软件修改影响分析研究与进展

孙小兵[1,2],李 斌[1,2],陈 颖[1],李必信[3],文万志[4]
(1.扬州大学信息工程学院,江苏扬州 225127;2.计算机软件新技术国家重点实验室(南京大学),江苏南京 210023;
3.东南大学计算机科学与工程学院,江苏南京 211189;4.南通大学计算机科学与技术学院,江苏南通 226019)

摘 要: 软件修改是软件维护的基本元素,对软件的任何修改会对软件的其他部分造成一些潜在的负面影响.软件修改影响分析就是用来识别软件修改可能带来的潜在影响.修改影响分析在软件维护、回归测试等方面都有着重要的应用.本文对近年来的修改影响分析技术进行调查,根据调查结果对修改影响分析技术进行分类和总结,为软件开发和维护人员选择适合的修改影响分析技术提供参考.通过对调查结果的分析,首先,提出了修改影响分析可从分析类型、方法、所支持的语言范型、阶段、层次五个角度进行分类;其次,对软件修改影响分析进行了展望,探讨了修改影响分析在基础理论、工具支持、评价机制、可拓展性、跟踪性等方面还需进一步深入的研究.

关键词: 软件修改; 修改影响分析; 综述
中图分类号: TP311 **文献标识码:** A **文章编号:** 0372-2112 (2014)12-2467-10
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2014.12.019

A Survey of Software Change Impact Analysis Techniques

SUN Xiao-bing[1,2], LI Bin[1,2], CHEN Ying[1], LI Bi-xin[3], WEN Wan-zhi[4]
(1. *School of Information Engineering, Yangzhou University, Yangzhou, Jiangsu* 225127, *China*;
2. *State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, Jiangsu* 210023, *China*;
3. *School of Computer Science and Engineering, Southeast University, Nanjing, Jiangsu* 211189, *China*;
4. *School of Computer Science and Technology, Nantong University, Nantong, Jiangsu* 226019, *China*)

Abstract: Software change is a fundamental ingredient of software maintenance. Changes made to software will inevitably cause some unpredicted and undesirable effects to other parts of the software. Software change impact analysis (CIA) is a technique for identifying the potential consequences of a change. It is found that CIA is very helpful in many aspects, including software maintenance, regression testing, etc. This paper attempts to give a survey of CIA techniques in recent years, and aims at providing references for developers to choose suitable CIA techniques. In this paper, we first introduce five classification approaches for CIA techniques. Then, we discuss some future challenges and directions for CIA in its fundamental theories, tools, evaluation, scalability, traceability, etc.

Key words: software change; change impact analysis; survey

1 引言

软件的固有特性就是适应性和修改性,修改可以是因为用户提出新的需求,软件使用过程中发现的错误,或是因为软件所使用的环境发生变化.对软件进行修改时,肯定会对软件的其他部分造成一些潜在影响,从而带来软件的不一致性.软件修改影响分析,简称影响分析,就是用来识别软件修改对软件其他部分带来的潜在影响[1].

软件修改实施前,需要进行修改影响分析,预估修改给软件带来的影响,辅助项目经理和开发人员评估和实施修改[1,2].修改影响分析结果可广泛应用在程序理解、程序调试、软件测量与度量、回归测试等领域[1~4].由于修改影响分析的重要性,近 20 年来,研究人员在这方面进行了大量的研究,在软件不同工件层次产生了各种修改影响分析技术,如需求和设计层次的静态影响分析、基于历史库的影响分析、动态影响分析等[1].而开发人员和维护人员在面对这些修改影响分析技术时,如

收稿日期:2013-04-27;修回日期:2014-04-16;责任编辑:梅志强
基金项目:国家自然科学基金(No.61402396,No.61472344);江苏省教育厅自然科学资金项目(No.13KJB520027);江苏省"六大人才高峰"项目(No.2011-DZXX-032);南京大学计算机软件新技术国家重点实验室开放基金(No.KFKT2014B13);扬州大学科技创新培育基金(No.2013CXJ025).

何根据具体的环境选择合适的修改影响分析技术和工具成了问题.另外,修改影响分析技术经过近 20 年的研究,其基础理论是否完善、是否存在一些成熟的修改影响分析工具支持,是否具有可信的、标准的评价机制等,这些都是修改影响分析技术发展需要解决的问题[5].因此,一方面,需要对近 20 年来出现的各种软件修改影响分析技术进行全面的调查和分类,这样可为实践人员在面对实际软件维护环境时选择所需的修改影响分析技术提供参考;另一方面,探讨软件修改影响分析领域还存在的问题,为研究人员的进一步研究提供建议.

2 修改影响分析基础

2.1 修改影响分析基本概念

在软件演化过程中,软件修改必不可少.而软件修改影响分析是软件修改过程中最重要的活动之一[1,2,6].修改影响分析过程的输入是修改集,经过修改影响分析,计算影响集.一些相关的基本概念如下:

定义 1(修改集) 修改集是用户所提出的修改请求的集合或是软件中要修改的元素集合,修改集可以是高层次的需求或设计规约,也可以是代码层次的类、方法或变量.

定义 2(影响集) 影响集是受到修改集中元素潜在影响的元素集合.影响集中的元素可以是受到修改影响的文档规约、代码,也可以是受到影响的测试用例.

2.2 修改影响分析常用方法

修改影响分析经过 20 余年的发展,越来越成熟,常用的修改影响分析技术如下:

(1)基于模型的影响分析[7~10].它依赖于跟踪抽象层的模型中各元素间的依赖关系进行影响分析,基于模型的影响分析通常处于需求层或设计层,不需接触源代码,而如下的一些影响分析技术则要提供源代码,基于源代码进行分析.

(2)基于静态语法依赖关系的影响分析[11~16].它首先构造源程序的依赖关系中间表示,通常用图建模,然后遍历图,在图中只要从修改元素节点处可达的元素都被认为是潜在影响的元素.

(3)基于耦合关系的影响分析[17~20].该技术依赖于一些耦合度量的计算,如程序的结构耦合、概念耦合等;修改影响分析就是识别那些与修改集中元素有耦合关系的元素作为影响集.

(4)基于执行的影响分析[21~23].它是通过执行程序以收集程序运行时的相关信息,如轨迹信息、覆盖信息以及执行关系等来进行影响分析.

(5)基于修改历史库挖掘的影响分析[24~28].该技术依赖于程序的历史库信息,提取程序元素间的修改耦合关系;然后根据这些程序元素在历史库中的修改耦合关系预测当前修改带来的影响.

2.3 修改影响分析评价

评价修改影响分析技术的有效性目前主要考虑精确性和安全性.如下两个概念通常用来计算影响分析技术的精确性和安全性:

定义 3(预估影响集[EIS]) 预估影响集就是通过某种影响分析技术计算的影响集.它可能包括那些实际并没有受到影响的元素,也可能遗漏那些实际受影响的元素.

定义 4(实际影响集[AIS]) 实际影响集是一个理想性的概念,即真正受到影响的元素集合.这些元素往往要进行二次修改、验证或再测试.

修改影响分析的目标就是计算一个与实际影响集 AIS 比较接近的预估影响集 EIS.当前用于评价修改影响分析准确性使用最多的是信息检索领域的两个度量指标[29]:查准率(precision)与查全率(recall),定义如下:

$$\text{Precision} = \frac{|\text{AIS} \cap \text{EIS}|}{|\text{EIS}|} \quad \text{Recall} = \frac{|\text{AIS} \cap \text{EIS}|}{|\text{AIS}|}$$

查全率用于衡量修改影响分析技术是否丢失一些有用信息,如一些实际受影响的元素没有被修改影响分析技术预测到;而查准率用于衡量修改影响分析技术是否产生了大量的没有受到影响的元素.如果查全率越高,说明丢失的有效的影响元素越少,这样越有利于提高维护人员对预测结果的信心;而如果查准率越高,维护人员能够高效地进行修改的实施和波动效应的检查.因此,一个好的修改影响分析技术应该准确地预测全部真正受到修改影响的元素.

3 修改影响分析分类

修改影响分析经过 20 余年的发展已产生了大量的研究成果.本文从 5 种不同的视角对修改影响分析技术进行分类(见表 1).

从修改影响分析类型看,可分为静态影响分析、动态影响分析和在线影响分析.静态影响分析考虑了程序的所有可能的行为,该方法比较保守、安全,得到的影响集规模较大,从而带来不精确性.而动态影响分析考虑程序的实际运行环境,只考虑程序的部分输入,计算的影响集规模较小,从而比较精确.另外,有些学者对动态影响分析做了一些改进,提出在线影响分析方法,它与动态影响分析的区别是,动态影响分析是在程序执行完再分析程序运行时所收集的信息计算影响集;而在线影响分析是在程序执行过程中,在收集程序运行相关信息的同时计算其影响集.

从修改影响分析的方法来看,目前有基于跟踪性

和基于依赖关系的分析[12,29].基于跟踪性的分析方法从两方面考虑系统元素间的关系,一是同一层次内,如需求、设计、代码与测试,考虑各层次上元素间的关系;二是在不同层次间,寻找某一层次的元素在其他层次所对应的元素,建立层次之间的元素跟踪关系.基于依赖关系的分析是通过分析程序的语法关系来获得程序元素之间的语义依赖关系.目前大多影响分析方法采用基于依赖关系的分析,这些依赖关系可以是代码层的类、方法或语句之间的依赖关系,也可以是需求层或设计层上模型元素间的依赖关系.

表 1　修改影响分析分类

视角	类型	相关参考文献
分析类型	静态	[12,13,24,25,27~29,43]
	动态	[21,22,23,30,45]
	在线	[50,51]
分析方法	基于跟踪性	[30,32~35]
	基于依赖关系	[7~9,14,22,46,53,55]
分析支持语言	面向过程语言	[21-23]
	面向对象语言	[46,52,53,56]
	面向方面语言	[28,38,39]
	面向服务语言	[40,41]
分析阶段	需求	[39~41]
	设计	[7~9]
	编码	[13,21~24,26~28,42~44,52]
分析层次	基于模型	[7~9,39~41]
	基于代码	[22~24,26,27,35~38,42~44]

从修改影响分析所在的层次来看,影响分析可在抽象的模型层,也可在代码层进行.当在抽象的模型层分析时,影响分析通常处于需求或设计阶段,借助于统一建模语言、对象约束语言、或用例图进行分析;当影响分析在代码层分析时,可通过提取程序元素间的依赖关系计算某个修改所带来的潜在影响.由于代码层所提供的信息更丰富,从而计算的影响集也更加精确.

从程序设计和语言规范来看,影响分析可以是针对面向过程语言、面向对象语言、面向方面语言、或是面向服务的语言.针对各种不同语言的程序,分析不同语言的具体特性提取相应的关系进行分析.对于新出现的语言,如面向方面的语言,可构造新的模型或依赖图,也可通过扩充传统的模型进行分析.

另外,影响分析可处于不同的阶段,如需求阶段、设计阶段、代码阶段.其他一些学者给出了与此类似的分类方法,他们从开发与管理人员的角色进行分类,这些角色包括需求工程师、项目规划人员、设计者、维护人员、以及测试人员[30].对于需求工程师来说,他们使用影响分析来维护文档;对于项目规划人员,他们使用影响分析来理解修改需求,决定是否接受这次修改请求;对于设计者,通过影响分析,他们负责安排和分配修改任务;而维护人员则负责具体的修改任务以及执行修改传播分析;对于测试人员,他们根据影响分析结果选择哪些测试用例进行回归测试,以及增加哪些测试用例覆盖旧的测试用例未覆盖的修改部分和影响部分,从而保证程序的一致性.

因此,从不同的角度都可以对修改影响分析技术进行分类,而这些修改影响分析分类角度不是相互排斥的.本文考虑各种修改影响分析分类间的关系,得到如图 1 所示分类框架.本文将按如下的分类框架介绍各种修改影响分析技术.首先,将修改影响分析分为基于跟踪性的分析和基于依赖关系的分析;然后,将基于依赖关系的分析分为基于模型的影响分析和基于代码的影响分析;最后,对基于代码的影响分析进行分类,分为静态影响分析、动态影响分析、面向对象影响分析、面向方面影响分析、以及面向服务影响分析.

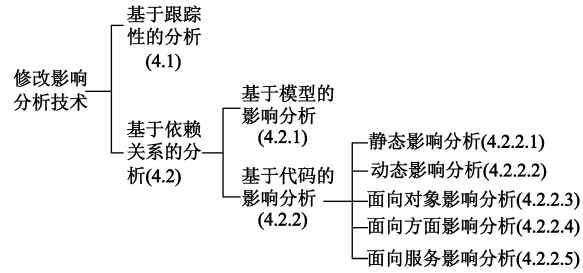

图1　修改影响分析技术分类框架

4　修改影响分析技术

4.1　基于跟踪性的修改影响分析

跟踪性指的是“能从前向和后向两个方向跟踪某个组件在生命周期各个阶段所对应的元素”[31].对软件所做的修改不仅影响到软件的源代码部分,还影响到需求文档、设计文档以及测试用例等.基于跟踪性的影响分析能够跟踪同一层次内各元素间的关系,也能跟踪某一层次的某个元素在另一个层次所对应的元素.当用户提出某个修改时,通过这种跟踪性分析,可将各个层次受到潜在影响的元素都识别出来.

在文献[32]中,Strens 等人提出的修改分析技术,其中包括修改影响分析.他们利用敏感性分析来预测两种类型的风险:哪些需求不稳定,以及在设计层中哪些领域易受需求的修改而受到影响.在修改影响分析过程中,将需求修改和设计规范作为输入,通过影响分析,计算出来的结果包括:设计层受影响的元素、产生的冲突、波动效应以及对成本的影响.

Fyson 等人提出一种新的中间表示方法——效应传

播图用于影响分析[33].该图能捕获某一阶段内各元素间的依赖关系,也能跟踪各阶段间元素间的关系.该技术可识别需求层受影响的文档,代码层受影响的代码,以及设计层受影响的元素.

O'Neal 在其博士论文中也提出一种基于跟踪性的修改影响分析方法[34].首先利用需求跟踪关系确定受影响的元素;然后定义一种度量参数计算每个需求更改的严重程度;最终产生的结果是按严重程度排序的各个需求的修改.有了这样的度量,管理人员可评价实施各个修改的风险.

Briand 等人从另一组角度进行修改影响分析[35].他们首先识别某个抽象层次的修改,然后计算其对精化层次元素的影响.这个精化层次捕获了更高抽象层次的用户意图.他们先建立这两个层次各元素间的跟踪关系,然后通过识别这种跟踪关系在精化层次上受影响的元素.

Lucia 等人从影响分析的角度阐述了不同的跟踪管理技术[36].他们将跟踪管理方法分为 3 类:纵向和横向的跟踪性关系,模糊和明确的跟踪性关系,基于结构和基于知识的跟踪性关系.他们还从这 3 个角度给出了相对应的影响分析方法.

4.2 基于依赖关系的修改影响分析

基于依赖关系的影响分析方法是目前修改影响分析最常用的技术,它利用某种程序分析技术获得程序实体间的语义关系[37].这种影响分析方法可基于抽象模型,也可基于代码.我们将从这两方面介绍目前基于依赖关系的影响分析方法.

4.2.1 基于模型的修改影响分析

修改影响分析可应用于需求或设计层次,这样管理人员可更早地做出修改决策.这些分析无需接触源代码,直接依赖于抽象的模型,该方法称为基于模型的影响分析.在需求层,影响分析通常基于用例图(UCM)作为中间模型来分析;在设计层,统一建模语言(UML)和对象约束语言(OCL)常作为中间模型来分析.表 2 列举了各种基于模型的影响分析技术,主要从他们的分析层次,中间模型进行比较.

表 2 基于模型的影响分析技术

层次	中间模型	文献
设计层	UML & OCL	[7]
从设计到代码层	UML & 运行时调用图	[38]
规约层	UCM	[39]
规约层	UCM	[40]
需求层	UCM & FCA	[41]

Briand 等人在文献[7]中提出了利用 UML 作为中间表示进行影响分析的方法.该方法首先验证 UML 图间的一致性;然后识别修改前和修改后的 UML 中被修改的元素;另外,还利用 OCL 定义一系列影响规则,而影响分析正是基于这些约束规则来计算影响集结果.

Chen 等人从工程管理的角度提出一种基于对象、面向属性的影响分析方法[38].该方法可处理软件各产品的修改,如软件组件、需求、设计文档、程序实体等,而且所计算的影响集元素也是各种不同的产品.他们的影响分析方法主要利用 UML 和运行时的调用图来计算其影响集.

另外,还有其他一些基于模型的影响分析是在需求或规约层进行.Hassine 等人在规约层提出了识别需求修改所带来的潜在影响的方法[39],他们从 UCM 中提取规约信息进行预测性影响分析.在需求层,修改集中元素可以是某个组件、职责或一个场景,而与元素相关的场景可用来构造影响分析所需的依赖关系.

Hewitt 等人将前向切片技术与依赖关系分析技术组合起来进行影响分析[40].首先依赖关系是根据规约信息进行分析;然后利用前向切片来计算切片,而所得到的切片结果即为影响集.

Shiriet 等人将用例图和形式概念分析(FCA)结合起来进行影响分析[41].他们首先利用建模技术使得需求模型可以执行;然后通过执行用例图场景构造相应的轨迹;再通过形式概念分析收集这些轨迹信息,构造相应的运行依赖关系.其所计算的影响集包括与轨迹修改相关的影响集,受影响的测试用例和用例场景.

基于模型的影响分析技术常应用在体系结构层次,这样可帮助管理人员更早地预测修改的可行性,修改带来的影响,所需修改成本、资源等.

4.2.2 基于代码的修改影响分析

基于代码的影响分析是在软件最终产品——源代码上进行.当知道详细修改计划时,在代码层次计算的影响集会更精确.代码层次的影响分析可以是静态的,也可以是动态的,还可是混合类型的,即静态与动态的组合.表 3 是一些基于代码层次的影响分析技术及其比较,这些影响分析技术从分析的粒度层次、使用的中间表示、所支持的语言、分析类型这几方面进行比较.

4.2.2.1 静态影响分析

如前所述,静态影响分析考虑了程序的所有可能输入和行为,是比较保守和安全的方法.静态修改影响分析通过静态分析程序(或其演化历史库)的语法和语义关系,建立程序的中间表示,如调用图、程序依赖图等;然后根据程序中间表示上的依赖关系进行影响分析.传统的静态影响分析方法主要有调用图上的传递闭包计算和静态切片计算两种方法[42~46].这部分将介绍最近出现的一些静态影响分析技术.

表 3　基于代码的影响分析技术

粒度层次	中间表示	支持语言	类型	文献
类,类成员	面向对象数据依赖图	面向对象	静态	[52]
方法	完整的路径有向无环图	面向过程	动态	[21]
测试用例	静态或动态调用图,语法依赖关系	面向对象	静态 + 动态	[53,54]
方法	控制调用图,控制流路径	面向过程,面向对象	静态	[13]
方法	执行后关系	面向过程,面向对象	动态	[22]
方法	程序依赖图,覆盖位向量	面向过程,面向对象	静态 + 动态	[23]
方法	静态面向方面调用图	面向方面	静态	[28]
类成员	动态依赖图	面向对象	动态	[55]
业务组件,方法	调用图,传播图	面向服务	静态	[40]
语句	面向方面系统依赖图	面向方面	静态	[39]
方法		面向过程,面向对象	在线	[50,51]
语句	分解切片概念格	面向过程	静态	[47]
方法	影响图	面向过程	静态	[56]
代码行,源文件	历史存储表	面向过程,面向对象	静态	[24]
源代码	面向方面交互图	面向方面	动态	[57]
包、类、方法、语句	层次依赖图	面向对象	静态	[14]
类与类成员	面向对象类与成员依赖图	面向对象	静态	[59]
类与方法	类与方法依赖格	面向对象	静态	[60]
方法	类与方法依赖格,调用图	面向对象	静态	[15]

Badri 等人提出一种利用调用图计算更加精确的影响集的方法[13].他们对调用图进行了改进,称为控制调用图.该控制调用图无需运行程序,就能捕获与组件调用相关的控制流路径;而影响分析就是基于该控制流路径计算,所计算的影响集可到达方法粒度层次,且比传统的基于调用图的影响分析更精确.

Tonella 等人提出一种过程内的影响集计算方法[47].他们利用分解切片概念格作为中间表示进行影响分析.他们所提出的方法目前还仅局限于过程内语句修改的影响集计算.

程序中某个元素的修改可能会传播给其他元素,因此可以研究程序元素的传播机制进行影响分析.Breech 等人归纳了这些影响机制,并利用这些影响机制构造了一个称为影响图的中间表示;然后基于该影响图计算影响集.通过比较分析,他们所计算的影响集精确性得到很大改进.

软件修改历史库中存在大量的程序依赖信息,通过数据挖掘的方法可将这些依赖关系挖掘出来进行分析是目前一种流行的方法.例如,CVS 版本控制系统是一个支持团队开发的版本控制系统,其记录了团队中各成员对程序的修改历史.Jashki 等人从软件修改历史库中挖掘出各个源文件的修改频率,然后存储于一个矩阵,构造程序文件的聚类,再通过该聚类计算影响集[25].该影响分析方法有效降低了影响分析的复杂度,且计算影响集的过程也更高效.Canfora 等人也利用挖掘修改历史库信息的方法进行影响分析[24].他们的影响分析在两个层次进行:代码层和源文件层.通过计算一个新的修改请求和源代码实体间的相似性来进行影响分析,其中那些相似度最大的程序实体将作为最易受影响的部分.另外,Hattori 等人提出了一种混合的方法计算影响集[26].他们首先利用普通的修改影响分析方法计算一个初始影响集;然后对初始影响集中每个元素计算其与修改集中元素在程序修改历史库中的相关性.这样的混合方法有效地提高了影响集的精确性.

4.2.2.2　动态影响分析

由于静态影响分析技术的不精确性,一些学者提出了动态影响分析方法.而动态影响分析主要通过运行软件系统,收集程序运行时的相关信息,如执行轨迹信息、覆盖信息、执行关系信息等;然后分析这些信息进行影响集的计算.通常动态影响分析的结果规模较小,也比较精确;不过其所需的成本代价较高,安全性得不到保证,即不能覆盖程序中一些真正受修改影响的部分.

Law 等人提出一种利用完整的路径剖面图获得程

序运行时轨迹的方法(PathImpact)[21].首先,他们利用压缩算法处理这些轨迹,得到完整的有向无环路径图[48];然后对该图进行前向遍历和后向遍历,前向遍历和后向遍历这两部分所得的结果组成最终的影响集.这种动态影响分析的方法能够有效提高影响集的精确性,但所需代价较高.

PathImpact 收集动态信息时有些信息是冗余且重叠的,从而需大量的空间来存储这些路径信息,Apiwattanapong 等人提出一种只需存储动态影响分析所需的信息,然后对这些信息进行分析的方法(CollectEA)[22].他们提出执行后关系这一概念,该技术的核心就是收集程序运行时各个方法的执行后关系,然后基于这种执行后关系进行影响分析.CollectEA 方法成本较小,不过不如 PathImpact 方法计算的影响集精确.

Orso 等人通过收集程序的覆盖信息进行影响分析(CoverageImpact)[23].首先,将那些覆盖修改集元素路径中所有的方法添加到初始影响集中;然后计算修改方法的前向切片,取他们的交集,该交集即为影响集.一些学者对这三种动态影响分析技术进行了实验比较[49].CollectEA 与 PathImpact 方法所计算的影响集比 CoverageImpact 更精确;而 CoverageImpact 所需的成本最低,CollectEA 所需的成本比 PathImpact 要小,综合看来 CollectEA 是较理想的影响分析方法.

动态影响分析必须在程序执行完再对所收集的信息进行分析,而实际上可在程序执行的同时对所收集的信息进行影响分析,该方法称为在线影响分析.Breech 等人对 Law 等人的动态影响分析方法做了一些修改,变为在线影响分析[50].该方法可计算任何数量的程序运行情况,其所计算的影响集的精确性与动态影响分析基本相当.

4.2.2.3 面向对象修改影响分析

随着面向对象语言以及面向对象设计方法的广泛应用,目前出现了很多面向对象语言程序的影响分析方法,由于面向对象语言程序与传统的结构化程序有着许多不同特征,如封装、继承、多态等特性的引入,传统的影响分析方法不再适用于面向对象程序的分析.

Lee 等人提出一系列影响度量参数评价某个修改所带来的影响[52].首先,他们建立数据流图和控制流图;然后将这两个图转变成面向对象数据依赖图,通过对该图的闭包计算来计算影响集,所计算的影响集的粒度是在类和类方法层次.该方法主要还是沿用传统的影响分析技术,没有考虑到面向对象的一些特殊特性,因此所计算的影响集不是很精确.

影响分析计算的影响集不仅可以是受影响的源代码,也可用来分析原测试套件中哪些测试用例受影响,用于选择回归测试用例.Ryder 等人就分析了基于修改去识别回归测试用例[53,54].他们首先将用户所提出的代码修改进行原子分解;然后在这些原子修改间建立某种偏序依赖关系便于后面建立语法正确的中间程序;最后通过分析,识别哪些测试用例受该修改影响以及某个测试用例失效是由于哪个修改造成的.

动态影响分析也可应用于面向对象程序.Huang 等人提出了针对面向对象程序的一种动态方法[55].他们建立一种动态依赖图,通过分析该动态依赖图计算影响集,该影响集的粒度层次是类成员层次,具有较好的精确性.

Kim 等人提出一种分布式环境下面向对象程序的修改影响分析方法[56].该方法利用一种分布式程序依赖图建立类、类成员、设计文档、服务器等元素间的依赖关系;然后在该分布式程序依赖图上根据所定义的防火墙技术计算影响集.

本小组也针对面向对象程序影响分析进行了大量的研究,并提出了几种面向对象程序的影响分析方法[57~60].

首先,提出一种基于层次切片,逐步求精的修改影响分析技术[14].该技术主要针对面向对象 Java 程序,计算从包层次、类层次,到方法层次、及语句层次的影响集结果;并且对于不同层次的影响集结果,其精确性不同,即:粒度越小,其精确性越高.

当前的修改影响分析主要基于程序中间表示(依赖图)上的可达性计算来得到影响集结果,但这些修改影响分析技术都没有考虑具体的修改类型,因而其结果的精确性受到影响.实际上不同的修改类型对程序的其他部分会有不同的传播机制,有的修改类型的发生并不影响程序的其他元素,而有的修改类型会给程序的其他部分造成影响.因此,在修改影响分析时考虑元素的修改类型能提高修改影响分析结果的精确性.因而对修改类型进行分类,并分析他们的传播机制,然后根据他们的传播机制进行修改影响分析[59],这样可得到更加精确的修改影响分析结果.

以上介绍的影响分析技术的输入和输出都是同一粒度层次.例如,给定类层次的修改集,计算类层次的影响集.但实际中,开发人员可能希望从类层次的修改得到更加细粒度层次的影响集,这样更有助于修改成本的预测.针对这一情况,我们提出一种基于概念格的跨层次的修改影响分析技术[60].该技术的输入是类层次的修改集,计算的结果是方法层次的影响集,所使用的方法是基于概念格上的层次特性来计算.另外,在此基础上更进一步,即如果知道更细粒度层次的方法修改时,如何计算影响集.当前的方法是基于调用图的遍历来计算,但结果非常不精确,实际使用效果很差.我们提出将概念格和调用图结合起来计算更加精确的影响集[15].

4.2.2.4 面向方面修改影响分析

面向对象技术并没有从本质上解决软件系统的复用性和易维护性.现实中的软件系统存在许多横切关注点,如日志记录、异常处理、安全性检查等,它们的实现代码和其他业务逻辑代码混杂在一起,并散落在软件不同地方,这样模块的可重用性大大降低,使软件系统的易维护性和复用性受到极大限制.由此产生了面向方面编程(AOP)技术,它是面向对象编程的延续和改进.这种编程模式提取散落在软件系统各处的横切关注点代码,并模块化,归整到一起,进一步提高软件的可维护性、复用性和可扩展性[61].因此,面向方面的程序相比传统的面向对象程序来说,又出现了一些新的特性和概念,如方面(aspects),关注点(concern)等.这样上面所述的一些影响分析技术并不能很好地对面向方面程序进行分析.目前这方面工作还较少,主要是赵建军研究小组在这方面所做的一些研究.他们主要对传统的技术进行改进,提出基于面向方面切片的修改影响分析技术[62].后来,他们参照 Ryder 等人的面向对象程序影响分析技术[53,54],针对面向方面程序提出相应的面向方面的影响分析技术[63,64],该技术也是用于识别哪些测试用例受修改影响以及某个测试用例失效是由于哪个修改造成的.

4.2.2.5 面向服务修改影响分析

面向服务的计算代表了分布式计算和软件开发的最新发展方向.它在面向对象、基于构件的开发、分布式对象计算及 Web 技术基础上,是一种新的软件开发、部署和集成模式,其目的在与有效解决在分布、动态、异构环境下,数据、应用和系统集成的问题[65,66].由于该领域相对其他面向对象的规范出现较晚,目前关于面向服务的修改影响分析技术还比较少.

Xiao 等人提出一种用于支持面向服务应用的修改影响分析方法[67].该方法从业务过程规约和源代码两个层次进行影响分析.首先识别哪些业务过程组件需进行修改;其次,利用业务过程规约中的控制和数据规约关系计算业务过程中哪些元素受到影响;然后,对上一步受到影响的元素进行映射,映射到源代码层次哪些元素受到影响;最后,在源代码层次利用调用和继承等依赖关系计算代码层次的影响集.

另外,Liu 等人还对 SOC 环境中的业务流程执行语言(Business Process Execution Language, BPEL)程序进行影响分析[68].该技术用于识别业务流程执行语言程序中哪些测试用例受到影响.首先,该方法定义一些影响规则识别 BPEL 并发控制结构修改时哪些测试路径受到影响,这些受影响的测试路径即为影响集;其次,根据影响集结果将这些受影响的测试路径分为可重用、可修改、过时、及需要添加的测试路径.

5 修改影响分析研究展望

随着软件的大量使用,大部分软件已不是从头开始开发,而是通过维护来不断适应环境的变化、用户的需求、技术的更新、及市场的变化等.软件产品本身的固有特性就是演化性.在软件演化过程中,修改影响分析起着越来越重要的作用.尽管修改影响分析研究已有二十余年,但其远非成熟.新的开发方法和设计规范不断涌现,影响集计算的不精确性等都对修改影响分析带来巨大的挑战.下面列举了当前影响分析所面临的一些挑战以及仍需解决的问题:

基础理论支撑　到目前为止,还没有形式化的修改影响分析定义,最常用的定义还是 Bohner 和 Arnold 在 1996 年给的定性定义,另外还缺乏相关的理论来支持修改影响分析技术.

工具支持　尽管已出现了一些支持修改影响分析的工具,不过大部分工具目前还处在原型或实验性阶段,还没有出现比较公认的商业化的影响分析工具.

评价机制　经过了 20 多年的研究,出现了各种影响分析技术,到目前为止都是从精确性、安全性及成本来比较各影响分析技术的优劣,但影响分析技术所涉及的远非这三个因素,还没有一个较好的评价机制去评估各种影响分析技术.

影响分析的拓展性　目前软件变的越来越庞大,越来越复杂,如异构系统,这些软件可能同时包括几种不同的设计规范、设计方法和语言,并且是分布式、动态变化的,到目前为止影响分析还没到达这个层次,因此迫切需要新的技术来解决异构系统或面向服务系统的影响分析.

精确性与安全性之间的权衡　目前影响分析可以是静态分析,也可是动态分析,静态分析比较简单,成本较低,安全性好,但很不精确,因此很难应用于实际;而动态影响分析精确性较好,但其往往丢失一些真正受到影响的元素,这影响到维护人员对系统修改与维护的信心,因而如何选择一种折中方案很有必要.

影响分析的跟踪性　目前大部分影响分析技术所计算的影响集都局限在某一层次,如代码层次的修改集计算代码层次的影响集、需求层次的修改集计算需求层次的影响集.但是,对软件的修改不只是局限在某一层次范围,而是包括整个软件,如需求文档、设计、代码、测试等都受到修改的影响.因此需要对修改影响分析技术的跟踪性进行研究,讨论修改给整个软件的各个层次元素带来的影响.

6 结语

软件修改实施前,需进行修改影响分析,准确评估

修改给软件带来的影响,辅助项目经理和开发人员评估、实施、验证修改.近 20 年来,修改影响分析领域研究产生了各种技术.本文从修改影响分析类型、方法、所支持的语言范型、阶段、层次五种视角对修改影响分析进行分类,并根据分类介绍了各种影响分析方法,为开发人员和维护人员提供了选择合适修改影响分析技术的参考.尽管修改影响分析取得了长足的发展,但其还存在一些问题,如不完善的修改影响分析理论,不成熟的工具支持,缺乏可拓展、可跟踪的修改影响分析技术,没有可信的评价标准,缺少既精确又安全的修改影响分析技术等.因此,未来修改影响分析技术方面的研究将综合考虑各种软件分析技术,结合不同的软件数据源(需求文档、代码、测试用例集、演化历史库等)对软件进行分析,计算软件各层次全面的、考虑安全性和精确性的折中的修改影响分析结果,这样软件修改影响分析结果才可信、可用.

参考文献

[1] B Li, X Sun, H Leung, S Zhang. A survey of code-based change impact analysis techniques[J]. Journal of Software Testing, Verification and Reliabiblity, 2013, 23(8): 613 – 646.

[2] X Sun, H Leung, B Li, B Li. Change impact analysis and changeability assessment for a change proposal: An empirical study[J]. Journal of Software and System, 2014, 96(10): 51 – 60.

[3] X Sun, B Li, Q Zhang. A change proposal driven approach for changeability assessment using formal concept analysis[A]. International Computer Software and Applications Conference [C]. Turkey: IEEE, 2012. 328 – 333.

[4] X Sun, B Li, C Q Zhang. Using FCA-based change impact analysis for regression testing[A]. International Conference on Software Engineering and Knowledge Engineering[C]. USA: KSI, 2012. 452 – 457.

[5] X Sun, B Li, B Li, W Wen. A comparative study of static CIA techniques[A]. Asia-Pacific Symposium on Internetware[C]. China: ACM, 2012.

[6] S S Yau, J S Collofello, T MacGregor. Ripple effect analysis of software maintenance[A]. International Conference on Computer Software and Applications[C]. USA: IEEE, 1978. 60 – 65.

[7] L C Briand, Y Labiche, L O Sullivan. Impact analysis and change management of UML models[A]. International Conference on Software Maintenance[C]. Netherlands: IEEE, 2003. 256 – 265.

[8] L C Briand, Y Labiche, L O'Sullivan, M M Sowka. Automated impact analysis of UML models[J]. Journal of Systems and Software, 2006, 79(3): 339 – 352.

[9] P H Tang. Design based change impact modeling for object-oriented software[D]. New York: State University of New York, 2003.

[10] 王映辉,王立福.软件体系结构演化模型[J].电子学报,2005,3(8):1381 – 1386.
Y H Wang, L F Wang. Research about model and ripple effect analysis of software architecture evolution[J]. Journal of Electronics, 2005, 3(8), 1381 – 1386. (in Chinese)

[11] X Sun, B Li, C Tao, W Wen, S Zhang. Change impact analysis based on a taxonomy of change types[A]. International Conference on Computer Software and Applications[C]. Korea: IEEE, 2010. 373 – 382.

[12] S Bohner, R Arnold. Software Change Impact Analysis[M]. Los Alamitos, CA, USA: IEEE Computer Society Press, 1996.

[13] L Badri, M Badri, S D Yves. Supporting predictive change impact analysis: a control call graph based technique[A]. Asia-Pacific Software Engineering Conference[C]. Taipei, Taiwan: IEEE, 2005. 167 – 175.

[14] X Sun, B Li, C Tao, S Zhang. HSM-based change impact analysis of object-oriented java programs[J]. Chinese of Journal Electronics, 2011, 20(2): 247 – 251.

[15] B Li, X Sun, H Leung. Combining concept lattice with call graph for impact analysis[J]. Advances in Engineering Software, 2012, 53(11): 1 – 13.

[16] B Li, Q Zhang, X Sun, H Leung. Using water wave propagation phenomenon to study software change impact analysis [J]. Advances in Engineering Software, 2013, 58(4): 45 – 53.

[17] H Kagdi, M Gethers, D Poshyvanyk. Blending conceptual and evolutionary couplings to support change impact analysis in source code[A]. Proceedings of the IEEE Working Conference on Reverse Engineering[C]. Boston, USA: IEEE, 2010. 119 – 128.

[18] H Kagdi, M Gethers, D Poshyvanyk. Integrating conceptual and logical couplings for change impact analysis in software[J]. Empirical Software Engineering, 2013, 18(5): 933 – 969.

[19] M Revelle, M Gethers, D Poshyvanyk. Using structural and textual information to capture feature coupling in object-oriented software[J]. Empirical Software Engineering, 2011, 16 (6): 773 – 811.

[20] D Poshyvanyk, A Marcus, R Ferenc, T Gyimóthy. Using information retrieval based coupling measures for impact analysis [J]. Empirical Software Engineering, 2009, 14(1): 5 – 32.

[21] J Law, G Rothernel. Whole program path-based dynamic impact analysis[A]. International Conference on Software Engineering[C]. Hilton Portland: IEEE, 2003. 308 – 318.

[22] T Apiwattanapong, A Orso, M Harrold. Efficient and precise dynamic impact analysis using execute after sequences[A]. International Conference on Software Engineering[C]. Missouri, USA: IEEE, 2005. 432 – 441.

[23] A Orso, M J Harrold. Leveraging field data for impact analysis and regression testing[A]. In Proceedings of the ACM SIG-

SOFT Symposium on Foundations of Software Engineering [C]. Helsinki, Finland: ACM, 2003. 128 – 137.

[24] G Canfora, L Cerulo. Fine grained indexing of software repositories to support impact analysis[A]. International workshop on Mining Software Repositories[C]. Shanghai, China: ACM, 2006. 105 – 111.

[25] M A Jashki, R Zafarani, E Bagheri. Towards a more efficient static software change impact analysis methods[A]. In Proceedings of the 8th ACM SIGPLAN-SIGSOFT Workshop on Program Analysis for Software Tools and Engineering[C]. Atlanta, Georgia: ACM, 2008. 84 – 90.

[26] L Hattori, G F Cardoso, M Sampaio. Mining software repositories for software change impact analysis: a case study[A]. In Proceedings of the 23rd Brazilian Symposium on Databases [C]. Campinas, Brazil: ACM, 2008. 210 – 223.

[27] AYing, G C Murphy, R Ng, M C Chu-Carroll. Predicting source code changes by mining change history[J]. IEEE Transactions on Software Engineering, 2004, 30(9): 574 – 586.

[28] M Gethers, H Kagdi, B Dit, D Poshyvanyk. An adaptive approach to impact analysis from change requests to source code [A]. Proc of 26th IEEE/ACM International Conference on Automated Software Engineering[C]. Lawrence, KS, USA: IEEE/ACM, 2011. 540 – 543.

[29] C J van Rijsbergen. Information Retrieval[M]. London: Butterworths, 1979.

[30] A Knethen, M Grund. QuaTrace: A tool environment for automatic impact analysis based on traces[A]. International Conference on Software Maintenance [C]. Amsterdam, The Netherlands: IEEE, 2003. 246 – 255.

[31] O Gotel, C W Finkelstein. An analysis of the requirements traceability problem[A]. Proceedings of the First International Conference on Requirements Engineering [C]. Colorado, USA: IEEE, 1994. 94 – 101.

[32] M R Strens, R C Sugden. Change analysis: A step towards meeting the challenge of changing requirements[A]. Symposium on Engineering of Computer Based Systems [C]. Friedrichshafen, Germany: IEEE, 1996. 278 – 283.

[33] M Fyso, C Boldyreff. Using application understanding to support impact analysis[J]. Journal of Software Maintenance: Research and Practice, 1998, 10(2): 93 – 110.

[34] J S O'Neal. Analyzing the Impact of Changing Software Requirements: A Traceability-Based Methodology [D]. Louisiana: Louisiana State University and Agricultural & Mechanical College, 2003.

[35] L C Briand, Y Labiche, T Yue. Automated traceability analysis for UML model refinements[J]. Information and Software Technology, 2009, 51(2): 512 – 527.

[36] A De Lucia, F Fasano, R Oliveto. Traceability management for impact analysis[A]. Frontiers of Software Maintenance[C]. Beijing, China: IEEE, 2008. 21 – 30.

[37] A Podgurski, L Clarke. A formal model of program dependences and its implications for software testing, debugging, and maintenance[J]. IEEE Transactions on Software Engineering, 1990, 16(9): 965 – 979.

[38] C Y Chen, C W She, J D Tang. An object-based, attribute-oriented approach for software change impact analysis[A]. International Conference on Industrial Engineering and Engineering Management[C]. Singapore: IEEE, 2007. 577 – 581.

[39] J Hassine, J Rilling, J Hewitt, R Dssouli. Change impact analysis for requirement evolution using use case maps[A]. International Workshop on Principles of Software Evolution[C]. Lisbon, Portugal: IEEE, 2005. 81 – 90.

[40] J Hewitt, J Rilling. A light-weight proactive software change impact analysis using use case maps[A]. International Workshop on Software Evolvability[C]. Budapest, Hungary: IEEE, 2005. 43 – 46.

[41] M Shiri, J Hassine, J Rilling. A requirement level modification analysis support framework[A]. IEEE Workshop on Software Evolvability[C]. Paris, France: IEEE, 2007. 67 – 74.

[42] M Weiser. Program slicing[A]. International Conference on Software Engineering[C]. CA, USA: IEEE, 1981. 439 – 449.

[43] B Li, X Fan, J Pang, J Zhao. A model for slicing JAVA programs hierarchically [J]. Journal of Computer Science and Technology, 2004, 19(6): 848 – 858.

[44] B W Xu, J Qian, X Zhang, Z Wu, L Chen. A brief survey of program slicing [J]. ACM SIGSOFT Software Engineering Notes, 2005, 30(2): 1 – 36.

[45] X Zhang, R Gupta, Y Zhang. Efficient forward computation of dynamic slices using reduced ordered binary decision diagrams [A]. International Conference on Software Engineering[C]. Scotland, UK: IEEE, 2004. 502 – 511.

[46] R Gupta, M Soffa, J Howard. Hybrid slicing: integrating dynamic information with static analysis[J]. ACM Transactions on Software Engineering and Methodology, 1997, 6(4): 370 – 397.

[47] P Tonella. Using a concept lattice of decomposition slices for program understanding and impact analysis[J]. IEEE Transactions on Software Engineering, 2003, 29(6): 495 – 509.

[48] J Larus. Whole program paths[A]. Proceedings of SIGPLAN Conference on Programming Language Design and Implementation[C]. Georgia, USA: ACM, 1999. 1 – 11.

[49] A Orso, T Apiwattanapong, J Law, G Rotherme, et. al. An empirical comparison of dynamic impact analysis algorithms [A]. International Conference on Software Engineering[C]. Scotland, UK: IEEE, 2004. 491 – 500.

[50] B Breech, A Danalis, S Shindo, L Pollock. Online impact analysis via dynamic compilation technology[A]. International Conference on Software Maintenance[C]. Alberta, Canada: IEEE, 2004. 453 – 457.

[51] B Breech, M Tegtmeyer, L Pollock. A comparison of online and dynamic impact analysis algorithms[A]. Ninth European Conference on Software Maintenance and Reengineering[C]. Manchester, UK: IEEE, 2005. 143 – 152.

[52] M Lee, A Offutt, R Alexander. Algorithmic analysis of the impacts of changes to OO software[A]. Proceedings of the Technology of Object-Oriented Languages and Systems[C]. CA, USA: IEEE, 2000. 61 – 70.

[53] B G Ryder, F Tip. Change impact analysis for object oriented programs[A]. ACM Workshop on Program Analysis for Software Tools and Engineering[C]. Utah, USA: ACM, 2001. 46 – 53.

[54] X Ren, F Shah, F Tip, B Ryder, O Chesley. An object-based, attribute-oriented approach for software change impact analysis [A]. International Conference on Object Oriented Programming, Systems, Languages and Applications[C]. Vancouver, Canada: ACM, 2004. 432 – 448.

[55] L L Huang, Y Song. Precise dynamic impact analysis with dependency analysis for OO programs[A]. Proceedings of the International Conference on Software Engineering Research, Management and Applications [C]. Busan, Korea: IEEE, 2007. 374 – 384.

[56] K Kim, J N Park, Y Yoon. A graph of change impact analysis for distributed object-oriented software[A]. Proceedings of the International Conference on Fuzzy Systems[C]. Seoul, Korea: IEEE, 1999.

[57] 孙小兵,李必信,陶传奇. 基于 LoCMD 的软件修改分析技术[J]. 软件学报,2012,23(6):1368 – 1381.
X Sun, B Li, C Tao. Using LoCMD to support software change analysis[J]. Journal of Software, 2012, 23(6): 1368 – 1381. (in Chinese)

[58] X Sun, B Li, S Zhang, C Tao. Using lattice of class and method dependence for change impact analysis of object oriented programs[A]. Proc of the Symposium on Applied Computing[C]. Taiwan, China: ACM, 2011. 1439 – 1444.

[59] X Sun, B Li, C Tao, W Wen, S Zhang. Analyzing impact rules of different change types to support change impact analysis [J]. International Journal of Software Engineering and Knowledge Engineering, 2013 23(3): 259 – 288.

[60] B Li, X Sun, J Keung. FCA-CIA: An approach of using FCA to support cross-level change impact analysis for object oriented Java programs[J]. Information & Software Technology, 2013, 55(8): 1437 – 1449.

[61] G Kiczales, E Hilsdale. Aspect-oriented programming[A]. Proceedings of the 8th European Software Engineering Conference Held Jointly with 9th ACM SIGSOFT in Ternational Symposium on Foundations of Software Engineering[C]. Vienna, Austria: ACM, 2001. 313.

[62] J Zhao. Change impact analysis for aspect-oriented software evolution[A]. Proceedings of the International Workshop on Principles of Software Evolution[C]. Orlando Florida, USA: IEEE, 2002. 108 – 112.

[63] S Zhang, Z Gu, Y Lin, J Zhao. Change impact analysis for aspectj programs[A]. International Conference on Software Maintenance[C]. China: IEEE, 2008. 87 – 96.

[64] S Zhang, Z Gu, Y Lin, J Zhao. Celadon: a change impact analysis tool for aspect-oriented programs[A]. International Conference on Software Engineering[C]. Leipzig, Germany: IEEE, 2008. 913 – 914.

[65] H Demirkan, R Kauman, J Vayghan, H Fill, D Karagiannis, P Maglio. Service-oriented technology and management: Perspectives on research and practice for the coming decade[J]. Electronic Commerce Research and Applications, 2009, 7(4): 356 – 376.

[66] N Gold, C Knight, A Mohan, M Munro. Understanding service-oriented software[J]. IEEE Software, 2004, 21(2): 71 – 77.

[67] H Xiao, J Quo, Y Zou. Supporting change impact analysis for service oriented business applications[A]. International Workshop on Systems Development in SOA Environments[C]. Minneapolis, MN, USA: IEEE, 2007.

[68] H Liu, Z Li, J Zhu, H Tan. Business process regression testing [A]. Proceedings of the 5th International Conference on Service-Oriented Computing [C]. Vienna, Austria: Springer, 2007. 157 – 168.

作者简介

孙小兵 男,1985 出生,江苏姜堰人,博士,讲师,硕士生导师,主要研究领域为软件维护与演化.

李 斌 男,1965 出生,江苏靖江人,博士,教授,博士生导师,主要研究领域为数据分析,智能计算.

陈 颖 女,1985 出生,江苏姜堰人,硕士研究生,主要研究领域为数据分析.

李必信 男,1969 出生,安徽庐江人,博士,教授,博士生导师,主要研究领域为软件建模、分析、测试与验证.

文万志 男,1982 出生,安徽桐城人,博士生,主要研究领域为软件错误定位,软件演化与维护.