基于加权 PageRank 算法的关键包识别方法

潘伟丰[1,2],李 兵[2,3],马于涛[2,3],姜 波[1]
(1.浙江工商大学计算机与信息工程学院,浙江杭州 310018;2.武汉大学软件工程国家重点实验室,湖北武汉 430072;
3.武汉大学计算机学院,湖北武汉 430072)

摘 要: 识别软件中的关键实体对于人们理解软件,控制和降低维护费用具有重要意义.然而现有的工作基本都是针对关键类识别的,针对关键包、方法/属性等的研究甚少;同时现有的工作也未能揭示关键类与软件外部质量属性间的关系.为丰富现有的工作,本文提出了一种基于加权 PageRank 算法的关键包识别方法.该方法用加权有向软件网络模型抽象包粒度软件系统,提出新度量 PR(PackageRank)从结构角度量度节点重要性,并引入加权的 PageRank 算法计算该度量值.数据实验部分以六个开源 Java 软件为例,分析了包的 PR 值与常用复杂网络中心性指标(介数中心性、接近中心性、度数中心性等)间的相关性;使用加权的 SIR(Susceptible-Infectious-Recovered)模型分析了 PR 所识别关键包的传播影响,并与其它相关方法进行比较,验证了本文方法的有效性;最后,以其中两个软件为例,分析了包的 PR 值与包可理解性间的关系,进一步验证了本文方法的有效性.

关键词: 关键包; PageRank 算法; 软件网络; 程序理解
中图分类号: TP311.5 **文献标识码:** A **文章编号:** 0372-2112 (2014)11-2174-10
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2014.11.008

Identifying the Key Packages Using Weighted PageRank Algorithm

PAN Wei-feng[1,2], LI Bing[2,3], MA Yu-tao[2,3], JIANG Bo[1]
(1. *School of Computer Science and Information Engineering, Zhejiang Gongshang University, Hangzhou, Zhejiang* 310018, *China*;
2. *State Key Laboratory of Software Engineering, Wuhan University, Wuhan, Hubei* 430072, *China*;
3. *School of Computer, Wuhan University, Wuhan, Hubei* 430072, *China*)

Abstract: Identifying key entities has many implications for software understanding and controlling and reducing maintenance costs. However the existing methods only focus on identifying key classes. Little work has been done on the identification of key entities at the other levels. Further the existing work also failed to reveal the relationships between key classes and external quality attributes. In this paper, we introduce a novel method IDEEP (IDEntifying kEy Packages using weighted PageRank algorithm) to identify the key packages. IDEEP uses a weighted and directed software network to describe packages and their dependencies, proposes a new metric PR (PackageRank) to quantify the package importance, and introduces a weighted PageRank algorithm to compute PR values. Our experiments are carried out on six Java software systems. First we analyze the correlation between PR values and other centrality metrics such as betweenness, closeness and degree. Second we use a weighted version of the susceptible-infectious-recovered model to examine the spreading influence of each node. The results show that our method is better than other six methods. Further, we reveal the relationships between key packages and their understandability and show that the key packages identified by our method are more meaningful from a software engineering perspective.

Key words: key package; PageRank algorithm; software network; program comprehension

1 引言

软件结构是决定软件质量的重要因素[1].然而,软件的最初结构往往不能适应新的需求,错误的修正、性能的提高、软件运行环境的变更等因素都促使软件做出必要的调整.软件维护已成为软件生存期中最长的一个阶段,其所花费的人力、物力等资源占了软件总费用的 90%以上[2].分析和理解软件是软件维护的重要部分[3~5].为了对软件实施变更,维护人员通常需要耗费总时间的 30%至 60%进行软件理解[4,5].特别当所维护的系统规模很大并且维护人员对系统不熟悉时,分析和理解软件更是难上加难.因此,提供有效的技术(或有意义的软件度量)简化维护人员的工作,指导其对软件的分析和理解,对于控制和降低软件维护费用具有重要意义[1,4].

收稿日期:2013-08-09;修回日期:2014-02-08;责任编辑:马兰英
基金项目:国家 973 重点基础研究发展计划(No.2014CB340401);国家自然科学基金(No.61202048,No.61273216,No.61272111);浙江省自然科学基金(No.LQ12F02011,No.LY13F020010);软件工程国家重点实验室开放基金(No.SKLSE-2012-09-21)

要分析和理解软件,首先要解决的问题是从何处(哪些软件实体)开始理解[1].软件系统是一类人工复杂系统,通常包括多个相互作用的实体,从不同的粒度看,主要包括:数据对象、方法、模块、类、构件、子系统等.通过这些实体间的交互、协作来实现预期的功能[6].近年来,一些研究者将复杂网络的方法引入到软件拓扑结构分析中,用软件的网络模型抽象面向对象(Object-Oriented,简称 OO)软件系统,发现 OO 软件的结构并不是随机和无序的,大多数都展现出"小世界"(small-world)和"无标度"(scale-free)等复杂网络特征[6~10].研究表明,无标度网络具有一个显著的特点,即:节点因其在网络中的位置不同而具有不同的重要性[11,12].关键节点对于流言和疾病传播的控制、市场规划的制定、资源的优化使用及信息的高效传播都具有重要意义[12].因此,选择从软件的关键实体开始理解软件结构是一种切实可行的方法[1].但是,在理解一个不熟悉的复杂软件系统时,维护人员该如何识别这些关键的软件实体呢?

针对这一问题,研究者们开展了一些工作:Zaidman 和 Demeyer[4]通过跟踪软件的执行过程,计算一些动态耦合度量值,并构建类依赖关系图,最后将该图作为 HITS(hyperlink-induced topic search)算法的输入,识别系统中的关键类.数据实验表明,该方法能够比较准确地识别关键类.但是该方法需要事先定义一些运行场景,收集系统的执行轨迹,时间和空间开销比较大.更进一步,他们又提出了一种基于静态耦合度量的关键类识别方法,该方法无需运行系统,在一定程度上缓解了空间开销,但是在实验系统上时间开销仍然得 1 个多小时.李兵等[13]将软件系统在类粒度抽象成加权软件网络,通过仿真算法分析节点在该网络中的传播影响,并据此评价类的重要性.数据实验表明,该方法可以有效地识别系统的关键类.Wang 和 Pan[14]将软件系统在类粒度抽象成软件网络,并引入复杂网络中的中心性指标(介数中心性、接近中心性、度数中心性等)量度类的重要性,并分析了各种指标在识别关键类上的异同.周毓明等[15]将类及类间依赖关系抽象成依赖图,比较了 PageRank 算法、HITS 和介数中心性指标在识别关键类上的差异.更进一步,他们又基于依赖图计算 h 指数及其衍生指数,并据此评价类的重要性[16],在一定程度上改进了 Zaidman 等的工作.

现有的工作虽然为识别软件的关键实体提供了一些解决方法,但基本都是针对关键类识别的.针对软件其它粒度实体(如方法/属性、包等)的研究甚少.同时对所识别的关键类与软件的外部质量属性(如可理解性)间的关系也未见讨论.有鉴于此,本文提出了一种基于加权 PageRank 算法的关键包识别方法 IDEEP(IDEntifying kEy Packages using weighted PageRank algorithm).IDEEP 将 OO 软件在包粒度抽象成加权有向软件网络,提出了一个新的度量指标 PR(PackageRank)量度节点在网络中的重要性,并提出用加权 PageRank 算法计算该度量值.在此基础上,本文以六个开源 Java 软件为例,分析了包的 PR 值与复杂网络中心性指标间的相关性;使用加权的 SIR(Susceptible-Infectious-Recovered)模型分析了 PR 所识别关键包的传播影响,并与其它相关方法进行比较,验证了本文方法的有效性;最后,分析了包的 PR 值与包可理解性间的关系,并与其它相关方法进行比较,进一步验证了本文方法的有效性.

2　IDEEP 方法

从结构化程序设计到 OO 程序设计,再到面向构件/服务的软件开发方式,软件设计和实现的重点已从实现局部的编程难题,转向了如何将代码进行有效的组织[17].特别是对于大规模复杂软件系统而言,结构的组织尤为重要.类是代码元素的抽象,对于理解和维护系统是至关重要的,比较适合小型软件的代码组织,但是对于规模较大的软件,仅用类来组织就未免粒度太细了,以至于难以获得对系统的较好理解[18].一般而言,粒度越细越微观,个性化程度就越高;反之,粒度越粗越宏观,共性越明显[19].因此,对于规模较大的软件用包将相关的类组织起来,有助于人们对系统的理解[18].本文用软件网络模型抽象包粒度软件的拓扑结构,提出了一种基于加权 PageRank 算法的关键包识别方法 IDEEP.以 IDEEP 提供的方法量度包的重要性,并选择重要性排名靠前的包开始分析和理解软件有助于简化维护工作.图 1 给出了 IDEEP 方法的基本框架.以下各小节将对框架中的主要部分进行说明,并给出相关概念的定义.

2.1　面向对象软件

本文主要关注 OO 软件,并使用开源的 Java 软件作为研究对象.使用开源软件作为研究对象,主要考虑到开源的系统比较容易获取源代码,同时没有版权问题,可以自由使用,便于实验的重复.使用 Java 语言编写的软件主要是基于以下几点考虑:

(1)OO 技术自上个世纪 90 年代开始已经成为广泛使用的开发范型,网络上(如开源项目托管网站 Sourceforge、Google Code 等)存储着大量开源的用 Java 语言编写的软件系统,实验对象容易获取.

(2)这些软件采用 OO 开发模式,具有相对清晰的内部结构,易于编写工具提取软件实体及它们之间的关系.

(3)我们的分析工具可以较好地解析 Java 语言编写的软件.

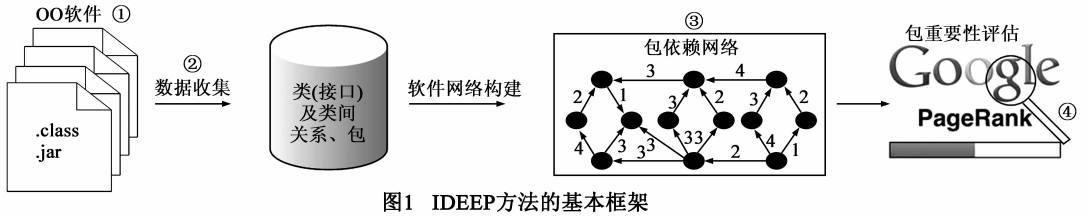

图1 IDEEP方法的基本框架

2.2 数据收集

IDEEP 方法的第一步是收集数据.通过解析 Java 字节码文件(.class 文件和.jar 文件)提取软件实体及它们之间的依赖关系(依赖关系详见 2.3 节软件网络的定义).本文所有的数据都是由我们自主研发的软件网络分析工具 SNAT[6,20](Software Network Analysis Tool)自动提取的.SNAT 可以实现从字节码文件生成多个粒度的特定软件网络,然后对软件网络进行一系列的度量、分析和可视化.

2.3 软件网络模型

如果将系统中的各个实体视为节点,实体间的相互关系表示为节点间的(有向)边,软件结构实质上表现为一种内部互连的复杂网络拓扑的形态.随着规模的剧增,软件的"网络化"趋势越来越明显,即软件系统可以表示为抽象的复杂网络模型.

目前,软件网络主要集中于三个粒度[6]:方法/属性粒度、类/接口粒度和包粒度.本文旨在识别软件中的关键包,因此我们将使用包粒度的软件网络抽象系统内包及包间依赖关系.下面首先给出包粒度软件网络的定义.

定义 1 包依赖网络(Package Dependency Network,简称 PDN):PDN 可以用一个加权有向网络表示,即:PDN = (N_p, E_p).其中:N_p 是节点的集合,表示一个特定软件中所有包的集合;E_p 是有向边的集合,表示包之间的依赖关系,即:$E_p = \{ < p_i, p_j > \}, p_i \in N_p, p_j \in N_p$.在 PDN 中包之间的依赖关系是从这些包所包含的类之间的依赖关系中提取的,即:不在同一个包中的两个类间若存在依赖关系,则相应的两个包之间也存在依赖关系.每条边 $< p_i, p_j >$ 被赋予一个整数值 w_{ij},代表该边两端(包)节点间依赖关系的强弱.PDN 的连接矩阵 ψ 描述了包节点间的连接关系:

$$\boldsymbol{\psi}_{ij} = \begin{cases} w_{ij} < p_i, p_j > \in E_p \wedge w_{ij} \geq 1 \\ 0, \text{otherwise} \end{cases} \quad (1)$$

其中:$\boldsymbol{\psi}$ 是一个 $A \times A$ 的矩阵,$A = |N_p|$ 表示节点的数量.若 $\boldsymbol{\psi}_{ij} = w_{ij}$,则第 i 个包节点和第 j 个包节点间存在有向边 $< p_i, p_j >$ 相连,否则不存在边.

我们使用边上的权值来量度包间依赖关系的强弱,使得构建的软件网络模型更能充分的刻画真实系统的结构特征.但是这同时带来了一个新的问题:如何计算边上的权值呢?如定义 1 所述,我们通过两个包内部类之间的依赖关系来量度这两个包之间关系的强弱.为了计算边上的权值,我们引入类粒度的软件网络抽象类及类间依赖关系.

定义 2 类依赖网络(Class Dependency Network,简称 CDN):CDN 可以用一个无权有向网络表示,即:CDN = (N_c, E_c).其中:N_c 是节点的集合,表示一个特定软件中所有类的集合(本文中若不明确指出,在谈及类时都包含了接口);E_c 是有向边的集合,表示类之间的依赖关系,即 $E_c = \{ < c_k, c_l > \}, c_k \in N_c, c_l \in N_c$.在 CDN 中,类 c_k 和类 c_l 在以下四种情况下会存在有向边 $< c_k, c_l >$:

(1)类 c_k 通过关键词 extends 继承自 c_l.

(2)类 c_k 通过关键词 implements 实现了接口 c_l.

(3)类 c_k 具有一个类型为 c_l 的属性.

(4)类 c_k 的对象的方法调用了类 c_l 对象的方法.

令操作符 $\text{gcp}(c_k)$ 返回类 c_k 所属的包名.如果 $< c_k, c_l > \in E_c \wedge \text{gcp}(c_k) \neq \text{gcp}(c_l)$,其中 $c_k \in N_c, c_l \in N_c$,那么 $< p_k, p_l > \in E_p, p_k = \text{gcp}(c_k), p_l = \text{gcp}(c_l)$.

基于 CDN,我们可以通过式(2)计算边 $< p_i, p_j >$ 上的权重 w_{ij}:

$$w_{ij} = \left| \bigcup_{m \in \text{gpc}(i)} R_m \cap \text{gpc}(j) \right| \quad (2)$$

其中:操作符 $\text{gpc}(i)$ 返回包 i 包含的类的集合;R_m 表示在 CDN 中从类 m 出发,沿着箭头所指方向可以到达的所有类节点的集合;$|*|$ 返回集合 $*$ 中元素的个数.

图 2 显示的是一个 Java 代码片段及其相应的 CDN 和 PDN.我们将以边 $< p_1, p_2 >$ 上的权值计算过程为例详细说明如何计算 PDN 边的权值.因为 p_1 包内的类 p_1.classY 和 p_1.classZ 都直接依赖于包 p_2 中的类 p_2.classW,但是 p_2 中的类都没有直接依赖于 p_1 中的类,所以 p_1 和 p_2 间仅存在有向边 $< p_1, p_2 >$.同时,因为 $\text{gpc}(p_1) = \{p_1.\text{classY}, p_1.\text{classZ}\}$, $R_{p_1.\text{classY}} = \{p_2.\text{classW}, p_2.\text{classX}\}$, $R_{p_1.\text{classZ}} = \{p_1.\text{classY}, p_2.\text{classW}, p_2.\text{classX}\}$, $\text{gpc}(p_2) = \{p_2.\text{classW}, p_2.\text{classX}\}$,所以 $w_{p_1,p_2} = |\{p_2.\text{classW}, p_2.\text{classX}\} \cup \{p_1.\text{classY}, p_2.\text{classW}, p_2.\text{classX}\} \cap \{p_2.\text{classW}, p_2.\text{classX}\}| = 2$.

2.4 PageRank 算法及包重要性量度

本文基于软件网络结构,并借鉴 PageRank 算法排

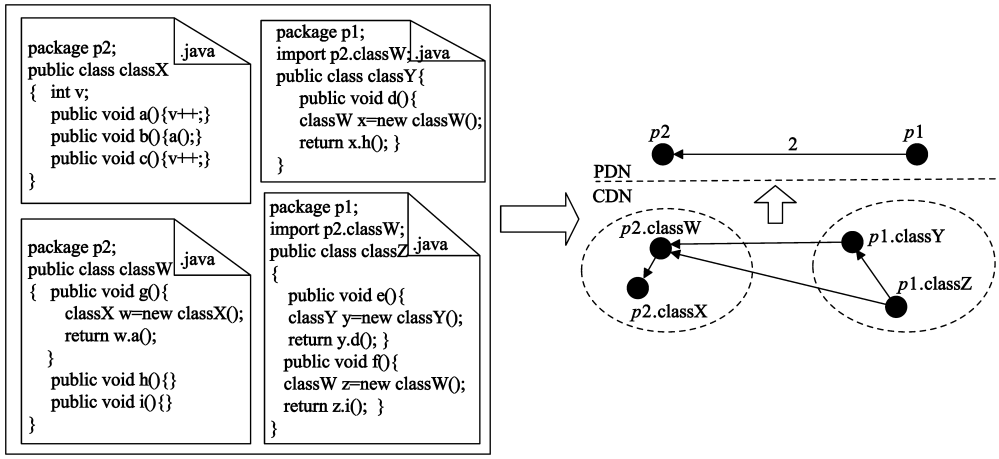

图2　代码片段及其相应的CDN和PDN

序网页的思想,挖掘软件中的关键包.因此,本节首先简要介绍 PageRank 算法,然后给出本文量度包重要性的指标及相应的指标计算方法.

2.4.1　PageRank 算法

PageRank 算法是由 Standford 大学的研究人员 Brin 和 Page 提出的,用于对搜索引擎的页面质量进行评价,为每个网页赋予一个衡量其重要性的权威值(称为 Pr 值),最早被用于检索结果的排序,后来也应用于网页抓取、文档聚类、词义消歧、图像检索等诸多方面[21].

PageRank 算法的基本思想是:从网页 A 指向网页 B 的链接被看作是页面 A 对页面 B 的支持投票,并基于"由较多重要网页链接的网页,必定是重要的网页"的想法来判断所有网页的重要性.在 PageRank 中,网页的重要性是通过该网页被访问到的概率 $Pr(t)$(通常称为 rank 值)来量度的.Brin 和 Page 为用户在 Web 上的浏览行为建立了一个模型:假设浏览者当前在浏览某一网页,下一步他将以概率 d(一般称为阻尼系数)顺着超链接点击访问,或者以概率 $1-d$ 从任意的一个新的页面开始访问.在该模型下,某一个页面 u 被访问到的概率 $Pr(u)$只与指向它的页面的概率有关,并通过式(3)计算:

$$Pr(u) = \frac{1-d}{\mathrm{MAX}} + d\sum_{v\in I(u)}\frac{Pr(v)}{|O(v)|} \tag{3}$$

其中:阻尼系数 d 一般根据经验设置,MAX 为页面的总数,$I(u)$是页面 u 的链入页面集,$O(v)$是页面 v 的链出页面集,$|S|$为集合 S 中元素个数.$Pr(u)$反映了页面 u 的重要程度,在 PageRank 算法中将之作为页面质量的评价参数,它采用的方法是给每个网页一个初始值,然后利用式(3)进行迭代运算,直到计算出的值收敛为止.

2.4.2　包重要性量度

在软件系统中,被频繁使用的包和不常被使用的包往往具有不同的重要性.如果一个包被频繁使用,一旦这个包存在错误,错误可以直接影响大量的包,而不常被使用的包中的错误其直接影响的包往往较少.特别当被频繁使用的包之间互相使用时,错误的影响将更加深远.所以一个有较多重要包使用的包往往是重要的包.

当软件用 PDN 抽象时,PDN 中的包节点相当于网页,包之间的有向边相当于网页间的超链接.因此,PageRank 算法的思想可以很自然地扩展到 PDN 中,挖掘软件中重要的包,以指导软件的理解.然而本文提出的 PDN 是一个加权的有向网络,最初的 PageRank 算法适应于无权的有向网络,为了使式(3)适应加权的情况,我们对其做了适当的修改,提出了一个用于量度包重要性的指标 PackageRank(简称 PR 值),计算公式如式(4):

$$\mathrm{PR}(p_i) = \frac{1-d}{|N_p|} + d\sum_{p_j\in I(p_i)}\frac{w(p_j,p_i)\mathrm{PR}(p_j)}{|\mathrm{WO}(p_j)|} \tag{4}$$

其中:$\mathrm{PR}(p_i)$是包 p_i 的 PR 值;d 是阻尼系数示;$|N_p|$是包集合 N_p 中包的个数;$I(p_i)$是包 p_i 的链入包集;$\mathrm{WO}(p_j)$是包 p_j 的所有链出边的权值和;$w(p_j,p_i)$是 p_j 和 p_i 两个包之间边 $<p_j,p_i>$ 上的权值.在本文中 d 设置为 0.85(该值是 PageRank 算法的常用取值),所有包的初始 PR 值为 1,然后利用式(4)进行迭代运算,直到计算出的值收敛为止.迭代过程见算法 1.

算法 1　PackageRank 算法

输入:PDN;

输出:包名及其相应的 PR 值;

1. 初始化 sum,SqDev 为 0,PR(i),PRbak(i) (i = 1,2,…,$|N_p|$)为 1,$\varepsilon = e^{-6}$

2. WHILE 条件永真 {

3.　for i = 0 to $|N_p|$ do

4.　　for j = 0 to $|N_p|$ do

5.　　if $w(j,i)>0$ then //存在边

```
6.          sum + = (PR(j) * w(j,i))/WO(j)
7.        end if
8.      end for
9.      PR(i) = (1 - d)/|Np| + d * sum
10.     SqDev + = ‖PR(i) - PRbak(i)‖    //求方差
11.     PRbak(i) = PR(i)//PRbak 为 PR 值的备份
12.     sum = 0
13.   end for
14.   if(sqDev < ε)
15.     return 包名及其相应的 PR 值
16. end while
```

算法 1 可以在 $O(|N_p|^2)$的时间复杂度内求得规模为$|N_p|$的 PDN 中所有包的 PR 值.

3 实验

为了分析本文方法的有效性,我们针对多个开源软件系统设计了对比实验.

3.1 研究问题

本文的实验关注于回答如下三个研究问题:

问题 1 PR 值与其它复杂网络中心性指标间的关系?

如前所述,一些研究者已经将复杂网络中量度节点中心性的指标引入软件中量度类的重要性.我们希望知道本文提出的 PR 度量指标与这些中心性指标间的相关性,即 PR 与各个指标在结果上的相似程度.

问题 2 IDEEP 方法相比于其它同类方法在识别关键节点方面是否更有效?

目前的方法都是针对关键类识别的,我们将用部分现有的方法识别关键包,并与本文提出的方法进行比较,从性能角度比较各种方法在关键节点识别方面的性能,从而说明本文方法的有效性.

问题 3 IDEEP 方法识别的关键包与软件外部质量属性的关系?

在真实的应用环境中检验一个度量指标的有效性是很必要的,我们必须揭示度量指标与软件外部质量属性间的关系[22].因此,我们希望知道包的 PR 值与包的可理解性之间的关系.

3.2 目标系统

本实验以开源软件项目 Azureus(http://azureus.sourceforge.net/)、Tomcat(http://tomcat.apache.org/)、JMeter(http://jmeter.apache.org/)、JFreeChart(http://www.jfree.org/jfreechart/)、XGen(http://sourceforge.net/projects/xgen/)、Jakarta ECS(http://jakarta.apache.org/ecs/)为实验对象.其中:Azureus 是一款用 Java 编写的 BitTorrent 客户端应用程序;Tomcat 是一款用 Java 编写的免费的 Web 服务器;JMeter 是一个桌面应用,用于压力测试和性能测量;JFreeChart 是一款用 Java 编写的开放的图表绘制类库;XGen 是一个从结构化的文本输入创建文本输出的工具;ECS 是一个用于建标记语言文档的 Java API.之所以选择这些项目是因为它们曾多次成为一些工作的研究对象,并且它们来自不同的领域,在一定程度上可以使所获得的结论具有一定的普遍意义.我们从项目的版本控制库获得相应版本的源代码.每个项目的基本数据如表 1 所示.

表 1 目标系统的基本情况

项目	版本号	代码行	包数	类数	方法/属性数
Azureus	3.0.1.4	307,021	428	5,102	47,434
Tomcat	6.0.18	161,933	166	2,331	39,158
JMeter	2.0.1	78,304	290	3,477	45,936
JFreeChart	1.0.12	137,034	107	1,959	29,453
XGen	0.5.0	5,457	21	74	848
Jakarta ECS	1.4.2	28,691	13	390	6,016

3.3 实验过程

我们使用 SNAT 解析实验软件系统,收集软件结构信息,构建 CDN,并基于 CDN 构建相应的 PDN.因为复杂网络中很多中心性指标的计算都要求节点所在网络是弱连通的,所以本文只关注 PDN 的最大弱连通子图.

为了说明目标系统 PDN 的一些结构特征,我们使用复杂网络研究中常用的统计参数分析了这些 PDN,结果如表 2 所示.其中:$|N_p|$、$|E_p|$分别代表网络的节点数和边数;$<k>$是网络节点的平均度;d 是网络的直径;L 是平均路径长度;C 代表聚类系数;H 是网络的异质度.参数的定义详见文献[23].因为很多指标都是针对无向网络的,在计算的过程中我们忽略了边的方向.

表 2 目标系统 PDN 的统计参数

项目	$\|N_p\|$	$\|E_p\|$	$<k>$	d	L	C	H
Azureus	419	2,865	13.675	5	2.494	0.472	1.445
Tomcat	156	635	8.141	6	2.963	0.549	1.015
JMeter	282	1,529	10.844	7	3.263	0.556	0.980
JFreeChart	96	437	9.104	9	3.015	0.600	0.907
XGen	17	30	3.529	3	1.971	0.451	0.837
Jakarta ECS	11	15	2.727	2	1.727	0.571	0.899

3.4 实验结果及分析

本节将围绕 3.1 节提出的三个研究问题分析实验得到的结果.

3.4.1 PR 值与其它复杂网络中心性指标间的关系

IDEEP 对一个特定系统的包进行重要性评价,赋予一个 PR 值,PR 值越大的包越重要.这一过程实质是对包重要性的一种排序,PR 值越大排名越靠前.

在统计学中,等级相关系数常用于量度两种排序结果的相似性及相关性的强弱. Kendall 等级相关系数 τ[24]是其中比较常用的一个指标. τ 的取值范围在 -1 到 1 之间:$\tau=1$ 表示两个排序结果拥有一致的等级相关性;$\tau=-1$ 表示两个排序拥有完全相反的等级相关性;$\tau=0$ 表示两个排序是相互独立的.一般有三种方式计算 τ:Tau-a τ_A、Tau-b τ_B 和 Tau-c τ_C.鉴于不同的包可能有相同的 PR 值(具有相同的等级),我们使用 τ_B 量度 PR 与其它复杂网络中心性指标在对包重要性排序时排序结果的相似关系.

假设 X、Y 是两个随机变量(也可以看作两个集合),都有 n 个元素,第 $i(1<=i<=n)$个元素分别用 x_i 和 y_i 表示. X 与 Y 中的对应元素组成一个元素对集合 $XY,(x_i,y_i)\in XY(1<=i<=n)$. 当集合 XY 中任意两个元素(x_i,y_i)与(x_j,y_j)的排名相同时(即当出现情况 1:$x_i>x_j$ 且 $y_i>y_j$ 或情况 2:$x_i<x_j$ 且 $y_i<y_j$),这两个元素就被认为是一致的. 当出现情况 3(情况 3:$x_i>x_j$ 且 $y_i<y_j$)或 4 时(情况 4:$x_i<x_j$ 且 $y_i>y_j$),这两个元素被认为是不一致的. 当出现情况 5(情况 5:$x_i=x_j$)或 6 时(情况 6:$y_i=y_j$),这两个元素既不是一致的也不是不一致的. τ_B 定义如下:

$$\tau_B=\frac{C-D}{\sqrt{(N_3-N_1)(N_3-N_2)}} \tag{5}$$

其中:C 表示 XY 中拥有一致性的元素对数;D 表示 XY 中拥有不一致性的元素对数;$N_3=n(n-1)/2$;$N_1=\sum_i^s t_i(t_i-1)/2$;$N_2=\sum_{j=1}^t u_j(u_j-1)/2$. N_1、N_2 分别是针对集合 X、Y 计算的:将 X 中的相同元素分别组合成小集合,s 表示小集合数,t_i 表示第 i 个小集合所包含的元素数.同理,我们可以在集合 Y 上求得 N_2.

图 3 显示了 PR 与其它中心性指标(包括介数(Betweenness)中心性、接近(Closeness)中心性、度数(Degree)中心性、k-核心(k-core)和 Pr)[23]在对目标系统包重要性量度时结果间的关系图.因为除 Pr 外,其它几个指标都是针对无向图的,在求这几个指标时我们忽略了边的方向.同时,我们使用 τ_B 量度 PR 与这些中心性指标排序结果的相似关系(结果如表 3 所示).从图 3 和表 3 可见,除了 ECS 外,在其它五个系统中,PR 与所有的中心性指标间具有显著性水平 0.01 的强正相关性.在 ECS 中,除了与 k-core 不存在明显的相关性外,PR 与其它几个中心性指标间具有显著性水平 0.05 的强正相关性.

表 3　PR 与其它中心性指标间的 τ_B

项目	Betweenness	Closeness	Degree	k-core	Pr
Azureus	0.370**	0.179**	0.436**	0.349**	0.884**
Tomcat	0.580**	0.323**	0.539**	0.416**	0.874**
JMeter	0.484**	0.294**	0.474**	0.349**	0.889**
JFreeChart	0.646**	0.359**	0.528**	0.391**	0.887**
XGen	0.543**	0.521**	0.659**	0.590**	0.960**
Jakarta ECS	0.578*	0.610*	0.610*	0.484	0.800**

注:** 表示 0.01 的显著性水平(双侧检验),* 代表 0.05 的显著性水平(双侧检验)

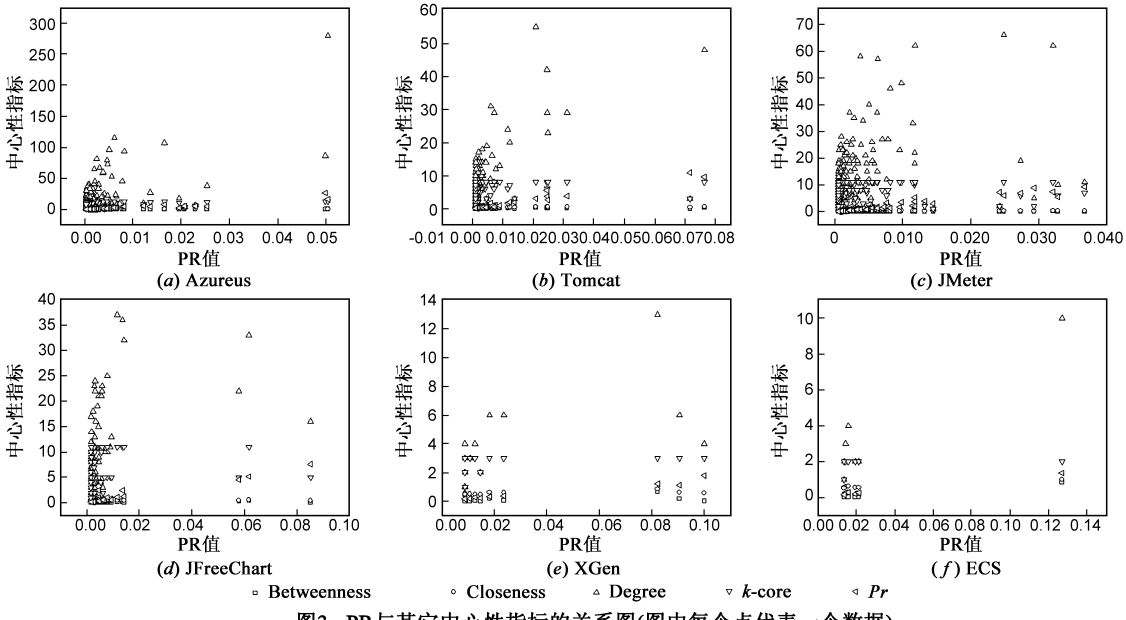

图3　PR与其它中心性指标的关系图(图中每个点代表一个数据)

为了分析在目标系统中,PR 与哪个中心性指标最相近,我们使用 Friedman 测试[25]对表 3 的数据进行进

一步分析(结果如表 4 所示). Friedman 测试广泛用于评价不同算法间的性能. 然而,在文献[25]中算法求解的是最小化问题,因而排名值越大算法性能越好. 然而本文中是 τ_B 越大,相关性越强,因此排名值越小相关性越强. 从表 4 可见,PR 排名与 *Pr* 最接近,与 Closeness 差别最大.

表 4 相关性强弱的排名

τ_B	排名
PR 与 *Pr*	1.000
PR 与 Betweenness	2.600
PR 与 Degree	2.600
PR 与 *k*-core	3.800
PR 与 Closeness	5.000

注:Friedman 测试的数据是表 3 中除 ECS 系统外的数据,因为 ECS 和其它系统 τ_B 的显著度水平不同

3.4.2 IDEEP 方法相比于其它同类方法在识别关键节点方面是否更有效?

在复杂网络研究中,常通过分析 top-*k*(重要性排名靠前的 *k* 个节点)个节点的传播能力(spreading influence)来检验方法的有效性[26~28]. 为此,本文将引入 SIR (Susceptible-Infected- Recovered)模型. SIR 模型在复杂网络中得到了广泛的研究,如用于研究疾病传播、金融危机扩散、舆论传播等[26~28]. 在软件工程中,类似的算法被用于分析变更传播[29]和错误传播[30]等.

在 SIR 模型中,个体(节点)存在三种状态:易感态(S),感染态(I)和恢复态(R). 处于 S 态的个体还未被疾病感染,但是若接触到处于 I 态的个体,则会以一定的概率被感染而处于 I 态;处于 I 态的个体已被疾病感染,同时又有能力将疾病传染给处于 S 态的个体. 处于 R 态的个体曾被疾病感染,而后被治愈而处于 R 态,并且具有了免疫能力,不会再次感染上疾病,也不会将疾病传播给其它个体. 但是在 SIR 模型中,处于 I 态的个体通常以固定的概率感染与其接触的 S 态个体. 然而 PDN 是一个加权有向网络,我们将引入一种融入了边权影响的传播概率计算方法(类似的计算边权的方式曾用于金融危机扩散研究中[28]). 本文中边权通过式(6)计算:

$$p_{ij} \propto m \cdot w_{ij} / \sum_i w_{ij} \tag{6}$$

其中:p_{ij}代表处于 I 态的个体 *i* 感染处于 S 态的接触个体 *j* 的概率;w_{ij}是个体 *i* 和 *j* 间边的权值;*m* 是一个放大系数,用于抽象疾病的强弱,它可以取任意正数. 在软件系统中,疾病可以看成是软件中的变更、错误(或缺陷).

本文提出了 SIR 模型的一个新变体——wSIR (weighted SIR)模型,它通过边权调整疾病的传播过程. wSIR 首先将所有个体的状态设为 S. 然后,选择要计算重要性的节点 *i*,将其状态设为 I. 该节点 *i* 将以一定的概率(通过式(6)计算)将疾病传染给与其直接接触(在网络中存在指向节点 *i* 的边)的处于 S 态的个体,将它们的状态从 S 设为 I. 而触发这一次传播的个体将设为 R 态. 在以后的步骤中,这一过程将不断重复,所有处于 I 态的个体都会以一定的概率(通过式(6)计算)影响直接接触的 S 态个体,直到网络中没有处于 I 态的个体为止. wSIR 模型和文献[29]中的模型都可以看成 SIR 在软件工程中的应用,两者最大区别在于:(1)文献[29]的变更传播概率是假设的且所有边都相等,而本文中的概率是根据边权计算的;(2)文献[29]未考虑变更的严重性,而本文通过引入 *m* 考虑了该类因素的影响.

在计算某个节点的重要性时,我们会独立的运行 wSIR 模型 1,000 次,然后计算平均受感染节点数,并将其作为对该节点重要性的评价. *m* 是 wSIR 模型的唯一参数,尽管 *m* 可以设置成任意的正整数,但若 *m* 设置的太小,受疾病节点影响的节点数都较小,边权差异带来的影响不明显;若 *m* 设置的太大,大部分疾病节点都几乎能影响所有的节点,边权差异带来的影响也不明显. 本文针对不同的系统,*m* 取自不同的范围,即:Azureus($m \in [2,13]$),Tomcat($m \in [3.5,15]$),JMeter($m \in [4,14]$),JFreeChart($m \in [3,29]$),XGen($m \in [1.5,5]$),Jakarta ECS($m \in [1,11]$). 其中下限和上限分别是影响范围最大的节点影响的节点数占总节点数 1/4 和 3/4 时 *m* 的最小取值(*m* 从 1 开始,以 0.5 的步长尝试取值).

软件的关键实体一般占了很小的比例,在类粒度的研究中只占了 1.5% – 2.0%[4]. 同时,检查大多数关键实体带来的代价也是无法接受的. 所以需要选择一个合适的关键实体的数量. 在识别关键包时,按照度量值将所有包降序排列时,我们针对不同的系统规模选择了排名靠前的少量的包作为关键包:Azureus 前 15 个;Tomcat 前 10 个;JMeter 前 10 个;JFreeChart 前 10 个;XGen 前 7 个;ECS 前 3 个.

图 4 显示了在计算某一节点重要性时受影响的节点数随 *m* 变化的曲线. 在求平均受影响节点数时,在取排名靠前的节点时,若碰到最后几个节点的排名一样时,我们是在这些具有相同排名的节点中随机采样 100 次,求平均受影响的节点数. 从图 4 可见,平均受影响节点数随着 *m* 的增长不断增加,并且在所有的系统中,由 PR 识别的关键包都能比其它方法识别的关键包致使更多的节点受影响. 可见,相比其它方法,PR 在识别关键包上更有效. 同时可以注意到,因为系统 XGen 和 ECS 的规模比较小,在图 4(*e*)在中,PR 和 *Pr* 的曲线重合,

Degree 和 Betweenness 的曲线也重合;在图 4(*f*)中,除 PR 外其它几条曲线也重合.通过跟踪 wSIR 模型的执行,我们发现这几条曲线重合的原因在于:我们关注的是排名靠前的关键包,在这几个重合的曲线中,排名靠前的几个包尽管相对排名不同,但都是同样的几个包.

3.4.3 IDEEP 方法识别的关键包与软件外部质量属性的关系?

我们使用 PR 量度包的重要性.然而度量指标的有效性必须在真实的应用环境中检验.本节将仅以 XGen 和 ECS 两系统为对象分析包的 PR 值与包的可理解性之间的关系(因为其它四个系统可理解性方面的数据未曾报道过,所以无法使用).

表 5 和表 6 分别显示的是 XGen 和 ECS 中包的 PR 值.尽管 XGen 原有 21 个包,ECS 原有 13 个包,但因文献[31]只讨论 XGen 中的六个包,ECS 中的十二个包,我们也仅考虑这些包.若某个包下有子包,则将子包的 PR 值累加到该包的 PR 值中,如:在计算 workzen.xgen.ant 的 PR 值时,我们将其下的子包 workzen.xgen.ant.legacy 的 PR 值也加到 workzen.xgen.ant 的 PR 值中.同时,因为 ECS 中的 org.apache.ecs.factory 和 org.apache.ecs.storage 是是孤立点,没有与其它包发生依赖关系,令它们的 PR 值为 0.

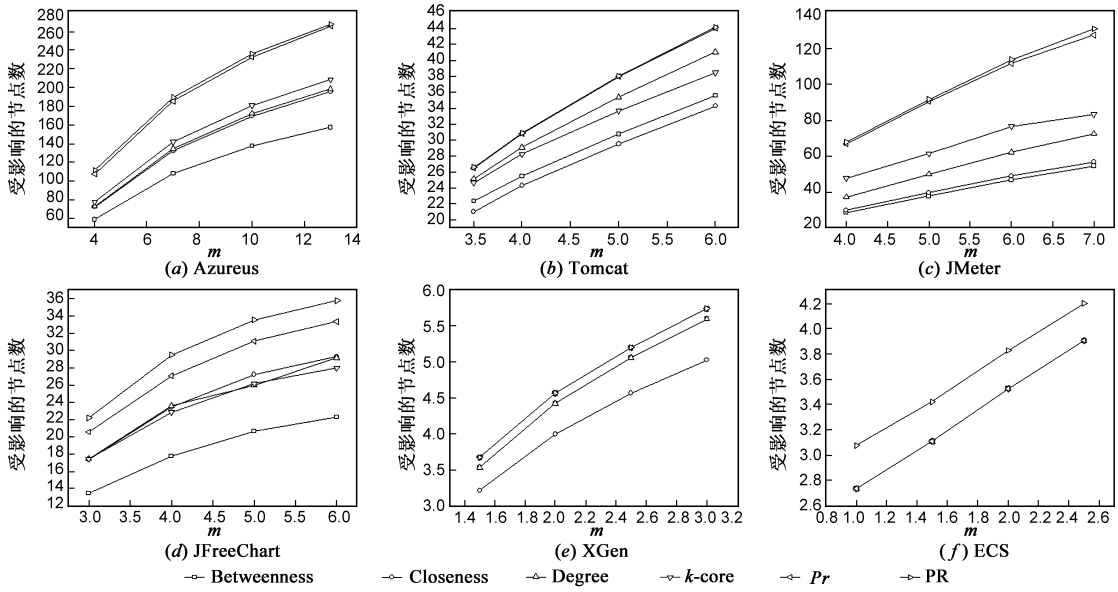

图4　受影响节点数随*m*取值的变化曲线

表 5　XGen 中各包的 PR 值

包名	PR	包名	PR
Workzen.xgen.ant	0.008333334	Workzen.xgen.model	0.05914352
Workzen.xgen.engine	0.011875001	Workzen.xgen.test	0.058333338
Workzen.xgen.loader	0.028796297	Workzen.xgen.util	0.025922455

表 6　ECS 中各包的 PR 值

包名	PR	包名	PR
org.apache.ecs.examples	0.012500000	org.apache.ecs.rtf	0.012500000
org.apache.ecs.factory	0.000000000	org.apache.ecs.storage	0.000000000
org.apache.ecs.filter	0.050276300	org.apache.ecs.vxml	0.012500000
org.apache.ecs.html	0.180721800	org.apache.ecs.wml	0.012500000
org.apache.ecs.html2ecs	0.012500000	org.apache.ecs.xhtml	0.174819020
org.apache.ecs.jsp	0.012500000	org.apache.ecs.xml	0.015729759

PR 作为包重要性的量度,是软件的一个内部属性,它应该与软件的外部属性(可理解性、可维护性、正确性等)存在一定的关联.Gupta 和 Chhabra[31] 曾对 XGen 和 ECS 两个系统所有包的可理解性做过人为的分析,并对理解一个包需要的代价做过打分,分越高,代价越大,结果如表 7 所示.

表 7　理解包所需的代价

包名	代价	包名	代价
Workzen.xgen.ant	2.0	org.apache.ecs.html	9.0
Workzen.xgen.engine	2.3	org.apache.ecs.html2ecs	1.3
Workzen.xgen.loader	5.3	org.apache.ecs.jsp	1.6
Workzen.xgen.model	6.0	org.apache.ecs.rtf	2.3
Workzen.xgen.test	4.3	org.apache.ecs.storage	2.0
Workzen.xgen.util	2.6	org.apache.ecs.vxml	1.3
org.apache.ecs.examples	1.3	org.apache.ecs.wml	8.0
org.apache.ecs.factory	1.0	org.apache.ecs.xhtml	9.0
org.apache.ecs.filter	2.0	org.apache.ecs.xml	4.0

为了分析 PR 值与包可理解性之间的关系,我们计算了两者之间的 Spearman 相关性系数,结果如表 8 所示.同时,表 8 还包含了其它六种方法与包可理解性之间的 Spearman 相关性系数.从表 8 可见,PR 与包可理解性之间存在显著水平 0.01 的强正相关性,比文献[31]

中的 PCM 方法要更强.说明随着包 PR 值的增大,理解包所需的代价也越大.同时我们发现,其它五种方法与包理解性之间不存在明显的相关性.可见,相比其它方法,用 PR 识别的关键包,理解它们的代价往往也是较高的.

表 8 Spearman 相关性测试

方法	Spearman 相关系数
PR	0.707**
PCM	0.730*
Betweenness	0.288
Closeness	0.141
Degree	0.249
k-core	0.199
Pr	0.321

注:** 表示 0.01 的显著性水平(双侧检验),* 代表 0.05 的显著性水平(双侧检验),无星标的表示不存在明显的相关性

4 结论和下一步工作

本文从软件静态结构出发,对关键包的识别问题进行了研究:根据类、包及其间依赖关系构建了加权有向软件网络模型,基于该模型提出了量度包重要性的指标 PR 及其计算方法,从而实现从软件代码中自动识别关键包.开源 Java 软件上的实例研究表明:本文的方法可以有效的识别软件中的关键包,相比于其它多种方法具有更好的性能.

结构复杂性是软件复杂性的重要方面,与软件的外部质量属性具有密切联系,同时也极大地影响着人们对于软件的理解及软件的维护费用.本文的主要贡献在于将软件结构信息融入包重要性量度中,并用加权的 PageRank 算法求解包的重要性,实现关键包识别,目标是帮助开放人员理解软件,降低软件维护费用.

未来将考虑如下研究工作:(1)用更多不同领域、不同规模的 Java 软件验证本文方法的有效性;(2)分析包重要性与软件其它质量属性(如可靠性、易变性等)间的关系,进一步检验度量指标在实际环境中的有效性和适应性;(3)结合软件执行信息提高关键包识别性能.

参考文献

[1] Yau S S, Collofello J S. Some stability measures for software maintenance[J]. IEEE Transactions on Software Engineering, 1980, SE-6(6): 545-552.

[2] Guimaraes T. Managing application program maintenance expenditure [J]. Communication of ACM, 1983, 26(10): 739-746.

[3] Corbi T A. Program understanding: Challenge for the 90s [J]. IBM Systems Journal, 1990, 28(2): 294-306.

[4] Zaidman A, Demeyer S. Automatic identification ofkey classes in a software system using Web mining techniques [J]. Journal of Software Maintenance and Evolution: Research and Practices, 2008, 20(6): 387-417.

[5] Ko A J, Myer B A, Coblenz M J, et al. An exploratory study of how developers seek, relate, and collect relevant information during software maintenance tasks [J]. IEEE Transaction on Software Engineering, 2006, 32(12): 971-987.

[6] Pan W F, Li B, Ma Y T, et al. Multi-granularity evolution analysis of software using complex network theory [J]. Journal of Systems Science and Complexity, 2011, 24(6): 1068-1082.

[7] 李兵,王浩,李增扬,等.基于复杂网络的软件复杂性度量研究 [J].电子学报,2006,34(12A):2371-2375.
Li B, Wang H, Li Z Y, et al. Software complexity metrics based on complex networks [J]. Acta Electronica Sinica, 2006, 34(12A): 2371-2375. (in Chinese)

[8] Potanin A, Noble J, Frean M, et al. Scale-free geometry in object-oriented programs [J]. Communications of the ACM, 2005, 48(5): 99-103.

[9] 吕金虎,王红春,何克清.复杂动力网络及其在软件工程中的应用 [J].计算机研究与发展,2008,45(12):2052-2059.
Lv J H, Wang H C, He K Q. Complex dynamical networks and their applications in software engineering [J]. Journal of Computer Research and Development, 2008, 45(12): 2052-2059. (in Chinese)

[10] 李辉,赵海,徐久强,等.基于 k-核的大规模软件宏观拓扑结构层次性研究 [J].电子学报,2010,38(11):2635-2643.
Li H, Zhao H, Xu J Q, et al. Research on hierarchy of large-scale software macro-topology based on k-core [J]. Acta Electronica Sinica, 2010, 38(11): 2635-2643. (in Chinese)

[11] Albert R, Jeong H, Barabási A L. Error and attack tolerance in complex networks [J]. Nature, 2000, 406(6794): 378-382.

[12] Kitsak M, Gallos L, Havlin S, et al. Identification of influential spreaders in complex networks [J]. Nature Physics, 2010, 6(11): 888-893.

[13] Li D W, Li B, He P, et al. Ranking the importance of classes via software structural analysis [A]. Proceedings of the 2011 International Conference on Future Communication, Computing, Control and Management [C]. Phuket, Thailand, Springer, 2011. 441-449.

[14] Wang M C, Pan W F. A comparative study of network centrality metrics in identifying key classes in software [J]. Journal of Computational Information Systems, 2012, 8(24): 10205-10212.

[15] 周毓明,徐宝文.基于依赖结构分析的类重要性度量方法 [J].东南大学学报:自然科学版,2008,38(3):380-

384.
Zhou Y M, Xu B W. Dependence structure analysis-based approach for measuring importance of classes [J]. Journal of Southeast University (Natural Science Edition), 2008, 38(3): 380 – 384. (in Chinese)

[16] 王木生,卢红敏,周毓明,等.利用 h 指数及其衍生度量识别关键类 [J].计算机科学与探索,2011,5(10):809 – 903.
Wang M S, Liu H M, Zhou Y M, et al. Identifying key classes using h-index and its variants [J]. Journal of Frontiers of Computer Science and Technology, 2011, 5(10): 809 – 903. (in Chinese)

[17] 潘伟丰,李兵,邵波,等.基于软件网络的服务自动分类和推荐方法研究 [J].计算机学报,2011,34(12):2355 – 2369.
Pan W F, Li B, Shao B, et al. Service classification and recommendation based on software networks [J]. Chinese Journal of Computers, 2011, 34(12): 2355 – 2369. (in Chinese)

[18] Gupta V, Chhabra J K. Package level cohesion measurement in object-oriented software [J]. Journal of the Brazilian Computer Society, 2012, 18(3): 251 – 266.

[19] 韩言妮,李德毅,陈桂生.软件网络的多粒度拓扑特性分析及其应用 [J].计算机学报,2009,32(9):1711 – 1721.
Han Y N, Li D Y, Chen G S. Analysis on the topological properties of software network at different levels of granularity and its application [J]. Chinese Journal of Computers, 2009, 32(9): 1711 – 1721. (in Chinese)

[20] Pan W F, Li B, Ma Y T, et al. Measuring structural quality of object-oriented softwares via bug propagation analysis on weighted software networks [J]. Journal of Computer Science and Technology, 2010, 25(6): 1202 – 1213.

[21] Brin S, Page L. The anatomy of a large-scale hypertextual web search engine [J]. Journal of Computer Networks and ISDN Systems, 1998, 30(1 – 7): 107 – 117.

[22] Basili V, Briand L, Melo W. A validation of object-oriented design metrics as quality indicators [J]. IEEE Transaction on Software Engineering, 1996, 22(10): 751 – 761.

[23] 汪小帆,李翔,陈关荣.网络科学导论 [M].北京:高等教育出版社,2012.
Wang X F, Li X, Chen G R. Network science: an introduction [M]. Beijing: High Education Press, 2012. (in Chinese)

[24] Kendall M. A new measure of rank correlation [J]. Biometrika, 1938, 30(1 – 2): 81 – 93.

[25] Garcia S, Fernandez A, Luengo J, et al. Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power [J]. Information Sciences, 2010, 180(10): 2044 – 2064.

[26] Newman M E J. Spread of epidemic disease on networks [J]. Physical Review E, 2002, 66: 016128.

[27] Colizza V, Barrat A, Barthelemy M, et al. The role of the airline transportation network in the prediction and predictability of global epidemics [J]. Proceedings of the National Academy of Sciences of the United States of America, 2006, 103(7): 2015 – 2020.

[28] Garas A, Argyrakis P, Rozenblat C, et al. Worldwide spreading of economic crisis [J]. New Journal of Physics, 2010, 12: 113043.

[29] 张莉,钱冠群,李琳.基于变更传播仿真的软件稳定性分析 [J].计算机学报,2010,33(3):440 – 451.
Zhang L, Qian G Q, Li L. Software stability analysis based on change impact simulation [J]. Chinese Journal of Computers, 2010, 33(3): 440 – 451. (in Chinese)

[30] 潘伟丰,李兵.基于软件网络错误传播分析的软件质量量度 [J].中南大学学报(自然科学版),2012,43(11):4339 – 4347.
Pan W F, Li B. Software quality measurement based on error propagation analysis in software networks [J]. Journal of Central South University (Science and Technology), 2012, 43(11): 4339 – 4347. (in Chinese)

[31] Gupta V, Chhabra J K. Package coupling measurement in object-oriented software [J]. Journal of Computer Science and Technology, 2009, 24(2): 273 – 283.

作者简介

潘伟丰(通信作者)　男,1982 年 12 月出生于浙江省杭州市,毕业于武汉大学获工学博士学位,现为浙江工商大学副教授,主要研究方向为软件工程、服务计算、复杂网络和智能计算.
E-mail: panweifeng1982@gmail.com

李　兵　男,1969 年 3 月出生于湖北省武汉市,武汉大学教授,博士生导师,主要研究方向为软件工程、云计算、人工智能和复杂网络.
E-mail: bingli@whu.edu.cn