(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 104133766 B
(45)授权公告日 2017.01.04

(21)申请号 201410330742.0

(22)申请日 2014.07.11

(65)同一申请的已公布的文献号
申请公布号 CN 104133766 A

(43)申请公布日 2014.11.05

(73)专利权人 西安交通大学
地址 710049 陕西省西安市咸宁西路28号

(72)发明人 郑庆华 李剑 王志文 屈宇 刘烃

(74)专利代理机构 西安通大专利代理有限责任公司 61200
代理人 陆万寿

(51)Int.Cl.
G06F 11/36(2006.01)

(56)对比文件
JP 特开2005-326953 A,2005.11.24,
CN 102594909 A,2012.07.18,
侯雨桥等.基于调用结构的软件可生存性评估方法.《中南大学学报(自然科学版)》.2013,第44卷第443-448页.
陈国强等.基于进化多目标优化的复杂网络社团检测.《中国科技论文在线》.2012,第1-9页.
Qinghua Zheng等.A Novel Method on Software Structure Evaluation.《2011 IEEE 2nd International Conference on Software Engineering and Service Science》.2011,第251-254页.

审查员 李维

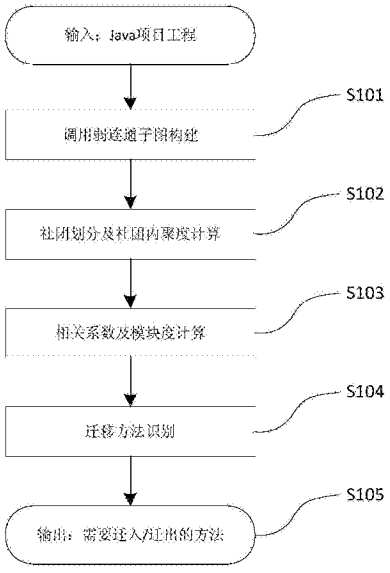
权利要求书2页 说明书8页 附图3页

(54)发明名称

基于多目标社团发现的软件系统可维护性评估与提升方法

(57)摘要

本发明提出了一种基于多目标社团发现的软件系统可维护性评估与提升方法,包括:1)构建最大弱连通子图;2)划分方法调用最大弱连通子图并计算每个类的不同方法社团内聚度;3)计算不同方法社团内聚度的斯皮尔曼相关系数,选取相关系数最大的两个社团划分算法来计算软件系统的模块度,评估软件系统的可维护性;4)基于每个类的方法社团内聚度,使用自适应的阈值过滤算法,过滤出一些内聚度较低的类,这些类中某些方法与其它类有紧密联系,迁移这些方法,从而提高类的内聚度和软件系统的模块度,提升软件系统的可维护性。本发明不仅提升了软件系统中类的内聚度,而且提升了整个软件系统的模块度,从而提升了软件系统的可维护性。



CN 104133766 B

1.基于多目标社团发现的软件系统可维护性评估与提升方法，其特征在于，包括以下步骤：

步骤S101：使用静态代码分析工具Understand和复杂网络分析包igraph，实现对Java软件系统方法调用的最大弱连通子图的生成；

步骤S102：运用四种基于不同目标的社团划分算法划分步骤S101所生成的最大弱连通子图，并计算每个类的方法社团内聚度；

步骤S103：计算四种社团划分算法结果的斯皮尔曼相关系数，最后挑选出最大斯皮尔曼相关系数的两个算法来计算类内聚度和模块度；

步骤S104：使用阈值过滤算法，找出一些内聚度较低类中可以迁移的方法，输出这些迁入\迁出的方法，提高这些类的内聚度，提升了整体的模块度，从而提升了JAVA软件系统的可维护性。

2.根据权利要求1所述的基于多目标社团发现的软件系统可维护性评估与提升方法，其特征在于，步骤S101具体包括以下步骤：

步骤S201：对待分析的Java软件系统解析.java文件，获取项目中定义的方法列表；

步骤S202：利用静态代码分析软件Understand分析Java软件系统源代码，获取所有方法调用对；

步骤S203：根据步骤S201中获取到的Java软件系统中的方法列表以后，对比步骤S202获取到的所有方法调用对，去掉冗余方法，最终获得软件自身方法调用对；

步骤S204：根据软件自身方法调用对生成软件方法调用网络图；

步骤S205：利用复杂网络分析包igraph过滤掉软件方法调用网络图的孤立的节点，生成软件系统方法调用的最大弱连通子图。

3.根据权利要求1所述的基于多目标社团发现的软件系统可维护性评估与提升方法，其特征在于，步骤S102具体包括以下步骤：

步骤S301：运用开源的复杂网络分析包igraph中的四种社团划分算法对得到的方法调用的最大弱连通子图进行划分；所述四种社团划分算法为基于模块度优化、知识地图、多级分层和标签传播的社团划分算法；

步骤S302：根据步骤S301的划分结果统计出Java软件系统中每个类的所属社团编号；

步骤S303：获取每个类所属各个社团的方法数；

步骤S304：计算出每个类的四种社团划分算法对应的方法社团内聚度；

方法社团内聚度的定义为：令某个类C共有m个方法，经过某个社团划分算法后，m个方法分布在N个社团中，第k个社团中分布有n_k个方法，k＝1，…，N，令n_1为n_k中的最大值，那么可以得到类C基于该社团划分算法的方法社团内聚度为n_1/m。

4.根据权利要求1所述的基于多目标社团发现的软件系统可维护性评估与提升方法，其特征在于，步骤S103具体包括以下步骤：

步骤S401：计算每两个社团划分算法之间的斯皮尔曼相关系数；

步骤S402：挑选6组斯皮尔曼相关系数中的最大值，记录这个最大值所对应的两种社团划分算法；

步骤S403：运用步骤S402中的两种社团划分算法计算JAVA软件系统的平均模块度和每个类的平均社团内聚度；

步骤S404:输出JAVA软件系统的平均模块度和其中每个类的平均社团内聚度。

5.根据权利要求1所述的基于多目标社团发现的软件系统可维护性评估与提升方法,其特征在于,步骤S104具体包括以下步骤:

步骤S501:判断社团内聚度的过滤阈值是否大于1,如果是则转入步骤S508,否则转入步骤S502;

步骤S502:在步骤S404输出的每个类的平均社团内聚度中查找一个社团内聚度大于过滤阈值的类C_1;

步骤S503:判断类C_1中是否存在方法在其他社团中,如果是则转入步骤S504,否则转入步骤S502查找符合条件的另一个类;

步骤S504:在其他社团中查找社团内聚度大于过滤阈值的类C_2;

步骤S505:判断步骤S503中的方法是否与类C_2具有调用关系,如果是则转入步骤S506,否则转入步骤S502;

步骤S506:保存所有满足条件的类C_1、C_2及步骤S503中的方法和方法数量,转入步骤S507;

步骤S507:过滤阈值递增0.05,转入步骤S501;

步骤S508:根据步骤S506中保存的记录中的方法数量列表,计算并保存每两个相邻阈值间方法数量差值,找出方法数量差值最大的两个相邻阈值,最后得到从两个阈值中较大的那个阈值处开始,满足要求的方法数量显著降低;

步骤S509:输出方法数量显著降低时所对应的过滤阈值下的所有类C_1、C_2以及相应的方法。

6.根据权利要求5所述的基于多目标社团发现的软件系统可维护性评估与提升方法,其特征在于,步骤S501中第一次设置的初始过滤阈值为0.5或0.6。

7.根据权利要求5所述的基于多目标社团发现的软件系统可维护性评估与提升方法,其特征在于,将步骤S509输出的类C_1中相应的方法迁入类C_2中,以提升整个JAVA软件系统的可维护性。

基于多目标社团发现的软件系统可维护性评估与提升方法

技术领域：

[0001] 本发明属于可信软件和软件测试交叉领域，特别涉及一种软件系统可维护性评估与提升方法。

背景技术：

[0002] 软件系统由众多模块组成，依靠模块间的相互调用完成各种复杂的功能，软件模块性的好坏将直接影响软件功能的实现、系统的运行效率以及整体稳定性。软件工程的目标是运用新技术和正确的管理方法来提高软件的质量和生存率，使软件系统具有更高的可维护性、可靠性和可重用性。

[0003] 面向对象技术采用数据抽象、封装、多态性、信息隐藏、重用等机制，增强了软件的可维护性，提高了生产效率。为了评估和提升面向对象软件设计的质量，研究人员提出了一些指标，这些指标从不同角度评估软件系统，比如内聚度、耦合度和复杂度。随着软件功能需求的日益增加，开发条件的不断改进，软件系统变得越来越复杂，软件系统已经被研究人员验证符合复杂网络的特征，在复杂网路中，一些节点内部的连接密度大于他们和网络中其他节点的连接密度，这样的结构被称为社团结构，软件系统的社团结构和软件的可维护性密切相关，社团发现算法能够很好的检测出复杂网络中的社团结构，社团结构可以用来计算软件系统模块度并评估软件系统可维护性。

[0004] 开发出高内聚和低耦合的软件系统被广泛认为是模块性比较好的软件结构，内聚就是模块之间的相互关系，高内聚意味着软件的各个模块内聚紧密联系，为此研究人员提出了一系列的类内聚度度量方法，比如：LCOM1，Coh，TCC，LCC，SCOM，CCM等等，这些方法基本上都建立在类中方法对类中实例变量使用和共享的基础上，只考虑了类中方法与变量之间、方法与方法之间的关系，然而这些度量方法并不能有效的对软件系统可维护性进行评估和提升。

发明内容：

[0005] 本发明的目的在于提出一种基于多目标社团发现的软件系统可维护性评估与提升方法，评价并提升软件系统的可维护性。本发明方法通过分析软件调用网络的社团结构及对应的模块度指标来评价并提升软件的模块性；对不同文件的代码进行重组，提高软件系统中类的内聚度，从而提升软件的模块性。

[0006] 为了实现上述目的，本发明采用如下技术方案：

[0007] 基于多目标社团发现的软件系统可维护性评估与提升方法，包括以下步骤：步骤S101：使用静态代码分析工具Understand和复杂网络分析包igraph，实现对Java软件系统方法调用的最大弱连通子图的生成；步骤S102：运用四种基于不同目标的社团划分算法划分步骤S101所生成的最大弱连通子图，并计算每个类的方法社团内聚度；步骤S103：计算四种社团划分算法结果的斯皮尔曼相关系数，最后挑选出最大斯皮尔曼相关系数的两个算法来计算类内聚度和模块度；步骤S104：使用阈值过滤算法，找出一些内聚度较低类中可以迁

移的方法，输出这些迁入\迁出的方法，提高这些类的内聚度，提升了整体的模块度，从而提升了JAVA软件系统的可维护性。

[0008] 优选的，步骤S101具体包括以下步骤：步骤S201：对待分析的Java软件系统解析.java文件，获取项目中定义的方法列表；步骤S202：利用静态代码分析软件Understand分析Java软件系统源代码，获取所有方法调用对；步骤S203：根据步骤S201中获取到的Java软件系统中的方法列表以后，对比步骤S202获取到的所有方法调用对，去掉冗余方法，最终获得软件自身方法调用对；步骤S204：根据软件自身方法调用对生成软件方法调用网络图；步骤S205：利用复杂网络分析包igraph过滤掉软件方法调用网络图的孤立的节点，生成软件系统方法调用的最大弱连通子图。

[0009] 优选的，步骤S102具体包括以下步骤：步骤S301：运用开源的复杂网络分析包iGraph中的四种社团划分算法对得到的方法调用的最大弱连通子图进行划分；所述四种社团划分算法为基于模块度优化(fg)、知识地图(im)、多级分层(ml)和标签传播(lp)的社团划分算法；步骤S302：根据步骤S301的划分结果统计出Java软件系统中每个类的所属社团编号；步骤S303：获取每个类所属各个社团的方法数；步骤S304：计算出每个类的四种社团划分算法对应的方法社团内聚度；方法社团内聚度的定义为：令某个类C共有m个方法，经过某个社团划分算法后，m个方法分布在N个社团中，第k个社团中分布有n_k个方法，k＝1，…，N，令n_1为n_k中的最大值，那么可以得到类C基于该社团划分算法的方法社团内聚度为n_1/m。

[0010] 优选的，步骤S103具体包括以下步骤：步骤S401：计算每两个社团划分算法之间的斯皮尔曼相关系数；步骤S402：挑选6组斯皮尔曼相关系数中的最大值，记录这这个最大值所对应的两种社团划分算法；步骤S403：运用步骤S402中的两种社团划分算法计算JAVA软件系统的平均模块度(和每个类的平均社团内聚度；步骤S404：输出JAVA软件系统的平均模块度和其中每个类的平均社团内聚度。

[0011] 优选的，步骤S104具体包括以下步骤：步骤S501：判断社团内聚度的过滤阈值是否大于1，如果是则转入步骤S508，否则转入步骤S502；步骤S502：在步骤S404输出的每个类的平均社团内聚度中查找一个社团内聚度大于过滤阈值的类C_1；步骤S503：判断类C_1中是否存在方法在其他社团中，如果是则转入步骤S504，否则转入步骤S502查找符合条件的另一个类；步骤S504：在其他社团中查找社团内聚度大于过滤阈值的类C_2；步骤S505：判断步骤S503中的方法是否与类C_2具有调用关系，如果是则转入步骤S506，否则转入步骤S502；步骤S506：保存所有满足条件的类C_1、C_2及步骤S503中的方法和方法数量；步骤S507：过滤阈值递增0.05；步骤S508：根据步骤S506中保存的记录中的方法数量列表，计算并保存每两个相邻阈值间方法数量差值，找出方法数量差值最大的两个相邻阈值，最后得到从两个阈值中较大的那个阈值处开始，满足要求的方法数量显著降低；步骤S509：输出方法数量显著降低时所对应的过滤阈值下的所有类C_1、C_2以及相应的方法。

[0012] 优选的，步骤S501中第一次设置的初始过滤阈值为0.5或0.6。

[0013] 优选的，将步骤S509输出的类C_1中相应的方法迁入类C_2中，以提升整个JAVA软件系统的可维护性。

[0014] 本发明一种基于多目标社团发现的软件系统可维护性评估与提升方法，包括如下步骤：

[0015] S101)基于词法语法、控制流分析技术，对待分析的目标代码进行扫描，获取软件

系统的方法列表，构建方法间的调用网络图，图的节点为方法，边为方法间的调用关系，然后对方法调用网络图进行预处理，去掉软件外部方法调用、所有的孤立节点，最后得到一个最大弱连通子图；

[0016] S102)分别运用基于模块度优化、知识地图、多级分层和标签传播的社团划分算法划分出最大弱连通子图的社团结构，基于不同算法划分出的社团结果计算出每个类的方法社团内聚度；

[0017] S103)分别计算不同方法社团内聚度的斯皮尔曼相关系数验证这些社团划分算法是否获得类似的结果或者各自独立于对方，然后选择相关系数最大的两个算法，基于这两个算法计算每个类方法社团内聚度平均值和每个软件系统的模块度，用模块度的值来评估软件系统的可维护性；

[0018] S104)基于每个类的方法社团内聚度，使用自适应的阈值过滤算法，过滤出一些内聚度较低的类，如若这些类中某些方法与其它类有紧密联系，迁移这些方法，从而提高类的内聚度和软件系统的模块度，提升软件系统的可维护性；

[0019] S105)输出方法迁移后内聚度提升的类。

[0020] 本发明进一步的改进在于：所述步骤S101)中软件系统的方法列表获取方法为：利用词法分析技术扫描Java软件源代码，将软件源代码转换为字符流，并通过词法分析技术得到的字符流，构建程序语法树，最后对该语法树进行分析，获取软件系统中方法的定义，包括方法名，方法传递参数，返回值，将相关方法信息保存入数据库。

[0021] 本发明进一步的改进在于：所述步骤S101)中软件系统的方法调用对获取方法为：利用控制流分析技术，获取目标代码所有方法调用对。对比数据库中的方法定义表对方法调用对进行分析与抽取，去除软件外部的方法调用对，最后根据分析结果，对软件方法调用对进行记录，记录格式为：类中方法的包名和方法名；被调用者的包名和方法名->调用者的包名和方法名。

[0022] 本发明进一步的改进在于：所述步骤S101)中软性系统的方法调用的最大弱连通子图生成方法为：依次读取软件方法调用对记录，对类中新方法名标记为方法调用网络图的新节点，对新的方法调用对标记为方法调用网络图的边，循环迭代这个过程生成方法调用网络图，最后去除孤立的节点，生成软件系统方法调用的最大弱连通子图。

[0023] 本发明进一步的改进在于：所述步骤102)中分别实现基于模块度的、知识地图的、多级分层和标签传播的社团划分算法，得到方法社团划分结果，记录格式为：包名和方法名，所属社团编号。根据划分结果计算每个类的方法社团内聚度(MCC)：

[0024] 令某个类C共有m个方法，经过某个社团划分算法后，m个方法分布在N个社团中，第k个社团中分布有n_k个方法(k＝1,…,N)，令n_1为n_k中的最大值，那么可以得到类C基于该社团划分算法的方法社团内聚度为n_1/m。

[0025] 本发明进一步的改进在于：所述步骤103)中分别计算不同方法社团内聚度的斯皮尔曼相关系数验证这些社团划分算法是否获得类似的结果或者各自独立于对方，然后选择相关系数最大的两个算法，基于这两个算法计算每个类方法社团内聚度平均值和每个软件系统的模块度，用模块度的值来评估软件系统的可维护性。

[0026] 本发明进一步的改进在于：所述步骤104)基于每个类的方法社团内聚度，使用自适应的阈值过滤算法，过滤出一些内聚度较低的类，如若这些类中某些方法与其它类有紧

密联系，迁移这些方法，从而提高类的内聚度和软件系统的模块度，提升软件系统的可维护性。

[0027] 本发明进一步的改进在于：所诉步骤4)中自适应的阈值过滤算法为：

[0028] 令某个类C_1一共有N_1个方法，其中有n_{11}个方法在同一社团Com_1中，并且满足：$类C_1的方法社团内聚度=\frac{n_{11}}{N_1}>Threshold$，在剩余的N1-n11个方法中，如果存在一个或者多个方法，其所在的社团Com_2中，有n_{21}个方法属于另一个类C_2，C_2一共有N_2个方法，并且满足：1)$类C_2的方法社团内聚度=\frac{n_{21}}{N_2}>Threshold$，2)这一个或者多个方法与类C2有调用关系。记录下满足此要求的类C_1、C_2及相应的方法和方法数量。阈值以0.05依次递增，保存每个阈值下的类C_1、C_2及相应的方法和方法数量，最后分析阈值从哪个值处开始满足要求的方法数量显著降低，记录此阈值下的类C_1、C_2及相应的方法。

[0029] 本发明进一步的改进在于：所述步骤105)中，记录的类C_1、C_2及相应的方法，把这些方法从类C_1迁移到类C_2中去，从而提升类C_1、C_2的内聚度，输出这些类和需要迁移的方法。

[0030] 相对于现有技术，本发明具有以下优点：

[0031] (1)本发明利用软件系统具有复杂网络特性，构建了软件系统方法间的调用网络图，图的节点为方法，边为方法间的调用关系。运用这个方法调用网络图不仅考虑了类中方法的调用关系，而且考虑了类与类之间的方法调用关系。

[0032] (2)本发明利用复杂网络中的社团结构理论，运用了不同目标对象的社团发现算法，提出了一种基于社团发现算法来计算面向对象软件系统的类的方法社团内聚度，从软件系统模块性的角度评估软件系统的可维护性。

[0033] (3)本发明提升了软件系统各模块内部一些类的内聚度，不仅提升了软件系统中类的内聚度，而且提升了整个软件系统的模块度，从而提升了软件系统的可维护性。

附图说明

[0034] 图1为基于多目标社团发现的软件可维护性评估与提升方法整体流程图；

[0035] 图2为生成方法调用网络最大弱连通图的流程图；

[0036] 图3为计算类的方法社团内聚度流程图；

[0037] 图4为过滤较低内聚度的类和其中方法流程图；

[0038] 图5为过滤算法的流程图。

具体实施方式

[0039] 以下结果附图详细说明本发明基于多目标社团发现的软件系统可维护性与提升方法的具体实施方式。

[0040] 请参阅图1所示，本发明提出一种基于多目标社团发现的软件系统可维护性评估与提升方法，包括以下步骤：

[0041] 步骤S101：使用静态代码分析工具Understand和复杂网络分析包igraph，实现对Java软件系统方法调用的最大弱连通子图的生成；

[0042] 结合图2，具体而言，在输入待分析的Java软件系统后，实现生成该Java软件系统调用网络的最大弱连通子图。

[0043] 步骤S201:对待分析的Java软件系统解析.java文件,获取项目中定义的方法列表;

[0044] 步骤S202:利用静态代码分析软件Understand分析Java软件系统源代码,获取所有方法调用对;

[0045] 步骤S203:根据步骤S201中获取到的Java软件系统中的方法列表以后,对比步骤S202获取到的所有方法调用对,去掉冗余方法,最终获得软件自身方法调用对;

[0046] 步骤S204:根据软件自身方法调用对生成软件方法调用网络图;

[0047] 步骤S205:利用复杂网络分析包igraph过滤掉软件方法调用网络图的孤立的节点,生成软件系统方法调用的最大弱连通子图。

[0048] 例如输入一个开源的Java语言开发的文本编辑器jEdit,利用静态代码分析工具Understand分析获取方法调用图,利用词法语法分析方法扫描源代码,过滤掉属于JDK中的方法,最后得到一个软件系统内部的方法调用对,例如:

[0049] "EditPane.java:EditPane.propertiesChanged()"->"JEditTextArea.java:JEditTextArea.setRightClickPopup(JPopupMenupopup)"[label="EditPane.java:485"];保存在callgraph.dot中,运用复杂网络分析包igraph,生成最大弱连通子图。

[0050] 步骤S102:运用四种基于不同目标的社团划分算法划分步骤S101所生成的最大弱连通子图,并计算每个类的方法社团内聚度。具体流程如图3所示:

[0051] 步骤S301:运用开源的复杂网络分析包iGraph中的四种社团划分算法(基于模块度优化(fg)、知识地图(im)、多级分层(ml)和标签传播(lp)的社团划分算法)对得到的方法调用的最大弱连通子图进行划分;

[0052] 步骤S302:根据步骤S301的划分结果统计出Java软件系统中每个类的所属社团编号;

[0053] 步骤S303:获取每个类所属各个社团的方法数;

[0054] 步骤S304:根据方法社团内聚度的定义(令某个类C共有m个方法,经过某个社团划分算法后,m个方法分布在N个社团中,第k个社团中分布有n_k个方法(k=1,…,N),令n_1为n_k中的最大值,那么可以得到类C基于该社团划分算法的方法社团内聚度为n_1/m),计算出每个类的四种社团划分算法对应的方法社团内聚度。

[0055] 针对上一步得到的方法调用最大弱连通子图,分别运用iGraph中的基于模块度优化(fg)、知识地图(im)、多级分层(ml)和标签传播(lp)的社团划分算法包,得到每个类中每个方法所属的社团结构,如下表格运用lp社团划分算法得到部分结果,左边为上一步方法调用对中的方法,右边为所属的社团编号,

[0056] 表1:社团划分后的结果

[0057]

EditPane.java:EditPane.handleMessage(EBMessagemsg)	74
EditPane.java:EditPane.getTextArea()	74
EditPane.java:EditPane.nextBuffer()	74
EditPane.java:EditPane.propertiesChanged()	34
JEditTextArea.java:JEditTextArea.setRightClickPopup(JPopupMenu popup)	34
JEditTextArea.java:JEditTextArea.getText()	34
JEditTextArea.java:JEditTextArea.getSelectionEnd()	34

[0058] 接着分析得到每个类所属的社团，例如分析EditPane类得到结果为EditPane:74,74,74,74,74,74,74,74,74,74,74,74,74,74,74,74,34可以得到EditPane这个类有16个方法在社团74中，只有1个方法在社团34中，那么运用lp社团划分算法得到的这个类的方法社团内聚度

[0059] $$MCC_{lp}(EditPane)=\frac{16}{17}=0.94$$

[0060] 类似的可以得到其他三个社团划分算法的社团内聚度$MCC_{im}=0.29$，$MCC_{fg}=0.53$，$MCC_{ml}=0.35$。

[0061] 步骤S103：计算四种社团划分算法结果的斯皮尔曼相关系数，最后挑选出最大斯皮尔曼相关系数的两个算法来计算类内聚度和模块度，具体流程如图4所示：

[0062] 步骤S401：计算每两个社团划分算法之间的斯皮尔曼相关系数；

[0063] 步骤S402：挑选6组斯皮尔曼相关系数中的最大值，记录这这个最大值所对应的两种社团划分算法；

[0064] 步骤S403：运用步骤S402中的两种社团划分算法计算JAVA软件系统的平均模块度(两种社团划分算法所计算模块度的均值)和每个类的平均社团内聚度(两种社团划分算法所计算社团内聚度的均值)；

[0065] 步骤S404：输出JAVA软件系统的平均模块度和其中每个类的平均社团内聚度。

[0066] 根据上一步得到的结果计算，计算每两个算法结果的斯皮尔曼相关性，例如im和fg这两个算法得到的部分社团内聚度结果为：

[0067] 表2：模块度计算

[0068]

项目	fg模块度	im模块度	lp模块度	ml模块度
jEdit	0.720	0.626	0.726	0.747

[0069] 表3：相关性计算

[0070]

方法	im算法	fg算法	等级x_i	等级y_i	d_i	d_i^2
EditPane	0.29	0.52	1	1	0	0
JEditTextArea	0.29	0.89	2	4	-2	4
BufferUpdate	0.5	0.75	3	3	0	0
FavoritesVFS	0.625	0.625	4	2	2	4
XmlParser	0.81	0.91	5	5	0	0

[0071] 1.排列第二列数据，创建新列等级X_i并赋以等级值1,2,3,...n；

[0072] 2.然后，排列第三列数据，创建新列Y_i并相似地赋以等级值1,2,3,...n；

[0073] 3.创建第i列d_i保存两个等级列的差值(x_i和y_i)；

[0074] 4.创建最后一列d_i^2保存等级差d_i的平方。

[0075] 最后算出斯皮尔曼相关系数$\rho = 1 - \frac{6\sum d_i^2}{n(n^2-1)}$，如下表算出每两个算法的相关系数，

[0076] 表4:斯皮尔曼相关系数计算结果

[0077]

项目	斯皮尔曼相关系数					
	fg-im	***fg-lp***	***fg-ml***	***im-lp***	***im-ml***	***lp-ml***

[0078]

jEdit	0.632	0.721	0.759	0.679	0.708	0.796

[0079] 得到最大的相关系数为lp和ml算法，算出MCC(EditPane)＝(0.94+0.35)/2＝0.64，最后得到jEdit模块度＝(0.726+0.747)/2＝0.7365。

[0080] 步骤S104:使用阈值过滤算法，找出一些内聚度较低类中可以迁移的方法，输出这些迁入\迁出的方法，提高这些类的内聚度，提升了整体的模块度，从而提升了JAVA软件系统的可维护性。具体流程如图5所示：

[0081] 步骤S501:判断社团内聚度的过滤阈值是否大于1，如果是则转入步骤S508，否则转入步骤S502;初始过滤阈值0.5或0.6；

[0082] 步骤S502:在步骤S404输出的每个类的平均社团内聚度中查找一个社团内聚度大于过滤阈值的类C_1；

[0083] 步骤S503:判断类C_1中是否存在方法在其他社团中，如果是则转入步骤S504，否则转入步骤S502查找符合条件的另一个类；

[0084] 步骤S504:在其他社团中查找社团内聚度大于过滤阈值的类C_2；

[0085] 步骤S505:判断步骤S503中的方法是否与类C_2具有调用关系，如果是则转入步骤S506，否则转入步骤S502；

[0086] 步骤S506:保存所有满足条件的类C_1、C_2及步骤S503中的方法和方法数量；

[0087] 步骤S507:过滤阈值递增0.05；

[0088] 步骤S508:根据步骤S506中保存的记录中的方法数量列表，计算并保存每两个相邻阈值间方法数量差值，找出方法数量差值最大的两个相邻阈值，最后得到从两个阈值中较大的那个阈值处开始，满足要求的方法数量显著降低；

[0089] 步骤S509:输出方法数量显著降低时所对应的过滤阈值(步骤S508中得到的两个阈值中较大的那个阈值)下的所有类C_1、C_2以及相应的方法。

[0090] 将步骤S509输出的类C_1中相应的方法迁入类C_2中,这样就能够提升整个JAVA软件系统的可维护性。

[0091] 根据上一步得到的结果,当阈值大小为0.9时,EditPane类被划分到两个社团,社团74和34中,有大于阈值0.9的94.11%的方法在社团74中,而在社团34中EditPane类有的一个propertiesChanged()方法在社团34中,在社团34中找到JEditTextArea类有大于阈值0.9的93.30%的方法在社团34中,并且根据方法调用关系对得到EditPane的方法propertiesChanged()与JEditTextArea类中的方法有如下调用关系:

[0092]

EditPane.propertiesChanged() ->	JEditTextArea.setRightClickPopup(JPopupMenu popup)
EditPane.propertiesChanged() ->	JEditTextArea.propertiesChanged()
EditPane.propertiesChanged() ->	JEditTextArea.setCaretBlinkEnabled(booleancaretBlinks)
EditPane.propertiesChanged() ->	JEditTextArea.setQuickCopyEnabled(booleanquickCopy)
EditPane.propertiesChanged() ->	JEditTextArea.setElectricScroll(intelectricScroll)
EditPane.propertiesChanged() ->	JEditTextArea.getPainter()

[0093] 显然EditPane的方法propertiesChanged()与JEditTextArea类有密切的联系,则应该将EditPane的方法propertiesChanged()迁入JEditTextArea类中,从而将EditPane类的lp算法社团内聚度提升到1,JEditTextArea的lp算法社团内聚度提升到0.94,将所有类过滤过以后,jEdit的lp算法模块度从0.726提升到0.813,提升了软件系统的模块度,从而提升了软件系统可维护性。

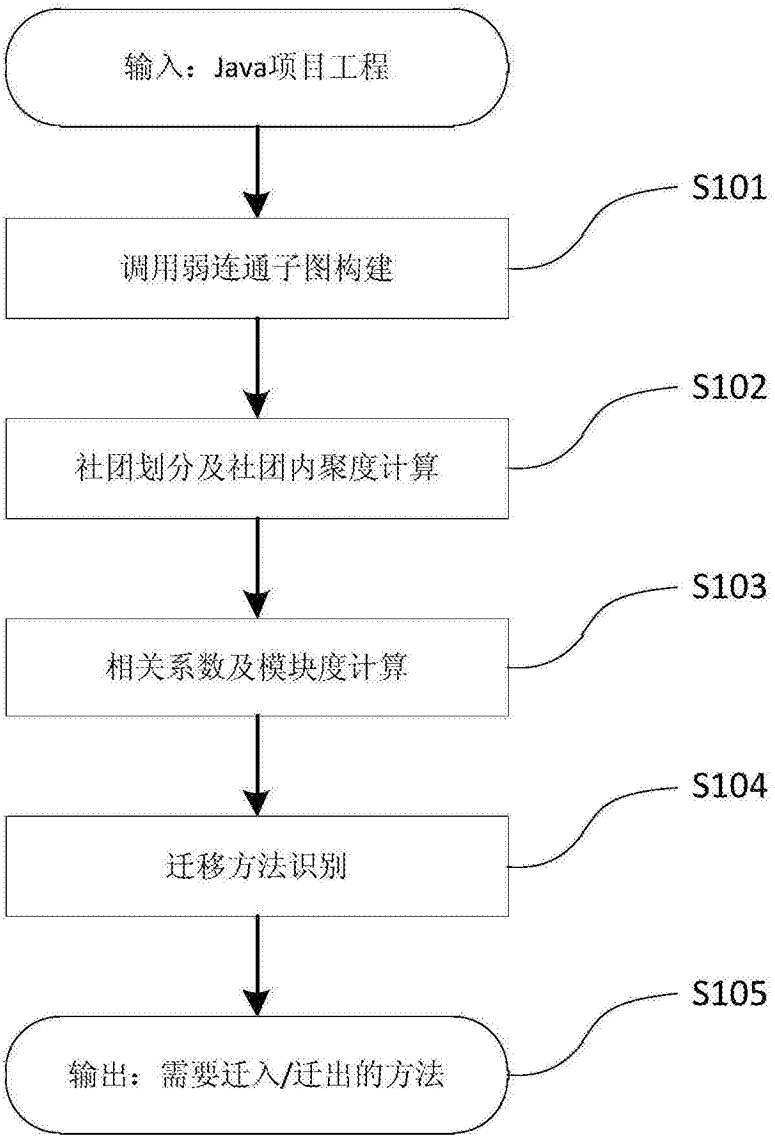

输入：Java项目工程
S101
调用弱连通子图构建
S102
社团划分及社团内聚度计算
S103
相关系数及模块度计算
S104
迁移方法识别
S105
输出：需要迁入/迁出的方法

图1

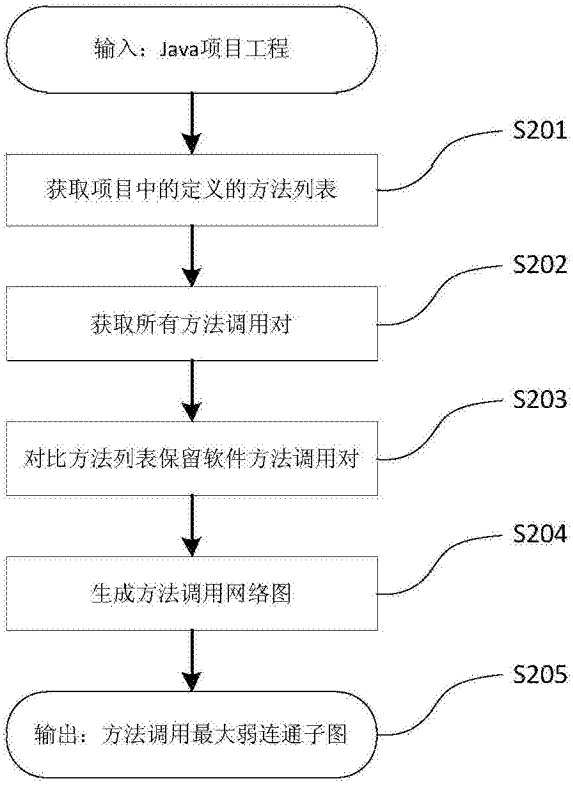

输入：Java项目工程
获取项目中的定义的方法列表
S201
获取所有方法调用对
S202
对比方法列表保留软件方法调用对
S203
生成方法调用网络图
S204
输出：方法调用最大弱连通子图
S205

图2

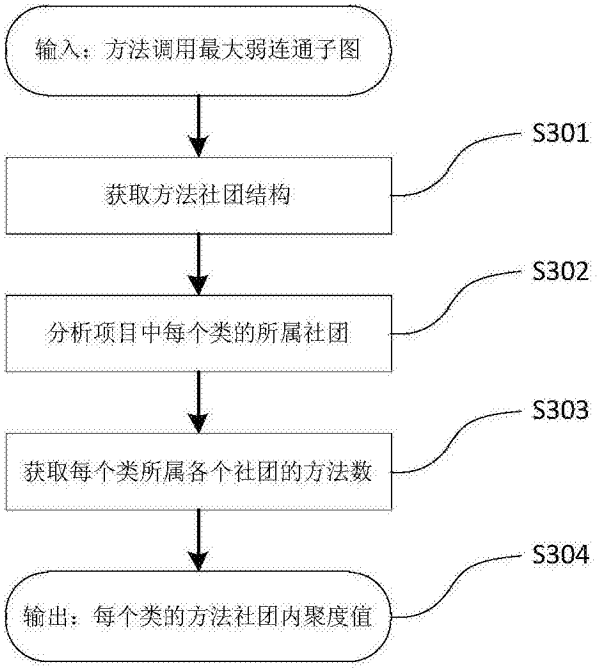

输入：方法调用最大弱连通子图
获取方法社团结构
S301
分析项目中每个类的所属社团
S302
获取每个类所属各个社团的方法数
S303
输出：每个类的方法社团内聚度值
S304

图3

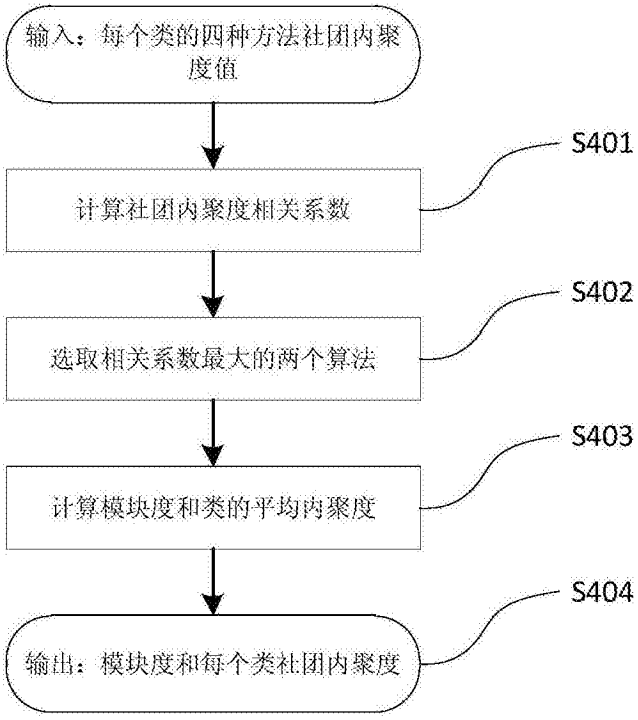

输入：每个类的四种方法社团内聚度值
计算社团内聚度相关系数
S401
选取相关系数最大的两个算法
S402
计算模块度和类的平均内聚度
S403
输出：模块度和每个类社团内聚度
S404

图4

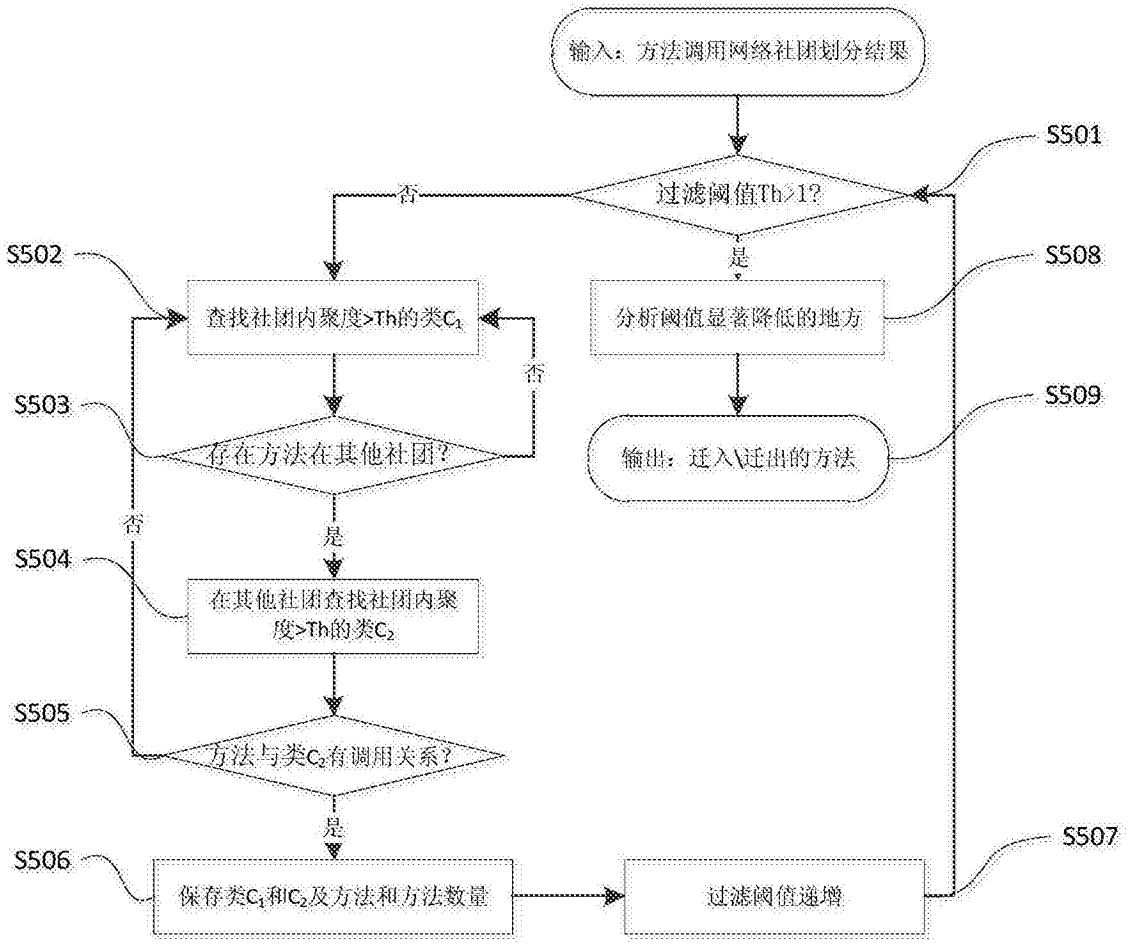

输入：方法调用网络社团划分结果
S501
过滤阈值Th>1?
否
是
S502
查找社团内聚度>Th的类C1
S508
分析阈值显著降低的地方
否
S503
存在方法在其他社团?
S509
输出：迁入\迁出的方法
否
是
S504
在其他社团查找社团内聚度>Th的类C2
S505
方法与类C2有调用关系?
是
S506
保存类C1和C2及方法和方法数量
过滤阈值递增
S507

图5