# 基于 $k$-核的大规模软件宏观拓扑结构层次性研究

李 辉[1,2],赵 海[1,2],徐久强[2],李 博[2],李 鹏[2,3],王家亮[2]
(1. 医学影像计算教育部重点实验室(东北大学),辽宁沈阳 110819;
2. 东北大学信息科学与工程学院,辽宁沈阳 110819;3. 辽宁大学信息学院,辽宁沈阳 110036)

**摘 要:** 大规模软件宏观拓扑结构的层次性研究对软件体系结构的理解与控制具有重要作用.本文首先根据大量大规模开源软件结构的核数统计数据,发现其层次结构的扁平和相对稳定,并论述了以 $k$-核分析大规模软件结构层次性的合理性.在此基础上,以 Eclipse 3.4.2 版本作为实证分析,对其进行了层次划分,分析了各层层内和层间连接关系,讨论了高层区域与低层区域的拓扑特征.最后,以 Eclipse 多个版本作为实证分析,从网络规模、度分布、小世界特征等方面对最高层结构演化进行了研究.

**关键词:** 软件结构; 层次; $k$-核; 无尺度; 最高层
**中图分类号:** TP393 **文献标识码:** A **文章编号:** 0372-2112 (2010) 11-2635-09

# Research on Hierarchy of Large-Scale Software Macro-Topology Base on $k$-core

LI Hui[1,2], ZHAO Hai[1,2], XU Jiu-qiang[2], LI Bo[2], LI Peng[2,3], WANG Jia-liang[2]
(1. *Key Laboratory of Medical Image Computing (Northeastern University), Ministry of Education, Shenyang, Liaoning* 110819, *China*;
2. *School of Information Science and Engineering, Northeastern University, Shenyang, Liaoning* 110819, *China*;
3. *School of Information, Liaoning University, Shenyang, Liaoning* 110036, *China*)

**Abstract:** Research on hierarchy of large-scale software macro-topology plays an important role in comprehension and control of software Architecture structure. In this paper, we find that the large-scale software hierarchical structures are flat and relative stable based on a large number of statistics data on coreness of open source software structures, and argue the rationality of analyzing the large-scale software structure hierarchy using the means of $k$-core. On this basis, we use a version of Eclipse 3.4.2 as empirical analysis, divide it into layers, and analyze the internal and external connected relation, then argue the topology characteristics of the high-layer regions and low-layer regions. Finally, we use several versions of Eclipse as empirical analysis to study the evolution of highest layer form some aspects as network scale, degree distribution, and small-world characteristic.

**Key words:** software structure; hierarchy; $k$-core; free-scale; highest layer

## 1 引言

随着软件规模的增长以及所涉及问题的复杂度的提高,软件开发人员在处理软件系统复杂性方面所遇到的困难在不断增多,处理问题的难度呈几何式增长,由此所带来的软件产品质量控制问题严重困扰着软件开发人员,成为现代软件工程研究必须面对的挑战[1,2].

现代大规模软件的庞大规模和高复杂性决定了对软件结构的研究应从宏观层面入手,将软件结构作为一个有机整体进行研究,这样可以避免过多的纠缠于结构中琐碎细节;而从宏观层面上对层次性的研究对软件结构的理解与控制具有至关重要的意义,一直是软件工程领域研究的重要问题[3].

早在上世纪 90 年代初,一些经典的软件度量方法就已经开始对软件结构的层次性进行了研究,比如 Chen&Liu 度量方法[4]、C&K 度量方法[5]以及 MOOD 度量方法[6]分别提出了对软件结构的继承性度量,为软件产品的质量检测和控制做出了重要贡献.研究人员通过度量软件体系结构来理解和控制软件层次结构的方法,在软件工程实践中也得到了良好的应用.

但是随着软件规模的不断增长,这些度量方法都逐渐显现出若干共性的不足,软件规模越大度量本身产生的错误也随之增多,而且它们过于关注软件结构的微观特性,对结构的宏观层面关注略显不足,并且很难全面

收稿日期:2009-08-26;修回日期:2010-03-24
基金项目:国家自然科学基金(No.60973022);高等学校科技创新工程重大项目培育资金(No.708026);中央高校基本科研业务费专项资金资助(No.N090604013)

反映软件系统的结构特性.因此,如何突破软件规模的限制,并且从宏观拓扑层面上对软件结构进行度量研究,是研究人员必须解决的难题.

从 2002 年开始,一些复杂系统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拓扑图,从网络拓扑的层面研究了软件结构的整体特性,拉开了结合复杂网络理论研究软件结构的序幕[7~10].复杂网络与软件工程结合的软件结构研究,解决了许多传统软件度量方法难以解释的问题,逐渐成为软件工程研究人员的共识,被认为是解决现代软件危机的一种关键方法,为软件产品度量研究及质量控制探索了一条新的道路.

在对 Internet 拓扑结构的研究中,利用 $k$-核对 Internet 进行层次性分析被认为是降低复杂性的有效方法,并取得了若干研究成果[11~13].大规模软件系统其结构通常体现出与 Internet 结构相似的一种自组织的多层次特征,因此 $k$-核也被用来表征大规模软件系统存在的层次结构,并作为描述宏观拓扑结构的一种可视化方法[15],但是进一步对软件结构层次分解以及量化研究目前还很少.本文面向大规模软件,利用 $k$-核及相关特征量统计和定量分析大规模软件其结构的层次性,通过理解和发现大规模软件结构复杂性背后蕴含的规律,可以为大规模软件结构度量提供一种可行的方法.

## 2 大规模软件宏观拓扑结构网络化表示

用复杂网络的分析方法分析大规模软件宏观拓扑结构,就必须实现软件结构与复杂网络的映射.本文将大规模软件宏观拓扑结构抽象为一个由点集和边集组成的图,具体来说,将软件源代码中的类、接口、结构体等基本程序单元抽取为图中的节点,将节点间的继承、关联、泛化、依赖等关系抽取图中的边,这里将起点和终点均相同的重边作为一条边处理.这些点集和边集组成的图实现了软件结构与网络拓扑之间的映射[16],可以在这基础上用复杂网络的分析方法对其进行定量的分析.

## 3 软件结构的 $k$-核

### 3.1 软件结构的核数

**定义 1** $k$-核($k$-core):图的 $k$-核是指反复去掉度值小于或等于 $k$ 的节点及与其连接的边之后所剩余的子图.

**定义 2** 核数(Coreness):节点的核数(以 $C$ 表示)表示包含该节点的最深的核,即节点存在于 $k$-核中,但是在($k+1$)-核中被移除,则节点的核数为 $k$.节点核数中的最大值称为图的核数($k$-$core_{max}$)[11].

节点的核数 $C$ 表示节点在图中的深度,描述了网络拓扑的层次特征[14].根据 $k$-核和核数的定义,对选取的 Linux 内核、Eclipse、Firefox 等知名开源软件的源代码进行度量,这些软件由于可用性、可靠性等各方面普遍受到较高的评价,具有较高的可信性.首先计算结构中各节点的核数 $C$,进一步得到软件结构网络拓扑图的核数 $k-core_{max}$.表 1 统计了这些软件结构的核数的度量结果,并列出了它们的节点数($N$)和边数($M$).

**表 1 软件结构的核数统计**

| 软件 | 节点数 $N$ | 边数 $M$ | 核数 $k-core_{max}$ |
|---|---|---|---|
| Linux 内核 2.4.36 | 17604 | 32721 | 7 |
| Firefox 2.0.0.1 | 10595 | 18739 | 6 |
| Kaffe 1.1.6.91 | 7849 | 12120 | 6 |
| MySQL 6.0.6 | 3793 | 5368 | 6 |
| VTK 5.0.4 | 2121 | 3292 | 6 |
| Tomcat 4.1.18 | 1452 | 1731 | 6 |
| Eclipse 3.2.2 | 9453 | 8452 | 5 |
| Koffice 1.5.0 | 4580 | 5892 | 5 |
| Openafs 1.4.8 | 1127 | 1230 | 5 |
| PostgreSQL 8.2.3 | 1060 | 1313 | 5 |
| Kopete 0.10.1 | 1010 | 1031 | 5 |
| Doxygen 1.5.1 | 717 | 1181 | 5 |
| Abiword 2.4.6 | 1993 | 3100 | 4 |
| Freemind 0.9.0 | 713 | 933 | 4 |
| Pingus 0.7.2 | 450 | 683 | 4 |
| Filezilla 3.2.3 | 431 | 529 | 4 |

可以看到,软件结构虽然结构复杂,但是核数 $k-core_{max}$ 普遍不高,这里统计的开源软件核数分布在 4~7 之间.相关文献[17]指出,同样是人工复杂系统,AS 级 Internet 的核数达到了 20~25,远远高于软件结构的核数.这说明,一方面表现出软件结构具有明显的层次特征,另一方面也说明目前软件系统结构的层次数目是非常有限的,层次结构相对扁平.软件结构的核数分布在一个相对小的范围,有助于评价现有软件结构的设计是否合理,以及在重构中结构是否稳定.

### 3.2 核数演化

在软件系统生命周期中,用户需求及所处环境等因素的变化,会促使软件系统不断重构和升级,软件系统的结构也在不断演化[18].随着软件系统的重构,软件结构中核数 $k$-$core_{max}$ 的变化也在一定程度上体现了结构拓扑的变化,尤其是层次演化趋势,对软件结构分析具有重要意义.

这里选取 Eclipse、Linux 内核与 Abiword 作实证分析,这三个软件都有较长的发展历史,版本数较多,分别由 Java、C 和 C++ 编写,具有代表性.通过对上述 3 个软件系统在不断重构中其结构的核数演化分析,可以进一步理解软件结构尤其是层次结构的变化趋势.

上述三个软件系统其结构的核数演化趋势如图 1 所示,分别统计了 Eclipse 从 2.0.1 到 3.4.2 期间、MySQL 从 3.23.51 到 6.0.6 期间,以及 Abiword 从 1.2.0 到 2.6.8 期间具有里程碑意义及重要版本的源代码结构核数值.其中,横坐标表示选取软件版本在同一时间序列下的发行次序,纵坐标表示其源代码结构的核数.

从图 1 可以明显看出,在上述三个软件系统的重构中,其源代码结构的核数变化不大,MySQL 的核数一直在 5 ~ 6 之间,Eclipse 一直稳定在 4-5 之间,而 Abiword 的核数一直为 4,非常稳定.软件结构核数的稳定,也使得软件结构的层次结构相对稳定,软件具有相对稳定的层次结构,为我们从不变中寻找变化、研究其层次性提供了便利.

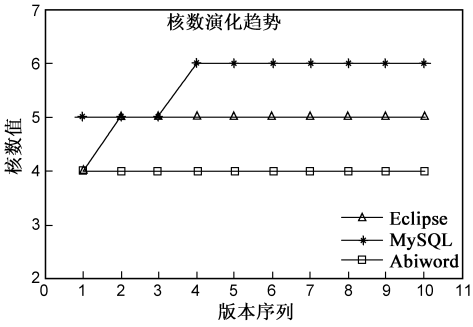



图1　Eclipse、MySQL与Abiword的核数演化

## 4　软件结构中节点的核数

### 4.1　节点核数的分布

得到软件结构所有节点的核数之后,进一步考察其节点核数 $C$ 的分布特征.经过统计分析发现,软件结构的节点核数分布与节点度分布相似,即大部分节点的核数都很小,少数的节点具有较大的核数.以 Eclipse 3.4.2 版本、MySQL 6.0.6 和 Abiword 2.2.7 版本为例,其节点核数分布在对数坐标系下如图 2 所示,其中横坐标为核数,纵坐标为该核数分布的概率密度 $P(C)$.

从图 2 可以看出,三者的核数分布均呈现出无尺度分布特征,核数分布不均匀,核数不超过 2 的节点占了节点总数的大多数,高核数节点只占节点总数的很小一部分.通过三个软件结构其节点的核数分布,大致可以看到从低核数节点数目到高核数节点数目逐渐下降.具有高核数值的节点,在软件整体结构以及演化过程中,具有相对比较重要的作用.

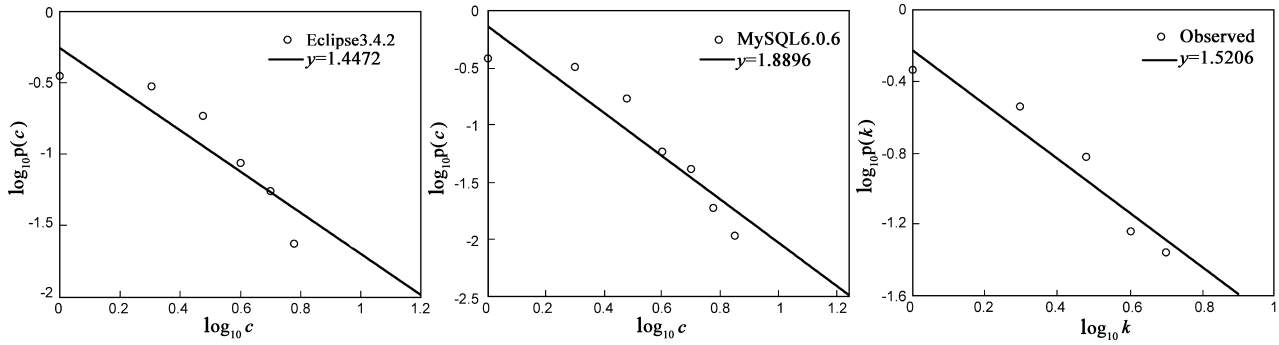


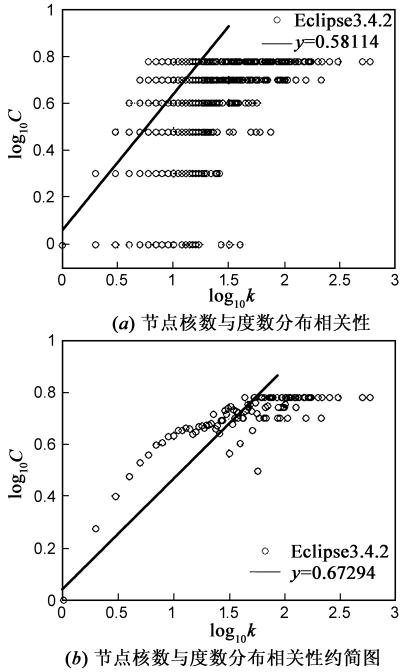

图2　Eclipse 3.2.2版本、MySQL 6.0.6版本和Abiword 2.2.7版本的核数分布

### 4.2　节点的核数与度数相关性

软件结构的节点核数分布与节点度分布相似,接下来讨论软件结构的节点核数与度值的相关性.以 Eclipse 3.4.2 版本为例,其节点核数与度值相关性如图 3 所示,其中,在对数坐标系下,横坐标为节点的度值 $k$,纵坐标为节点的核数 $C$.

从图 3($a$)可以看到,尽管软件结构的节点核数分布与度分布较为相似,但是并非所有度值高的节点其核数就一定很高,也存在一部分高度值节点的核数较小,这种特征符合 $k$-核和核数的定义.特殊网络结构如星形网络,其中心节点的度值可能很大,但由于其连接的所有度值为 1 的节点的核数都为 0,其中心节点的核数也仅为 1.

另外,软件结构中各核内的度值较为分散,需要适当约简数据以便更清晰的表达节点和核数与度值相关性.具体来说,当节点具有相同度值时,取其核数的平均值作为纵坐标,如图 3($b$)所示.由约简图可以看到,Eclipse 3.2.2 版本的结构中,低度值节点的核数与其度值具有一定的正相关性;但是在节点度较高的区域,核




($a$) 节点核数与度数分布相关性

($b$) 节点核数与度数分布相关性约简图

图3　Eclipse 3.4.2版本节点核数与度数相关性

数不再随节点度值增大而增大.高度值节点主要集中在高核区域内,但是也有一部分高度值节点分布在低核区域,再次说明并非所有高度值都具有高核数.同时,以 $k$-核分析软件结构其层次性更直观,比以节点度分析更有效.

## 5 软件结构分层分析

Eclipse 是一个开放源代码的、基于 Java 的可扩展开发平台,自从正式发布之后得到了广泛应用,并深受好评,是开源软件的优秀代表.鉴于工作量和篇幅的原因,下面的工作我们都以 Eclipse 软件作为实证分析.本节以 3.2 小节选取的 Eclipse 软件的一个切片—3.4.2 版本作为实证研究,对软件的层次性进行度量与分析.

### 5.1 层次划分

根据 $k$-核的定义,网络的 $k$-核是反复移除所有度值小于等于 $k$ 的节点及其与之连接的边后所剩余的节点和边的集合,被移除的节点以及这些节点之间的连接就构成了($k$-1)-层.这样就可以首先从网络拓扑中反复地去除度为 1 的节点,直到剩余的所有节点度值都大于 1,于是就得到 0-层;然后,再反复地移去度值为 2 的节点,直到网络拓扑中所有节点的度值都大于 2,得到 1-层;依此类推,直到网络拓扑中再无节点为止,即得到网络结构的最高层,从而从低层到高层将网络拓扑结构层层分解,算法描述如下:

(1)计算网络中所有节点的度值,并为每个节点构造一个列表用来存放该节点的邻居节点;(2)令 $i=1$;(3)从网络中移除度值 $k \leqslant i$ 的所有节点及其与之连接的边,将移除的节点置于队列 $L_i$ 中,并将它们的邻居节点的度值减 1;(4)重复步骤(3),直到再无节点可被移除,此时当前队列 $L_i$ 所有节点核数为 $i$-1,队列 $L_i$ 中的节点数计作 $l_i$;(5)如果剩余网络不为空,则令 $i=i+1$,执行步骤(3);(6)如果剩余网络为空,则 $i$-1 为网络的核数,队列 $L$ 中的每个元素 $L_i$ 中存放的就是($i$-1)-层中的节点集 $N$;(7)$L_i$ 中节点集以及连接 $L_i$ 中任意两节点的边,构成了($i$-1)-层网络结构.

这样就得到了软件系统的层次集合 $L:\{L_i\}, i=1, \cdots k\text{-}core_{\max}$;及每层节点集合 $N:\{n_i\}, i=1, \cdots l_i$.

图 4 统计了 Eclipse 3.4.2 版本各层节点数与边数.可以看到,从最高层到最低层,层内节点数目近似呈指数减少,呈现出“金字塔”状层级结构.同时,可以看到各层边的数目并不随着节点数目的增加而增加,总体上变化不大,考虑到各层节点数从高层到低层逐层增多,边数从高层到低层是逐渐稀疏化的.

### 5.2 层间关系

统计 Eclipse 3.4.2 版本各层节点分别与其它层节点及层内节点的连接数目,如表 2 所示,可以看到每一层都与自身所在层和其它层存在连接关系,并且连接数目不尽相同.然后进一步计算得到与各层连接数所占的比例,统计结果如表 3 所示.可以看到,层内节点的连接数所占比例普遍要比与其它层连接要大,这一方面说明层内节点存在更强的协作关系,另一方面也验证了层次划分方法的合理性.此外,当前层节点与相邻层节点的连接数所占比例相比其它层也普遍要大一些,说明软件结构各相邻层之间存在着相对较密切的联系,易于结构的层次控制.

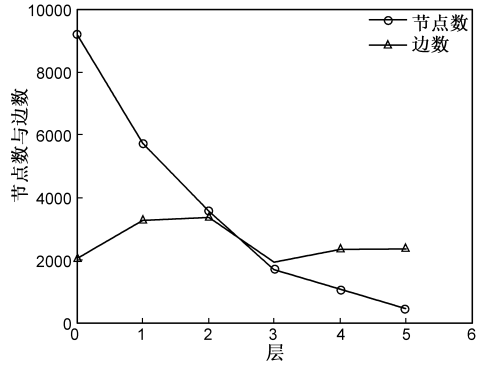



**图4 Eclipse 3.4.2版本各层节点数与边数**

**表 2** Eclipse **3.4.2** 每层节点分别与各层节点连接数统计

| 层 | 0-层 | 1-层 | 2-层 | 3-层 | 4-层 | 5-层 |
|---|---|---|---|---|---|---|
| 0-层 | 2035 | 1642 | 1001 | 612 | 740 | 521 |
| 1-层 | 1642 | 3253 | 2513 | 1331 | 1692 | 1704 |
| 2-层 | 1001 | 2513 | 3361 | 1738 | 2144 | 2269 |
| 3-层 | 612 | 1331 | 1738 | 1921 | 1891 | 2257 |
| 4-层 | 740 | 1692 | 2144 | 1891 | 2322 | 2294 |
| 5-层 | 521 | 1704 | 2269 | 2257 | 2294 | 2341 |

**表 3** Eclipse **3.4.2** 每层节点分别与各层节点连接比例统计

| 层 | 0-层 | 1-层 | 2-层 | 3-层 | 4-层 | 5-层 | 总计 |
|---|---|---|---|---|---|---|---|
| 0-层 | 31.06% | 25.06% | 15.28% | 9.34% | 11.30% | 7.95% | 100% |
| 1-层 | 13.53% | 26.81% | 20.71% | 10.97% | 13.94% | 14.04% | 100% |
| 2-层 | 7.68% | 19.29% | 25.80% | 13.34% | 16.46% | 17.42% | 100% |
| 3-层 | 6.28% | 13.65% | 17.83% | 19.70% | 19.39% | 23.15% | 100% |
| 4-层 | 6.68% | 15.27% | 19.34% | 17.06% | 20.95% | 20.70% | 100% |
| 5-层 | 4.58% | 14.97% | 19.93% | 19.82% | 20.15% | 20.56% | 100% |

**表 4** Eclipse **3.4.2** 每层节点分别与各层节点的加权连接数统计

| 层 | 0-层 | 1-层 | 2-层 | 3-层 | 4-层 | 5-层 | 总计 |
|---|---|---|---|---|---|---|---|
| 0-层 | 73.29 | 94.64 | 93.12 | 121.04 | 233.67 | 384.24 | 1000 |
| 1-层 | 23.33 | 73.97 | 92.22 | 103.86 | 210.80 | 495.82 | 1000 |
| 2-层 | 11.31 | 45.44 | 98.07 | 107.84 | 212.38 | 524.96 | 1000 |
| 3-层 | 7.59 | 26.44 | 55.71 | 130.92 | 205.76 | 573.58 | 1000 |
| 4-层 | 8.53 | 31.23 | 63.83 | 119.79 | 234.83 | 541.79 | 1000 |
| 5-层 | 5.82 | 30.45 | 65.44 | 138.40 | 224.59 | 535.30 | 1000 |

另外,从图 4 可知,各层间的节点数是由外层向内层逐层减少的,考虑到节点数因素,定义各层节点间加权连接数 $W$,表示当前层与各层节点联系的紧密程度,定义如下:

**定义 3** 加权连接数(Weighted Connection):指当前

层的节点与某一层的节点间连接数与后者节点数的比值,占当前层与所有各层连接数与后者节点数的比值只和的比例,公式为:

$$w_{i,j}=\frac{\frac{e_{i,j}}{l_j}}{\sum_{i=0,j=0}^{k-core_{\max}}\frac{e_{i,j}}{l_j}}\times 1000 \tag{1}$$

其中,$k-core_{\max}$表示软件结构的核数,$i,j=0,1,\cdots,k-core_{\max}$,$e_{i,j}$表示核数为 $i$ 的当前层节点与核数为 $j$ 所在层节点的连接数目,$l_j$ 表示核数为 $j$-层的节点数目.表 4 统计了 Eclipse 3.4.2 版本各层节点的加权连接数(为了方便数据处理,将所有值乘以 1000),可以看到:

(1)对各层所受的影响力来说,各层受其所在层节点的影响,比受其它层对该层节点的影响普遍要强,各层内部节点间联系较为紧密.

(2)对每层与各层的联系紧密程度来说,由最低层向最高层,该层与各层节点的联系逐渐增强,体现出"阶级"特征的层级控制关系.

同时,通过表 2 和表 3 统计数据也可以看出,无论是每层节点间的内部连接还是与各层的外部连接均不尽相同,核数较高的层和核数较低的层存在明显区别.这里将核数大于 $k/2$ 的视为高层区域,小于等于 $k/2$ 的视为低层区域.可以看到,对 Eclipse3.4.2 版本而言,高层区域边数/节点数的值大于 1,低层区域边数/节点数的值小于 1.

### 5.3　高层区域与低层区域比较

**定义 4**　度分布(Degree Distribution):度分布指的是网络中节点的度的分布情况,用分布函数 $P(k)$来描述.$P(k)$是指随机选定一个节点 $i$,恰好使得 $k_i=k$ 的概率.节点度呈幂律分布,是无尺度网络区别于随机网络和规则网络最重要的特征[19].度分布可以被用来表征系统结构的不均匀性,定性分析结构信息的不确定性.在度分布图中,度分布的幂指数 $r$ 描述了无尺度网络度分布的均衡程度[20].幂指数 $r$ 越大,度分布的区间越集中;幂指数 $r$ 越小,度分布的区间越发散,网络度分布越"不均匀".本文采用 pdf-degree 幂律分布来表征节点的度分布,即节点度 degree 的概率密度 pdf(Probability density function)为 degree 的负幂指数函数.对于 pdf-degree 有 $P(k)\sim k^{-\gamma}$,其中 $k$ 为度值,$P(k)$为度值为 $k$ 的节点出现的概率,在对数坐标系下,有 $\log_{10}P(k)\sim(-\gamma)\log_{10}k$.

图 5 描述了在对数坐标系下的 Eclipse 3.4.2 各层节点的 pdf-degree 分布,可以看到,各层节点 pdf-degree 分布具有幂律分布特征,说明由各层节点构成的网络结构仍然是无尺度网络.但是,其拟合效果却是由低层到高层各层逐渐降低的;并且其幂律分布系数也由低核到高核逐渐降低,即由低层到高层各层节点度分布逐渐趋向"不均匀",这是与节点核数与度数的相关性分析结果是一致的.

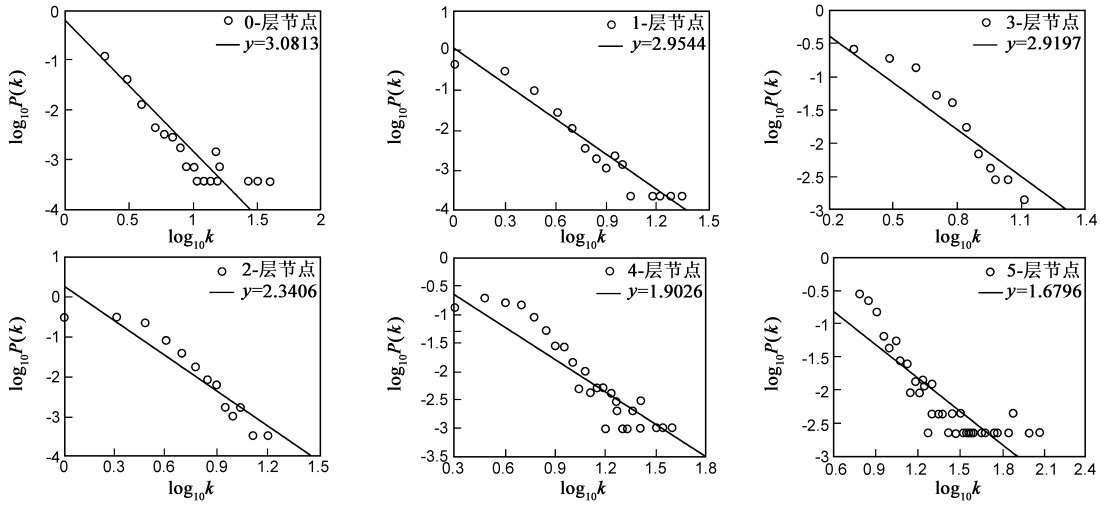



**图5　Eclipse 3.4.2版本各层节点的pdf-degree分布**

无尺度网络的聚集性是一个非常重要的特征,通常用聚集系数来表征,反映了网络中节点间的内聚程度以及组织分层的趋势.

**定义 5**　聚集系数(Clustering Coefficient):节点 $i$ 的聚集系数,也称局部聚集系数,为该节点所有邻居节点之间连接数目占可能的最大连接数目的比例.其公式为[21]:

$$cc_i=\frac{2E_i}{n_i(n_i-1)} \tag{2}$$

其中 $n_i$ 为节点 $i$ 的邻居节点的个数,$E_i$ 为这 $n_i$ 个节点之间存在的连接的数目.

网络的平均聚集系数,简称聚集系数,是节点总数为 $N$ 的网络中所有节点聚集系数的平均值,其公式为:

$$Clu = \frac{1}{N}\sum_{i=1}^{N} cc_i \quad (3)$$

网络拓扑的聚集特性,主要由于低度值节点倾向与高度值节点连接的异配性质所导致.而高度值节点之间的连接又比较紧密,所以网络整体上表现为节点拥有紧密相连的邻居.统计表明,大部分真实大规模网络中的节点倾向于聚集在一起,对于节点规模为 $N$ 的网络,其聚集系数远大于 $O(N^{-1})$,具有高聚集特性.

各层节点的聚集系数如图 6 所示,由低层到高层,各层节点聚集系数逐渐增大.说明由低层到高层,各层节点体现出更强的聚集性,反映了软件系统中组成元素的内聚程度从低层到高层逐层增强,聚集程度的层次化趋势较为明显,符合软件工程中"高内聚、低耦合"原则.

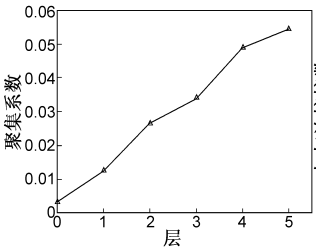



图6 Eclipse3.4.2版本各层节点和聚集系数

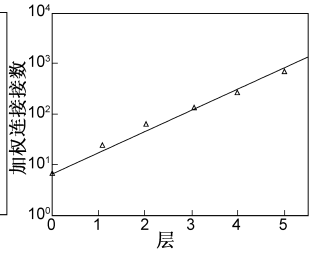



图7 最高层节点与各层加权连接数

统计 Eclipse 3.4.2 版本每层节点数 $N$、每层所有节点的度值之和 $K$,然后计算每次所有节点度值之和占软件结构所有节点度值之和的比例 $R$,统计结果如表 5 所示.可以看到,低层的节点数占了节点总数的大多数,高层的节点数所占比例较少.但是,4-层和 5-层的节点占节点总数的 6.91%,其度值之和却超过了所有节点度值之和的 1/3,体现出明显的"富人俱乐部"特征.在软件结构中,核数高的节点通常除了度数很高之外,同时也间接的和各层中大量的节点有着依赖关系.最高层的节点数尽管只有 450,其节点度值之和却占了所有节点度值之和的 17.34%,最高层节点的相当一部分边连接于软件结构中各层的节点,说明最高层在结构中影响力巨大.

因此可知,低层区域其节点度的无尺度分布特征更明显,高层区域其节点的聚集性更明显.

表 5 Eclipse 3.4.2 结构分层度值及所占比例统计

| 层 | 节点数 $N$ | 度值之和 $K$ | 度值之和占总度值比例 $R$ |
|---|---|---|---|
| 0-层 | 9215 | 8586 | 10.85% |
| 1-层 | 5758 | 15388 | 19.44% |
| 2-层 | 3568 | 16387 | 20.70% |
| 3-层 | 1678 | 11671 | 14.74% |
| 4-层 | 1051 | 13405 | 16.93% |
| 5-层 | 450 | 13727 | 17.34% |

### 5.4 最高层分析

从上节表 5 可以看到,最高层尽管节点数较少,但是度值所占比例却很大,在软件整体结构中处于支配地位.而且层内节点间协作非常密切,这些核心节点间的协作为软件系统搭建了核心框架,为软件系统其它模块提供基础服务.

通过表 4 中最高层节点与各层节点的连接数 $E$,可以看到最高层节点与每一层都存在连接关系,说明最高层对各层存在直接影响;还可以进一步看到,最高层对各层节点的连接数由外层向内层逐层增加,说明最高层对各层的控制由外向内逐层增强.最高层与各层间的加权连接数在坐标系中描点,如图 7 所示,其中纵坐标取对数.可以看到,各点经过拟合大致呈线性增长,也就是说,最高层对各层的影响力由外层向内层呈指数增长.

另一方面,最高层是软件结构核心框架,其对软件整体结构的影响要大于其它层节点,因为这些节点需要更多的与各层节点进行协作和数据交互,也就承担了更多的工作责任.同时最高层节点出现错误对整体结构的影响也极大,这也就是其脆弱性的所在.所以,保证最高层节点的正常工作并提高其鲁棒性是非常重要的.

## 6 软件结构最高层演化

### 6.1 最高层拓扑

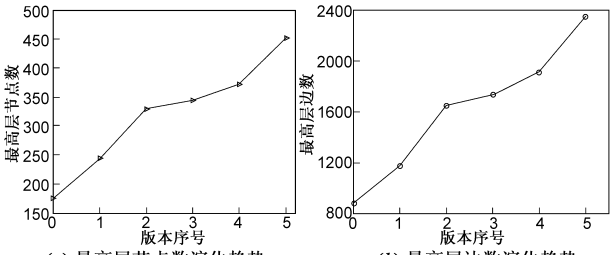
目前,软件系统结构越来越复杂化,往往超出了开发人员的控制,因此,理解和掌握软件演化趋势将是控制软件开发的关键因素[22].作为软件系统的核心框架结构,最高层结构演化的研究更应当得到重视.

对大规模软件结构最高层的演化研究需要选取在同一时间序列下的若干版本,除考虑时间跨度外,主要应考虑软件发布版本的主版本号和次版本号更新的因素.本节依然选取 Eclipse 软件作为实证分析,这里选取 Eclipse 软件的 2.1.3、3.0.1、3.1.1、3.2.2、3.3.1.1 和 3.4.2 五个版本的最高层结构进行演化分析.

通过前文讨论可知,最高层在网络结构中的影响力巨大.进一步分析上述 Eclipse 各版本的结构,其高核数节点主要有两类,一类是在软件开发中复用次数较多的节点,这些节点的度值一般比较大,通常是软件系统的基础模块,比如包 org.eclipse.swt.widgets、org.eclipse.swt.graphics、org.eclipse.ui.internal,且 org.eclipse.swt.widgets 包中几乎所有的类都集中在最高层,包括 Button、Text、Label、Control 等基础类,是 Eclipse 最基础的组件;另一类,则是连接软件结构不同层间的节点,这些节点可能度值并不大,但是在结构中影响力巨大,承担着层间数据的传递,如 org.eclipse.ui.dialogs.SaveAsDialog 类.这些节点在结构演化中始终处于最高层,构成了软件结构的核心框架,并且保持稳定.

### 6.2　最高层结构规模

网络结构的规模与网络中节点数与边数大小密切相关.图 8 分别统计了 Eclipse 个版本结构的节点数与边数的演化情况,可以看到,随着其整体结构演化,最高层节点数和边数是稳定增加的.进一步统计可知,每个版本的最高层新增节点均来自于上一版本的次高层(核数大小仅小于最高层节点所在层).另外,各版本最高层节点数和边数在整体结构中所占比例如表 6 所示,最高层节点数所占比例均在 2%左右,边数所占比例维持在 5%-6%,均变化不大,说明最高层规模始终随着整体结构规模的变化而变化.




(*a*) 最高层节点数演化趋势　(*b*) 最高层边数演化趋势

**图8　最高层节点数与边数演化趋势**

**表 6　最高层节点数与边数占整体结构比例演化趋势**

| Eclipse 版本 | 2.1.3 | 3.0.1 | 3.1.1 | 3.2.2 | 3.3.1.1 | 3.4.2 |
|---|---|---|---|---|---|---|
| 最高层节点数所占比例 | 1.96% | 1.93% | 2.21% | 1.94% | 1.93% | 2.12% |
| 最高层边数所占比例 | 5.39% | 5.04% | 5.97% | 5.29% | 5.33% | 5.91% |

### 6.3　最高层节点平均度

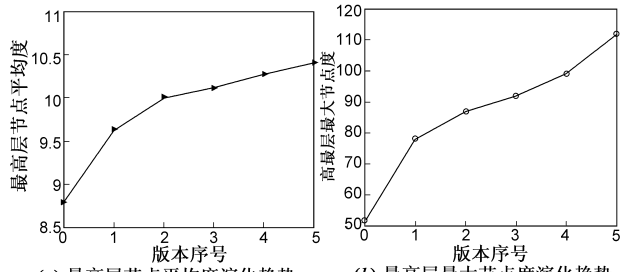
**定义 6**　节点平均度(Average of Degrees):网络中所有节点的度的平均值称为网络的节点平均度,记为 $< k >$:

$$< k > = \frac{1}{N} \sum_{i=1}^{N} k_i \tag{4}$$

其中,$N$ 表示节点总数,$k_i$ 表示第 $i$ 个节点的度值.节点平均度反映了网络拓扑的平均连通性能,这个值越大,表明网络拓扑的平均连通性能越好.

软件系统其最高层结构不存在度值为 0 的节点,是一个完全连通的图.最高层是软件系统的核心结构,其节点间具有密切的协作关系,为软件系统提供基础服务.对软件系统来说,节点平均度 $< k >$ 能够表征软件复用的程度.这里,以节点平均度 $< k >$ 表征最高层的复用程度.

Eclipse 在结构演化中各版本的最高层节点平均度 $< k >$ 如图 9($a$)所示,可以看到 $< k >$ 值呈稳定增长的趋势,说明最高层在软件系统重构中被越来越多的复用.最高层节点作为系统的核心结构,其 $< k >$ 值增长更多的来自软件系统重构过程中的模块复用,这既符合软件工程思想,也符合复杂网络演化中的"偏好依附"原则,即新加入的节点总是优先连接度值较大的节点.图 9($b$)统计了 Eclipse 各版本最高层中最大节点度值,可以看到最大节点度呈持续增大的趋势,这正是软件复用的结果.




(*a*) 最高层节点平均度演化趋势　(*b*) 最高层最大节点度演化趋势

**图9　最高层节点平均度及最大节点度值演化趋势**

### 6.4　最高层节点度分布

我们已经知道,Eclipse 3.4.2 版本的最高层结构是一种无尺度网络,其节点度呈幂律分布特征,这里依然采用 pdf-degree 幂律分布来表征最高层节点的度分布,各版本的 pdf-degree 分布如图 10 所示.可以看到,各版本最高层结构均呈现出无尺度特征,并且其幂律分布系数 $r$ 随着软件演化呈下降趋势,说明在结构演化过程中最高层节点度分布趋向于"不均匀".我们知道,对复杂网络来说,对于随机节点的实效,无尺度网络比随机网络更加健壮,体现出宏观结构的稳定性.而且,对软件系统开发来说,除去一些预算、平台迁移等不确定因素,为了获得更好的协作能力和稳定的结构,节点度分布趋向"不均匀".因此,对最高层来说,其演化过程中结构逐渐趋向于稳定.

### 6.5　最高层的小世界网络特征

网络的小世界特征与拓扑结构的聚集系数和平均最短路径长度两个因素密切相关[23],下面是最短平均路径长度的定义.

**定义 7**　平均最短路径长度(Average Shortest Path Length):网络中两个节点 $n_i$ 和 $n_j$ 之间的距离 $d_{ij}$ 定义为连接这两个节点的最短路径上的边数.节点数为 $N$ 的网络的平均最短路径长度 $D$ 定义为任意两个节点间距离的平均值,公式为:

$$D = \frac{2}{N(N+1)} \sum_{i \geqslant j} d_{ij} \tag{5}$$

如前文所述,聚集系数反映了网络拓扑中节点之间的内聚程度以及层次化的趋势.对于节点规模为 $N$ 的网络,如果聚集系数远远大于 $O(N^{-1})$,说明网络具有高聚集特性.Eclipse 各版本最高层结构的聚集系数 $Clu$ 和最短路径长度 $D$ 统计如图 11 所示.

从图 11($a$)可以看出,各版本最高层聚集系数 $Clu$ 值远大于 $O(N^{-1})$,说明最高层结构具有高聚集特性;其次最高层的聚集系数一直明显大于整体结构的聚集系数,说明最高层具有比整体结构更高的聚集特性;而

且,聚集系数 *Clu* 值总体上呈逐渐增大的趋势,说明随着软件系统重构,最高层结构的聚集性逐渐提高,符合软件工程中"高内聚、低耦合"的思想.

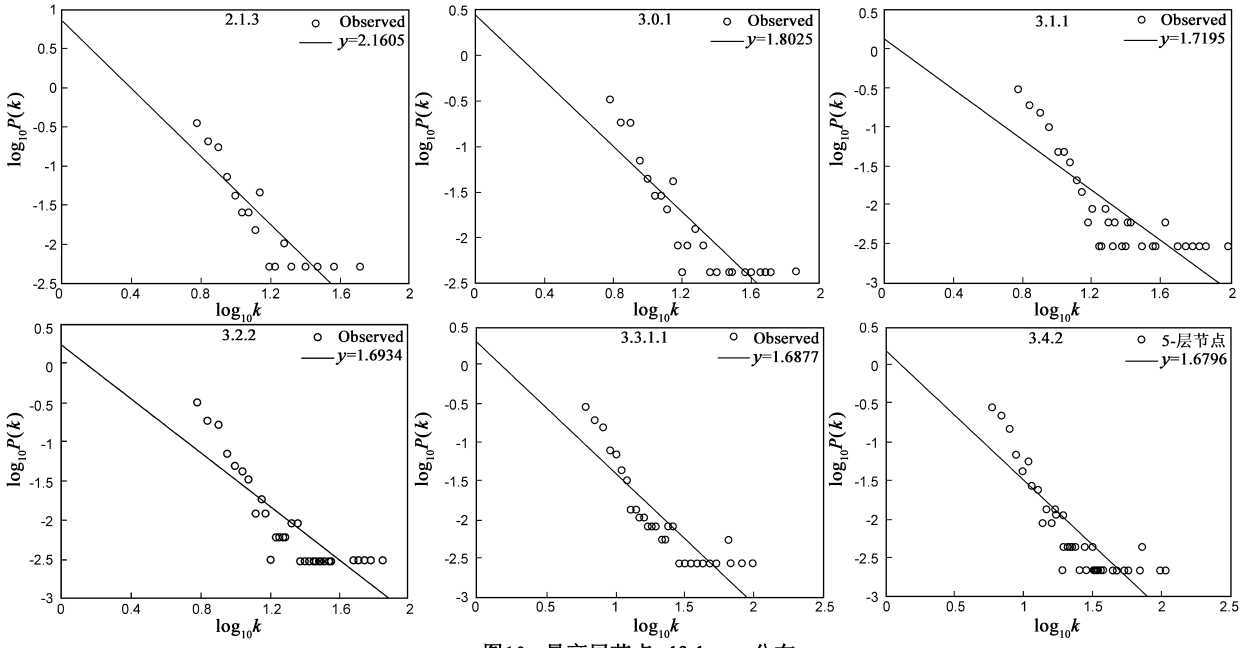
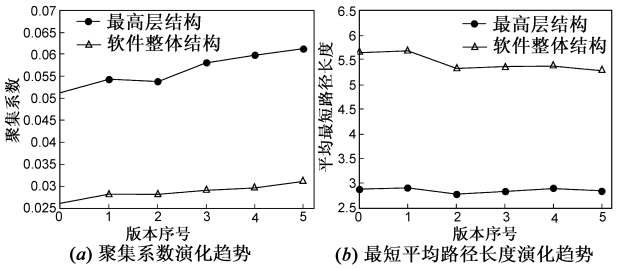



**图10 最高层节点pdf-degree分布**




(*a*) 聚集系数演化趋势 (*b*) 最短平均路径长度演化趋势

**图11 最高层与整体结构的聚集系数与最短平均径长度演化**

另外,从图 11(*b*)可以看到,最高层的平均最短路径长度 *D* 的变化幅度并不大,一直在 2.8 左右,明显低于整体结构,这与其高聚集特性是相对应的.大量统计数据表明,较大的聚集系数和较小的平均最短路径长度成为很多软件系统的显著特征,而且具有小世界网络特征.比较最高层与整体结构,可知最高层相比整体结构具有更大的聚集系数 *Clu* 和更短的平均最短路径长度 *D*,小世界网络特征更加明显.

## 7 结论

对于大规模软件来说,软件的架构比起对程序的算法设计等其重要性变得日益明显,对其结构层次性的研究对软件体系结构的控制具有重要意义,同时有助于更加深入的理解软件系统的内部组织机理.本文利用 *k*-核对大规模软件结构层次性的研究分析,对其及系统结构的约简提供了有力支持,同时也有助于我们进一步认识其结构的复杂性本质.

本文首先根据大量大规模开源软件结构的核数统计,发现大规模软件具有相对扁平的层次结构,并且软件结构的核数随着软件演化保持稳定.然后统计了软件结构中节点的核数分布,分析了节点的核数与度值相关性,认为以 *k*-核分析软件层次结构更加直观.在此基础上,通过 *k*-核定义对软件结构进行了层次划分,通过各层层内与层间连接,发现每层层内节点存在更强的协作关系,同时相邻层间的联系相对较为密切,易于结构的层次控制;通过各层层内与层间的加权连接数统计,发现每层与各层的联系,由最底层到最高层逐层增强.而且,对各层来说,低层区域各层节点度分布无尺度特征更明显,高层区域各层聚集性更明显;其中,最高层作为软件结构的核心框架,同时体现出对各层的巨大影响力和自身的脆弱性.最后,对最高层的演化进行了量化分析,发现最高层节点规模逐渐增大,节点平均度的提高体现出模块复用程度的提高,节点度分布体现出结构逐渐趋向"不均匀",节点间聚集性也在不断提高,符合软件工程中"高内聚、低耦合"的原则,并且结构趋向于有序化.并且,我们在对其它大规模软件结构的研究中也得到了相同的结论.

目前,对大规模软件的层次性研究还没有进入到一个很深入的阶段,各层之间的协作与控制中蕴含的规律还有待进一步研究.另外,对大规模软件来说,当最高层的规模等因素达到一定程度时,必然会从中孕育出更核心的局部结构,从而导致最高层结构发生一次剧变,这一过程及其对软件整体结构的影响是下一步工作需要研究的重点.

**参考文献:**

[1] Arun S, Rajesh K, Grover PS. Estimation of quality for software components: an empirical approach[J]. ACM SIGSOFT Software Engineering Notes, 2008, 33(6): 1 – 10.

[2] Nachiappan N, Brendan M, Victor B. The influence of organizational structure on software quality: an empirical case study [A]. Proceedings of the 30th International Conference on Software Engineering[C]. New York, USA: ACM, 2008. 521 – 530.

[3] Robert B, Elaine W, Thomas O. Predicting the location and number of faults in large software systems[J]. IEEE Transactions on Software Engineering Archive, 2005, 31(4): 340 – 355.

[4] Chen JY, Lu JF. New metric for object-oriented design[J]. Information and Software Technology, 1993, 35(4): 232 – 240.

[5] Chidamber S R, Kemerer C F. A metrics suite for object oriented design[J]. IEEE Transactions on Software Engineering, 1994, 20(6): 476 – 493.

[6] Brito F, Abreu E. The MOOD metric set[A]. In: Proceedings of the ECOOP'95 Workshop on Metrics[C]. Aarhus, Denmark: Springer, 1995.

[7] Jenkins S, Kirk SR. Software architecture graphs as complex networks: a novel partitioning scheme to measure stability and evolution[J]. Information Sciences, 2007, 177(12): 2587 – 2601.

[8] 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 ZY, et al. Software complexity metrics based on complex networks[J]. Aeta Electronica Sinica, 2006, 34(12A): 2371 – 2375. (in Chinese)

[9] Cai W, Zhao H, Zhang HH, et al. Static structural complexity metrics for large-scale software[J]. Special Issue on Software Engineering and Complex Networks of Dynamics of Continuous, Discrete and Impulsive Systems Series B, 2007, 14(S6): 12 – 17.

[10] Liu J, Lu JH, He KQ. Characterizing the structure quality of general complex software networks[J]. International Journal of Bifurcation and Chaos, 2008, 18(2): 605 – 613.

[11] Tomasz L. Size and connectivity of the k-core of a random graph[J]. Discrete Mathematics, 1991, 91(1): 61 – 68.

[12] Janson S, Luczak M J. Asymptotic normality of the k-core in random graphs[J]. Annals of applied probability, 2008, 18(3): 1085 – 1137.

[13] Oliver R. The k-core and branching processes[J]. Combinatorics, Probability and Computing, 2008, 17(1): 111 – 136.

[14] Zhang HH, Zhao H, Cai W, ZhaoM, Luo GL. A qualitative method for analysis the structure of software systems based on k-core[J]. Special Issue on Software Engineering and Complex Networks of Dynamics of Continuous, Discrete and Impulsive Systems Series B, 2007, 14(S6): 18 – 24.

[15] Zhang HH, Zhao H, Cai W, Zhao M, Luo GL. Visualization and cognition of large-scale software structure using the k-core analysis[A]. Intelligent Information Hiding and Multimedia Signal Processing[C]. Washington, D. C., USA: IEEE, 2008. 954 – 957.

[16] Ma YT, He KQ, Du DH. A qualitative method for measuring the structural complexity of software systems based on complex networks[A]. In: 12th Asia-Pacific Software Engineering Conference[C]. Washington, D. C., USA: IEEE, 2005. 257 – 263.

[17] 张昕,赵海,王莉菲,等.AS 级 Internet 拓扑分析[J].通信学报,2008,29(7):50 – 61.
Zhang X, Zhao H, Wang LF, et al. Analysis on the internet AS-level topology[J]. Journal on Communications, 2008, 29(7): 50 – 61. (in Chinese)

[18] Lee Y, Yang J, Chang KH. Metrics and evolution in open source software[A]. 7th International Conference on Quality Software [C], Washington, D.C., USA: IEEE, 2007. 191 – 197.

[19] Albert R, Barabási AL. Statistical mechanics of complex networks[J]. Rev Mod Phys, 2002, 74(1): 47 – 97.

[20] Newman MEJ. The structure and function of complex network [J]s. SIAM Rev, 2003, 45(2): 167 – 256.

[21] Dorogovtsev SN. Clustering of correlated networks[J]. Physical Review E, 2004, 69(2): 027104.

[22] Myers C. R. Software systems as complex networks: structure, function, and evolvability of software collaboration graphs[J]. Physics Review E, 2003, 68(4): 1 – 15.

[23] Strogatz S, Watts D. Collective dynamics of 'small world' networks[J]. Nature, 1998, 393(6684): 440 – 442.

**作者简介:**

**李　辉**　男,1983 年 1 月出生于山东省淄博市,东北大学博士研究生,主要研究方向为大规模软件度量.
E-mail: lih2002@126.com

**赵　海**　男,1959 年 3 月出生于辽宁省沈阳市,东北大学教授,博士生导师,主要研究方向为普适计算、复杂网络、软件工程、嵌入式技术、传感器网络等.
E-mail: zhaoh@mail.neu.edu.cn