Software Engineering and Applications 软件工程与应用, 2013, 2, 151-156
http://dx.doi.org/10.12677/sea.2013.26026 Published Online December 2013 (http://www.hanspub.org/journal/sea.html)

Research on Forecasting Structural Risks of Large-Scale Software Based on Ripple Degree*

Zheng Liu[1], Qian Zhang[2], Hai Zhao[1]

[1]College of Information Science and Engineering, Northeastern University, Shenyang
[2]Technology Strategy & Development Department, Neusoft Corporation, Shenyang
Email: liuzheng@mail.neu.edu.cn

Received: Nov. 15th, 2013; revised: Dec. 4th, 2013; accepted: Dec. 11th, 2013

Abstract: Because of the association between nodes in software network, the ripple effect exists in software network. In this paper, the distributions of forward and reversal ripple degree in many networks of open source software are analyzed firstly, and then a metric formula that evaluates the significance is presented after focusing on those nodes that have high ripple degree. According to the metric results, we can select the vulnerable nodes, rigid nodes and a "bridge" node in software structure and this method can provide guidance for design and remodeling of software structure.

Keywords: Ripple Degree; Forecasting Structural Risk; Software Structure; Software Network

基于波及度的软件结构风险预测方法的研究*

刘　铮[1]，张　骞[2]，赵　海[1]

[1]东北大学信息科学与工程学院，沈阳
[2]东软集团技术战略与发展部，沈阳
Email: liuzheng@mail.neu.edu.cn

收稿日期：2013 年 11 月 15 日；修回日期：2013 年 12 月 4 日；录用日期：2013 年 12 月 11 日

摘　要：由于软件网络中各节点之间的相互调用与关联，波及效应普遍存在于软件网络之中。本文针对大量的开源软件的网络结构进行研究，首先分析了正向波及度和逆向波及度在软件网络中的分布规律，然后对波及度较大的节点进行重点研究，提出了波及度显著性度量公式，并根据度量结果发现软件结构中的脆弱节点、僵化节点和“桥梁”节点，为软件结构设计与重构提供指导。

关键词：波及度；结构风险预测；软件结构；软件网络

1. 引言

大规模软件由于其功能复杂性所带来的结构复杂性已成为其固有的特性之一，这使得传统的软件质量控制方法往往难以得到良好的效果，为软件工程研究领域提出了新的挑战。生理学家丹特－恰尔夫认为：“人类的大脑结构是一个复杂网络，表现出非常强烈的小世界特征，能够形成最有效的连结。”作为人类大脑思维活动的产物，软件系统成为大脑结构的一个分形，其内部结构也表现出了明显的复杂网络特征。从 2002 年开始，一些复杂系统科学和统计物理领域的研究人员对大量面向对象软件系统的类图进行了研究，将类及类间关系抽象为有向图，从网络拓

*资助项目：教育部基本科研业务费项目(N110304003)。

扑的层面对软件网络的整体特性进行了研究，拉开了借鉴复杂网络理论研究软件结构的序幕[1-3]。近年来，将复杂网络的理论应用在对大规模软件系统的结构和行为特性的研究中，把软件系统抽象为一类人工复杂网络，用网络的观点来重新审视软件系统的方法得到了越来越多研究者的认同。研究人员通过建立软件网络模型，研究软件网络的拓扑特征，揭示了软件结构的一些普遍特性，为探索大规模软件系统的结构特征提供了有力的支持，也为软件复杂性控制和软件可信性度量研究提供了有利的指导和帮助。

20 世纪 90 年代初，随着基于 Internet 的分布式大规模软件系统的进一步应用，软件人员将软件功能变化影响分析提到了比较关注的位置[4]。Bohner[5,6]提出了一个软件变化分析的过程框架，并首次引用“波及效应”一词来形象的描述软件变化的影响。以 Ahmed Breech[7-9]为代表的一些研究人员在面向对象软件系统中从软件结构实体(如函数或变量等)变化的角度研究实体变化对其它与其相关的实体的影响。Chen[10]等人研究设计文档、软件组件、外部数据和需求变化之间的关系，建立了一个基于对象和面向属性的软件变化影响模型。随着软件复用技术的兴起及 UML 建模技术的广泛应用，王映辉[11,12]在 Bohner 的基础上，利用可达矩阵实现了对基于构件的软件体系结构演化与波及效应分析。

软件复杂功能的实现在于软件中各功能模块的关联与调用，而模块间的组织结构则直接影响到软件系统的总体性能。在软件开发与维护中，关注系统内部结构的合理性将降低软件出错的风险，减少系统维护的费用。本文主要针对面向对象软件网络中节点变化所产生的波及效应进行分析，确定节点的波及范围和规律，并对系统中可能存在缺陷的结构进行风险预测，为软件的设计和维护提供指导。

2. 软件网络中的波及效应

2.1. 软件网络模型的建立

本文的主要研究对象是面向对象的软件系统，其网络模型的建立过程是：首先将源代码抽象为系统的类图，然后将类作为网络中的节点，类之间的交互关系就是节点间的边，由此抽象出软件网络的拓扑模型。由于类之间的关系包括继承、关联、聚合、依赖等，不同关系下的类之间联系的紧密程度也不同，为了能够更准确地描述软件系统结构，对软件网络中的边赋予不同的权重，即采用加权网络来表示软件系统的内部结构。软件网络模型定义如下：

$$N=(V,A,T,W) \tag{1}$$

其中，$V(\text{Vertices})=\{v_i \mid i=1,2,\cdots,n\}$，表示网络中 n 个节点的集合，每个节点对应于软件系统中的一个类；$T(\text{Type})=\{t \mid t\in\{G,U,D\}\}$，此处的 G(Generalization)代表泛化关系，U(Usage)代表使用关系，D(Dependence)代表依赖关系；$A(\text{Arcs})=\{(v_i,v_j,t) \mid v_i\in V,v_j\in V,t\in T,i\neq j\}$ 代表网络中边的集合，每条边对应于软件系统中两个类之间的关系，由于在实际的软件系统中，两个类之间的关系可能不是唯一的，例如类 A 与类 B 之间既存在依赖关系，又存在使用关系，因此每条边的类型 t 是一个复合值；$W(\text{Weight})=\{w \mid w=\max(w(t)),t\in T\}$ 代表每条边的权重。

权值的大小是根据两个节点之间相互联系的紧密程度，也就是相互作用的两个节点之间关系的强弱定的。在软件的加权网络中，边的权值越大，表示相互作用的两个节点联系越紧密，这样的性质符合加权网络中相似权的性质。在研究复杂网络节点之间联系紧密程度时，应使用相似权，并满足以下条件：相似权在[0,1]之间；权重为 0，则无边连接；权重为 1，回到无权网络的定义。在 UML 类图中，类之间的关系共有 9 中不同的描述，但是其中的普通关联、限定关联、关联类、聚合和组成等关系在 UML 结构上没有明显的区别，只能通过语义加以区分，因此在本文的研究中将上述几种关系统一归纳为使用关系，同时将继承、绑定与实现关系统一归纳为泛化关系。根据 UML 类图中类间关系的紧密程度，将依赖关系的权重赋值为 0.1，使用关系的权重赋值为 0.4，继承关系的权重赋值为 0.7。当两个节点之间有两种或两种以上关系存在时，其权值取最大值。

2.2. 波及效应对软件的影响

在软件网络中，波及效应的表现是当某一节点出现故障或发生改变时，这种变化会随着与其他节点的关联而迅速蔓延，进而影响到系统的其他部分。僵化性和脆弱性是腐化软件中两种常见的特征[13]，它们主要是由于节点的波及范围过大或受波及影响的范围

过大而造成的。其中僵化性是指难以对软件进行改动，即使是单一简单的改动也会在有依赖关系的模块中产生连锁反应。这种作用范围越大，系统就越僵化。僵化性在软件网络中的具体表现为节点的波及效应大，当其改动或发生故障时会有大量与其相关的节点不得不做出相应的改动，因此僵化设计的系统必然会导致其维护成本大大提高。

脆弱性是指在对软件中的一个模块进行改动时，程序中许多在概念上与该模块没有关联的地方都会出现问题，而要修正这些问题，又会引出更多的问题。脆弱性在软件网络中主要表现在那些受波及范围比较大的节点上，这些节点在结构上与大量节点具有相关性，因此任何一个相关节点的改变都会对它造成影响。由此可见，研究软件网络中的波及效应问题可以对那些可能造成软件僵化或脆弱的内部结构进行预测，在软件结构设计中对其进行优化，并为软件的维护与重构提供指导。

2.3. 波及度定义

从波及效应的定义来看，软件网络中的节点既可以成为这种波及效应的制造者，也可以是受波及效应影响的被波及对象，因此，在对波及效应进行研究的过程中，将波及效应在节点上的体现分为两种：正向波及度和逆向波及度。其定义如下。

定义 1：正向波及度：从软件网络中的节点 vi 出发，沿着正向边可以到达的所有节点的最短路径上权值之积的和。记为：

$$RF_{WG}(v_i)=\sum_{j\neq i}^{v_j\in TF(v_i)} W_{ij},\ W_{ij}=\prod_{i\neq j}^{w_{mn}\in SP_{F(ij)}} w_{mn} \qquad (2)$$

其中，$TF(v_i)$为节点 v_i 通过正向边可以到达的所有节点集合。W_{ij} 为 v_i 与 $TF(v_i)$中某一节点 v_j 的总权值，由节点 v_i 与 v_j 之间正向最短路径中所有边权值的乘积计算得来。当 v_i 的出度为 0 时，$RF_{WG}(v_i)=0$。

节点的正向波及度反映了节点在软件网络中受其它节点影响的程度，即当与其相关的节点发生改变或故障时对该节点造成影响的可能性。通常正向波及度较大的节点容易导致软件系统出现脆弱性，是在软件结构设计中需要被高度关注的部分。

定义 2：逆向波及度：从软件网络中的节点 v_i 出发，沿着逆向边可以到达的所有节点的最短路径上权值之积的和。记为：

$$RR_{WG}(v_i)=\sum_{j\neq i}^{v_j\in TR(v_i)} W_{ij},\ W_{ij}=\prod_{i\neq j}^{w_{mn}\in SP_{R(ij)}} w_{mn} \qquad (3)$$

其中，$TR(v_i)$为节点 v_i 通过逆向边可以到达的所有节点集合。W_{ij} 为 v_i 与 $TR(v_i)$中某一节点 v_j 的总权值，由节点 v_i 与 v_j 之间逆向最短路径中所有边权值的乘积计算得来。当 v_i 的入度为 0 时，$RR_{WG}(v_i)=0$。

逆向波及度反映了该节点发生改变或故障时，会对其它与其相关联的节点造成的影响程度，最直接的反映了节点的波及效应。一般来说，逆向波及度大的节点可能会导致软件系统出现僵化性，因此在软件结构的设计和维护中需要引起关注。

3. 波及度的规律分析

根据波及度的定义，对选取的 125 个开源软件样本进行波及度分析。这些软件样本均为面向对象软件，功能涵盖了开发工具、应用软件、系统软件、娱乐软件及编译软件等，开发语言包括 C++、java 和 C#。

首先根据正、逆向波及度的定义对软件网络中节点的波及度进行计算，并以 0.5 为区间间隔，对节点波及度的区间分布频率进行统计，结果发现，随着区间值的增大，节点正、逆向波及度的出现频率都呈现出首先迅速下降，然后趋于平稳，最终趋向于 0 的变化趋势。

由波及度的定义可知，正向波及度大的节点大多位于系统的较高层，在构造上依赖了大量的其它节点，是系统中的脆弱环节，一旦其相关的节点出现问题，都可能会受到影响。因此这样的节点数量在整个系统中不能过多，以免导致系统脆弱性的出现。相应地，逆向波及度大的节点大多是构成系统的基础类或接口，位于系统的底层，被大量的节点直接或间接的关联，这样的节点一旦发生改动，将会导致大范围的节点不得不做出相应的调整。如果这样的节点数量过多，必然会导致系统的僵化。

从上面的分析可以看出，节点波及度的分布规律很好地反映了软件的设计原则。对于单个节点来说，若其正向波及度较大，说明其复用了大量的底层节点，功能相对强大，内部结构相对复杂，容易受到波及，改动或出现故障的概率较大，为了避免造成更大范围的影响，应该尽量避免对这样的节点进行复用或

关联，所以其逆向波及度不应过大。同理，逆向波及度较大的节点由于其在系统中的重要地位，保持其稳定的结构至关重要，因此应该尽量位于其它节点的波及范围之外，即正向波及度要小。由此可以看出，正、逆向波及度应该具有一定的负相关性。

为了验证这一点，对各样本软件中节点的正、逆向波及度分布联合统计，结果发现，正向波及度大的节点其逆向波及度都不大，反之亦然。以 firefox 软件为例，如图 1 所示。

由图可以看出，大部分节点的正、逆向波及度都比较小，他们位于图中的左下角；少数正向波及度大的节点其逆向波及度较小；而逆向波及度大的节点其正向波及度较小，这些节点位于坐标轴附近。当然，在一些软件中，还会出现另外一类节点，它们的正、逆向波及度都比较大，比如软件 Azureus 中的个别节点，如图 2 所示。这类节点是软件设计中应该尽量避免的。

4. 波及度的显著性度量及结构风险预测

通过对节点波及度分布的分析可以发现，软件中大部分节点的正、逆向波及度都比较小，只有少部分节点具有较大的正向波及度或逆向波及度，而同时具有较大的正、逆向波及度的节点非常少，甚至为 0。从软件设计的角度出发，具有单向波及度较大值的节点是不可避免的。由于正向波及度大的节点受波及的可能性大，属于“脆弱节点”，所以在故障维护时应首先考虑对这些节点的排查；逆向波及度大的节点往往会引起广泛的波及效应，属于“僵化节点”，所以在软件维护或升级中应该尽量减少对它们的改动。而同时具有较大的正、逆向波及度的节点是软件结构中的“隐患节点”，它们对软甲你的稳定性造成了较大的威胁，在设计的过程中应该尽量避免。

由于波及效应的存在，软件系统的稳定性受到了极大的挑战，大规模软件的维护成本也不断提高。为了能够提高软件的可维护性，降低维护成本，在软件的设计阶段就应该关注其结构的合理性，尽量降低软件中节点波及效应的影响，对结构中存在的风险因素进行预判。

在对样本软件的节点波及度分析中发现，每款软件的正、逆向波及度的最大值相差很大，随后我们对软件规模和正、逆向波及度平均值，正、逆向波及度

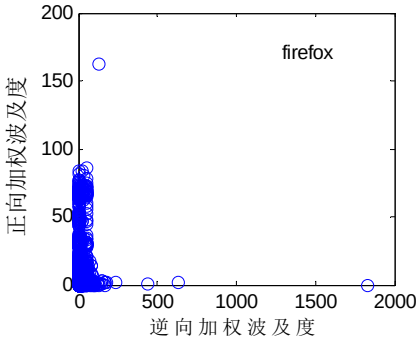

Figure 1. Joint distribution of forward and reversal ripple degree in firefox
图 1. Firefox 软件网络中节点的正逆向波及度联合分布图

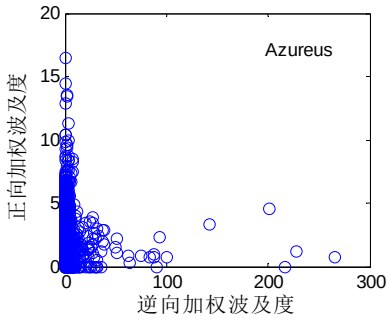

Figure 2. Joint distribution of forward and reversal ripple degree in Azureus
图 2. Azureus 软件网络中节点的正逆向波及度联合分布图

最大值分别进行拟合，结果发现软件规模与软件的平均波及度水平和波及度范围没有明显的相关性。为了能够找出软件结构中的“脆弱节点”、“僵化节点”和“隐患节点”，本文采用显著性度量方式，对上述三种节点进行判断，其结果将为软件结构设计提供参考。

4.1. 正向波及度显著性度量

不同的软件，其正向波及度数值范围相差很大，为了找出每个软件中正向波及度较大的节点，提出了正向局部波及度显著性度量公式，用来对节点波及度水平进行度量。

$$Z_{RF^W(v)} = \frac{RF_{WG}(v) - RF_{WG}avg}{RF_{SD}^{W}} \quad (4)$$

其中，$RF_{LG}(v)$是节点的正向波及度，$RF_{LG}avg$ 是整个软件系统的平均正向波及度，RF_{SD} 是正向局部波及度的标准偏差。从软件样本中随机选取六款软件按照上述公式进行波及度显著性度量，其统计结果如图 3 所示。

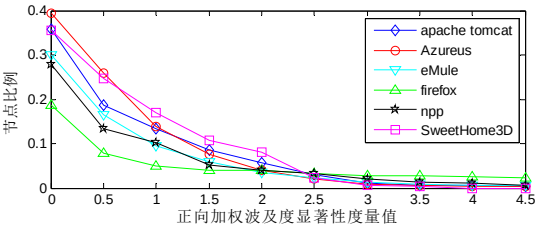

Figure 3. Statistics of forward ripple degree's significant value
图 3. 正向波及度显著性度量统计图

图中横坐标是波及度显著性度量值，纵坐标是大于对应 $Z_{RF}(v)$的节点数所占总节点数的比例。从图中可以看出，软件的正向波及度显著性度量值分布曲线表现出了相似的变化趋势，即随着度量值的增大，对应的节点比例迅速下降，当度量值大于 2.5 之后，对应的节点比例变化趋于平稳，维持在 3%以下。随后，对软件样本中的其余样本进行了相同的度量，得到了相同的统计规律。

通过查阅相关软件的源代码和其公开的开发文档发现，这些度量值超过 2.5 的节点在构造上依赖或关联了大量的其他类，功能大多比较复杂，被修改的次数相对较多，属于“脆弱节点”。这类节点在后期的软件维护中必须引起相关人员的特殊关注。

4.2. 逆向波及度显著性度量

与正向波及度类似，不同软件的节点逆向波及度范围不尽相同，因此提出了逆向波及度显著性度量公式对节点逆向波及度水平进行度量。

$$Z_{RR^W(v)} = \frac{RR_{WG}(v) - RR_{WG}avg}{RR_{SD}^W} \tag{5}$$

其中，$RR_{WG}(v)$是逆向波及度，$RR_{WG}(avg)$是系统的平均逆向波及度，RR_{SD}^W 是逆向波及度的标准偏差。

利用逆向波及度显著性度量公式对软件样本中的 125 款软件进行了度量，统计分析之后发现所有的软件逆向波及度显著性度量值都表现出了相似的变化规律。如图 4 所示，以随机选取的六款软件的统计结果为例，随着度量值的增大，对应的节点比例首先迅速下降，后趋于平稳。其中，度量值大于 0 的节点比例都在 10%以上，当度量值增大到 2.5 的时候，对应的大于该值的节点比例下降到 3%以下。通过查看源代码，发现这些度量值大于 2.5 的节点都是软件系统中的基础类或接口，被大量的节点继承、依赖或引

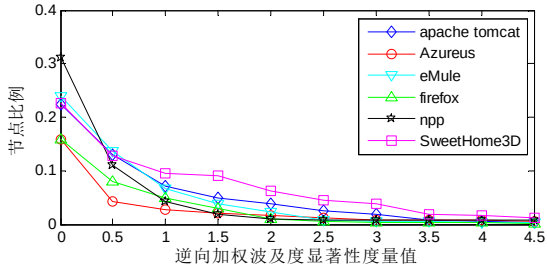

Figure 4. Statistics of reversal ripple degree's significant value
图 4. 逆向波及度显著性度量统计图

用，属于“僵化节点”，因此，应该尽量保证它们的稳定性，在后期的维护过程中尽量少做改动。

4.3. 联合波及度显著性度量

软件网络中正、逆向波及度都比较大的节点是系统结构中的关键节点，由于它们的构造基于大量的底层节点，同时又被大量的高层节点所关联，因此往往是结构中的桥梁，也是隐患最大的环节。为了对这部分节点的波及效应进行分析，提出了节点联合波及度的计算公式：

$$R_{WG}(v) = RR_{WG}(v) * (1 + RF_{WG}(v)) \tag{6}$$

联合波及度反映了软件网络中由于节点直接或间接发生改变所带来的波及范围，其中，$RR_{WG}(v)$为节点的逆向波及度，$RF_{WG}(v)$为节点的正向波及度。1 + $RF_{WG}(v)$代表了节点 v 发生改变的波及效应(包括其自身)。节点联合波及度的显著性由以下公式计算：

$$Z_{R_{WG}(v)} = \frac{R_{WG}(v) - R_{WG}avg}{R_{SD}^W} \tag{7}$$

其中，$R_{WG}(v)$为节点的联合波及度，$R_{WG}avg$ 为该软件的平均联合波及度，R_{SD}^W 为软件联合波及度的标准偏差。

对样本软件中的 125 款软件的节点联合波及度的显著性进行度量并统计，结果发现所有软件的联合波及度显著性度量值的分布规律基本一致，仍以六款随机软件为例，其度量值分布如图 5 所示。

从图中可以看出，所有软件的节点联合波及度显著性度量值都大于−0.5，而且对于每款软件，其 80%的节点的度量值都分布在区间[−0.5, 0.5]内。对着度量值的增大，对应的节点比例逐渐减小。

图 6 是图 5 的局部放大图，从中可以看出，当 $Z_{R_{WG}(v)} \geq 2$ 时，节点比例都小于 4%，当 $Z_{R_{WG}(v)} \geq 4$ 时，

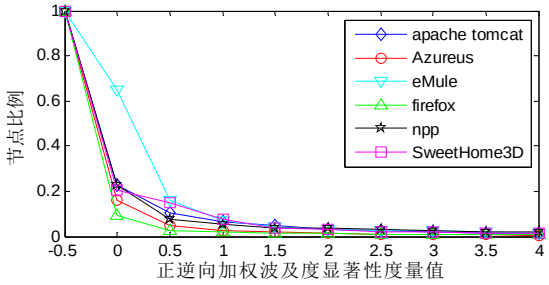

Figure 5. Statistics of joint ripple degree's significant value
图 5. 联合波及度显著性度量统计图

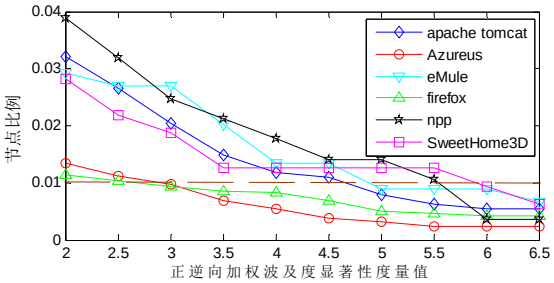

Figure 6. Statistics of joint ripple degree's significant value
图 6. 联合波及度显著性度量统计图

节点比例都小于 2%，当 $Z_{R_{WG}(v)} \geq 6$ 时，节点比例都小于 1%。根据测试报告和 CVS 记录，当节点的联合波及度显著性度量值大于等于 6 时，它们出现问题或被修改的次数远远高于其他节点，所以，选取 6 作为度量值的阈值，当节点的联合波及度显著性度量值大于 6 时，该节点为软件网络中的关键节点，充当了桥梁的作用，其一旦受到攻击，整个网络将会受到极大威胁，甚至引起整个软件系统的瘫痪。

在软件开发过程中，对已经建立的软件结构及时进行波及效应的分析和度量，可以尽早发现结构中的脆弱节点、僵化节点和“桥梁”节点，在保证软件功能可实现的前提下对其进行优化，将结构风险降到最低，并对这些节点加以关注，为软件的后续维护提供参考依据。

5. 结论

软件网络中的波及效应反映了软件内部结构之间的交互影响程度，当软件发生变化时，这种变化会通过模块或类间的关联关系蔓延到更大范围，对软件的多个模块或类造成影响。软件网络中，节点的正向波及度反映了该节点受其它节点影响的可能性，正向波及度越大，受影响的可能性越大，此为脆弱节点；逆向波及度反映了该节点对其它节点造成影响的可能性，逆向波及度越大，对其它节点的影响越大，此为僵化节点；而正、逆向波及度都比较大的节点被称为“桥梁”节点，在结构中起到了关键的连接作用，是结构中最值得关注的敏感部位。通过对波及度的分析，确定基于显著性的度量方法，用于在软件结构中找到上述三种节点。波及度显著性度量的结果可以在软件的开发初期指导软件系统的结构设计，在软件测试和运行的过程中为软件维护提供参考。

参考文献 (References)

[1] Vasa, R., Schneider, J.G., Woodward, C., et al. (2005) Detecting structural changes in object oriented software systems. *Proceedings of the* 2005 *International Symposium on Empirical Software Engineering*, 479-486.

[2] Vasa, R., Schneider, J.-G. and Nierstrasz, O. (2007) The inevitable stability of software change. *Proceedings of the International Conference on Software Maintenance*, Paris, 4-13.

[3] Girolamo, A., Newman, L.I. and Rao, R. (2005) The structure and behavior of class networks in object-oriented software design. www.eecs.umich.edu/~leenewm/documents/classnetworks.pdf

[4] 程平, 刘伟, 陈艳 (2010) 基于矩阵变换的软件可信性演化波及效应. *系统工程理论实践*, **5**, 778-785.

[5] Bohner, S.A. (1996) Impact analysis in the software change process: A year 2000 perspective. *Proceedings of the* 1996 *International Conference on Software Maintenance*, 42-51.

[6] Bohner, S.A. (2002) Software change impacts: An evolving perspective. *Proceedings of the International Conference of software Maintenance*, 263-272.

[7] Hassan, A.E. and Richard, C.H. (2004) Predicting change propagation in software system. *Proceeding of the* 20*th IEEE International Conference on software Maintenance*, **9**, 284-293.

[8] Breech, B., Danalis, A., Shindo, S., et al. (2004) Online impact analysis via dynamic compilation technology. *Proceeding of the* 20*th IEEE International Conference on Software Maintenance*, **9**, 453-457.

[9] Malik, H. and Hassan, A.E. (2008) Supporting software evolution using adaptive change propagation heuristics. *Proceeding of the* 20*th IEEE International Conference on Software Maintenance*, **10**, 177-186.

[10] Chen, C.Y., She, C.W. and Tang, J.D. (2007) An object-based, attribute-oriented approach for software change impact analysis. *Proceeding of the IEEE International Conference on Industrial Engineering and Engineering Management*, **12**, 577-581.

[11] 王映辉, 张世琨, 刘瑜 (2004) 基于可达矩阵的软件体系结构演化波及效应分析. *软件学报*, **8**, 1107-1115.

[12] 王映辉, 王立福, 张世琨 (2006) 一种软件需求变化追踪方法. *电子学报*, **34**, 1428-1432.

[13] Martin, R.C. (2003) 敏捷软件开发 - 原则、模式与实践. 清华大学出版社, 北京, 88-132.