软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2013,24(12):2814−2829 [doi: 10.3724/SP.J.1001.2013.04397]

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

复杂软件系统的软件网络结点影响分析*

汪北阳[1,2], 吕金虎[3]

[1](软件工程国家重点实验室(武汉大学),湖北 武汉 430072)

[2](长江大学 计算机科学学院,湖北 荆州 434023)

[3](中国科学院 数学与系统科学研究院 系统控制重点实验室,北京 100190)

通讯作者: 汪北阳, E-mail: wang_wwang@whu.edu.cn

摘 要: 目前一些研究利用复杂网络理论揭示了软件网络的特性,为人们从系统的角度了解软件的结构提供了方法.但这些研究的一些结论却与软件的实际表现有着较大的差异.分析了软件网络结点的特性,揭示了产生上述差距的部分原因;并提出一种加权软件网络模型,以更准确地描述软件网络结点间的依赖关系;在这个加权软件网络模型的基础上,分析了软件的实际依赖关系及几个统计特性;分析了各统计特性与软件网络结点影响的关系;进一步提出了软件网络关键结点的概念;同时,在分析各种结点影响的基础上,提出了 4 个合理的假设;最后,通过对两款软件的实验来验证这 4 个假设的有效性.

关键词: 加权软件网络;复杂软件系统;软件结构;软件缺陷传播;软件维护

中图法分类号: TP311 **文献标识码**: A

中文引用格式: 汪北阳,吕金虎.复杂软件系统的软件网络结点影响分析.软件学报,2013,24(12):2814−2829. http://www.jos.org.cn/1000-9825/4397.htm

英文引用格式: Wang BY, Lü JH. Software networks nodes impact analysis of complex software systems. Ruan Jian Xue Bao/ Journal of Software, 2013,24(12):2814−2829 (in Chinese). http://www.jos.org.cn/1000-9825/4397.htm

Software Networks Nodes Impact Analysis of Complex Software Systems

WANG Bei-Yang[1,2], LÜ Jin-Hu[3]

[1](State Key Laboratory of Software Engineering (Wuhan University), Wuhan 430072, China)

[2](School of Computer Science, Yangtze University, Jingzhou 434023, China)

[3](The Key Laboratory of Systems and Control, Academy of Mathematics and Systems Science, The Chinese Academy of Sciences, Beijing 100190, China)

Corresponding author: WANG Bei-Yang, E-mail: wang_wwang@whu.edu.cn

Abstract: The complex network theory has been used to reveal some typical features of software networks. It provides a new way for us to understand the software structure from the system view. However, there exist some gaps between the theoretical results and the practical performance of software systems. This paper aims to reveal some essential causes for the above difference by analyzing the characteristics of software network nodes. This paper proposes a novel weighted network model to much more accurately describe the dependencies among the software network nodes. Based on this model, this paper analyzes the actual dependencies of the software network nodes and several statistical characteristics. Also, this paper further analyzes the relationships between these statistical characteristics and the nodes impact. Furthermore, this paper introduces the concept of the key node and four fundamental hypotheses. Finally, this paper verifies the effectiveness of the above four hypotheses by designing the experiments on two software systems. This study provides a guide to research in defects propagation, software reliability and software integration testing.

Key words: weighted software network; complex software system; software structure; software defects propagation; software maintenance

* 基金项目: 国家自然科学基金(61025017, 11072254, 61203148)

收稿时间: 2012-07-05; 修改时间: 2012-11-06; 定稿时间: 2013-03-27

Adams 等人指出,在实际的软件开发过程中,复杂软件系统中占很大比例的潜在性软件缺陷只会导致很少的软件失效,而观察发现的绝大多数的软件失效是由极少部分的潜在性缺陷所引起的.他们的研究中有一条非常重要的结论:清除软件中的大部分缺陷,对软件的可靠性只有微不足道的影响;只有当清除的缺陷是属于那些占极少比例的“大规模(large size)”缺陷,软件的可靠性才会出现显著的增强[1].

Fenton 等人指出,在软件发布前的测试中,少数模块包含有大多数的缺陷,而在软件的运行状态下,发现的绝大多数缺陷包含在极少数的模块中;然而,这个现象既不能用模块的规模也不能用模块的复杂性来解释[1].文献[1]指出,以前所声称的模块规模与模块的缺陷密度有关以及通常意义的复杂性度量可以很好地预测错误倾向性及缺陷倾向性模块的说法是没有证据支持的.同时,文献[1]还肯定了一个结论:在软件发布前的测试中发现的缺陷,要比软件使用 12 个月所发现的缺陷要多很多.

文献[1]是通过对实例软件的统计得出了上述结论,但为什么会出现这样的现象,文献并没有从软件的构成结构上给出合理的解释.根据软件产品质量成本理论,如果能找出那些影响软件运行的缺陷倾向性模块,忽略对软件的实际运行没有或影响的概率很小的缺陷,则测试软件和查找这些缺陷的成本将会极大地减少.这样,在节约软件开发成本的同时又不会在软件质量上体现出明显的降低.从软件开发的商业角度来讲,这项研究具有较好的质量和成本平衡的指导意义.

软件的结构对软件质量的影响非常深远.正如 Torres 在文献[2]中指出:软件的结构是软件容错的基础.事实上,软件研究及开发人员一直追求的“高内聚、低耦合”特性在软件开发实践中只是一个美丽的愿望,因为根本无法避免软件的高耦合[3].因此,合理地设计软件的结构,对软件的可靠性和稳定性以及软件的维护等质量特性都有着极为重要的影响.

在研究软件的结构时,屏蔽软件的实现细节,从系统的角度研究软件的结构,研究者和开发人员可以更清晰地理解软件结构的性质.近年的研究表明,大规模的软件系统属于复杂系统[4−12].利用软件的组成单元和它们之间的依赖关系组成的网络称为软件网络,软件网络是一个具有幂率分布的无标度网络,具有小世界特性[3−11].这样的特性,使软件网络的结点间具有小的平均最短路径以及结点连接的偏好性[12].

对于软件网络的研究,常用的方法是抽取构成软件系统的某种粒度的单元(包、组件、类或模块等)为结点,将单元之间的依赖关系(调用、继承、消息等)作为边构成软件网络.目前,研究认为,软件的缺陷、变更等的传播因为软件结构是一种复杂网络,所以其传播代价比较小[7];因为涟漪效应,缺陷、变更等很容易扩散到其他的结点[13].从理论上分析以及仿真的手段证明来看,上述关于软件缺陷、变更的传播代价的结论是有说服力的.然而,Adams 及 Fenton 等人的文献对实际软件缺陷传播所研究发现的现象,却与上述这些研究的结论有着较大的差异.针对这样的矛盾,我们研究了大量的软件系统的网络结构,发现产生上述矛盾的一些问题所在:目前,在以复杂网络理论为指导的软件变更、缺陷传播的研究中,其构成软件网络的结点大都是构成软件的模块、类、包等;通过这些单元的依赖关系构成软件网络的边来反映结点之间的连通性.在这样的网络拓扑上研究软件结点之间的影响(变更、缺陷的传播、耦合度等),没有考虑软件网络自身的特殊性.由于软件网络的结点有其特殊的结构,在以上述的研究中所采用的软件网络模型与真实的软件网络有着很大的不同.故发现,在以软件的模块、类、包等为软件网络的结点时,夸大了软件网络的传播性.这样的传播性夸大在很多复杂网络上都有表现,比如传染病的传播性、Internet 上的病毒等.虽然社会网络和 Internet 都是典型的复杂网络,但在这些复杂网络上的传染病和病毒并不像复杂网络理论给出的那样会瞬间传播到大部分的结点上.

本文的研究对软件网络作为一种复杂网络,为什么其结点的缺陷不会以很小的代价传递到所有连通的结点中这个现象的部分原因进行解释;分析并证实不同结点对软件缺陷、变更等传播的影响.本文研究所构建的软件网络充分考虑了方法层的依赖关系;结点聚集信息(缺陷、变更等)的强度和结点传播信息的强度等;根据所构建的加权软件网络,建立软件网络各结点的影响测度值矩阵,此矩阵存储了对应的软件网络的加权依赖关系,根据该矩阵,可以获得软件网络的加权网络特性,这些加权网络特性更准确地反映了真实软件网络的各种特性.

李兵等人在研究软件网络的结构时提出,应该更关注软件网络结点的方法层,因为方法层的细节影响了结点传播的概率[14].本文研究与他们研究的区别在于:软件网络结点影响力的因素不仅在于其以什么样的概率传

递结点自身的缺陷,还在于该结点受其他结点的影响大小.同时,本文不仅研究了软件网络中相邻结点的相互影响程度,也研究了结点在整个网络中的影响程度,因为软件的缺陷具有出现错误和产生错误的地点不相邻特性.

在研究软件缺陷传播的过程中我们发现,软件运行时错误发生所在的模块以及缺陷产生所在的模块各有其自身的特征,从而提出了关键结点的概念.关键结点概念的提出,能够较好地解释文献[1]观察到结点中错误的聚集途径和表现规律.研究结果也合理地解释了 Adams 及 Fenton 等人的文献所观察到的软件缺陷传播及测试时和运行时表现不一致的现象.本研究希望通过结合软件网络结点所特有的组成特性,揭示软件网络依赖关系的本质,对复杂软件系统的设计、维护和测试提供帮助.

为了更清晰地介绍本文的研究,第 1 节简要介绍软件网络.第 2 节介绍软件网络的依赖关系,通过例子揭示软件网络在以不同粒度的软件构成单元为结点时,软件网络的不同连通性.第 3 节介绍加权软件网络模型——结点影响加权软件网络((weighted software network for node impact,简称 WSNNI),以及在此模型上建立的结点间影响测度值矩阵.第 4 节介绍在此模型基础上的加权软件网络统计特性,并分析各统计特性所反映的软件网络中结点的影响,在分析这些影响的同时,本文给出 4 个合理的假设.第 5 节通过构建两款软件的 WSNNI,并以软件缺陷的传播规律为实验的验证对象,验证第 4 节给出的假设,在实验的基础上分析和揭示文献[1]中提出的软件缺陷表现的软件结构本质.最后,我们希望通过本研究所揭示的软件网络连通性的本质,对指导软件开发和研究做出一点贡献.

1 软件网络

软件结构用网络形式来描述时,可以在不同粒度上抽取网络,包括包、类、方法、属性等.将上述不同粒度抽象为结点、他们之间的依赖关系抽象为边,由这些结点和边所构成的软件结构就是软件网络.为解释软件网络,下面用图 1[7]来描述软件静态结构的形成.

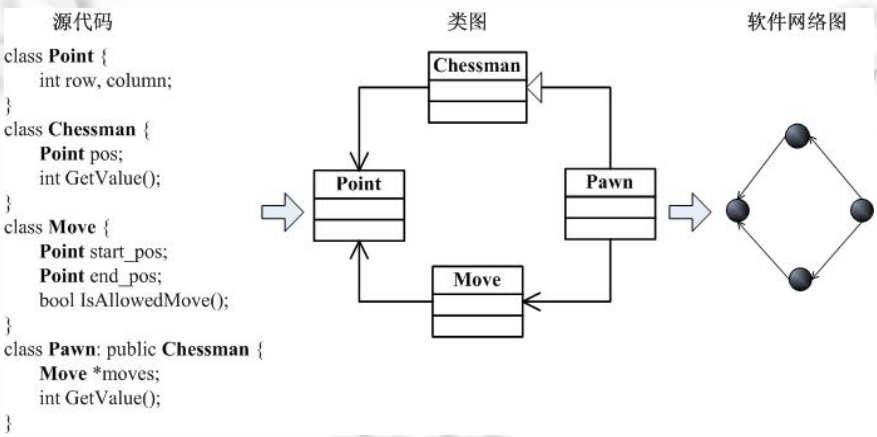


Fig.1 Software network at class level

图 1 类层次的软件网络

软件网络用二元组来定义:

$$\Omega_s=(W_s,E_s),$$

其中,$W_s=\{s_i\}(i=1,2,\ldots N)$是 N 个类的集合,E_s 是类之间边/连接的集合.

2002 年,Valverde 等人[15]首先研究了软件网络,他们将软件工程中的类图作为研究对象,用无向网络表示软件系统,网络中的结点为类,边是类之间的继承(inheritance)和关联(association)关系.他们分析了 JDK(Java development kit)1.2[16]的统计特性,结果表明:这两个软件系统的结构都展现出非常明显的“小世界”和“无尺度”特征.

Myers[6]指出,软件网络中边的方向也有意义,仅用无向网络模型可能会推导出错误的软件工程原则.他建立了软件系统的有向网络模型.为直观地反映软件网络,我们用 DependencyFinder[17]软件抽取了 JEDIT4.3 在类层次的依赖关系,并用 Pajek[18]软件画出了 JEDIT4.3 的网络图,如图 2 所示.

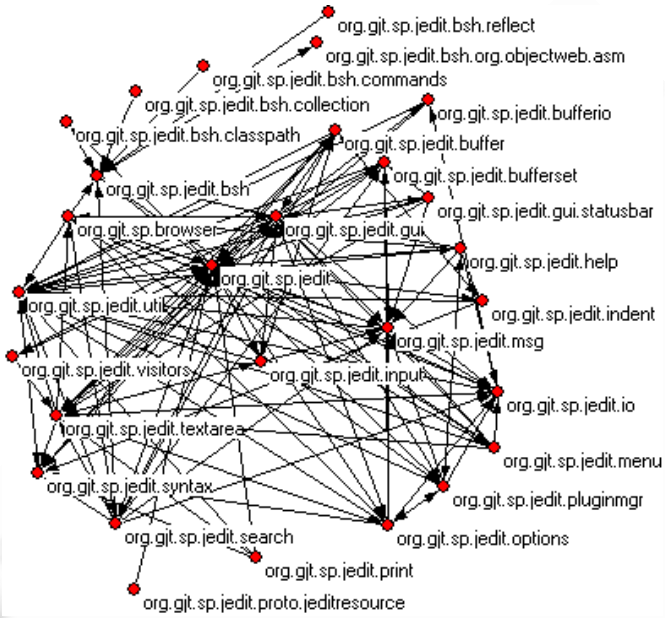


Fig.2 Software network of Jedit 4.3

图 2 Jedit4.3 的软件网络

2 软件系统的依赖关系及其拓扑结构

基于图 1 这样的软件网络,其依赖关系反映了一种全连通图的性质.如前所述,当我们考虑软件的实际特性时,这样的图是否为全联通呢?下面以图 3~图 5 来说明这个问题.

图 3 中的结点 *A*,*B*,*C* 是以类为单元抽取的结点.图中箭头所指表示被依赖的对象.显然,类 *A*,*B*,*C* 之间只是存在着部分依赖.那么,是否 *C* 的变更或缺陷一定会影响 *A* 呢?

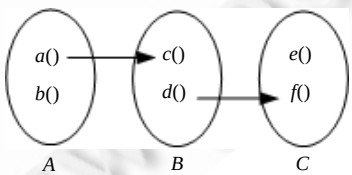


Fig.3 Schematic diagram of software system dependency

图 3 软件系统的依赖结构示意图

图 4 为一般软件网络研究以类为单元所抽取的如图 3 所示的软件结构拓扑图.显然,从图 4 中可以看到的结点 *C* 和结点 *A* 是连通的.但从图 3 所反映的情况看,结点 *C* 的缺陷只能从其方法 *f* 传递到 *B* 的方法 *d* 上,而 *d* 的缺陷不能传递到 *A*.因此,*C* 的缺陷无法传递到 *A*.显然,通过图 4 这样的结构来反映软件网络的依赖是不能真实地反映软件构成单元之间的依赖关系的.那么,在这样的软件网络上研究软件网络结点之间的影响显然也不会准确.从图 3 中还可以看到,如果结点 *C* 的方法 *e*()有缺陷,也不能传递到结点 *B*.那么,*B* 和 *C* 之间的连通也是部分的.为了能准确地反映软件构造单元之间的影响,需要构造一个能够合理反映软件构造单元依赖的软件网络模型.

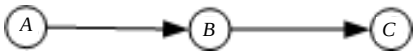


Fig.4 Topological structure of software network at class level

图 4 以类为单元的软件网络拓扑结构

图 5 为从方法层依赖关系所抽取的图 3 所示的软件结构拓扑图.从图 5 中所反映的软件网络真实拓扑来看,当 *C* 结点有错误或有变更出现时,并不直接影响结点 *A*.对于结点之间的部分依赖的关系,比如结点 *B* 只依赖结点 *C* 的方法 *f*(),而不依赖其方法 *e*(),这样的关系在本文中我们用软件网络的权值加以表示.下面介绍本文提出的加权软件网络模型.

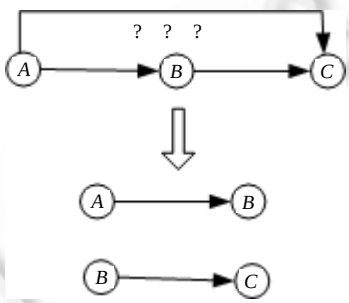


Fig.5 Real topological structure of part-dependency software network

图 5 部分依赖软件网络的真实拓扑结构

3 结点影响加权网络 WSNNI

3.1 结点影响加权网络WSNNI定义

为了便于描述,本文定义加权软件网络为软件结点影响加权网络 WSNNI,见定义 1.

定义 1(软件结点影响加权网络). WSNNI 用来反映软件网络中结点间的相互影响,用一个二元组来定义.二元组中包含一个软件网络的加权有向图,一个存储结点间影响测度值的矩阵.WSNNI 的定义如下:

$$\text{WSNNI}=(G,M_I) \tag{1}$$

其中,$G=(N,W)$,包括 N 个结点以及一组带有权重的边 W.在加权网络中,权重分为相异权和相似权两种.相异权表示权值越大,两点间的距离越远,关系越疏远;而相似权表示权值越大,两点间的距离越小,关系越密切.本文采用的是相似权重,权值为 0 表示结点之间没有依赖存在.由于本文采用有向加权网络,故 w_{ij} 和 w_{ji} 并不相同.M_I 为存储有向图 G 中结点之间的影响测度值的矩阵.边的权重 W 及结点间影响测度的矩阵 M_I 在下面分别详细介绍.

3.2 权值的定义

为准确刻画本文加权网络中权值的定义,以图 6 说明.图 6 中,结点 *A* 中有 3 个方法同时依赖于 *B* 中的方法 *c*(),所以结点 *A* 只会受到方法 *c*()的影响,而结点 *B* 依赖 *C* 中的两个方法.我们可以观察得出,图 3 中 W_{AB} 和图 6 中所示的 W_{AB} 都只依赖与 *B* 中的一个方法,但显然,图 6 中结点 *A* 中 3 个方法都依赖于 *B* 中的方法 *c*(),而图 3 中所示结点 *A* 只有一个方法依赖与 *B* 中的方法 *c*().图 7 给出了一个例子,图 7(a)和图 7(b)分别在图 3 和图 6 的基础上在结点 *A* 前连入了一个结点 *X*.从图 7 可以看出,图 7(a)的结点 *X* 受结点 *B* 的影响明显要比图 7(b)中 *X* 受结点 *B* 的影响小.故,为了能更全面地反映结点之间的影响,本文对两点间边的权值 W_{ij} 的定义如下:

定义 2(两个结点间边的权值). 设 Nf_i 为结点 i 的方法数之和,Nf_{ij} 为结点 i 依赖于结点 j 的方法数之和,Nf_j 为结点 j 的方法数之和,Nf_{ji} 为结点 j 被结点 i 依赖的方法数之和,则权值 w_{ij} 的计算方法如下:

$$w_{ij}=\frac{Nf_{ij}}{Nf_i}\times\frac{Nf_{ji}}{Nf_j} \tag{2}$$

边权值的定义对我们研究结点间的影响非常重要.对公式(2)的意义,我们用如下实例加以解释:

如图 3 所示的软件网络,其 w_{ab} 的值为 $w_{ab}=\frac{1}{2}\times\frac{1}{2}=0.25$,其 w_{bc} 的值为 $w_{bc}=\frac{1}{2}\times\frac{1}{2}=0.25$;再如图 6 所示的软件网络,其 w_{ab} 的值为 $w_{ab}=1\times\frac{1}{2}=0.5$,其 w_{bc} 的值为 $w_{bc}=\frac{2}{2}\times\frac{2}{2}=1$.从图 3 和图 6 的相邻边权值的计算,我们可以看出相邻结点间的影响程度,显然,如果某条边的权值为 1,则表明两点间的一个全依赖,比如类的全继承等.

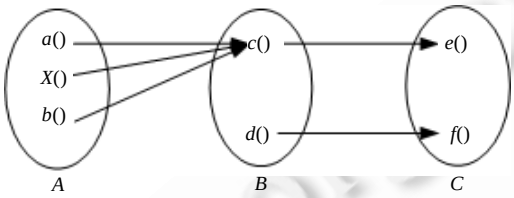


Fig.6 Schematic diagram of the weight

图 6 权的图示

3.3 WSNNI的结点间影响测度值矩阵

权值 W_{ij} 的定义不仅体现了相邻结点受影响的情况,在扩展结点 i 和 j 为不相邻的结点情况下,更体现了其研究不相邻但连通的两个结点之间影响的情况.对比图 7 中结点 X 与结点 B 之间的影响:

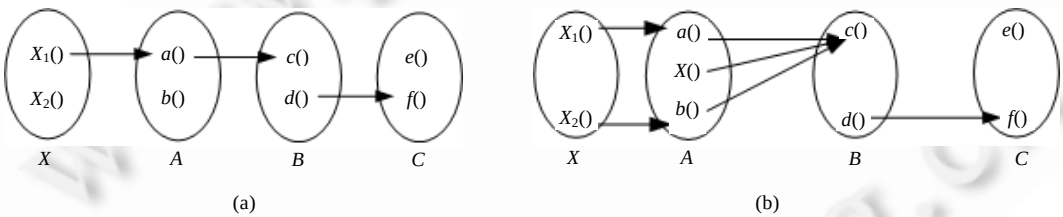


Fig.7 Schematic diagram of non-adjacent connected nodes

图 7 非相邻连通结点影响示意图

从图 7 反映的情况可以看出,结点 B 对结点 X 的影响的概率显然是不一样的.利用上面本文定义的权值,可以很好地测度这种影响.为了说明两个不直接相邻的结点之间的影响,下面给出一条路径上不相邻两点之间的影响测度:设两个不直接相邻(但连通)的结点 i 和 j,其中,结点 i 通过某条路径间接依赖于结点 j,设表示结点 j 对结点 i 的影响测度用符号 I_{ij}.先设从结点 i 沿某条路径到结点 j 的依赖关系的权值集合为 SWR_{ij}(沿 i 到 j 的某路径上边权值的集合),则 I_{ij} 的计算算法如下:

输入:SWR_{ij};

输出:I_{ij}.

1. 初始化 I_{ij} 为 1;w_x(w_x 为指向 SWR_{ij} 元素的权值变量)指向 SWR_{ij} 的第 1 个元素;
2. 如果 w_x 没有指向 SWR_{ij} 的最后一个元素,则 $I_{ij}=I_{ij}\times w_x$;否则,跳到第 4 步;
3. w_x 指向下一个元素;重复第 2 步的执行;
4. 输出 I_{ij};

在计算了各个结点间的影响测度值后,下面给出在 WSNNI 模型下的结点间影响测度值矩阵 M_I 的定义.

定义 3(WSNNI 模型下结点间影响测度值矩阵 M_I). 结点间影响测度值矩阵 M_I 存储有向图 G 中各结点间的影响测度值 I_{ij}.矩阵的行按有向图 G 中结点集合 N 的某个顺序排列,列的顺序和行的顺序一致;行和列交接处为行号所代表的结点受列号所代表的结点的影响测度值.

为直观说明 M_I 的定义,下面分别给出图 7 中的 M_I:

- 图 7(a)所示的 $M_I=\begin{bmatrix}1 & 0.25 & 0.065 & 0.0156\\ 0 & 1 & 0.25 & 0.065\\ 0 & 0 & 1 & 0.25\\ 0 & 0 & 0 & 1\end{bmatrix}$;
- 图 7(b)所示的 $M_I=\begin{bmatrix}1 & 0.677 & 0.333 & 0.0833\\ 0 & 1 & 0.5 & 0.125\\ 0 & 0 & 1 & 0.25\\ 0 & 0 & 0 & 1\end{bmatrix}$.

在 M_I 中,用 I_{ij} 表示结点 i 到 j 的影响测度值,$0\leqslant I_{ij}\leqslant 1$.$I_{ij}$ 为 0,表示一个结点没有依赖于另一个结点;值为 1,表示一个结点完全依赖于另一个结点.其中,结点本身对自己的依赖是完全的,所以在矩阵中,正对角线上的值都为 1.

4 WSNNI 的统计特性及相应的结点影响分析

加权网络也符合复杂网络的特性[19],下面给出在加权软件网络 WSNNI 模型下的几个加权网络的统计特性.根据这些统计特性,分析软件网络中结点的影响.在分析各结点影响的基础上,我们给出了 4 种假设.

4.1 结点强度和结点强度分布

4.1.1 结点强度(S_i)定义及结点影响分析

结点的度定义为网络上与该结点相连的其他结点的数目.在有向网络上,结点的度分为出度和入度.但在加权复杂网络上,简单地用结点的度来表示结点的统计特性显然就不合适了.加权软件网络中,与结点度 k_i 相对应的扩展就是结点强度或结点权 S_i[19].

定义 4(加权软件网络的结点强度). 设 N_i 为结点 j 的邻接点的集合,w_{ij} 为结点 i 到邻接点 j 的权值,则结点强度表示为

$$S_i=\sum_{j\subseteq N_i} w_{ij} \tag{3}$$

在 WSNNI 模型上,对于结点强度的意义分析如下:

WSNNI 是一个加权的有向网络,出边代表了结点对相邻结点的依赖,入边代表了结点被相邻结点的依赖;边的权值代表了结点与相邻结点依赖的程度,故要定义 WSNNI 模型下的结点强度,边的权值是一个非常重要的因素.S_i 的意义表示结点强度既考虑结点的近邻数,又考虑该结点和近邻结点间的权重.但在本文提出的 WSNNI 模型下,公式(3)所表示的结点强度显然是不能准确表达结点在软件网络中的影响意义的,因为结点强度是由结点的边的数量和边的权重共同产生的结果.同样大小的结点强度可能反映两个不同的信息:

1) 边的数量比较多,而边的权重较小.这样的结点将对整个网络中更多的结点产生影响,但对单个邻近结点的影响相对较小;
2) 边的数量较少,而边的权重值较大.这样的结点对其邻近的结点产生了较大的影响,但对整个网络中所影响的结点的数量相对较少.

如果不将这两种情况区别开来,使用结点强度很难清楚地反映结点在软件网络中的影响.为了清楚地反映结点在软件网络中的影响,我们给出一个在 WSNNI 模型下加权软件网络的点强度 S_i 的变形.

结点强度定义是公式(3)所定义的加权网络结点强度的一个变形,目的是能够根据结点强度了解结点在网络中的影响情况.

定义 5(WSNNI 模型下结点强度). 设结点 i 的度为 k_i,i 的所有边的平均权值为 w_{ia},则 WSNNI 模型下结点强度的定义为

$$S_i=k_i\times w_{ia} \tag{4}$$

公式(3)和公式(4)中的 S_i 其值是相等的,推导如下:

因为 $w_{ia}=\frac{\sum_{j\subseteq N_i} w_{ij}}{k_i}$,所以 $S_i=k_i\times w_{ia}=k_i\times\frac{\sum_{j\subseteq N_i} w_{ij}}{k_i}=\sum_{j\subseteq N_i} w_{ij}$.由此可见,公式(4)的结点强度只是公式(3)结点强度的变形.但在公式(4)中,突出了结点强度与结点度(边的数量)及结点的平均权值的关系.

结点强度的计算采用的权值是本文第 3.2 节中定义的权值,该权值的定义反映了结点在网络中的全局影响.在软件测试和维护中,我们有时需要了解结点与相邻结点的影响关系,这与结点依赖的方向有关.其相关的统计特性为结点的入点强度和出点强度.

定义 6(入点强度). 入点强度 S_{ii} 定义为指向结点 i 的边(结点 i 被其他结点依赖的关系)的平均权值 w_{iia} 与结点 i 的入度 k_{ii}(依赖于结点 i 的其他结点的数量,这里等同于指向结点 i 的边的数量)的乘积.即:

$$S_{ii}=k_{ii}\times w_{iia} \tag{5}$$

分析:结点的入点强度越大,对该结点的调用、继承或与之通信的结点更多或对其方法的依赖越多.

假设 1:结点的入点强度越大,在软件网络中该结点对其他结点的影响越大,结点的缺陷、变更等传播到其邻接点的概率越大.

定义 7(出点强度). 与入点强度相对,出点强度 S_{io} 定义为由结点 i 指向其他结点的边(结点 i 依赖于其他结点的关系)的平均权值 w_{ioa} 与结点 i 的出度 k_{io}(结点 i 依赖于其他结点的数量,这里等同于结点 i 指向其他结点的边的数量)的乘积.即:

$$S_{io}=k_{io}\times w_{ioa} \tag{6}$$

分析:结点出点强度越大,则依赖其他结点越多,积累其邻接点缺陷或变更等的概率越大.

假设 2:结点出点强度越大,该结点受其邻接结点的影响越大,引入其邻接结点缺陷和变更等的概率越大.

4.1.2 结点强度分布(P_s)及结点影响分析

结点强度分布 $P(s)$描述复杂网络中一个结点强度是 s 的概率,这个统计特性反映加权有向网络的一个全局特性.结点强度分布定义如下:

定义 8. 设 IS_i 和 OS_i 分别代表结点 i 的入点强度和出点强度,则入点强度和出点强度分布为

$$P_{in}(s)=\sum_{i=1}^{m}\Pr\{IS_i=s\} \tag{7}$$

$$P_{out}(s)=\sum_{i=1}^{m}\Pr\{OS_i=k\} \tag{8}$$

其中,m 表示结点集合中结点的总数.

结点影响分析:

显然,在加权软件网络中,结点强度分布可以反映出软件中的结点相互影响的强度分布,也反映了软件依赖关系的复杂性.在软件网络中,结点强度小的结点偏向于连接结点强度大的结点;有大量的结点其强度较小,也有少量的结点具有很大的强度[9,10].软件的优先连接模块导致了软件结点幂率分布的特性[20].Barabási 和 Albert 在文献[20]中指出,新加入的模块偏向于连接那些最早存在于网络中的模块,这导致大量的结点具有很有限的连接,而最先存在于网络中的结点却具有很高的连接度.

这个统计特性更多地从软件的演化中体现出来,很多的文献给出了验证,本文不再赘述.

4.2 加权路径长度和平均最短加权路径

在 WSNNI 中,为了反映在连通的路径上一个结点受其他结点的影响,我们给出了加权路径长度及其平均最短加权路径长度的定义.

4.2.1 加权路径长度(L)

加权网络没有明确的路径长度的概念,但我们可以用两个结点间的影响测度值来描述结点间传播缺陷、变更等的难易程度(传播代价).为了符合习惯的路径长度的定义,我们用影响测度值的倒数表示结点间的长度.这样,长度越大,结点间的传播代价也越大;反之亦然.

定义 9(WSNNI 模型上的加权路径长度). WSNNI 模型上结点 i 到 j 间的加权路径长度 L_{ij} 为 i 到 j 的影响测度值 I_{ij} 的倒数,即:

$$L_{ij}=\frac{1}{I_{ij}} \tag{9}$$

其中,$1\leqslant L_{ij}\leqslant\infty$.$L_{ij}=1$,表示 i 完全受 j 的影响,其传播代价也最小;$L_{ij}=\infty$,表示 i 与 j 之间没有直接的依赖关系.事实上,当两个结点之间的 I_{ij} 为 0 时,结点间没有任何层次上的连通,故 L_{ij} 的值为∞.

结点影响分析:

在第 3.3 节中,我们给出了 I_{ij} 的计算算法.可知,I_{ij} 是 i 到 j 上所经过的路径边权值的乘积,而且任何一条边上的权值都是小于或等于 1.故从 i 到 j 经过的边越多,则 I_{ij} 的值可能就越小,路径 L_{ij} 就会越大,传播代价就越大.可以看出,很多结点的缺陷传递给其他不相邻的结点的概率很小,这也符合了文献[1]给出的结论.事实上,在很多网络中都存在着这样的现象,比如传染病的传播途径、Internet 上的病毒等,并不像理论上那样很快就遍布到绝大多数的结点上一样.

4.2.2 平均最短加权路径(L_a)

定义 10. WSNNI 上结点的平均最短路径长度 L_a 为所有相连结点最短路径之和除以结点总数,即:

$$L_a=\frac{\sum L_{ij}}{\sum e_{ij}} \tag{10}$$

其中,$\sum L_{ij}$ 表示所有相连结点最短加权路径之和,$\sum e_{ij}$ 表示所有相连结点的总数.

结点影响分析:

在 WSNNI 上,平均最短路径是最短加权路径值的平均值.这里,平均路径值越小,表明整个软件网络中结点的相互影响就越大.平均最短路径反映了软件整体的耦合强度,耦合强度直接影响了软件的维护和变更的难度.一个软件网络,其平均最短路径越小(这里是指 WSNNI 网络模型下),网络结点的耦合强度越强,则结点间的缺陷、变更等传播代价越小;反之亦然.

无权网络在这里可以看成是其边的权值为 1 的加权网络的一种特殊形式.显然,无权网络的大部分边要比加权网络的权值大,无权网络的最短路径要比加权网络的最短路径小得多.故无权网络反映的软件网络的结点之间的影响要比加权软件网络大得多.

假设 3:实际软件的缺陷传播要比无权软件网络反映的少很多,WSNNI 模型更能准确地反映软件网络结点的影响强度.

4.3 关键结点

我们在研究软件网络中结点影响的过程中发现,有少数结点的出点强度和入点强度都较其他结点大很多,与其相邻结点的边的影响测度值都比较大.根据这些特点,本文提出了软件网络的关键结点.

4.3.1 关键结点定义

定义 11(关键结点). 关键结点 $N_{ki}=\{n_i|n_i$ 为这样的结点集合,其入点强度和出点强度都远大于网络中结点的平均入点强度和平均出点强度$\}$.

分析发现,这些结点具有很强的缺陷或变更聚集能力,同时也具备很强的缺陷或变更的传播能力.

4.3.2 关键结点对软件缺陷传播的影响分析

在软件工程实践中,当软件功能比较复杂或规模比较大时,一般会将这些复杂或大规模的模块不断地分解为较小的模块,直到模块足够简单.这种分而治之的策略,可以比较好地解决复杂问题.然而在软件集成的过程中,这些小的模块又会通过不同的方式连接,从而形成一个完整的功能模块.这样的工作使软件网络结点更多、路径更长,核心功能模块的度变得更大;随着软件的演化,软件中的模块倾向于产生更多的外部调用而不是内部调用[21];在上文中我们提到过,软件网络是一种复杂网络,新结点的加入更偏向于连接度更大的结点.所以,软件演化越复杂,软件网络中度大的结点,尤其是入度大的结点,将被更多的模块调用,入度会越大,关键结点在整个

软件网络中的影响也会越大.

事实上,根据文献[1]提出的研究结果,复杂的软件系统很多在测试中发现的缺陷,在软件运行很长一段时间后也没有表现出来;测试前发现的许多缺陷,在测试中也并没有在其他结点中表现出来.根据我们对软件网络实际的依赖关系分析,关键结点更广泛地与完成功能的其他结点关联,其他结点的缺陷或变更等,按照关联边的权值大小,以相应的概率传递到关键结点.这些关键结点直接与运行结点邻接,所以,关键结点的缺陷能以更大的概率传播到运行结点,从而表现出软件错误;但其他没有与运行结点直接连接的结点,其缺陷传播的加权路径长度要大很多,传播到运行结点的概率也就要小很多,因此,这些变更或缺陷在运行结点中表现出来的概率就小了很多.

假设 4:在软件测试时发现的绝大多数测试前缺陷都包含在关键结点中.

通过对 WSNNI 的定义及在其上的统计特性的定义和分析,本文给出了 4 个合理的假设.为了验证这些假设的正确性,接下来的第 5 节,我们将通过对两款软件的软件网络进行分析,并通过植入缺陷、变更的方法实证本文提出的 WSNNI 模型在分析软件网络结点影响时的有效性.

5 实验和数据分析

5.1 数据来源

本文选择开放源代码软件 Junit 和 Jedit 作为实验对象,利用工具抽取 Junit 和 Jedit 在包、类和方法粒度上的依赖关系,建立软件的 WSNNI 和结点的影响测度矩阵 M_I.并计算结点的入点强度、出点强度及这两款软件的 WSNNI 的结点平均最短路径;根据两款软件的各种计算结果来验证上述的 4 个假设的有效性.

本文选用这两款开源软件作为实验对象基于如下两个考虑:

1) Junit 和 Jedit 的设计都已很成熟,有着优秀的结构设计,便于缺陷的植入和测试;
2) Junit 和 Jedit 在规模上有着明显的区别(见表 1),便于在不同规模软件上验证本文提出的假设的有效性是否一致.

Table 1 Component units statistics information of Junit and Jedit

表 1 Junit 和 Jedit 的组成单元统计信息

单元 统计数量	包		类				方法(特征)			
	Junit	Jedit	Junit		Jedit		Junit		Jedit	
单元总数(结点)	30	29	248		1 132		1 562		12 105	
单元依赖总数(边)	0	0	来自类	指向类	来自类	指向类	来自方法	指向方法	来自方法	指向方法
			393	3 809	1 655	27 437	6 957	3 541	6 957	43 381

Junit 和 Jedit 的包、类、方法的统计信息见表 1.本文使用软件依赖统计工具 DependencyFinder[17]获得该表数据.在统计这些数据时,我们排除了 Java 自带的相关代码,以便更准确地反映所实验的软件自身的特性.表 1 中,对类和方法的边进行统计时,为了区分它们的方向(软件结点的依赖关系是有向的)我们用“来自”表示由其他的结点(类或方法)指向该结点,代表该结点被其他结点依赖;反之,用“指向”表示边由该结点指向其他结点,代表该结点依赖其他结点.

5.2 实验设置

5.2.1 实验验证项目

为了验证本文提出的 4 个假设的有效性,我们选用两款开源软件 Junit 和 Jedit,建立它们的 WSNNI,分别在 Junit 和 Jedit 中植入 326 和 1 481 个缺陷:

1) 计算它们的 M_I;
2) 计算各结点入点强度,验证假设 1;
3) 计算各结点的出点强度,验证假设 2;

4) 计算每个结点的最短平均加权路径,比较不同结点缺陷的传入数量占相连结点缺陷总数的比值,验证假设 3;

5) 比较关键结点与其他结点的影响测度值,计算关键结点和其他结点中发现的植入缺陷和变更的比率,验证假设 4.

5.2.2 实验验证方法

本文采用了对选用的 Junit 4.9 和 Jedit 4.3 源代码进行缺陷植入,并对缺陷在软件中的传播进行测试和统计.旨在揭示软件网络特性与软件组成单元的真实依赖关系,因此,实质是揭示软件结构的一些特性.故,实验所植入的缺陷为软件本身在设计阶段和编码阶段可能产生的缺陷,不考虑在需求分析和分析建模阶段产生的系统描述与用户需求不一致的功能性缺陷.在设计缺陷时,考虑到实验验证的项目是验证软件组成单元的实际依赖关系及软件网络结点通过这些依赖关系对软件中其他结点的影响,故植入缺陷的类型根据结点依赖的类型来确定.下面是本实验植入的缺陷类型及缺陷植入方法的说明:

- 植入缺陷类型及其传播的方式

(1) 数据缺陷.包括:

1) 被共享的数据定义错误、数据越界产生的错误.这些错误的传播主要体现在:被共享数据被不同的对象使用后产生了错误的结果,因而传播给其他的对象;

2) 因对公共数据的存取产生错误传播;

3) 通过参数调用其他结点的错误数据产生的错误传播;

4) 消息传递的错误数据导致的错误传播;

(2) 接口缺陷.包括因接口设计时产生的接口内的常量定义类型和实现接口时类参数类型不匹配等产生的错误传播;

(3) 类继承缺陷.包括:

1) 父类设计时产生的错误被子类继承而产生的错误传播;

2) 子类在继承父类时产生的错误传播给不同的对象或下一级子类时产生的错误传播.

- 缺陷植入方法

植入缺陷一般是在软件测试时,为了估算软件中缺陷的数量而采用的方法.这里,我们并不是为了估算软件中缺陷的数量而植入缺陷,而是观察缺陷的传播情况.为更真实地反映软件缺陷分布,我们在植入缺陷时按结点的代码行分配缺陷的数量.为了更真实地反映缺陷分布的情况,我们在植入缺陷时对选定的结点植入相应类型的缺陷时,在其内部是随机植入.

- 缺陷统计方法

在我们的实验中,统计每个模块的不同缺陷数是验证缺陷传播的必要工作.我们采用的预测方法是 Capture-Recapture 方法中的 M0 模型.方法如下:

(1) 对选定软件的每个模块植入一定数量的缺陷,采用 Chapman 估计器,计算出发现的缺陷占植入缺陷的比值,设为 r_i;

(2) 清除第(1)步植入的缺陷,测试本文提出的实验用缺陷,针对每个模块测试,统计每类缺陷数量,设为 f_i;

(3) 利用第(2)步得到的缺陷数量乘以第(1)步计算得到的缺陷发现率,估算每类被传播的缺陷在每个模块中的缺陷(不包括直接植入在该模块中的缺陷)设为 q_i,即 $q_i=r_i\times f_i$.

本文选用 Capture-Recapture 方法中的 M0 模型的理由如下:

Capture-Recapture 模型中有 4 类捕捉模型[22],分别是 M0,Mh,Mt 及 Mth.这些模型的区别在文献[22]中进行了详细的叙述.我们选择 M0 模型是基于如下考虑:

1) 我们的测试是在实验环境下,测试用例共同完成,对每个模块缺陷共同评审,因此满足 M0 模型的“评审员具有相同发现缺陷的能力”的条件;

2) 我们测试的缺陷是在实验环境下植入的,测试用例也是针对植入的缺陷,所以也基本满足“缺陷被发

现的概率相同”的条件;

3) 本文提出的实验目的是评估软件网络传播缺陷的能力,在相同实验条件下反映一个缺陷传播的相对关系,对缺陷发现的精确性要求相对较低.

综上所述,选用 M0 模型可以满足我们的实验要求.下面针对前文提出的 4 个假设分别给予验证和分析.

5.3 结果与分析

5.3.1 假设 1 的验证

这里,我们通过验证缺陷的传播来反映结点的影响强度.在给 Junit 和 Jedit 两款软件植入缺陷后,针对缺陷测试,统计缺陷传播到其他结点中的数量.图 8 为不同入点强度的结点将缺陷传播到其他结点的数量的统计散列图.

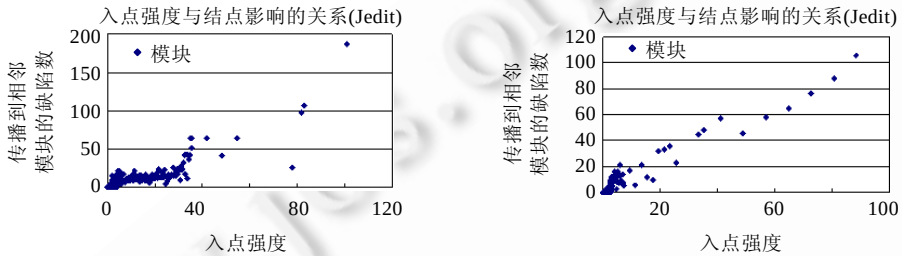


Fig.8 Relationship between S_{ii} and node impact

图 8 入点强度 S_{ii} 与结点影响关系

从实验的结果可以看出这样两个信息:

1) 入点强度大的结点很少,大量的结点入点强度都很小;

2) 入点强度大的结点其缺陷传播到相邻结点的数量明显要比入点强度小的结点要多很多.

从图 8 可以看出,结点的入点强度(S_{ii})明显地反映了软件网络中结点的入点强度大的结点传递了更多的缺陷给其邻结点.从 S_{ii} 的定义可知,结点的入点强度(S_{ii})等于结点入边平均权值与入边数量的乘积.在实际的分析中可以看到,结点强度很大的结点,其入边的平均权值和入边的数量都较大.因此,正如我们在实验中发现的那样,这些入点强度大的结点将自身的缺陷传递到了更多的结点中,这反映了入点强度大的结点被更多的结点所依赖的事实.同时也发现,入点强度大的结点,其大部分的单个相邻结点被传入了较多的缺陷,这反映出了入点强度大的结点,其每条入边的权值也较大的事实.在统计每个结点的缺陷在其他结点中出现的数量时,我们发现了类似的现象:两款软件中,入点强度很大的极少数结点的缺陷分布到了更多的结点中,并且有些被传入缺陷的结点(这些结点的边的权值较大)中被传入了更多的缺陷.这些依赖关系集中表现在类的继承上.同样,入点强度小的结点,它们传播给其相邻结点的缺陷要少很多.

由实验结果我们可以验证假设 1 是合理的.

5.3.2 假设 2 的验证

图 9 是结点的出点强度与结点受其邻结点缺陷影响的数量的实验统计散列图.

从公式(6)中可知,结点强度 S_{io} 定义为由结点 i 指向其他结点的边(结点 i 依赖于其他结点的关系)的平均权值 w_{ioa} 与结点 i 的出度 k_{io}(结点 i 依赖于其他结点的数量,这里等同于结点 i 指向其他结点的边的数量)的乘积.显然,结点的出点强度反映了结点依赖于其他结点的强度.出点强度由结点依赖于其他结点的数量(即出边的数量)和每条出边的权值来决定.

通过实验发现:

1) 软件网络中结点的出点强度越大,结点积累其他结点缺陷的数量越多;

2) 软件网络中大量结点的出点强度都很小,极少数的结点出点强度很大.

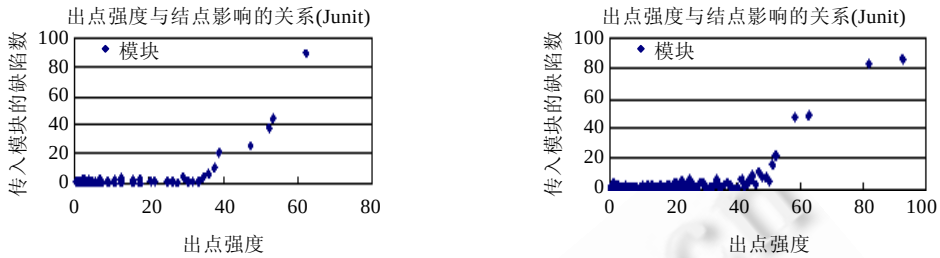


Fig.9 Relationship between S_{io} and node impact

图 9 出点强度 S_{io} 与结点影响关系

出点强度大的结点依赖了更多的其他结点,同时也汇聚了更多的来自于其他结点的缺陷.分析发现,这些出点强度大的结点其出边的权值往往也比较大.因此,这些结点更易于积累其他结点的缺陷.从图 9 可以看出这个趋势.同时,大量结点很少积累来自其他结点的缺陷,这些结点的出点强度较小;其相邻的结点数比较少,而且绝大多数结点的出边权值也比较小.这些依赖更多趋向于消息传递的依赖和一些由参数传递的数据的缺陷.事实上,这也是一些低耦合的设计表现.

显然,实验的结果可以证明假设 2 是有效的.

5.3.3 假设 3 的验证

无权网络将软件网络的各条边的影响认为是一样的.而 WSNNI 模型对每条边进行了加权,其权值为 0 到 1 之间(包括 0 和 1),无权网络可以看成是加权网络的一种特殊情况,即其边的权值为 1.以无权网络分析结点的影响强度,认为相邻结点的缺陷或变更等都会传播到本结点.通过分析我们认为,利用无权软件网络分析软件模块的影响强度(如缺陷、变更等的传播),夸大了各结点的影响强度.

从公式(9)可知,加权路径长是结点影响测度值的倒数.影响测度值越大,路径越短,则结点与其他结点的连通性越强;反之亦然.公式(10)定义了平均最短路径 $L_a=\frac{\sum L_{ij}}{\sum e_{ij}}$,平均最短路径是结点在整个连通的最大子图中与所有相连的结点的连通性的指标.因为平均最短路径代表了结点在其软件网络的最大子图中的连通性,因此,在我们的实验中利用它对比验证软件实际的缺陷传播性与结点最短路径之间的关系.

图 10 是实验统计的结点平均最短路径与结点实际被其相连通的结点传入的缺陷占其相连通结点缺陷总数的百分比,从图 10 中可以看出,软件的实际缺陷并不是如无权网络反映的那样完全传播给了相连的其他结点,而是与 WSNNI 模型下结点的平均最短路径关系密切.平均最短路径越长,其被其他结点传入的缺陷占所有缺陷的比例越低;反之亦然.显然,实验反映的结果并不是如无权网络模型所描述的那样,结点的缺陷会以较小的代价传递给其他结点,大部分结点传播其缺陷的概率较低,只有一些平均最短路径很小的结点传播其缺陷的概率比较高.因此,证明了无权网络模型夸大了软件传播缺陷的能力,而 WSNNI 模型能够更准确地反映软件结点传播缺陷的能力.

显然,实验证明假设 3 是有效的.

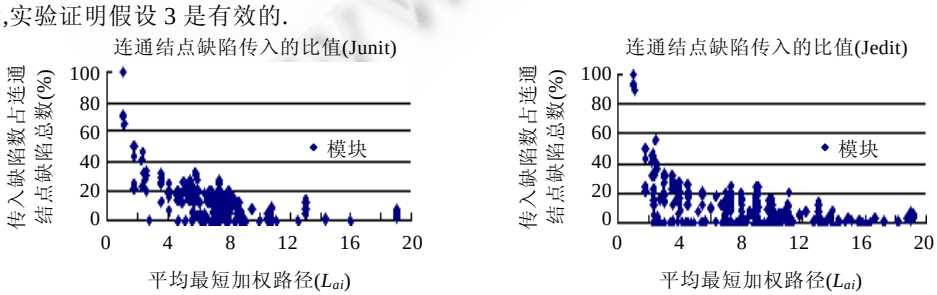


Fig.10 Relationship between L_{ai} and node impact

图 10 结点平均最短加权路径 L_{ai} 与结点影响的关系

实验结果展示,在软件设计时,软件的功能模块分解有助于软件缺陷的分散,并且减少缺陷影响的范围.当然,模块划分太多,导致软件系统的复杂性增强,这又会导致软件系统集成的复杂性增加.不过,这个问题不是本文讨论的范围.

5.3.4 假设 4 的验证

根据关键结点的定义,关键结点的入度和出度都较其他结点大很多.计算结果显示,关键结点与相邻结点边的影响测度值都比较大,故,我们用结点与相邻结点边的平均影响测度值来反映关键结点.利用上述的植入缺陷,对应每个结点与其相邻结点边的平均影响测度值,我们统计了缺陷传播强度.在第 3.3 节中,本文给出了结点影响测度值 I_{ij} 的计算方法.在定义中可知,因为每个相邻结点的边的权值 $0\leqslant w_x\leqslant 1$,$I_{ij}$ 是从一个结点到另一个结点的经过的路径上权值的乘积,故经过的路径越长,I_{ij} 的值越小.实验统计结果用图 11 所示的散列图表示,其结果是统计每个结点被其相连结点传入缺陷的个数与结点的测度值之间的对应关系.

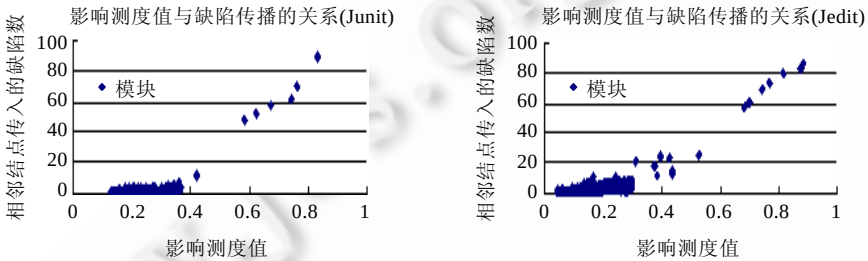


Fig.11 Relationship between key node and defect propagation

图 11 关键结点与缺陷传播

从图 11 显示的结果可以看出,有几个点的影响测度值明显要比其他结点的测度值大,这些结点就是关键结点.同样也可以看出,关键结点在测试中发现的由其邻接点传入的缺陷要远大于其他结点所测试出的缺陷数.我们在分析不同的软件网络时发现:这些关键结点大量依赖于其他结点,少数的结点甚至与其他每个结点都产生依赖关系;同时,这些关键结点也大量地被其他结点所依赖.通过实证统计和分析可以证实:这些关键结点是汇集并且传递其他结点缺陷的主要结点,其在软件网络中的影响显得非常大;而非关键结点的缺陷传播概率很低.

通过上面的验证,假设 4 是有效的.

通过假设 4 的验证看出:软件很多组成单元的缺陷传递给其他的结点的概率很低;而关键结点的缺陷由于具有相对很大的入点强度和出点强度,因此不仅很容易聚集其所依赖的相邻结点的缺陷,同时也很容易将这些缺陷传播到依赖关键结点的其他结点中去.这个实验结果很好地揭示了文献[1]提出的软件缺陷的相关表现.

6 结论及未来的工作

以系统的观点分析软件缺陷、变更等的传播规律,是一个很有效的方式.尤其是将复杂网络理论引入到软件工程的领域,提出软件网络概念,为我们认识软件的本质提供了新思路和新方法.目前的软件网络将软件的组成单元(包、类(模块)等)简单地抽象为网络的结点,研究的焦点在网络的边的一些特性上.然而,忽略软件网络结点本身的特性来研究软件网络,无法准确地反映软件网络的特性.事实上,目前一些基于软件网络的研究,理论上反映了一些软件网络的特性,但这些特性却与软件的实际表现相差甚远.为了找出这方面的原因,本文认真分析了软件网络结点的一些特性,提出了一个反映软件网络结点影响强度的加权软件网络模型 WSNNI.在此模型的基础上,本文给出了几个加权软件网络的统计特性;分析了这些统计特性与软件网络结点影响的关系;提出并验证了 4 个假设;采用软件错误植入的方式,本文验证了上述 4 个假设的合理性,从而揭示了文献[1]观察到的现象.在此研究的基础上,未来我们的相关工作是基于 WSNNI 的软件可靠性分析及软件结构优化研究,以及这些理论在软件的维护和测试中的指导.

References:

[1] Fenton NE, Ohlsson N. Quantitative analysis of faults and failures in a complex software system. IEEE Trans. on Software Engineering, 2000,26(8):797−814. [doi: 10.1109/32.879815]

[2] Torres-Pomales W. Software fault tolerance: A tutorial. Technology Report, NASA/TM-2000-210616, 2000. [doi: 10.1007/978-3-642-22655-7]

[3] Taube-Schock C, Walker RJ, Witten IH. Can we avoid high coupling?. In: Mezini M, ed. Proc. of the 25th European Conf. on Object-Oriented Programming (ECCOOP 2011). Berlin, Heidelberg: Springer-Verlag, 2011. 204−228. [doi: 10.1007/978-3-642-22655-7_10]

[4] Shi MJ, Li X, Wang XF. Evolving topology of Java networks. In: Proc. of the 6th IEEE World Congress on Intelligent Control and Automation. Dalian, 2006. 21−23. [doi: 10.1109/WCICA.2006.1712345]

[5] Wang L, Wang Z, Yang C, Zhang L, Ye Q. Linux kernels as complex networks: A novel method to study evolution. In: Proc. of the ICSM 2009. Edmonton, 2009. 41−50. [doi: 10.1109/ICSM.2009.5306348]

[6] Myers CR. Software systems as complex networks: Structure, function, and evolvability of software collaboration graphs. Physical Review, 2003,68(2):046116.1−046116.15. [doi: 10.1103/PhysRevE.68.046116]

[7] Liu J. Research on software structure analysis and optimization by applying complex network methodology [Ph.D. Thesis]. Wuhan: Wuhan University, 2007 (in Chinese with English abstract).

[8] Yan D, Qi GN. The scale-free feature and evolving model of large-scale software systems. ACTA PHYSICA SNICA, 2006,55(8): 3799−3806 (in Chinese with English abstract).

[9] Li B, Ma YT, Liu J, Ding QW. Advances in the studies on complex networks of software systems. Advances in Mechanics, 2008, 38(6):805−813 (in Chinese with English abstract).

[10] He KQ, Li B, Ma YT, Liu J, Peng R. Software Networks. Beijing: Science Press, 2008 (in Chinese).

[11] Han MC, Li DY, Liu CY, Li H. Networked characteristics in software and its contribution to software quality. Computer Engineering and Applications, 2006,20(3):9−10 (in Chinese with English abstract).

[12] Ma YT, He KQ, Li B, Liu J. Empirical study on the characteristics of complex networks in networked software. Ruan Jian Xue Bao/Journal of Software, 2011,22(3):381−407 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/3934.htm [doi: 10.3724/SP.J.1001.2011.03934]

[13] Zhang L, Qian GQ, Li L. Software stability analysis based on change impact simulation. Chinese Journal of Computers, 2010,33(3): 440−451 (in Chinese with English abstract).

[14] Pan WF, Li B, Ma YT, Qing YY, Zhou XY. Measuring structural quality of object-oriented softwares via bug propagation analysis on weighted software networks. Journal of Computer Science and Technology, 20102,5(6):1202−1213. [doi: 10.1007/s11390-010-1095-2]

[15] Valverde S, Cancho RF, SoléRV. Scale-Free networks from optimal design. Europhysics Letters, 2002,60(4):512−517. [doi: 10.1209/epl/i2002-00248-2]

[16] Sun. Java development kit 1.2. http://www.oracle.com/technetwork/java/javase/downloads/index.html?

[17] DependencyFinder. http://depfind.sourceforge.net

[18] Pajek. http://vlado.fmf.uni-lj.si/pub/networks/pajek/default.htm

[19] Yook SH, Jeong H, Barabási. Weighted evolving networks. Physical Review Letters, 2001,86(25):5835−5838. [doi: 10.1103/PhysRevLett.86.5835]

[20] Barabási AL, Albert R. Emergence of scaling in random networks. Science, 1999,286(5439):509−512. [doi: 10.1126/science.286.5439.509]

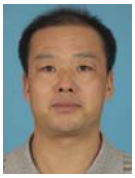
[21] Jenkins S, Kirk SR. Software architecture graphs as complex networks: A novel partitioning scheme to measure stability and evolution. Information Science, 2007,177(2007):2587−2601. [doi: 10.1016/j.ins.2007.01.021]

[22] Wang Q, Wu SJ, Li MS. Software defect prediction. Ruan Jian Xue Bao/Journal of Software, 2008,19(7):1565−1580 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/19/1565.htm [doi: 10. 3724/SP.J.1001.2008.01565]

附中文参考文献:

[7] 刘婧.基于复杂网络的软件结构分析及优化研究[博士学位论文].武汉:武汉大学,2007.

[8] 闫栋,祁国宁.大规模软件系统的无标度特性与演化模型.物理学报,2006,55(8):3799−3806.
[9] 李兵,马于涛,刘婧,丁琦伟.软件系统的复杂网络研究.力学进展,2008,38(6):805−813.
[10] 何克清,李兵,马于涛,刘婧,彭蓉.软件网络.北京:科学出版社,2008.
[11] 韩明畅,李德毅,刘常昱.软件中的网络化特征及其对软件质量的贡献.计算机工程与应用,2006,20(3):9−10.
[12] 马于涛,何克清,李兵,刘婧.网络化软件的复杂网络特性实证.软件学报,2011,22(3):381−407. http://www.jos.org.cn/1000-9825/3934.htm [doi: 10.3724/SP.J.1001.2011.03934]
[13] 张莉,钱冠群,李琳.基于变更传播仿真的软件稳定性分析.计算机学报,2010,33(3):440−451.
[22] 王青,伍书剑,李明树.软件缺陷预测技术.软件学报,2008,19(7):1565−1580. http://www.jos.org.cn/1000-9825/19/1565.htm [doi: 10.3724/SP.J.1001.2008.01565]



汪北阳(1973－),男,湖北仙桃人,博士,讲师,主要研究领域为软件工程,复杂网络,网络化软件,软件质量控制.
E-mail: wang_wwang@whu.edu.cn

吕金虎(1974－),男,博士,研究员,博士生导师,IEEE Fellow,主要研究领域为复杂系统与复杂网络,网络化软件,非线性电路与系统,系统生物学.
E-mail: jhlu@iss.ac.cn