计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2016.20140727
53(3):503-516,2016

一种软件演化活动波及效应混合分析方法

王 炜 李 彤 何 云 李 浩
(云南大学软件学院 昆明 650091)
(wangwei@ynu.edu.cn)

A Hybrid Approach for Ripple Effect Analysis of Software Evolution Activities

Wang Wei, Li Tong, He Yun, and Li Hao
(*Software School*, *Yunnan University*, *Kunming* 650091)

Abstract Ripple effect is defined as the process of determining potential effects to a subject system resulting from a proposed software evolving activity. Since software engineers perform different evolving activities to respond to different kinds of requirements, ripple effect analysis has been globally recognized as a key factor of affecting the success of software evolution projects. The precision of most existing ripple effect analysis methods is not as good as expectation and lots of methods have their inherent limitations. This paper proposes a hybrid analysis method which combines the dynamic and information retrieval based techniques to support ripple effect analysis in source code. This combination is able to keep the feature of high recall rate of dynamic method and reduce the adverse effects of noise and analysis scope by the domain knowledge which is derived from source code by information retrieval technique. In order to verify the effectiveness of the proposed approach, we have performed the ripple effect analysis and compared the analysis results produced by dynamic, static, text, historical evolving knowledge based methods with the proposed method on one open source software named jEdit. The results show that the hybrid ripple effect analysis method, across several cut points, provides statistically significant improvements in both precision and recall rate over these techniques used independently.

Key words ripple effect analysis; dynamic ripple effect analysis; domain knowledge based noise reduction; minimal complete set of evolution activities; association rule mining

摘 要 确定演化活动潜在影响的过程称之为波及效应分析.波及效应分析已经被公认为影响软件演化项目成败的一个关键因素.针对当前波及效应分析准确率不高、各方法存在固有缺陷的问题,提出了一种混合波及效应分析方法,该方法将动态分析方法与文本分析方法相结合,在保持高召回率的基础上,基于演化软件领域知识降低了噪声对分析结果的不利影响,约简了分析范围,提高了查准率.为验证方法的有效性,对开源软件 jEdit 分别使用动态、静态、基于文本、基于历史演化知识和混合分析方法进行波及效应分析.通过比对实验结果,表明混合波及效应分析方法具有较好的综合性能.

关键词 波及效应分析;动态波及效应分析;基于领域知识的降噪;演化活动最小完备集;关联规则挖掘

中图法分类号 TP311.5

收稿日期:2014-08-04;**修回日期**:2015-05-14

基金项目:国家自然科学基金项目(61462092,61262024,61379032);云南省自然科学基金重点项目(2015FA014);云南省自然科学基金面上项目(2013FB008)

This work was supported by the National Natural Science Foundation of China (61462092,61262024,61379032), the Key Program of the Natural Science Foundation of Yunnan Province (2015FA014), and the General Program of the Natural Science Foundation of Yunnan Province (2013FB008).

确定演化活动潜在影响范围的过程称之为波及效应分析[1].波及效应分析是可追踪性分析(traceability analysis)、演化代价估计(evolving cost estimation)、波及效应管理(ripple effects management)等研究的基础[2-7].波及效应分析存在"两高":高成本和高难度.据统计,在长生命周期软件系统中,50%~75%的成本用于软件维护和演化,其中一半以上用于故障定位和波及效应分析[8].文献[7]指出对 jEdit,ArgoUML 等中等规模的软件系统进行波及效应分析,查准率在 5%~18%.造成上述困境的原因有以下 4 点:1)与系统相关的需求文档、设计文档、用户手册、系统开发人员等资源遗失、不准确或无法访问.2)传统的波及效应分析是代码驱动,即假设需要实施演化活动的目标代码是已知的.实际情况是程序员无法直接确定演化活动的目标代码,只能从演化意图入手,确定演化活动的影响范围.3)软件系统规模越来越大,发生演化活动的频度越来越高,执行波及效应分析的时间间隔也变得越来越短.这要求分析者能够快速响应演化需求.4)待演化软件大多与其他软件或操作系统交织在一起构成一个复杂的生态系统,因此很难对待演化系统的边界进行明确划分,增加了分析的难度.

按研究成果出现的先后顺序,波及效应分析大致可以分为 4 类:静态波及效应分析方法(static ripple effect analysis)、动态波及效应分析方法(dynamic ripple effect analysis)、基于文本的波及效应分析方法(text based ripple effect analysis)和基于历史演化知识的波及效应分析方法(historical evolution knowledge based ripple effect analysis).

静态分析方法将源代码、设计文档等静态数据作为输入,通过获取静态数据间的控制流、数据流关系,实现波及效应建模、分析.1996 年 Bohner[9-10] 基于可达矩阵提出了最早的波及效应静态分析方法.随后,学者们使用不同工具(可达图、可达矩阵等)从不同粒度(构件、类、方法,变量)分别提出了各自的分析和预测方法[11-13].静态方法具有较强的易用性,但由于控制流和数据流在大型软件系统中十分复杂,无法自动判定确切的问题边界,需要大量人工介入,分析过程开销随着软件复杂程度呈指数级增长.分析结果的准确性依赖于开发人员对软件结构的理解程度.

动态波及效应分析方法将执行迹(execution traces)作为输入数据,通过捕获执行迹与用例间的映射关系实现波及效应的建模和分析.根据执行迹获取方式的不同,动态分析方法可以分为[14]:基于功能覆盖[5]和基于执行路径的分析方法[4,15].上述方法均需要设计并执行测试用例,区别在于测试用例是否形成对功能点或对程序执行路径的覆盖.相较于静态分析方法,此类方法可获得较高的召回率.但由于应用软件生态环境的复杂性,导致采集到的执行迹中包含了大量与软件功能不相干的噪声数据,查准率较低.因此,传统的动态分析方法需要对同一用例多次采集执行迹,通过比对实现去噪.这导致了动态分析方法同样需要大量人工介入,效率较低.

源代码、变更请求(change request)等软件实体(software artifacts)蕴含着大量的领域知识.基于文本的波及效应分析方法对变更请求[16-17]和源代码[18-19]中的领域知识进行抽取,通过对比领域知识与软件功能描述之间的相似度实现波及效应分析.对比静态和动态分析方法,基于文本的波及效应分析方法具有较高的查准率,但召回率较低.另外,该方法的有效性很大程度上依赖于变更信息的多寡、代码中变量命名的规范性以及用词习惯等因素,上述因素导致了该方法的普适性较差.

软件系统在日常运行过程中积累了大量诸如开发者来往邮件、软件维护日志等历史数据.这些历史数据中蕴含着丰富的软件体系结构、演化决策等知识.学者们对上述数据进行挖掘,并据此进行波及效应分析和预测[20-21].与基于文本的波及效应分析方法类似,此类分析方法性能同样受制于历史演化知识的多寡、检索关键词用词习惯等因素.

静态、动态、基于文本和历史演化知识的波及效应分析方法可以识别演化活动的波及范围,但大多无法对波及范围进行定量刻画.同时,各方法具有各自优点,但也存在固有缺陷.一种比较自然的想法是将上述方法进行整合,在充分发挥其优点的基础上,尽可能规避其固有缺陷.

本文提出的波及效应混合分析方法,具有 3 个特点:

1) 实现了对现有波及效应分析方法的无缝整合.在充分发挥动态分析方法召回率高的基础上,基于文本分析方法实现对执行迹的降噪,提高了查准率,缩减了分析范围.建立波及效应图实现量化分析方法.

2) 更贴近于实际应用.从故障现象入手是一种用例驱动的波及效应分析方法.

3) 传统动态波及效应分析方法需要对执行迹进行多次采集,而本文方法仅需要进行一次执行迹采集,降低了分析开销.

1　本文方法

如图 1 所示,本方法首先获取与演化活动用例对应的执行迹;其次基于文本分析方法,实现对执行迹的降噪;在此基础上识别代码间的波及关系,构造波及效应图,实现量化分析.

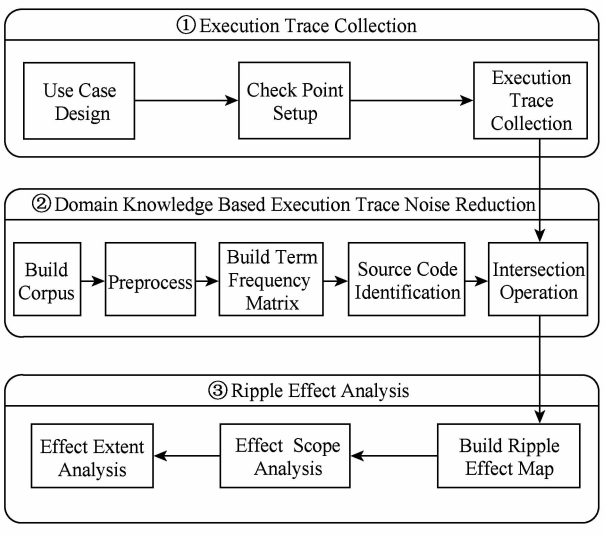

Fig. 1　Overview of hybrid ripple effect analysis method.

图 1　混合波及效应分析方法概况

1.1　执行迹获取

执行迹记录了软件在运行过程中消息的传递序列,反映了特定环境下软件系统的行为属性.

定义 1. 有限非空序列 $p=\langle u_1,u_2,\cdots,u_n\rangle$ 定义为一个进程,记录了软件系统各组成单元 u_i 间的消息传递序列.

根据粒度大小组成单元可分为:服务、构件、类、方法(函数)等.

定义 2. 有限非空集合 $S_{ET}=\{p_1,p_2,\cdots,p_m\}$ 为执行用例 U 时产生的执行迹,其中 p_i 为进程. 若 $|S_{ET}|=m$,则称 S_{ET} 是 m 元执行迹.

定义 3. 有限非空类集合 $S_{EC}=\bigcup_{i=1}^{m} p_i$,定义为与执行迹 S_{ET} 对应的代码集合,其中 $|S_{ET}|=m$.

执行迹的获取包含 3 个步骤:

1) 编写与演化活动相关的用例. 用例应能够反映演化意图,例如系统功能的删除、添加、迁移等.

2) 在源代码中设置检查点(check point)用以观察程序运行时消息的传递过程,并将带有检查点的代码进行编译形成可执行文件.

3) 执行用例,观测并收集与用例相关的执行迹.

表 1 为执行迹片段示例:

Table 1　Execution Traces Slice Sample of Thread main

表 1　进程 main 的执行迹片段示例

Call Stack Depth	Method Name	Class Name with Full Path
Ⅰ	process-Options	org. mozilla. javascript. tools. shell. Main
Ⅰ	init	org. mozilla. javascript. tools. shell. Global
Ⅱ	call	org. mozilla. javascript. ContextFactory
Ⅱ	init	org. mozilla. javascript. ScriptableObject
Ⅲ	clinit	org. mozilla. javascript. Context

与传统的动态分析方法需要对同一用例多次采集执行迹,进行人工比对实现降噪不同. 本方法仅需要进行一次执行迹的获取. 降噪过程由 1.2 节实现. 这大大节省了执行迹获取的开销,提高了整个分析过程的自动化程度.

1.2　基于领域知识的执行迹降噪

源代码的变量名、函数名、类(对象)名、API 函名、注释等蕴含了丰富的领域知识. 本节降噪过程描述如下:

1) 生成语料库. 为源代码中每一个类(函数)建立一个文本文件,提取类代码中蕴含领域知识的关键词存入该文件. 所有类对应的文本文件构成语料库.

表 2 为 1 个由 6 个文本文件组成的语料库示例. 其中每一行代表一个与源代码对应的文本文件.

Table 2　Corpus Sample

表 2　语料库示例

File Name	File Content
code1	this,table,document,word,document
code2	one,tableOpen,document,table,file,edit
code3	insert,insert,file,inserting,edit
code4	circle,line,circle
code5	picture,file,draw,xml
code6	draw,line,draw,the,graphic,about

2) 预处理. 语料库中存在很多与领域知识不相关或重复的信息,需要删除达到降噪的目的. 预处理包含去除停词(stop words)、分词、词干还原 3 部分. 去除停词将删除与领域知识不相关的词,例如表 2 中的 this,one,the,about 等词;分词操作将连续的字符序列按照一定的规则拆分成若干单词,例如将表 2 中的 tableOpen 拆分成为 table 和 open 两个

词;词干还原将意义相近的同源词和同一个单词的不同形态进行归并,例如将表 2 中的 inserting 还原成为 insert.

3) 生成词频矩阵.将经过预处理的语料转换为词频矩阵,该矩阵的行表示在语料库中出现过的单词,列是对一个类中关键词的计数.矩阵元素 t_{ij} 表示第 i 个单词在第 j 个类中出现的次数.表 3 给出了与表 2 对应的词频矩阵.

Table 3 Term Frequency Matrix Sample

表 3 词频矩阵示例

Key words	code1	code2	code3	code4	code5	code6
table	1	2	0	0	0	0
word	1	0	0	0	0	0
keywords document	2	1	0	0	0	0
open	0	1	0	0	0	0
insert	0	0	3	0	0	0
file	0	1	1	0	1	0
edit	0	1	1	0	0	0
line	0	0	0	1	0	1
circle	0	0	0	2	0	0
picture	0	0	0	0	1	0
draw	0	0	0	0	1	2
xml	0	0	0	0	1	0
graphic	0	0	0	0	0	1

4) 获取演化活动用例对应代码集合.将演化活动用例的文字性描述转化为向量作为输入,使用潜在语义索引(latent semantic indexing, LSI)对词频矩阵进行索引排序.设定相似度阈值,凡大于阈值的向量所对应的类组成演化活动用例对应的代码集合.

定义 4. 设 $\boldsymbol{Q}=(q_1,q_2,\cdots,q_n)$ 是演化活动描述对应的向量,$\boldsymbol{t}_{\cdot j}=(t_{1j},t_{2j},\cdots,t_{nj})^{\mathrm{T}}$ 表示词频矩阵的第 j 列向量,则二向量的相似度定义为:$Sim(\boldsymbol{Q},\boldsymbol{t}_{\cdot j})=\dfrac{\boldsymbol{Q}\cdot\boldsymbol{t}_{\cdot j}}{|\boldsymbol{Q}|\times|\boldsymbol{t}_{\cdot j}|}$.其中,$\cdot$ 表示向量内积,$||$ 表示向量的模.

例如开发人员提交的演化活动用例描述为“Crash when inserts a word into a document”.如表 3 所示,该用例描述对应的向量为 $\boldsymbol{Q}=(0,1,1,1,0,0,0,0,0,0,0,0,0)^{\mathrm{T}}$,通过计算 $\boldsymbol{Q}$ 与表 3 中词频矩阵的相似度,有以下结果:

$$Sim(\boldsymbol{Q},code1)>Sim(\boldsymbol{Q},code4);$$
$$Sim(\boldsymbol{Q},code2)>Sim(\boldsymbol{Q},code5);$$
$$Sim(\boldsymbol{Q},code3)>Sim(\boldsymbol{Q},code6).$$

传统基于文本的波及效应分析[22]使用经验相似度阈值 $\alpha=0.0075$.但笔者在实验中发现,当数据规模较大时,经验相似度阈值 $\alpha=0.0075$ 效果并不理想.因此,我们使用文献[23]推荐的方法,取相似度最高的 10%~15%的代码作为索引结果.

5) 求交运算.将执行迹对应的代码集合 S_{EC} 与本节产生的演化活动用例对应的代码集合求交集.凡是不在上述 2 集合中同时出现的类均认为是噪声类.噪声类要么没有在用例执行时被实际调用,要么与演化用例不相关.噪声类的去除减少了波及效应分析的数据量.将降噪结果作为 1.3 节波及效应分析的输入数据.

1.3 波及效应分析

本节将经过降噪处理的执行迹代码作为输入,依次执行 3 个步骤,对演化活动的波及效应进行量化分析:

1) 识别波及关系;

2) 构造波及效应图;

3) 波及效应量化分析.

1.3.1 波及关系识别

代码间的波及关系刻画了若一个类被修改则另一个类也会被修改的事实.

定义 5. 有限集合 $R_S=\{c_i\rightarrow c_j\mid c_i,c_j\in C\}$ 为波及系集合,其中 C 为类集合,$c_i\rightarrow c_j$ 是一条波及关系,c_i 称为波及关系的前件,c_j 称为波及关系的后件.若 $|c_i\cup c_j|=n$,则称为 n 元波及关系.

笔者认为波及关系是波及效应存在的根本原因.对面向对象软件执行演化活动时,对象间存在依赖、关联、泛化和实现关系均可以用波及关系表示.类 c_i 和 c_j 间的依赖关系表示了 c_i 发生变化而影响到 c_j 也发生变化的语义.因此,对类 c_i 执行演化活动时必然存在波及关系 $c_i\rightarrow c_j$.类 c_i 和 c_j 间的关联关系是一种特殊的依赖关系,既构成依赖关系的类 c_i 和 c_j 间相互存在依赖关系.因此,演化活动发生时,必然存在 2 个关联关系 $c_i\rightarrow c_j$,$c_j\rightarrow c_i$.泛化关系刻画了一般类与特殊类之间的继承关系.若类 c_i 和 c_j 间存在泛化关系,且 c_i 是特殊类、c_j 是一般类,则对 c_i 进行修改必然导致 c_j 的变化,因此必然存在波及关系 $c_i\rightarrow c_j$.实现关系是泛化关系和依赖关系的结合,表示了接口和实现它功能类之间的语义关系.接口 i_a 与类 c_i 间存在实现关系,对 i_a 进行修改必然导致类 c_i 要进行相应的修改,反之亦然.因此,若接口 i_a 和类 c_i 间存在实现关系,则存在波及关系 $i_a\rightarrow c_i$ 和 $c_i\rightarrow i_a$.在结构化软件系统中,函数间通过调用

关系实现具体计算功能.设函数 f_i 调用函数 f_j,且 f_j 返回计算结果给 f_i,显然存在波及关系 $f_i \rightarrow f_j$,$f_j \rightarrow f_i$ 关系.若 f_j 不返回计算结果给 f_i,则仅存在波及关系 $f_i \rightarrow f_j$ 关系.综上所述,针对任意演化活动,软件各类实体具有的关系均可以用波及关系刻画.

矩阵 $\boldsymbol{K}_{n\times n}$ 为波及关系的存储介质.以面向对象软件系统为例,类 i 与类 j 间存在波及关系,当且仅当 i 和 j 两个类间存在关联关系,或第 i 个类是第 j 个类的特殊类,或第 i 个类依赖第 j 个类,或 i 和 j 两个类间存在实现关系,记为 $k_{ij}=1$,否则 $k_{ij}=0$.波及关系的方向性由行和列分别表示,行为起始类,列为终结类.例如表 4 第 1 行、第 6 列为 1,表示 c_1 与 c_5 之间存在波及关系,且 $c_1 \rightarrow c_5$.

Table 4　Ripple Effect Relationship Matrix of Some Software

表 4　某系统的波及关系矩阵图

Classification	c_1	c_2	c_3	c_4	c_5	c_6
c_1	1	0	0	0	1	0
c_2	0	1	0	1	0	0
c_3	0	0	1	1	1	0
c_4	0	1	0	1	0	0
c_5	1	0	0	1	1	1
c_6	0	1	0	0	0	1

1.3.2　波及效应图构造

波及效应图是波及效应量化分析的基础.整个构造过程包括 2 个方面:边集合识别和波及效应图构造.

1) 边集合识别

波及效应图的边集合由波及关系集合充当.算法 1 描述了波及关系图边集合的识别过程.

算法 1. 波及效应图边集合识别算法.

输入:软件波及关系矩阵 $\boldsymbol{K}$、支持度 $Support$ 和置信度 $Conf$;

输出:波及效应图边集合 E.

① 计算波及关系矩阵的传递闭包 $\boldsymbol{K}^+$;

② 基于支持度 $Support$ 对 $\boldsymbol{K}^+$ 进行频繁类集挖掘;

③ 基于置信度 $Conf$ 对频繁类集中的波及关系进行识别,生成波及效应图边集合 E.

算法 1 中行①$\boldsymbol{K}^+=\boldsymbol{K}\vee\boldsymbol{SK}^1\vee\cdots\vee\boldsymbol{K}^n$ 定义为波及关系矩阵 $\boldsymbol{K}$ 的传递闭包矩阵,其中 $\boldsymbol{K}^i=\boldsymbol{K}\vee\boldsymbol{K}^{i-1}$.

算法 1 中行②③采用关联关系挖掘方法[24],实现频繁类集和波及关系的识别.根据算法要求,做如下定义:

定义 6. 传递闭包矩阵 $\boldsymbol{K}^+$ 行 i 中取值为 1 的元素组成的集合 $k_i=\{k_{i1},k_{i2},\cdots,k_{in}\}$ 定义为第 i 个类的演化事务.有限非空集合 $T=\{k_1,k_2,\cdots,k_n\}$ 是软件系统的演化事务集合.

集合 $k_i=\{k_{i1},k_{i2},\cdots,k_{in}\}$ 描述了对第 i 个类实施演化活动将导致 k_i 集合中的类也将进行演化活动.表 5 给出了与表 4 对应的传递闭包矩阵,可以看出在演化事务 k_1 中,若对 c_1 实施演化活动将导致 c_1,c_2,c_4,c_5,c_6 也需要进行演化.

Table 5　The Transitive Closure of Table 4

表 5　表 4 对应的传递闭包矩阵

id	c_1	c_2	c_3	c_4	c_5	c_6
k_1	1	1	0	1	1	1
k_2	0	1	0	1	0	0
k_3	1	1	1	1	1	1
k_4	0	1	0	1	0	0
k_5	1	1	0	1	1	1
k_6	0	1	0	1	0	1

定义 7. 类 c_i 的支持计数 c^i_{count} 定义为演化事务集合 $T=\{k_1,k_2,\cdots,k_n\}$ 中包含类 c_i 的演化事务数量.

表 5 中由于类 c_1 被事务 k_1,k_3 和 k_5 所包含,因此支持计数 $c^1_{\text{count}}=3$.

定义 8. 设 $T=\{k_1,k_2,\cdots,k_n\}$ 为演化事务集合,一条波及关系 $c_i \rightarrow c_j$ 的支持度定义为

$$Support=\frac{(c_i \cup c_j)^{i,j}_{\text{count}}}{|T|}.$$

定义 9. 设 $T=\{k_1,k_2,\cdots,k_n\}$ 为演化事务集合,一条波及关系 $c_i \rightarrow c_j$ 的置信度定义为

$$Conf=\frac{Support(c_1 \cup c_2)}{Support(c_1)}.$$

直观上来看,波及关系 $c_i \rightarrow c_j$ 的支持度定义为包含($c_i \cup c_j$)的演化事务在整个演化事务集合中的百分比.置信度刻画了演化事务集合既包含 c_i 又包含 c_j 的数量占所有包含 c_i 的事务的百分比.支持度取值越小说明此波及关系的偶然性越强,取值越大说明此波及关系的必然性越强.置信度描述了类 c_i 发生演化情况下类 c_j 也需要进行演化的条件概率 $p(c_j|c_i)$,置信度取值越高说明该波及关系前后件间的联系越紧密.

定义 10. 一个频繁类集是支持度高于 $Support$

的类组成的集合. 称一个基数为 k 的频繁类集合为 k-频繁类集,记为 F_k.

当 $Support=60\%$, $Conf=60\%$时,表 5 经过算法 1 计算可产生如下计算结果:

$F_1=\{(c_2),(c_4),(c_6)\}$, $F_2=\{(c_2,c_4),(c_2,c_6),(c_4,c_6)\}$, $F_3=\{(c_2,c_4,c_6)\}$,波及效应图边集合 $E=\{c_6\to c_4, c_4\to c_6, c_6\to c_2, c_2\to c_6, c_4\to c_2, c_2\to c_4, c_6\to(c_2,c_4), c_4\to(c_2,c_6), c_2\to(c_4,c_6)\}$,各波及关系的置信度分别为{1.0, 0.67, 1.0, 0.67, 1.0, 1.0, 1.0, 0.67, 0.67}.

2) 波及效应图构造

定义 11. 波及效应图 $G_{RE}=(V,E,L)$是一个带权有向图. 顶点集合 $V=\{v_j \mid v_j\in \bigcup_k F_k\}$,其中 F_k 是 k-频繁类集; E 为算法 1 输出结构;标记函数 L: $e_i\to Conf_i$, $Conf_i$ 定义为波及关系 $c_i\to c_j$ 的置信度, $e_i\in E$.

图 2 给出了与表 5 对应的波及效应图:

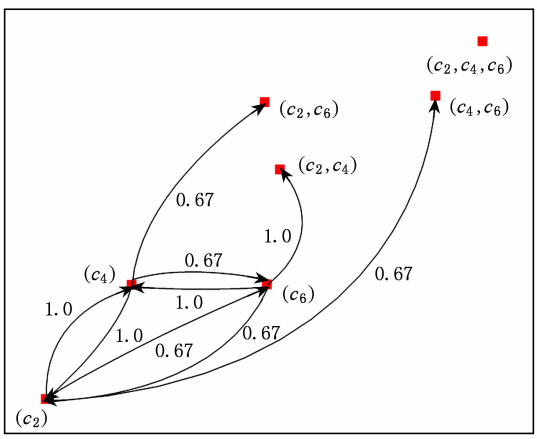

Fig. 2 Ripple effect graph of Table 5.

图 2 表 5 对应的波及效应图

软件演化的基本活动包括添加、删除、替换、和迁移 4 类活动[25]. 基于波及效应图,本文对上述 4 类演化活动做如下定义:

定义 12. 设 $G_{RE}=(V,E,L)$为波及效应图.

1) 设 $v\notin V$,谓词 $add(G_{RE},v)$表示在 G_{RE}中添加 v 以及与 v 存在波及关系的边,称为添加 v 活动;

2) 设 $v\in V$,谓词 $del(G_{RE},v)$表示从 G_{RE}中删去 v 以及与 v 存在波及关系的边,称为删除 v 活动;

3) 设 $v\in V$, $v'\notin V$,谓词 $sub(G_{RE},v',v)$表示用 v'替换 G_{RE}中的 v,并且删除与 v 存在波及关系的边,增加与 v'存在波及关系的边,称为替换 v 活动;

4) 设 $v\in V$,谓词 $mig(G_{RE},v)$表示修改与 v 存在波及关系的边的连接属性,称为迁移 v 活动.

图 3~6 分别描述了对图 2 实施的增加顶点 v、删除顶点(c_2,c_4,c_6)、用顶点 v'替换顶点(c_2,c_4,c_6)和迁移顶点(c_2,c_4,c_6)演化活动.

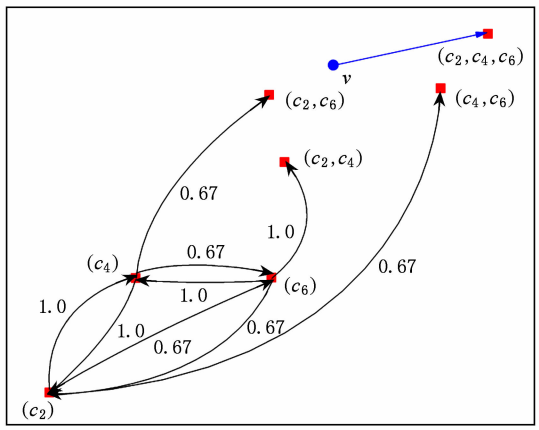

Fig. 3 Add vertex v to Fig. 2.

图 3 对图 2 添加顶点 v

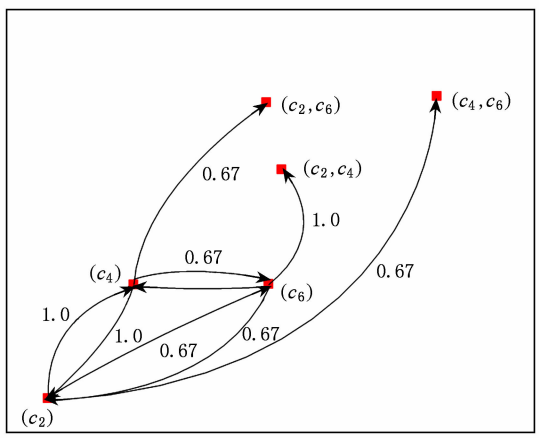

Fig. 4 Delete vertex (c_2,c_4,c_6) from Fig. 2.

图 4 对图 2 删除顶点(c_2,c_4,c_6)

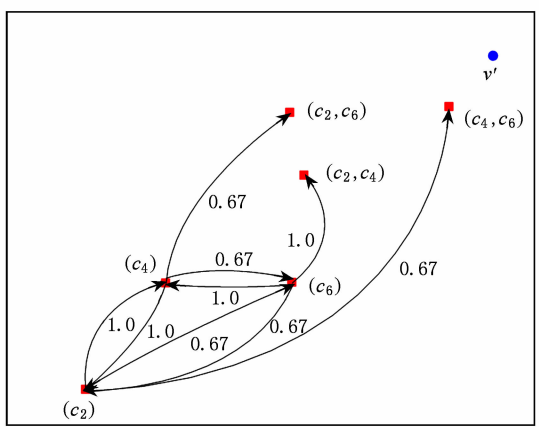

Fig. 5 Substitute vertex (c_2,c_4,c_6) with v' for Fig. 2.

图 5 对图 2 用顶点 v'替换顶点(c_2,c_4,c_6)

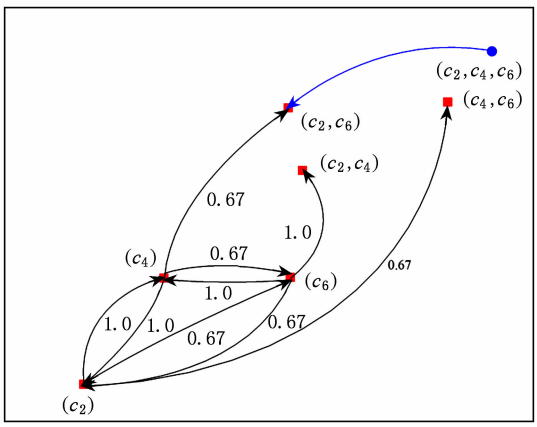

Fig. 6　Migrate vertex (c_2,c_4,c_6) for Fig. 2.

图 6　对图 2 迁移顶点(c_2,c_4,c_6)

1.3.3　波及效应量化分析

本节基于波及效应图从演化活动的波及范围和波及程度 2 方面进行量化分析.

定理 1. 集合$\{add(G_{RE},v),del(G_{RE},v)\}$构成软件演化活动集合的最小完备集.

证明. 要证明$\{add(G_{RE},v),del(G_{RE},v)\}$是软件演化活动集合的最小完备集,就必须证明:1)所有其他演化活动均可以使用 $add(G_{RE},v)$和 $del(G_{RE},v)$表示,即要证明 $sub(G_{RE},v',v)$和 $mig(G_{RE},v)$能被 $add(G_{RE},v)$和 $del(G_{RE},v)$表示. 2)若删除 $add(G_{RE},v)$或 $del(G_{RE},v)$其中任一活动,$sub(G_{RE},v',v)$和 $mig(G_{RE},v)$活动将不能表示.

1) 根据 $sub(G_{RE},v',v)$定义可知,替换活动可视为首先从 G_{RE}中删除 v 以及与 v 存在波及关系的边,然后添加 v'以及与 v'存在波及关系的边. 因此,$sub(G_{RE},v',v)$可认为是 $del(G_{RE},v)$和 $add(G_{RE},v')$活动的复合,可以用 $del(G_{RE},v)$和 $add(G_{RE},v')$表示.

2) 根据 $mig(G_{RE},v)$定义可知,$mig(G_{RE},v)$是 $v'=v$ 情况下 $sub(G_{RE},v',v)$的一个特例. 因此,$mig(G_{RE},v)$也可以用 $add(G_{RE},v')$和 $del(G_{RE},v)$表示;综上所述,$add(G_{RE},v')$和 $del(G_{RE},v)$是软件演化活动的完备集.

3) 根据以上分析删除 $add(G_{RE},v')$或 $del(G_{RE},v)$其中任意之一,$mig(G_{RE},v)$和 $sub(G_{RE},v',v)$均不能表达,因此 $add(G_{RE},v')$和 $del(G_{RE},v)$是软件演化活动集合的最小完备集.　证毕.

根据定理 1 可知,软件演化波及效应分析可以约简为对 $add(G_{RE},v')$和 $del(G_{RE},v)$演化活动波及效应的分析.

定义 13. 设波及效应图为 $G_{RE}=(V,E,L)$,$\forall v_1,v_2\in V$,若 v_1,v_2 间存在通路,记为 v_1Rv_2. v_1,v_2 间的通路长度 $Len(v_1,v_2)$称为 v_1 对 v_2 的影响深度. 若 $v_1=v_2$,则 $Len(v_1,v_2)=0$.

定理 2. 设波及效应图为 $G_{RE}=(V,E,L)$,$\forall v_1,v_2\in V$,若 v_1,v_2 间存在通路,则 $Len(v_1,v_2)\leqslant|V|-1$.

证明. 设 $W=v_0v_1\cdots v_l$ 为 G_{RE}中长度为 l 的通路,且始点为 v_0,终点为 v_l. 若 $l\leqslant|V|-1$,则 W 为满足要求的通路;否则,即 $l>|V|-1$,也就是 W 中的顶点数大于 G_{RE}的顶点数. 则必然存在 s,k,满足 $0\leqslant s<k\leqslant l$,使 $v_s=v_k$,即 W 中存在 v_s 到 v_k 的回路,根据定义 13 可知回路 $Len(v_s,v_k)=0$. 删除回路中除 v_s 以外的所有顶点得到新的通路 $W'=v_0v_1\cdots v_sv_{k+1}v_{k+2}\cdots v_l$,则 W'仍然是以 v_0 为始点、v_l 为终点的通路,且 W'的长度较 W 的长度至少少 1. 若 W'的长度 $r\leqslant|V|-1$,则 W'满足要求,否则对 W'重复上述过程. 由于 G_{RE}是有限图,因此经过有限步后,必得到长度小于等于$|V|-1$ 的通路.　证毕.

定义 14. 设波及效应图为 $G_{RE}=(V,E,L)$,$del(G_{RE},v')$为施加于 G_{RE}上的删除演化活动. 则删除演化活动的波及范围 $dea(G_{RE},v')=\{v\mid Len(v,v')\geqslant1$ 或 $Len(v',v)\geqslant1\}$

定义 15. 设波及效应图为 $G_{RE}=(V,E,L)$,$add(G_{RE},v')$为施加于 G_{RE}上的添加演化活动. 则添加演化活动的波及范围 $aea(G_{RE},v')=\{v\mid Len(v,v')\geqslant1$ 或 $Len(v',v)\geqslant1\}$.

通过波及范围的定义可以计算对某一代码所实施的演化活动所影响到的范围.

定义 16. 设波及效应图为 $G_{RE}=(V,E,L)$,$dea(G_{RE},v')$和 $aea(G_{RE},v')$分别为对节点 v'实施删除和添加演化活动的影响范围. 若 $v\in dea(G_{RE},v')$或者 $v\in aea(G_{RE},v')$,则 v 对 $v'\in V$ 波及程度定义为

$$ImpactDegree(v)=\max\left(id_i\,\middle|\,id_i=\prod_{j=1}^{k}Conf_j\right),i\in[1,n],$$

其中,id_i 是 v 到 v'的第 i 条路径上各边对应置信度的乘积,n 是 v 到 v'的通路数量.

波及程度是一个概率属性,该属性描述了对代码 v'实施的演化活动将导致代码 v 发生相应演化活动的最大概率. 取值越高说明波及效应发生的概率越大. 如图 2 所示,顶点(c_2)对顶点(c_2,c_4)的波及程度为 $0.67\times1.0=0.67$.

对于波及效应图 G_{RE}而言,对某代码实施演化活动不产生波及效应的条件是该代码在波及效应图上是一个孤立点.

定理 3. 设 $G_{RE}=(V,E,L)$ 为波及效应图，对 $v\in V$ 实施演化活动不产生波及效应，当且仅当 v 是 G_{RE} 的孤立点.

证明. 若顶点 v 是孤立点，则 v 无有向边与之连接，即 v 不与其他类(方法、函数)具有波及关系. 删除或添加 v 不会影响其他类(方法、函数)，当然也就不存在波及效应.

若删除和添加 v 不影响其他类(方法、函数)的执行，则说明 v 与其他类(方法、函数)无波及关系，即 v 与其他类(方法、函数)之间不存在边连结，则 v 是孤立点. 证毕.

如图 2 所示，点(c_2, c_4, c_6)就是孤立点，对该点实施演化活动对其他点不会产生波及效应.

2 实 验

由于本文方法与动态方法[5]、基于文本[18]和基于历史演化知识的波及效应分析方法[21]均属于用例驱动的方法，可以采用统一的指标衡量其性能. 在实验部分进行了定量对比. 静态波及效应分析方法[12]是代码驱动的，与本文方法不属于同一类，因此在实验部分对静态方法与本文方法进行了定性比对. 通过对比，完成 2 个问题的研究和分析：

研究问题 1. 与其他方法进行对比证明本文方法的有效性.

研究问题 2. 支持度和置信度参数对本文方法的影响及其作用机理.

2.1 实验过程

2.1.1 实验对象

为保证实验结果的客观性，采用开源软件 jEdit① 作为实验对象，其简要情况如表 6 所示：

Table 6 Brief Introduction of jEdit

表 6 jEdit 简介

Name	Version	Lines of Code	Class Number	Keywords Number
jEdit	4.3	103 896	531	4 372

2.1.2 实验过程

本文实验过程包含 5 个步骤：

1) 生成用例. 从 jEdit 缺陷追踪系统② 中抽取故障用例描述、演化结果信息. 如表 7 所示，*id* 是故障

Table 7 Fault Use Cases Description

表 7 故障用例描述

Serial Number	*id*	Usecase Description	Evolution Result
1	950 961	Folding handling newlines at the start of closed folds	org. gjt. sp. jedit. buffer. BufferAdapter. java; org. gjt. sp. jedit. buffer. BufferChangeListener. Adapter. java; org. gjt. sp. jedit. buffer. JeditBuffer. java; org. gjt. sp. jedit. textarea. BufferHandler. java; org. gjt. sp. jedit. textarea. RangeMap. java
2	1 292 706	Size of file open/save dialogs incorrect on dual displays	org. gjt. sp. jedit. GUIUtilities. java
3	1 538 702	PluginManager Dependency plugins are not activated	org. gjt. sp. jedit. GUIUtilities. java; org. gjt. sp. jedit. PluginJAR. java; org. gjt. sp. jedit. View. java; org. gjt. sp. jedit. gui. ErrorListDialog. ActionHandler. java; org. gjt. sp. jedit. gui. FloatingWindowContainer. java; org. gjt. sp. jedit. jEdit. java; org. gjt. sp. jedit. pluginmgr. ManagePanel. java; org. gjt. sp. jedit. pluginmgr. PluginManager. java; org. gjt. sp. jedit. pluginmgr. Roster. Remove. java
4	1 578 785	Provide a way of reloading changed buffers without prompting	org. gjt. sp. jedit. PerspectiveManager. java; org. gjt. sp. jedit. Buffer. java; org. gjt. sp. jedit. gui. BufferOptions. java; org. gjt. sp. jedit. jEdit. java; org. gjt. sp. jedit. options. BufferOptionPane. java; org. gjt. sp. jedit. options. BufferOptionPane. ActionHandler. java; org. gjt. sp. jedit. options
5	1 638 642	jEdit could use JSplitPane continuous layout	org. gjt. sp. jedit. GUIUtilities. java; org. gjt. sp. jedit. View. java; org. gjt. sp. jedit. browser. BrowserView. java ; org. gjt. sp. jedit. gui. OptionsDialog. java ; org. gjt. sp. jedit. gui. RegisterViewer. java; org. gjt. sp. jedit. help. HelpViewer. java ; org. gjt. sp. jedit. options. AppearanceOptionPane. java; org. gjt. sp. jedit. pluginmgr. InstallPanel. java

① http://www.jedit.org/

② http://sourceforge.net/p/jedit/bugs/search/

Continued (Table 7)

Serial Number	*id*	Usecase Description	Evolution Result
6	1 658 252	C mode incorrect bracket matching in multi-line defines	org. gjt. sp. jedit. syntax. ParserRule. java; org. gjt. sp. jedit. syntax. TokenMarker. java; org. gjt. sp. jedit. syntax. XmodeHandler. java
7	1 676 041	enabled jEdit treats every CamelHump part of a word as a word itself when moving the caret	org. gjt. sp. jedit. TextUtilities. java ; org. gjt. sp. jedit. options. EditingOptionPane. java ; org. gjt. sp. jedit. textarea. TextArea. java ; org. gjt. sp. jedit. textarea. TextAreaMouseHandler. java
8	1 913 979	HistoryTextField: array out of bounds when no history	org. gjt. sp. jedit. search. SearchDialog. java
9	2 696 564	Text Block Selection via Gutter Right-Click	org. gjt. sp. jedit. EditPane. java; org. gjt. sp. jedit. options. GutterOptionPane. java; org. gjt. sp. jedit. textarea. Gutter. java; org. gjt. sp. jedit. textarea. Gutter. MouseHandler. java
10	2 896 464	Plain text copy from HyperSearch results	org. gjt. sp. jedit. search. HyperSearchResults. java; org. gjt. sp. jedit. search. HyperSearchResults. MouseHandler. java; org. gjt. sp. jedit. search. HyperSearchResults. CopyToClipboardAction. java; org. gjt. sp. jedit. search. HyperSearchResults. ToStringNodes. java; org. gjt. sp. jedit. search. HyperSearchResults. ResultTreeTransferHandler. java

用例在缺陷追踪系统中的唯一标识,用例描述是对现象的文字描述,演化结果记录了修复故障用例涉及的类.

2) 获取执行迹. 采用 Flat3① 采集与用例对应的执行迹,将执行迹存储为 JPAD② 格式. 各用例对应执行迹概况如表 8 所示:

Table 8　Execution Trace Description

表 8　执行迹描述

Serial Number	*id*	Class Number in Execution Trace
1	950 961	126
2	1 292 706	110
3	1 538 702	171
4	1 578 785	164
5	1 638 642	187
6	1 658 252	132
7	1 676 041	164
8	1 913 979	148
9	2 696 564	165
10	2 896 464	179

3) 执行迹降噪. 使用 TMG③ 对 jEdit 源代码建立 TF-IDF 矩阵,如表 9 所示. 将每一个用例相对应的用例描述作为查询条件,采用 LSI 计算 TF-IDF 矩阵中与查询条件相似的类集合 S_{DC}. 求执行迹类集合 S_{EC} 与领域知识对应类集合 S_{DC} 的交集,实现执行迹降噪.

4) 生成波及关系. 将经过降噪处理的代码导入 starUML④ 生成类图,存储为 xml 格式数据. 处理该 xml 数据产生波及关系矩阵 $\boldsymbol{K}$,计算 $\boldsymbol{K}$ 的传递闭包,并构造波及效应图;

5) 与动态、基于文本和历史演化知识的波及效应分析方法进行定量比对,与静态波及效应分析方法进行定性比对.

Table 9　TF-IDF Matrix

表 9　TF-IDF 矩阵

Serial Number	*id*	Matrix Dimension (Keywords Number× Class Number)
1	950 961	5210×126
2	1 292 706	4863×110
3	1 538 702	5843×171
4	1 578 785	5954×164
5	1 638 642	6174×187
6	1 658 252	5449×132
7	1 676 041	5804×164
8	1 913 979	5681×148
9	2 696 564	5854×165
10	2 896 464	5974×179

2.2　评价标准

度量波及效应分析方法的好坏是看预测演化活动影响范围是否接近于实际影响范围. 在以往研究中提出多种指标作为量化分析结果好坏的标准[26-27].

① http://www. cs. wm. edu/semeru/flat3/index. html

② http://docs. oracle. com/javase/7/docs/technotes/guides/jpda/jpda. html

③ http://scgroup20. ceid. upatras. gr:8000/tmg/

④ http://staruml. sourceforge. net/en/

在众多的指标中查准率和召回率是使用最广的 2 个指标[26-29]. 其中,查准率反映了预测影响范围与实际影响范围的一致性,召回率反映了预测影响范围在实际影响范围中的占比. 高查准率意味着将花费较少的开销来确定演化活动的波及范围,高召回率意味着凡是通过算法推荐的波及范围均是可信的.

定义 17. 设有限非空集合 S_I,S_C 分别表示各波及效应分析方法检出代码集合和实际演化活动所影响的代码集合. 查准率、召回率分别定义如下:

$$P=\frac{|S_I\cap S_C|}{|S_I|}\times 100\%,$$

$$R=\frac{|S_I\cap S_C|}{|S_C|}\times 100\%.$$

针对本文提出方法,S_I 集合对应于波及效应图的所有顶点集合的并集,实际波及范围由表 7 中演化结果记录.

定义 18. 设 P_{our},R_{our} 分别表示使用本文方法进行波及效应分析的查准率和召回率,P_b,R_b 表示使用动态、基于文本或基于历史演化知识进行波及效应分析的查准率和召回率. 查准率增益、召回率增益分别定义如下:

$$P_{Gain}=\frac{P_{our}-P_b}{P_b}\times 100\%,$$

$$R_{Gain}=\frac{R_{our}-R_b}{R_b}\times 100\%.$$

查准率增益 P_{Gain} 和召回率增益 R_{Gain} 二个参数刻画了本文方法较其他方法的性能增幅.

定义 19. 设 P,R 分别表示查准率和召回率,则调和平均数 F 定义为

$$F=2\times\frac{R\times P}{P+R}\times 100\%.$$

由于查准率和召回率具有互逆关系,无法描述波及效应分析方法的综合性能,本文使用调和平均数 F(F-measure)度量波及效应分析方法的综合性能.

2.3 实验结果

2.3.1 与动态、文本和历史演化知识分析方法比对

当 $Support=0.05$ 时,各方法查准率如图 7 所示,本文方法平均查准率为 28.52%,动态分析方法平均查准率为 4.25%,基于文本的分析方法平均查准率为 5.63%,基于历史演化知识分析方法的平均查准率为 25.61%. 本文方法查准率较动态和文本 2 方法取得了较大提高,基于历史演化知识方法的查准率接近本文方法,特别是 $id=1\,538\,702$ 的错误用例,获得了 71.43%的查准率.

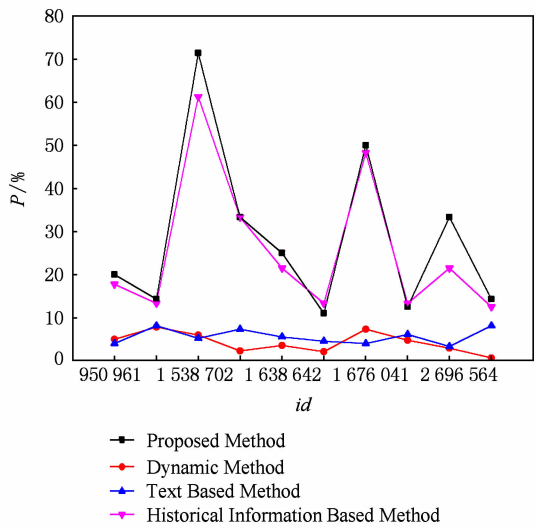

Fig. 7 Precession comparison graph.

图 7 查准率对比图

当 $Support=0.05$ 时,各方法召回率如图 8 所示,本文方法平均 $R=50.07\%$,动态分析方法平均召回率为 73.56%,基于文本的分析方法平均召回率为 71.56%,基于历史演化知识的分析方法平均召回率为 16.83%. 本文方法较基于历史演化知识的波及效应分析方法召回率取得了较大提高,但较其余 2 方法低.

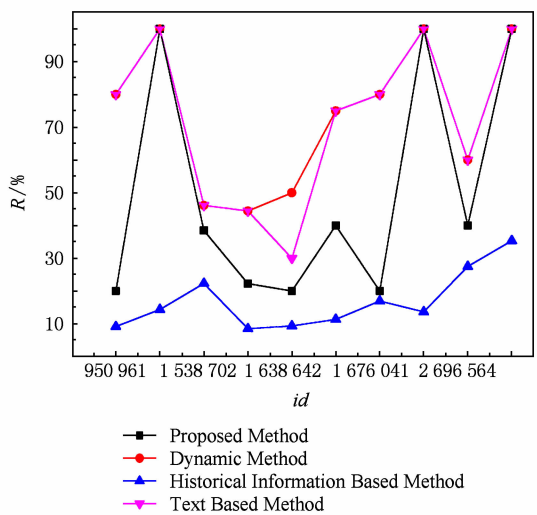

Fig. 8 Recall comparison graph.

图 8 召回率对比图

查准率增益 P_{Gain} 和召回率增益 R_{Gain} 比对结果如图 9 和图 11 所示. 在 $Support=0.05$ 情况下,本文方法相较于动态方法,查准率的平均增益为 760.02%;相较于文本方法,查准率的平均增益为 482.07%;相较于基于历史演化知识方法,查准率的平均增益为 10.19%. 对比动态方法召回率平均增

益为－35.67%,对比文本方法召回率平均增益为－33%,对比基于历史演化知识方法召回率平均增益为219.88%.

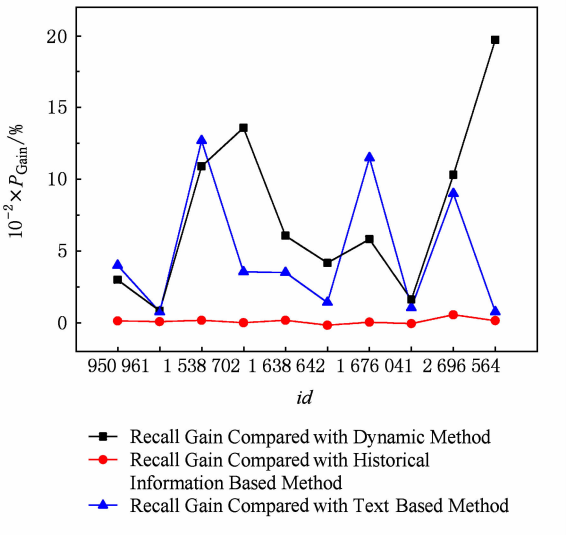

Fig. 9　Precession gain graph.

图 9　查准率增益图

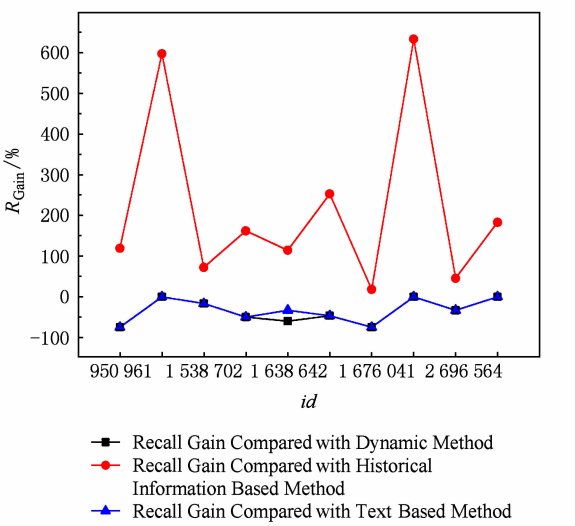

Fig. 10　Recall gain graph.

图 10　召回率增益图

调和平均数 F 比对结果如图 11 所示. 在 $Support=0.05$ 情况下,4 种方法平均调和平均数分别为 36.34%,8.04%,10.46%,20.31%.

2.3.2　与静态方法实验对比

由于演化活动用例具有直观性,同时也较容易观测和记录,因此即便是未经过专业训练的程序员也能使用本方法进行波及效应分析. 而静态波及效应分析方法是代码驱动的分析方法,需要在确定演化活动目标代码的前提下才能进行波及效应分析. 在缺乏有效软件设计文档、用户手册等数据支持的

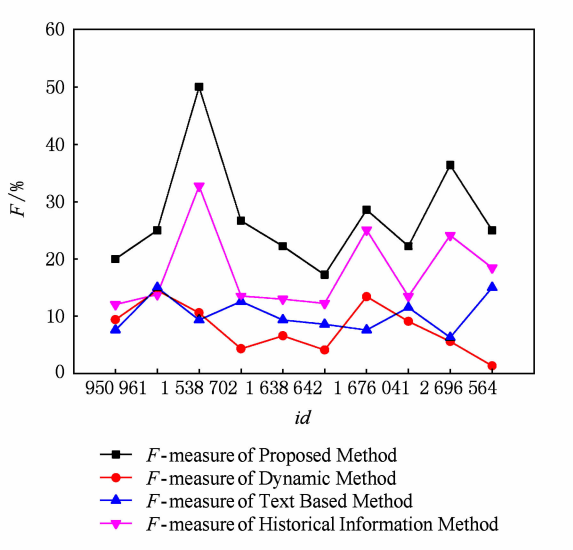

Fig. 11　F-measure graph.

图 11　调和平均数图

情况下,要明确演化活动的目标代码将是非常困难的,后继的波及效应分析也将难以实现. 这导致了本文提出方法较静态波及效应分析方法更贴近实际应用.

同时,静态波及效应分析方法需要将软件系统的所有源代码作为输入,构造可达图或可达矩阵实现波及效应分析. 整个分析过程随着代码量的增长呈指数级增长. 笔者采用文献[13]提出的可达矩阵对 jEdit4.3 进行波及效应分析. 结果显示 jEdit4.3 对应的可达矩阵是一个 531×531 的方阵,矩阵的生成时间(不含数据预处理)为 13 min 22 s(计算平台 CPU:Intel i5-4200,内存:4 GB,操作系统:Windows8). 由此可见静态波及效应分析方法对类似 jEdit 这样中小规模的软件系统分析有较大的计算开销. 而本文方法由于是对经过降噪的执行迹进行分析,在相同计算平台下,构造波及效应图计算时间(不含数据预处理时间)为 3 min 13 s. 因此本文方法较静态方法具有更高的效率.

2.3.3　波及效应量化分析

动态、静态、基于文本和基于历史演化知识的波及效应分析方法大多不具备量化分析能力. 在实验中为每一个用例建立了一个波及效应图,通过对波及效应图的分析可实现对任意演化活动波及范围的量化分析. 图 12 给出了故障用例 950 961 对应的波及效应图,图 12 中顶点颜色越深表示执行演化活动时越容易受波及. 表 10 给出了实施每一个演化活动时最容易受到波及的代码集合及其波及程度量化结果.

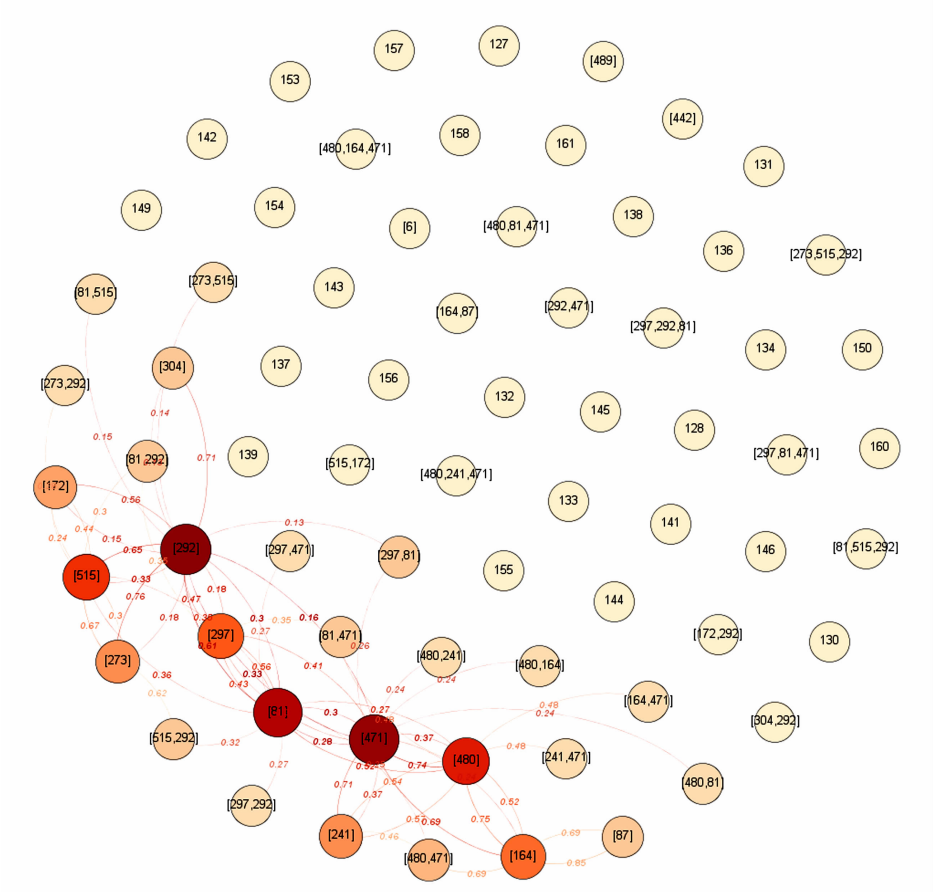

Fig. 12 Ripple effect graph of failure use case 950 961.

图 12 故障用例 950 961 对应的波及效应图

Table 10 Quantitive Ripple Effect Analysis for Each Error Use Cases

表 10 对每一错误用例的定量波及效应分析

id	The Most Trustwrothy Ripple Effect	Confidence
950 961	jedit. org. gjt. sp. jedit. textarea. BufferHandler. java→jedit. org. gjt. sp. jedit. textarea. DisplayManager. java	0.85
1 292 706	jedit. org. gjt. sp. jedit. browser. VFSFileNameField. java→jedit. org. gjt. sp. jedit. browser. VFSBrowser. java	0.94
1 538 702	jedit. org. gjt. sp. jedit. gui. DockableWindowFactory. java→jedit. org. gjt. sp. jedit. jEdit. java	0.94
1 578 785	jedit. org. gjt. sp. jedit. bufferset. BufferSetManager. java→jedit. org. gjt. sp. jedit. View. java	0.94
1 638 642	jedit. org. gjt. sp. jedit. ServiceManager. java→jedit. org. gjt. sp. jedit. jEdit. java	0.87
1 658 252	jedit. org. gjt. sp. jedit. bsh. BSHCastExpression. java→jedit. org. gjt. sp. jedit. syntax. TokenMarker. java	0.87
1 676 041	jedit. org. gjt. sp. jedit. textarea. FirstLine. java→jedit. org. gjt. sp. jedit. syntax. TokenMarker. java	0.74
1 913 979	jedit. org. gjt. sp. jedit. options. BrowserOptionPane. java→jedit. org. gjt. sp. jedit. indent. RegexpIndentRule. java	0.91
2 696 564	jedit. org. gjt. sp. jedit. bufferset. BufferSetAdapter. java→jedit. org. gjt. sp. jedit. syntax. TokenMarker. java	0.71
2 896 464	jedit. org. gjt. sp. jedit. pluginmgr. MirrorList. java→jedit. org. gjt. sp. util. IntegerArray. java	0.72

2.3.4 支持度、置信度参数的影响

为了验证支持度、置信度对本文方法的影响，选取故障用例 950 961 进行测试，在所有操作均相同的前提下，分析改变支持度和置信度的取值对查准率、召回率以及波及关系生成数量的影响. 详细结果如表 11、表 12 所示.

从表 11 可以看出，当支持度从 0.05 开始下降时，代码检出量增加，导致查准率持续降低，召回率保持不变；当 $Support=0.02$ 时，查准率增加至 13.33%，召回率也同时增加至 40%；随着支持度继续降低，导致查准率持续降低，召回率持续提高，并最终达到 100%. 通过实验可知，$Support=0.02$ 是

一个最优取值.在该取值下可以获得较高的查准率和召回率.从表12可以看出,当置信度持续降低,产生波及关系的数量不断增加.

Table 11　Relationship Between Support Precision and Recall

表11　查准率、召回率与支持度取值的关系

id	*Support*	*P*	*R*
950961	0.0500	0.2000	0.2000
	0.0400	0.1250	0.2000
	0.0300	0.0909	0.2000
	0.0200	0.1333	0.4000
	0.0150	0.1053	0.4000
	0.0100	0.0713	1.0000

Table 12　Relationship Between the Amount of Ripple Effect Relationship and Confidence

表12　波及关系数量和置信度取值的关系

id	Belief	Number of Association Relation
950961	0.9	0
	0.8	1
	0.7	6
	0.6	13
	0.5	19
	0.4	27
	0.3	39
	0.2	53
	0.1	61

通过分析可知,支持度取值将影响本文方法的查准率和召回率,伴随着支持度的降低,将导致查准率的降低、召回率的增高,但查准率的降低并不是一个线性过程.因此,支持度的取值需要对演化软件具有一定的了解.置信度取值将影响对波及效应进行量化分析,取值越小越能够对小概率波及效应进行量化分析.

3　结　　论

本文提出的混合波及效应分析方法,通过与动态、静态、基于文本和基于历史演化知识的波及效应分析方法进行比对实验,结果显示,本文方法是一种以适当牺牲召回率为代价从而大幅提高查准率的新方法.通过对调和平均数的比对,显示本文方法具有综合性能好的特点.

针对影响本文方法的2个重要因素:支持度和置信度,本文进行了影响机制分析.我们注意到,支持度设置将影响波及效应分析的查准率、召回率,进而影响查准率增益、召回率增益以及调和平均数.该参数的有效取值依赖于对软件系统的了解程度.置信度设置将影响对波及效应的量化分析,取值应该尽量小,以满足量化小概率波及效应.

在下一步的工作中,将对支持度、置信度参数自动设置问题展开研究工作,拟实现在提高查准率的同时,将召回率保持在一个合理的范围内.另外,为了验证本文方法的普适性,将对更多的开源项目进行验证.

参　考　文　献

[1] Li B, Sun X, Leung H, et al. A survey of code-based change impact analysis techniques [J]. Software Testing Verification & Reliability, 2013, 23(8): 613-646

[2] Dagenais B, Robillard M P. Using traceability links to recommend adaptive changes for documentation evolution [J]. IEEE Trans on Software Engineering, 2014, 40(11): 1126-1146

[3] Hill E, Pollock L, Vijay-Shanker K. Exploring the neighborhood with Dora to expedite software maintenance [C] //Proc of the 22nd IEEE/ACM Int Conf on Automated Software Engineering. New York: ACM, 2007: 14-23

[4] Law J, Rothermel G. Whole program path-based dynamic impact analysis [C] //Proc of the 25th Int Conf on Software Engineering. Piscataway, NJ: IEEE, 2003: 308-318

[5] Orso A, Apiwattanapong T, Law J, et al. An empirical comparison of dynamic impact analysis algorithms [C] //Proc of the 26th Int Conf on Software Engineering. Piscataway, NJ: IEEE, 2004: 491-500

[6] Ren X, Shah F, Tip F, et al. Chianti: A tool for change impact analysis of Java programs [C] //Proc of the 19th Annual ACM SIGPLAN Conf on Object-Oriented Programming, Systems, Languages, and Applications. New York: ACM, 2004: 432-448

[7] Gethers M, Dit B, Kagdi H, et al. Integrated impact analysis for managing software changes [C] //Proc of the 34th Int Conf on Software Engineering. Piscataway, NJ: IEEE, 2012: 430-440

[8] Seacord R C, Plakosh D, Lewis G A. Modernizing Legacy Systems: Software Technologies, Engineering Processes, and Business Practices [M]. Reading, MA: Addison-Wesley, 2003

[9] Bohner S A. Software change impacts-an evolving perspective [C] //Proc of the 18th Int Conf on Software Maintenance. Piscataway, NJ: IEEE, 2002: 263-272

[10] Bohner S A. Impact analysis in the software change process: A year 2000 perspective [C] //Proc of the 13th Int Conf on Software Maintenance. Piscataway, NJ: IEEE, 1996: 42-51

[11] Malik H, Hassan A E. Supporting software evolution using adaptive change propagation heuristics [C] //Proc of the 24th IEEE Int Conf on Software Maintenance. Piscataway, NJ: IEEE, 2008: 177-186

[12] Wang Yinghui, Zhang Shikun, Liu Yu, et al. Ripple-effect analysis of software architecture evolution based on reachability matrix [J]. Journal of Software, 2004, 15(8): 1107-1115 (in Chinese)
(王映辉, 张世琨, 刘瑜, 等. 基于可达矩阵的软件体系结构演化波及效应分析[J]. 软件学报, 2004, 15(8): 1107-1115)

[13] Hassan A E, Holt R C. Predicting change propagation in software systems [C] //Proc of the 20th IEEE Int Conf on Software Maintenance. Piscataway, NJ: IEEE, 2004: 284-293

[14] Orso A, Apiwattanapong T, Harrold M J. Leveraging field data for impact analysis and regression testing [J]. ACM SIGSOFT Software Engineering Notes, 2003, 28(5): 128-137

[15] Law J, Rothermel G. Incremental dynamic impact analysis for evolving software systems [C] //Proc of the 14th Int Symp on Software Reliability Engineering. Piscataway, NJ: IEEE, 2003: 430-441

[16] Canfora G, Cerulo L. Fine grained indexing of software repositories to support impact analysis [C] //Proc of the 3rd Int Workshop on Mining Software Repositories. New York: ACM, 2006: 105-111

[17] Weiss C, Premraj R, Zimmermann T, et al. How long will it take to fix this bug? [C] //Proc of the 4th Int Workshop on Mining Software Repositories. Los Alamitos, CA: IEEE Computer Society, 2007

[18] Qusef A, Bavota G, Oliveto R, et al. Recovering test-to-code traceability using slicing and textual analysis [J]. Journal of Systems and Software, 2014, 88(2): 147-168

[19] Schrettner L, Jász J, Gergely T, et al. Impact analysis in the presence of dependence clusters using static execute after in WebKit [J]. Journal of Software-Evolution and Process, 2014, 26(6): 569-588

[20] Cheluvaraju B, Nagal K, Pasala A. Mining software revision history using advanced social network analysis [C] //Proc of the 19th Asia-Pacific Software Engineering Conf. Piscataway, NJ: IEEE, 2012: 717-720

[21] Herzig K, Zeller A. Mining cause-effect-chains from version histories [C] //Proc of the 22nd IEEE Int Symp on Software Reliability Engineering. Piscataway, NJ: IEEE, 2011: 60-69

[22] Marcus A, Sergeyev A, Rajlich V, et al. An information retrieval approach to concept location in source code [C] //Proc of the 11th Working Conf on Reverse Engineering. Piscataway, NJ: IEEE, 2004: 214-223

[23] Ju Xiaolin, Jiang Shujuan, Zhang Yanmei, et al. Advances in fault localization techniques [J]. Journal of Frontiers of Computer Science and Technology, 2012, 6(6): 481-494 (in Chinese)
(鞠小林, 姜淑娟, 张艳梅, 等. 软件故障定位技术进展[J]. 计算机科学与探索, 2012, 6(6): 481-494)

[24] Liu Bing, Hsu W, Ma Yiming. Integrating classification and association rule mining [C] //Proc of the 4th Int Conf on Knowledge Discovery and Data Mining. Menlo Park, CA: AAAI, 1998: 1-7

[25] Li Yunchang, He Pinjie, Li Yulong. Techniques for Software Dynamic Evolution [M]. Beijing: Peking University Press, 2007 (in Chinese)
(李云长, 何频捷, 李玉龙. 软件动态演化技术[M]. 北京: 北京大学出版社, 2007)

[26] Arnold R S, Bohner S A. Impact analysis-towards a framework for comparison [C] //Proc of the 9th Int Conf on Software Maintenance. Piscataway, NJ: IEEE, 1993: 292-301

[27] Hattori L, Guerrero D, Figueiredo J, et al. On the precision and accuracy of impact analysis techniques [C] //Proc of the 7th IEEE/ACIS Int Conf on Computer and Information Science. Piscataway, NJ: IEEE, 2008: 513-518

[28] Poshyvanyk D, Marcus A, Ferenc R, et al. Using information retrieval based coupling measures for impact analysis [J]. Empirical Software Engineering, 2009, 14(1): 5-32

[29] Sun X, Li B, Tao C, et al. Change impact analysis based on a taxonomy of change types [C] //Proc of the 34th Annual Conf on Computer Software and Applications. Piscataway, NJ: IEEE, 2010: 373-382

Wang Wei, born in 1979. PhD, associate professor of Yunnan University. Member of China Computer Federation. His research interests include software evolution, and formal method.

Li Tong, born in 1963. PhD, professor of Yunnan University. Member of China Computer Federation. His research interests include software evolution and formal method.

He Yun, born in 1989. Master. Student member of China Computer Federation. His research interests include software engineering and software evolution.

Li Hao, born in 1970. PhD, professor of Yunnan University. His research interests include software engineering and cloud computing.