基于复杂网络的大型软件系统度量

王树森 顾 庆 陈 森 陈道蓄
(南京大学计算机科学与技术系软件新技术国家重点实验室 南京 210093)

摘 要 随着软件系统规模的不断增大,如何度量面向对象大规模软件系统成为一个亟待解决的问题。近年来迅速发展的复杂网络理论为解决该问题提供了一个新的视角。介绍了大规模软件系统中的复杂网络现象,从元素级、模块级、网络级3个不同的粒度提出基于复杂网络的软件系统的各种度量;实现了大型Java程序复杂网络描述和度量工具JPAC。JPAC可用于分析大型Java系统的结构,并计算基于复杂网络的各度量值。

关键词 复杂网络,软件系统,软件度量,JPAC

Metrics for Large-scale Software Systems Based on Complex Networks

WANG Shu-sen GU Qing CHEN Tao CHEN Dao-xu
(State Key Laboratory for Novel Software Technology, Department of Computer Science and Technology, Nanjing University, Nanjing 210093, China)

Abstract With the aggrandizement scale of software systems, how to measure the complexity of large-scale object-oriented software systems has been a challenge for software engineers. The theory of complex networks recently studied has offered a new perspective on solving this problem. We studied the phenomenon of complex networks in large-scale software system, and proposed metrics from three different levels, i. e. , unit-level, module-level and network-level. We built JPAC, a tool for characterizing and measuring the complexity of Java systems. JPAC can be used to analyze the structure of large-scale Java systems, and compute software metrics based on complex networks.

Keywords Complex networks, Software system, Software metrics, JPAC

1 引言

度量是工程技术领域中一个不可或缺的要素,近年来随着软件工程领域的长足发展,度量技术逐渐融入到软件工程领域,并成为良好软件工程的一个重要组成部分,因为"凡是不能度量的,便是不能对其进行控制的"[1]。度量有助于对软件开发中的各项活动进行了解和控制,有助于过程和产品的改进。随着软件工程的发展,对软件开发中的过程、产品和资源进行量化的观点已被越来越多的人认同,因此也推动了对软件度量的研究。传统的面向对象度量方法,如C&K方法[2]、MOOD方法[3]等主要关注于类,衡量类的内部复杂性,类与类之间的关系的复杂性,如继承复杂性、耦合复杂性等等。这些度量仅仅注重统计与对象类有关的数量或规模,对于大型软件而言,过于侧重细粒度的统计,而没有关注粗粒度的模块构件等方面的度量。

近年来复杂网络理论的提出为大规模软件系统的度量提供了一种新的视角。研究表明,很多社会网络、生态网络、Internet、WWW、软件系统调用图等都具有明显的复杂网络特征[4-7],包括小世界效应、无标度效应等。通过将实际中的社会网络、生态网络等视为复杂网络,研究其结构特征,研究者们发现了这些网络的形成和演化的很多规律。这些研究对社会学、生物学、物理学的发展已经产生了一定的影响。

研究表明,大型软件系统中也存在复杂网络现象[8-13]。大型软件系统中存在大量不同粒度的单元,对这些软件单元以及它们之间的交互关系建模,发现很多网络具有明显的复杂网络现象。将复杂网络普遍具有的性质和规律应用于软件中,从而研究软件系统的结构特征,可以更好地理解和控制软件系统。

本文首先分析大规模面对对象软件系统的类层次结构图,发现这些图具有明显的复杂网络特征。继而基于复杂网络理论,从元素级、模块级、网络级3个不同的粒度提出基于复杂网络的软件系统的各种度量;第2部分介绍了复杂网络理论和大型软件系统中的复杂网络特征;第3部分介绍了基于复杂网络的软件度量;第4部分介绍了大型Java程序复杂网络描述和度量工具JPAC。JPAC可以分析Java软件系统的类层次结构图,将具有复杂网络特征的大型软件系统划分为各模块,并计算基于复杂网络的各种度量值。

2 大型软件系统的复杂网络特征

2.1 复杂网络

系统的复杂性主要决定于元素之间的交互。只要能保持系统元素之间交互的基本性质,那么系统的基本特性就不会改变,因此网络模型是描述大型复杂系统的有效模型。大量最新研究发现:网络的拓扑结构决定着网络所拥有的特

到稿日期:2008-04-01 本文受国家"863"项目(编号:2006AA01Z177),江苏省自然科学基金基础研究项目(编号:BK2006115)资助。
王树森 硕士研究生,研究方向为软件工程,E-mail: wangss@dislab.nju.edu.cn;顾 庆 副教授,CCF会员,研究方向为分布式计算与并行处理、软件工程;陈 森 硕士研究生,研究方向为软件工程;陈道蓄 教授,博士生导师,研究方向为分布式计算与并行处理等。

性[14]。在以前的研究中，由于对网络拓扑结构性质知之甚少，科学家们往往忽略了网络的拓扑性质。最近几年，由于计算机数据处理和运算能力的飞速发展，科学家们发现大量真实系统的网络模型既不是规则网络，也不是随机网络，而是呈现出乎意料的统计特征。这样的网络是真实复杂系统的拓扑抽象，因此被称为复杂网络(complex networks)。

1998 年，Watts 等发现大量真实网络都具有小世界效应[4]；1999 年，Barabasi 和 Albert 指出许多现实世界中的复杂网络的连接度分布服从幂律分布[5]。由于幂律分布的网络不像随机网络和规则网络可以将平均度看作节点度的特征标度，因此将这类网络称为无标度网络。无标度特性刻画了复杂网络的不均匀复杂性，即大部分节点只有少数连结，而少数节点则拥有大量的连结。此后的研究表明，现实世界中许多复杂网络都具有小世界或无标度等特征。各种基于复杂网络特性的研究已成为一个极其重要而且富有挑战性的科学前沿方向。

复杂网络具有很多与规则网络和随机网络不同的统计特征，其中最重要的是小世界效应(small-world effect)和无标度特性(scale-free property)。后面我们将给出复杂网络各种特征的具体描述和定义。需要注意的是，经典的计算机科学中的“网络”是指加权图，而“复杂网络”作为一个专有名词是指无权图。后面中我们提到“网络”时都指无权图。

2.2 大型软件系统中的复杂网络特征

软件系统中存在不同粒度的单元，如方法(functions)、类(classes)、接口(interfaces)、类库(libraries)、编译单元(compiling units)、包(packages)等等。这些软件单元之间存在很多交互关系，如方法之间的调用关系，类之间的继承、聚合关系，编译单元之间的包含关系等等。对这些软件单元以及它们之间的交互关系建模，以软件单元为顶点，以它们之间的交互关系为边，建立网络模型。研究发现，复杂网络现象在不同粒度的单元之间以不同关系形成的各种网络模型中是普遍存在的。

Potanin 等[8]研究了几个大规模面向对象的软件系统，以对象为顶点，对象之间的引用关系为边构造网络。他们分析了几个流行的面向对象语言开发的软件系统，包括 C++，Java，Smalltalk 等，这些软件系统形成的网络都具有明显的复杂网络特征。Valverde 等[9]研究了大规模开源软件的软件体系结构图，以类为顶点，类之间的关系为边构造网络，研究其复杂网络特征。Myers[10]提出软件协作图(Software Collaboration Graph)的网络模型，研究开源软件中的无标度现象。Wheeldon 等[11]以大规模 Java 系统的不同属性、不同关系构造网络，发现了 12 种不同方式构造的网络都具有无标度效应。以包含源代码的文件为顶点，文件间的包含关系为边，研究者们发现 C/C++系统有无标度现象。Tamai 等[12]提出一个统计模型，统计了大型软件中类的方法数和方法的代码行数，分析并解释了其中的无标度现象。Concas 等[13]研究了数十个以 Smalltalk 语言开发的大型软件系统，计算了各种系统属性，包括变量名、方法名的分布，继承图，类和方法的大小，以及系统的体系结构图，发现其中的幂律分布并尝试从软件演化的角度解释这些现象。

3 基于复杂网络的软件度量

3.1 软件度量

如何提高软件的质量，始终是软件工程领域研究的重要方向。基于度量的量化管理是目前最有效的质量保证手段之一。良好的软件度量有助于对软件开发中的各项活动进行了解和控制，有助于过程和产品的改进。

传统的 McCabe 环复杂度是一种较为成功的结构化程序度量方法，其主要思想是基于计算线性独立路径条数来度量程序的复杂性[15]。Boehm 等[1]提出了定量地评价软件质量的概念，给出了 60 个质量度量公式，表明怎样用于评价软件质量，并且首次提出了软件质量度量的层次模型。其主要思想是将软件质量的概念分解为若干层次，对底层的软件再引入数量化的指标，从而得到软件质量的整体评价。Waters 和 McCall 等[1]提出了从软件质量要素(factor)、准则(criteria)到度量(metric)的 3 层次式软件质量度量模型。他们将软件质量要素降为 11 个，且给出了各要素的关系。他们的主要思想是，要素是软件质量的反映，而软件属性可用作评价准则，定量化地度量软件属性，从而反映软件质量的优劣。

目前面向对象思想和方法成为软件设计与开发的主流，但上述传统的度量方法无法应用于许多面向对象的特性。因此，出现了一些适合面向对象软件的度量方法。较为有代表性的是 C&K 方法[3]和 MOOD 方法[4]。这些度量主要关注于类，衡量类的内部复杂性、类与类之间的关系的复杂性，如继承复杂性、耦合复杂性等等。这些度量仅仅注重统计与对象类有关的数量或规模，对大型软件而言，过于侧重细粒度的统计，而没有关注粗粒度的模块构件等方面的度量。

3.2 基于复杂网络的度量

由于软件系统中存在大量的复杂网络现象，因此复杂网络的特征统计量可以衡量软件系统的结构特征。复杂网络的一些重要的属性可以表示复杂网络的拓扑结构特征。软件系统中的复杂网络也具有这些性质，这是各种复杂网络的共性。我们可以从元素级、模块级、网络级等 3 个层次度量软件系统的结构特征。图 1 给出了三层度量模型。

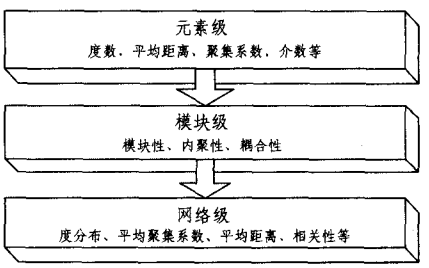

图 1 三层度量模型

为下面叙述方便，我们定义复杂网络 $G=(V,E)$，V 为结点集合，E 为边的集合。

(1)元素级的度量

元素指构成图的最基本的单元，包括顶点和边。顶点的特征统计量包括：

度数 结点度数是结点最基本的特征。度数可衡量结点的重要程度。结点的度数越大，其所代表的类在网络中与其他类交互得越多，该类重要程度就越大。

平均距离 网络中两点间的最短距离是指两个结点之间至少要经过的边的数目。某个顶点的平均距离指该顶点到其他所有顶点的最短距离的平均，定义为式(1)。对于非连通图，考虑到某些顶点不可达，定义可采用式(2)。

$$\overline{d_i}=\frac{\sum_{i\neq j}d(i,j)}{|V|} \tag{1}$$

$$\frac{1}{\overline{d_i}}=\frac{1}{|V|}\sum_{j\neq i}\frac{1}{d(i,j)} \tag{2}$$

平均距离表明了顶点在网络中的中心度。顶点的平均距离越小，则该顶点所代表的类与其它顶点所代表的类联系所需要经过的边越少，说明该顶点所代表的类的中心度越高。

聚集系数(Clustering coefficient)　结点的聚集系数用来描述网络中结点的聚集情况，即结点邻边的连接程度。其定义为结点邻边间实际存在的边数与可能存在的最大边数之比。若结点 i 有 n_i 条邻边，这些邻边之间有 m 条连边，则该结点的聚集系数为

$$C_i=\frac{m}{n_i(n_i-1)/2} \tag{3}$$

聚集系数表明了结点的相邻结点间的连接程度，可以衡量类之间的内聚性。顶点聚集系数越大，其所代表的类的相邻类间连接越紧密，类之间的内聚性越高。

介数(Betweenness centrality)　网络中任意可达的两点之间存在至少一条最短路径。对于某结点或边，通过其上的最短路径越多，则该结点重要程度越高，因此可定义结点或边的介数为

$$B_u=\sum_{i\neq j}\frac{\sigma(i,u,j)}{\sigma(i,j)} \tag{4}$$

其中 $\sigma(i,j)$ 指结点 i 和 j 之间总的最短路径的个数，$\sigma(i,u,j)$ 指结点 i 和 j 之间经过 u 的最短路径的个数。

介数表明结点的重要程度。介数越大，表明经过该结点的最短路径越多，因此该结点所对应的类为比较关键的类，该类的重要程度越高。

边的特征统计量包括：

介数(Betweenness centrality)　边的介数定义与顶点的介数定义相同，见式(4)。边的介数大小表示了边的重要程度。

(2)模块级的度量

社区结构是复杂网络最重要的特征之一[16-18]。很多实际网络表现出明显的社区现象，即社区内的连边相对较多，社区间的连边相对较少。在生物网络、社会网络中，社团可刻画网络的模块化特性。我们在研究大型软件系统时，也发现软件系统具有明显的社区现象。因此，我们可从模块级定义软件的度量。

模块性　Newman 与 Girvan 最早提出计算衡量社区模块性(Modularity)的计算公式[17]。该定义是基于以下两个事实：(1)社区内的结点之间的连边应尽可能多；(2)社区间的结点之间的连边应尽可能少。模块性定义为式(5)。

$$M=\sum_{s=1}^{r}\left[\frac{l_s}{L}-\left(\frac{d_s}{2L}\right)^2\right] \tag{5}$$

其中 $L=\sum|E|$ 为 G 的总边数，$\delta(v,w)=\begin{cases}1,\text{if}(v,w)\in E,\\0,\text{otherwise}\end{cases}$，$l_s=\sum_{v\in r_s,w\in r_s}\delta(v,w)$ 为社区 C_s 内各顶点间的边数总和，$d_s=\sum_{v\in r_s,w\in V}\delta(v,w)$ 为社区 C_s 内所有结点的总度数。

式(5)表明，模块性为社区内的边的比例与相对应的随机图的这些边的期望的比例之差。社区内的边数越大，社区间的边数越少，M 的值越大，社区的特征越明显。M 值越大表示很大比例的边属于选定的社区内部。

对于软件系统而言，结点的相互作用范围往往就是结点内在语义协同的成果，如包、构件等就是将许多类集合成一个更高层次的单位，形成一个高内聚、低耦合的类的集合。对于给定的包或构件，M 的值越大，表明该划分的模块性越好。

内聚性　模块内部的拓扑结构可能是多种多样的，有的模块可能有一个或几个中心点，其他结点均与中心点相连。有的模块则可能是完全分散的结构，每个结点的度数几乎相同。为了刻画模块内结点的特征，可以定义结点的 z 值(z-score)为式(6)：

$$z_i=\frac{\kappa_i-\overline{\kappa_C}}{\sigma_{\kappa_C}} \tag{6}$$

其中 κ_i 表示结点 i 在模块 C_i 中的边数，$\overline{\kappa_C}$ 为模块 C_i 内所有结点的 κ 的平均值，σ_{κ_C} 为模块 C_i 内所有结点的 κ 的标准方差。

z_i 的大小表示了结点 i 在模块内与其他结点的连接程度。z_i 的值越大，表明该结点在模块中越处于中心的位置，结点的内聚程度越高；z_i 的值越小，表明该结点在模块中越处于边缘的位置，结点的内聚程度越低。

耦合性　结点的边的分散程度由参与系数(participation coefficient)来定义。结点 i 的参与系数定义为

$$P_i=1-\sum_{s=1}^{r}\left(\frac{\kappa_{iC}}{k_i}\right)^2 \tag{7}$$

其中 κ_{iC} 为结点 i 与模块 C 内结点的边数，k_i 为结点 i 的总度数。结点的边越集中，则 $\sum_{s=1}^{r}\left(\frac{\kappa_{iC}}{k_i}\right)^2$ 的值越大，P_i 越小，表明耦合性越小；结点的边越分散，则 $\sum_{s=1}^{r}\left(\frac{\kappa_{iC}}{k_i}\right)^2$ 的值越小，P_i 越大，表明耦合性越高。

(3)网络级的度量

网络级的度量指全局特征的度量。可从如下几个特征量考察软件系统的整体结构特征。

度分布 $P(k)$　定义为图中度数为 k 的结点的个数占网络结点总个数的比值。也可理解为随机选择一个顶点，该顶点度数为 k 的概率。在随机图中，由于每条边生成的概率相同，因此随机图的度分布服从 Poisson 分布。而在实际的大型网络中，比如 World-Wide Web，Internet，JDK 的类图等等，研究发现这些图的度分布服从幂律分布 $P(k)\sim k^{-\gamma}$。由于幂律分布的无标度性，我们称这些网络具有无标度特性。

结点的度分布反映了网络的整体性质，因此是软件系统的一个整体的度量。参数 γ 的大小表明了网络的整体结构。一般而言，γ 取值为(1.0,3.0)之间。

平均距离　网络的平均距离指所有结点之间的最短路径的均值。软件系统的平均距离表明了类与类之间的平均距离，表明类之间的紧密程度。在基于继承关系形成的图中，平均距离可表示类之间的平均继承深度；在基于聚合关系形成的图中，平均距离可用来衡量类之间聚合的程度。

平均聚集系数　网络的平均聚集系数指所有结点的聚集系数的均值，反映了网络整体的内聚性。

相关性　正相关表明度数较大的结点倾向于与度数较大的结点相连，负相关表明度数较大的结点倾向于与度数较小的结点相连。网络的相关性与其演化过程密切相关。

4 大型 Java 程序复杂网络描述和度量工具 JPAC

4.1 复杂网络工具 JPAC

复杂网络目前比较流行的工具有 Pajek(Program for Large Network Analysis)和 JUNG(Java Universal Network/Graph Framework)。Pajek 功能强大,但有其自身的格式限制;JUNG 是基于 Java 的网络、图形可视化类库,是一个开源的可扩展的 Java API 框架。

为了更充分地研究和表示大型 Java 软件的结构特征,我们开发了大型 Java 程序复杂网络描述和度量工具 JPAC,其结构如图 2 所示。

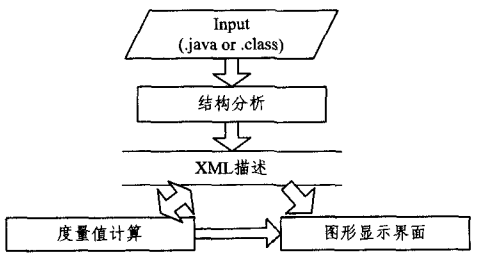

图 2 JPAC 结构图

JPAC 的输入是 Java 源代码文件,包括源码(.java)或字节码(.class)。JPAC 在分析源码的结构后,以每个类为顶点、以类之间的关系为边生成一个有向图,即 JSCG,下节中我们将描述该模型。JSCG 记录了我们所需要的软件系统的结构信息。生成 JSCG 后,JPAC 将结构信息用 XML 文件保存起来。基于 XML 文件所保存的信息,JPAC 可以将这些结构信息显示出来,也可以对其进行分析,分析其复杂网络特征,计算基于复杂网络的各度量,并将度量值存入 XML 文件中。JPAC 可以将这些度量信息与结构信息放在一起,在界面上显示。

4.2 软件系统网络模型 JSCG

大规模的软件系统是由若干类相互作用组成的,根据这些相互作用可以定义一个网络拓扑图,并对其进行分析,研究软件系统结构的复杂网络特性及其演化规律。Christopher R. Myers 提出了 Software Collaboration Graph (SCG)[10],描述了软件系统的类和类之间的相互关系,是一种简化了的类图(不考虑变量和方法)。

在 SCG 的基础上我们建立了符合 Java 语言的网络图结构,定义 Java SCG,简称 JSCG[19]。该图的结点是一个类、接口或者枚举类,边是类、接口或者枚举类之间的相互关系。这里主要考虑 3 种相互关系:继承关系(inheritance)、实现关系(implementation)、聚集关系(aggregation),且这些关系是有向的,因此 JSCG 是一个有向图。用数学语言精确地描述为:

定义 JSCG(Java Software Collaboration Graph)是一个图:$JSCG=\{V,E\}$,其中 $V=\{v \mid type\ of\ v\in CLA\}$

$CLA=\{class,interface,enum\}$

$E=\{(v1,v2)\mid v1,v2\in v\ (v1\ r\ v2\ r\in REL\}$

$REL=\{inherit,implement,aggregate\}$

E 是有向的,依据 REL 中的关系由 $v1$ 指向 $v2$。REL 中的关系是可选的,可以只考虑其中一个关系建立边,也可以考虑其中两个或全部考虑来建立边,根据不同需求进行分析。

因 Java 编译器为每个类生成一个字节码文件,就字节码进行分析将非常方便。我们基于字节码的分析方法来构建 JSCG。可以用两种分析字节码的方法:1)将字节码文件先装入 JVM,再从 JVM 中读取类信息;2)直接从字节码文件中读取类信息,读取后根据类与类之间的关系确定顶点之间的连边。逐个读入类字节码文件,直到结束。

图 3 是一个 JSCG 构建的例子。JSCG 构建成功后,可以很方便地对复杂网络的几个重要特征进行统计和分析。JDK 的代码结构也可以按照此方法建立其 JSCG。我们用 JPAC 工具分析 JDK 各版本(1.2—1.5)的 JSCG,验证 JDK 系统中的复杂网络现象,计算 JDK 系统的一些度量值。

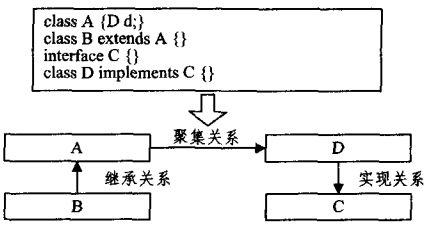

图 3 JSCG 的构建

4.3 计算度量值和 JPAC 界面显示

JPAC 可以计算前面提到的一些度量值。我们以 JDK 1.5.0_06 为例,画出其 JSCG 结构图,并计算基于复杂网络的度量值。

图 4 展示了 JDK1.5.0_06 的 JSCG 结构图。图中结点的位置与其中心度有关。结点越靠近中心,说明其所代表的类的中心度数越大。每个结点所代表的类的其他信息储存在 XML 文件中,选中某结点即可在图中显示出来。

图 5 展示了 JDK1.5.0_06 的度量值。界面上画出了入度、出度、总度数分布的双对数坐标图。可以看出,曲线都近似于一条缓慢下降的直线,用 Matlab 对这些曲线(双对数坐标)进行拟合,拟合结果如表 1 所列,γout,γin,γall 三项分别对应着 3 个双对数坐标曲线拟合后的斜率的绝对值。这就说明这些度分布符合幂律分布 $P(k)\propto k^{-\gamma}$,而 γout,γin,γall 分别对应着出度标度指数、入度标度指数、总度标度指数。

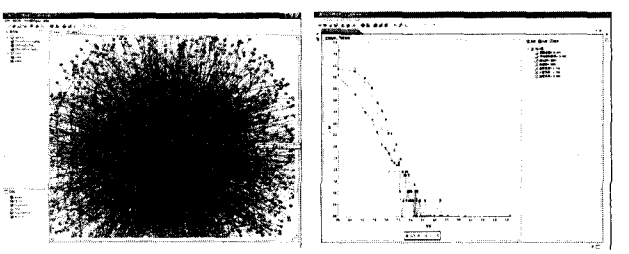

图 4 JDK1.5.0_06 的 JSCG 结构图 图 5 JDK1.5.0_06 的度量值

表 1 中显示了 jdk1.2.2,jdk1.3.0,jdk1.4.2,jdk1.5.0,jdk1.6.0 五个版本的 java 包的 JSCG 的特征数据,从中可以研究 JDK 等大型软件系统的演化规律。

表 1 JDK5 个版本的 JSCG 特征数据

JDK 版本号	1.2.2	1.3.0	1.4.2	1.5.0	1.6.0
节点数	1128	1236	1777	2138	2363
边数	2496	2751	3877	4708	5251
平均度数	4.4255	4.4515	4.3635	4.4041	4.4444
出度标度指数 γout	2.4154	2.4763	2.5506	2.6331	2.5877
入度标度指数 γin	1.0374	1.0722	1.0840	1.0721	1.0736
总度标度指数 γall	1.3139	1.3394	1.3682	1.3339	1.3510
平均距离	2.9225	2.9567	3.0958	3.0259	3.0215
平均簇系数	0.5803	0.5762	0.5655	0.5695	0.5657

(下转第 302 页)

Architecture // NODe 2004. LNCS 3263. Berlin Heidelberg: Springer-Verlag, 2004: 138-152

[5] Jansen A G J, Smedinga R, van Gurp J, et al. First class feature abstractions for product derivation// IEE Proc. -Softw. 2004, 151(4)

[6] Filman R E, Elrad T, Clarke S, et al. Aspect-Oriented Software Development. ISBN: 0-321-21976-7. October 2004

[7] AspectJ. in action: PRACTICAL ASPECT-ORIENTED PROGRAMMING. RAMNIVAS LADDAD. ISBN 1-930110-93-6. Printed in the United States of America

[8] Lee K, Kang K C. Feature Dependency Analysis for Product Line Component Design // ICSR 2004. LNCS 3107. Berlin Heidelberg: Springer-Verlag, 2004: 69-85

[9] 刘小龙，王忠. 面向方面编程研究. 开发研究与设计技术. 武汉工程大学计算机科学与工程学院

（上接第 290 页）

与 Pajek, JUNG 等大型网络分析工具相比，JPAC 采用 XML 文件保存网络的结构信息，具有简单易用的特点，通用性也更好。JPAC 功能强大，可以分析大型 Java 系统的结构，并生成与复杂网络相对应的随机网络，比较二者的异同。JPAC 可以计算基于复杂网络的各度量值，并用模拟退火算法分离出社区结构，亦即软件的拓扑意义上的模块结构。通过对生成的模块结构与原结构的比较，可以分析各种模块划分方法的优劣，从而对软件系统更好地理解、评估、预测、控制和改进。

结束语 如何提高软件的质量，始终是软件工程领域研究的重要方向。基于度量的量化管理是目前最有效的质量保证手段之一。随着面向对象方法的日益流行，软件系统规模的不断增大，如何度量面向对象大规模软件系统，成为一个亟待解决的问题。

本文运用复杂网络理论解决这个问题。研究发现，软件系统中存在大量复杂网络现象。通过研究复杂网络所共有的一些特征，从元素级、模块级、网络级 3 个不同的粒度提出基于复杂网络的软件系统的各种度量。介绍了大型 Java 程序复杂网络描述和度量工具 JPAC。JPAC 可以分析大型 Java 系统的结构，并计算基于复杂网络的各度量值。通过计算这些度量值，可以定量地刻画和分析软件的结构和行为，以便更好地理解、评估、预测、控制和改进。

用 JPAC 分析了 JDK 各版本，我们发现其 JSCG 图具有明显的复杂网络的特征。我们计算了基于复杂网络中度量，并分析了 JDK 的演化。

下一步的工作是研究尽可能多的软件系统，提出度量的定量标准，并指导于实际的软件开发中。我们可以计算出各结点类的 z-score 的值和参与系数的值，并按分散系数的值将各结点分类，从而与面向构件的软件开发有效地结合起来。我们的工作也可应用于遗产系统构件的识别与提取、构件的度量等问题。

参 考 文 献

[1] Fenton N E, Pfleeger S L. Software Metrics A Rigorous and Practical Approach. Second Edition. Boston: PWS Pub., 1997

[2] Chidamber S R, Kemerer C F. A Metrics Suite for Object-oriented Design. IEEE Trans. on Soft. Eng., 1994, 20(6): 476-493

[3] Brito e Abreu F. The MOOD Metrics Set // Proc. of ECOOP'95 Workshop on Metrics. Aarhus, Denmark, 1995

[4] Strogatz S H. Exploring complex networks. Nature, 2001, 410: 268-276

[5] Albert R, Barabasi A-L. Statistical mechanics of complex networks. Rev. Mod. Phys., 2002, 74: 47-97

[6] Dorogovtsev S N, Mendes J F F. Evolution of Networks: From Biological Nets to the Internet and WWW. Oxford: Oxford University Press, 2003

[7] Newman M E J. The structure and function of complex networks. SIAM Review, 2003, 45: 167-256

[8] Potanin A, Noble J, Frean M, et al. Scale-free Geometry in Object-oriented Programs. Comm. ACM, 2005, 48: 99-103

[9] Valverde S, Ferrer-Cancho R, Sole′ R. Scale-free Networks from Optimal Design. Europhysics Letters, 2002, 60: 512-517

[10] Myers C. Software Systems as Complex Networks: Structure, Function, and Evolvability of Software Collaboration Graphs. Physical Rev. E, 2003, 68

[11] Wheeldon R, Counsell S. Power Law Distributions in Class Relationships // Proc. Third IEEE Int'l Workshop Source Code Analysis and Manipulation. 2003

[12] Tamai T, Nakatani T. Analysis of Software Evolution Processes Using Statistical Distribution Models // Proc. Int'l Workshop Principles of Software Evolution (IWPSE). 2002: 120-123

[13] Concas G M. Power-Laws in a Large Object-oriented Software System. IEEE Trans. on Soft. Eng., 2007, 33(10): 687-708

[14] 李冰，王浩，等，基于复杂网络的软件复杂性度量研究. 电子学报，2006, 34(B12): 2371-2375

[15] McCabe T A. Complexity measure. IEEE Trans. on Soft. Eng., 1976, 2(4): 308-320

[16] Guimera R, Amaral L. Cartography of complex networks: modules and universal roles. Nature, 2005, 433(7028): 895-900

[17] Newman M E J, Girvan M. Finding and evaluating community structure in networks. Phys. Rev. E, 2004, 69: 026113

[18] Newman M E J. Modularity and community structure in networks // Proc. Natl. Acad. Sci. USA, 2006, 103: 8577-8582

[19] 陈焘，王树森. 基于复杂网络的 JDK 代码结构演化研究. NASAC, 2006

[20] 汪小帆，李翔，陈关荣. 复杂网络理论及其应用. 北京：清华大学出版社，2006