获得博士学位, 现为复旦大学计算机科学技术学院副教授、硕士生导师, 主要研究领域为软件维护, 自适应软件, 软件复用与产品线。在 RE、ICSM、ICSR、WCRE、IST Journal、JCST、计算机学报、软件学报、电子学报、计算机研究与发展等国内外会议及期刊上发表论文 30 余篇, 主持自然科学基金项目 1 项，参加 863 项目多项。CCF 高级会员。

ZHAO Wenyun was born in 1964. He received his M.S. degree from Fudan University in 1989. He is a professor and doctoral supervisor at Fudan University. His research interests include software engineering, electronic commerce, etc.

赵文耘(1964-), 男, 江苏常熟人, 1989 年在复旦大学获得硕士学位, 现为复旦大学计算机学院教授、博士生导师, 主要研究领域为软件工程、电子商务。在 RE、ICSM、ICSR、WCRE、IST Journal、JCST、计算机学报、软件学报、电子学报、计算机研究与发展等国内外会议及期刊上发表论文 50 余篇, 主持自然科学基金项目 2 项、863 项目多项。CCF 高级会员。