回归测试用例选择技术研究综述

陈 翔[1,2] 顾卫江[1] 徐 慧[1] 顾 庆[2] 陈道蓄[2]
(南通大学计算机科学与技术学院 南通 226019)[1] (南京大学软件新技术国家重点实验室 南京 210093)[2]

摘 要 回归测试用例选择(Regression Test Case Selection,RTS)问题是回归测试研究中的一个热点,旨在从已有测试用例集中选择出所有可检测代码修改的测试用例。但迄今为止,国内研究人员并未对RTS问题的已有研究成果进行系统总结和比较。首先在回归测试活动和测试用例划分基础上,引出RTS问题和相关假设。随后从源代码和模型角度对已有RTS技术进行分类,从源代码角度出发,又进一步将其细分为线性规划法、数据流分析法、图遍历法、程序切片法和防火墙法等。接着对常见评测数据集和评测指标进行总结,最后对该问题的未来研究方向进行了展望。

关键词 回归测试,测试用例选择,图遍历法,程序切片,线性规划
中图法分类号 TP311.5 **文献标识码** A

Regression Testing Selection Techniques: A State-of-the-art Review

CHEN Xiang[1,2] GU Wei-jiang[1] XU Hui[1] GU Qing[2] CHEN Dao-xu[2]
(School of Computer Science and Technology, Nantong University, Nantong 226019, China)[1]
(State Key Laboratory for Novel Software Technology at Nanjing University, Nanjing University, Nanjing 210093, China)[2]

Abstract Regression test case selection(RTS) is a hot research topic in the study of regression testing. This technique aims to identify modification-revealing test cases from existing test suite. But to date, researchers in China do not systematically summarize and compare existing research work for RTS problem. This paper firstly formulated the RTS problem and its underlying assumptions based on the classification on regression testing activities and test cases. It secondly classified existing RTS techniques into two categories: code-based RTS and model-based RTS. It further classified these code-based RTS techniques into subcategories, such as integer programming approach, data-flow analysis approach, graph-walk approach, program slicing approach, and firewall approach. It thirdly summarizd commonly-used experiment subjects and evaluation metrics. It finally suggested some potential future work of this topic.

Keywords Regression testing, Test case selection, Graph-walk approach, Program slicing, Integer programming

1 引言

开发人员在软件开发和维护过程中,因移除软件内在缺陷、完善软件已有功能、重构软件原有代码或提高软件产品运行性能等,需要修改代码并触发软件演化。随着新的软件开发方法和模型(如敏捷软件开发、极限编程和测试驱动开发等)的日益成熟和不断推广,软件演化频率迅速提高,因此亟需经济有效的方法来确保演化后软件产品的质量。目前测试人员一般通过回归测试来保证代码修改的正确性并避免代码修改给被测程序其他模块带来的副作用。但统计数据表明回归测试开销一般占整个软件测试预算的80%以上,并占整个软件维护预算的50%以上[1,2]。为了降低回归测试的开销,国内外研究人员对高效自动的回归测试技术展开了深入研究。其中对测试用例的高效维护是回归测试中的核心研究问题。一种简单维护策略是重新执行修改前程序 P 配套的测试用例集 T,但该策略确存在诸多不足,例如测试用例的持续添加会导致测试开销过大、已有的部分测试用例在修改后程序上无效等。针对上述问题,研究人员提出了一系列测试用例维护技术,包括失效测试用例的识别和修复技术、测试用例选择技术(RTS)、测试用例优先级排序技术、测试用例集缩减技术和测试用例集扩充技术等[3]。

本文综述的RTS技术可以有效缓解软件持续演化过程中测试用例集规模快速增长的问题。该技术在分析代码修改后,从已有测试用例集 T 中为修改后程序 P' 选择出部分测试用例。理想情况下,希望可以选择出所有可检测缺陷的测试用例。对RTS问题的研究最早可以追溯到Fischer在1977年发表的研究论文[4]。随后研究人员将RTS技术成功应用到不同测试阶段,包括单元测试、集成测试、系统测试和验收

到稿日期:2012-12-01 返修日期:2013-01-25 本文受国家重点基础研究发展计划(2009CB320705),国家高技术研究发展计划(2006AA01Z177),国家自然科学基金(60873027,61202006),江苏省高校自然科学研究项目(12KJB520014),南通市应用研究计划项目(BK2012023和BK2012027),南京大学计算机软件新技术国家重点实验室开放课题(KFKT2012B29),南通大学自然科学研究项目(03040844)资助。
陈 翔(1980—),男,博士,讲师,主要研究方向为软件测试和程序分析,E-mail:xchencs@ntu.edu.cn;顾卫江(1977—),男,硕士,讲师,主要研究方向为软件测试;徐 慧(1965—),女,博士,教授,主要研究方向为网络安全;顾 庆(1972—),男,博士,教授,博士生导师,主要研究方向为软件质量保障;陈道蓄(1947—),男,教授,博士生导师,主要研究方向为并行计算和软件质量保障。

测试。同时RTS技术适用的软件产品从面向过程语言(如Fortran和C)扩展到面向对象语言(如C++和Java)。

从20世纪70年代初提出RTS问题到现在,国内外研究人员对其日益关注和研究,提出了多种有效解决方案。Rothermel和Harrolad首次对该问题进行了总结并提出一种统一评估框架[5,6]。随后Yoo和Harman对回归测试中的测试用例集缩减、选择和优先级问题进行了分析[7]。但迄今为止,并没有国内研究人员对RTS问题进行过系统综述。

为了对该研究问题进行系统的分析、总结和比较,我们首先在IEEE、ACM、Spring、Wiley和Elsevier等数据库中进行检索,选择时采用的关键词包括:"Test Case Selection"和"Regression Test Selection"等。然后对选择出的论文,通过查阅参考文献来进一步识别出遗漏论文。最终我们选择出与该主题直接相关的75篇论文(截止到2012年11月)。图1按照论文发表的时间进行汇总,其中横坐标表示发表年份,纵坐标表示该年份发表的论文总数。从图1中可以看出该研究问题一直是回归测试中的一个研究热点,近6年发表的论文数占总论文数的35%。从发表的会议和期刊来看,绝大部分论文发表在软件工程领域的知名会议或期刊上,例如ICSM会议有18篇,COMPSAC会议有6篇,ICSE会议有4篇,STVR期刊有5篇,TSE期刊有4篇,TOSEM期刊有2篇,JSS期刊有2篇。

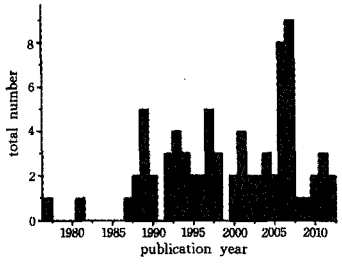

图1 论文在不同发表时间的数量分布

2 研究背景

2.1 回归测试活动和测试用例的划分

Leung和White对回归测试活动和测试用例的划分构成了回归测试后续研究的重要理论基础[2]。他们首先将回归测试活动分为两类:逐步式回归测试和纠正式回归测试。在逐步式回归测试中,程序P'对应的程序规约会发生变动;而在纠正式回归测试中,程序P'规约维持不变,而源代码会发生变动。

在回归测试活动划分基础上,Leung和White进一步将测试用例划分为5类:(1)Reusable测试用例。这类测试用例在P上覆盖的程序模块在P'上并未修改,所以在测试P'时并不需要执行这类测试用例,但在回归测试时仍需保存这类测试用例以用于随后版本的测试。(2)Retestable测试用例。这类测试用例在P上覆盖的部分程序模块在P'上被修改过,所以在测试P'时需要重新执行这类测试用例。(3)Obsolete测试用例。这类测试用例的失效原因包括:因程序规约的改动导致测试用例的输入和输出发生变动,该测试用例测试的功能在新版本上已经被移除,或该测试用例不能进一步提高P'的代码覆盖率。(4)New-structural测试用例。这类测试用例可以提高P'的代码覆盖率。(5)New-specification测试用例。这类测试用例可以测试因规约变动产生的新代码。

根据上述划分可知,已有测试用例集T中仅包含前3类测试用例,而后2类测试用例需要针对P'进行额外设计。在大部分回归测试研究中,研究人员主要关注的是纠正式回归测试。具体到RTS问题中,研究人员主要关注从已有测试用例集T中识别出Retestable测试用例。若需关注逐步式回归测试,则需要额外考虑采用特定测试用例生成技术来测试新的代码或规约。

2.2 测试用例选择问题描述和相关假设

RTS技术作为回归测试中缩减测试用例集规模的一种有效手段,主要借助白盒测试技术,在代码修改分析基础上完成对已有测试用例的选择,并确保选择出的测试用例与代码修改相关。Rothermel和Harrold首次对该问题给出了如下形式化描述[5,6]:

给定:修改前程序P和相应测试用例集T,修改后程序P'。

问题:寻找T的子集T'对P'进行测试,并且T'中的任意测试用例均是可检测代码修改的测试用例。

在理想情况下,RTS技术希望从T中选择出所有可检测缺陷的测试用例,对这类测试用例可定义如下[5,6]:

定义1(可检测缺陷的测试用例) 一个测试用例是可检测缺陷的测试用例,当且仅当它可以检测到P'中包含的缺陷,即测试用例的实际输出与预期输出不一致。

但不存在可以判断测试用例在P'上是可检测缺陷的测试用例的有效方法。当满足一些假设时,RTS技术可以选择出可检测代码修改的测试用例[6],并且这些测试用例均是可检测缺陷的测试用例。Rothermel和Harrold对这类测试用例定义如下[5,6]:

定义2(可检测代码修改的测试用例) 一个测试用例是可检测代码修改的测试用例,当且仅当测试用例在P和P'上的实际输出不一致。

需要满足的两个假设分别是:

假设1 P-Correct-for-T假设。假设测试用例集T中的任意一个测试用例t,均可在程序P上执行结束并输出正确结果。

假设2 Obsolete-Test-Identification假设。假设测试用例集T中的任意一个测试用例t,存在一个可以判断t对P'是否失效的有效方法。一个测试用例在P'失效,当且仅当测试用例的输入在P'无效,或者测试用例的输入和输出关系在P'无效。

若满足上述两个假设,首先借助一个有效方法可以从T中识别出所有的失效测试用例。对于余下的非失效测试用例,在纠正式回归测试中,若测试用例在P上可以执行结束并输出正确结果,则该测试用例在P'上也应执行结束并输出相同结果。所以可以通过寻找可检测代码修改的测试用例来寻找可检测缺陷的测试用例。但即使上述两个假设成立,仍无法找到一个可以在非失效测试用例中识别出可检测代码修改的测试用例的有效方法。所以Rothermel等人随之提出可覆盖代码修改的测试用例。他们将这类测试用例定义如

下[5,6]：

定义 3(可覆盖代码修改的测试用例) 一个测试用例是可覆盖代码修改的测试用例，当且仅当：(1)该测试用例覆盖了 P' 中新增或修改的代码。(2)或在程序 P 上覆盖了被 P' 删除的代码。

当满足假设 3 时，若一个非失效测试用例是可覆盖代码修改的测试用例，则该测试用例可能是可检测代码修改的测试用例。假设 3 可描述如下：

假设 3 Controlled-Regression-Testing 假设。除了代码因素，其他影响测试用例在 P 和 P' 上输出结果的因素均假设保持一致。

当同时满足假设 1—3 时，我们可以在非失效测试用例中通过识别可覆盖代码修改的测试用例来进一步识别出可检测缺陷的测试用例。这些不同类型测试用例间的包含关系如图 2 所示。

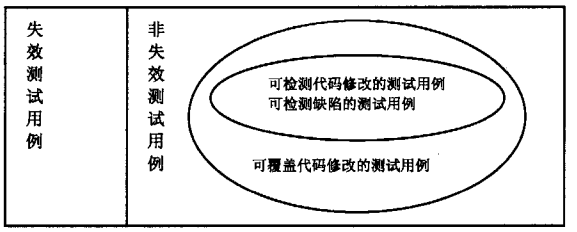

图 2 不同类型测试用例间的包含关系

但假设 3 在实际测试场景中难以成立。例如：同一程序在不同操作系统上的执行行为可能并不一致，程序执行过程中包含不确定性步骤等。但从已有研究工作来看，通过寻找可遍历代码修改的测试用例仍是解决 RTS 问题的一种可行手段。

3 研究工作分类

传统 RTS 问题的研究框架如图 3 所示。首先对修改前后程序针对控制流或数据流进行建模，在模型基础上通过比对识别出代码修改模块，然后通过代码修改影响分析，进一步识别出存在依赖关系的其他程序模块。最后从已有测试用例集中，结合代码修改影响分析结果选择出所有可覆盖代码修改的测试用例并用于程序 P' 的测试。

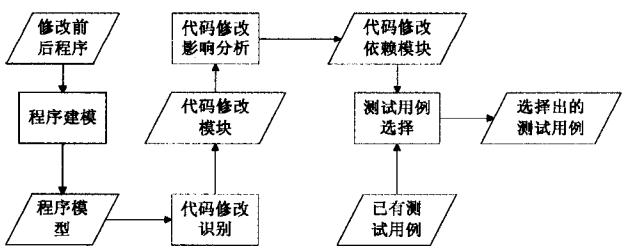

图 3 RTS 问题的研究框架

以上述研究框架为基础，本文将已有 RTS 技术分为两类：一类从源代码角度出发，适用于软件编码阶段；另一类从程序规约或模型角度出发，适用于软件设计阶段。

3.1 基于源代码的 RTS 技术

3.1.1 线性规划法

Fischer 等人针对 Fortran 编程语言首次对 RTS 问题展开研究[4,8]。他们首先将仅包含单一入口和出口的代码段称为基本语句块。然后基于基本语句块构造出两个矩阵：可达矩阵和测试用例依赖矩阵。其中可达矩阵描述了基本语句块间的可达关系，测试用例依赖矩阵描述了测试用例的基本语句块覆盖信息。

假设被测程序 P 包含 m 个基本语句块，对应测试用例集 T 包含 n 个测试用例，则线性规划模型为：

$$\text{Min } x_1+x_2+\cdots+x_n$$

$$\text{s. t.}$$

$$a_{11}x_1+a_{12}x_2+\cdots+a_{1n}x_n\geqslant b_1$$

$$a_{21}x_1+a_{22}x_2+\cdots+a_{2n}x_n\geqslant b_2$$

$$\cdots\cdots$$

$$a_{m1}x_1+a_{m2}x_2+\cdots+a_{mn}x_n\geqslant b_m$$

其中，变量 x_i 取值为 1，表示选择第 i 个测试用例对 P' 进行测试，否则取值为 0。系数 a_{ij} 取值为 1，表示第 j 个测试用例覆盖第 i 个基本语句块，否则取值为 0。变量 b_i 的取值基于可达矩阵，当分析出与代码修改相关的基本语句块后，通过可达矩阵，可以计算获得存在可达关系的其他基本语句块并将其取值设置为 1，即希望这些基本语句块被选择出的测试用例至少覆盖一次。

Hartmann 和 Robson 实现并扩展了上述方法，并将其用于 C 语言程序的测试[9,10]。区别之处在于他们用函数代替基本语句块。

但这类方法假设上述两个矩阵在修改前后程序上保持一致，即代码修改不会影响到程序的原有控制流结构。同时这类方法可能会遗漏一部分可检测代码修改的测试用例，即这类方法具有不安全性。

3.1.2 数据流分析法

这类方法对代码修改进行数据流分析，通过识别出新添、修改或删除的定义使用对(Definition-use Pair)来辅助测试用例的选择。Harrold 和 Soffa 提出一种适用于单元测试阶段的增量数据流分析法[11]，该方法可以复用测试用例在历史程序版本上的分析信息，并避免对整个系统进行数据流分析，从而有效降低回归测试开销。Taha 等人基于增量数据流分析法提出一种可以减少回归测试和缺陷定位开销的系统方法[12]。随后 Harrold 和 Soffa 提出一种可同时支持过程内和过程间的数据流分析法[13]，该方法可以有效处理全局变量、引用参数和递归函数等结构。Fischer 等人将这类方法成功用于电子表格程序的测试[14]。

但这类方法难以分析与数据流无关的代码修改。例如在程序 P' 中增加一个无参函数，或者修改后的函数返回语句不包含任何变量使用等。

3.1.3 图遍历法

图遍历法是目前 RTS 研究中的一种主流方法。Rothermel 和 Harrold 首次基于图遍历法提出了一系列 RTS 技术[15-19]。Rothermel 和 Harrold 首先将程序 P 和 P' 建模为控制依赖图(Control Dependence Graphs，CDG)并在图上同时执行深度优先遍历，识别出与代码修改相关的危险边集，依据测试用例在程序 P 上的覆盖信息来指导测试用例的选择，即如果测试用例覆盖了这些危险边，则选择出这些测试用例并用于 P' 的测试[15]。但该方法仅适用于单个过程。同时 CDG 并未考虑数据依赖关系，所以选择出的一些测试用例可能仅

覆盖了变量定义出现的语句，而未覆盖变量使用出现的语句。接着 Rothermel 和 Harrold 对上述方法进行扩充，提出基于程序依赖图（Program Dependence Graph，PDG）的过程内 RTS 技术和基于系统依赖图（System Dependence Graph，SDG）的过程间 RTS 技术[16]。其中 PDG 在单个过程内同时考虑了控制和数据依赖关系，SDG 则可以处理包含多个过程的程序。随后 Rothermel 和 Harrold 提出一种基于控制流图（Control Flow Graphs，CFG）的 RTS 技术并开发出 DejaVu 工具。相对于 CDG 来说，CFG 更容易对程序内部结构进行建模，同时该技术的选择精度也更高，但 CFG 的主要不足是并未考虑数据流信息[17,18]。这些 RTS 技术的有效性在不同的程序对象和测试用例集上均进行了实证研究[19]。Ball 从分支覆盖角度出发进一步提高了 RTS 技术的选择精度[20]。

随着新编程范式的提出，研究人员对已有 RTS 技术不断进行扩充。Rothermel 等人结合面向对象程序特征（以 C++ 程序为研究对象），将被测程序建模为过程间控制流图（Interprocedural Control Flow Graph）[21]。该模型通过添加调用边和返回边将不同的过程连接起来。Harrold 等人针对 Java 程序，提出一种相似的 RTS 技术[22]，该方法将被测程序建模为 Java 类间图（Java Interclass Graph），该模型可以有效地处理过程间控制流和面向对象语言的特征。具体来说，Java 类间图通过扩展 CFG 可以支持：(1)变量和对象类型；(2)内部和外部方法；(3)因内部方法调用内部或外部方法产生的过程间交互；(4)因外部方法调用内部方法产生的过程间交互；(5)异常处理。Beydeda 和 Gruhn 基于 Rothermel 等人的研究工作，通过将黑盒测试中的数据流信息添加到类控制流图（Class Control Flow Graph）来对面向对象程序进行测试[23]。Orso 等人则考虑采用不同粒度的模型来提高 RTS 技术的扩展性[24]。所提方法分两个阶段：(1)划分阶段：针对程序 P 和 P'，采用粗粒度方式对被测程序建模并完成快速分析。该阶段的分析主要基于类和接口间的依赖关系，如层次、聚类和使用关系等。其输入是与代码修改相关的类和接口，输出是与修改存在依赖关系的类和接口。(2)选择阶段：完成第一阶段后，将识别出的类和接口以细粒度方式建模并通过图遍历法识别出危险边集，同时从 T 中选择出覆盖过这些危险边的测试用例。

目前图遍历法已经成功应用于一些特定领域的软件测试工作，例如 AspectJ 程序、基于组件的软件和 Web 服务应用等。Xu 和 Rountev 针对 AspectJ 程序对 RTS 技术进行扩展[25]，即在 CFG 中考虑了方法和通知在连接点间的交互。赵建军等人同样针对 AspectJ 程序提出一种 RTS 方法[26]。Orso 等人将 RTS 方法用于测试基于组件的软件[27,28]，该方法基于软件组件的元数据和程序规约。他们提出两种方法：基于代码的 RTS 方法和基于规约的 RTS 方法。对于前一类方法，需要每个组件提供代码覆盖信息反馈接口。对于后一类方法，组件规约借助 UML 中的状态图进行描述。Lin 等人采用基于 Java 类间图的方法将 Web 服务转化为 Single-JVM 局部应用[29]。Ruth 等人将从 Web 服务开发人员中搜集到的一个粗粒度 CFG 作为整个应用的组成模块[30-32]。Tarhini 等人采用定时标记迁移系统（Timed Labeled Transition System）将标记状态机进行组装[33]。李必信等人将 BPEL 程序建模为 XBFG（eXtensible BPEL flow graph）并提出相应的 RTS 技术[34]。

3.1.4 *程序切片法*

Gupta 等人基于数据流分析法，应用程序切片技术识别出与代码修改存在依赖关系的定义使用对[35,36]，该方法在执行时间和存储空间上均具有一定优势。Agrawal 等人则提出基于 4 种程序切片的 RTS 技术[37]。一个测试用例在程序上的执行切片（Execution Slice）是该测试用例覆盖的语句集，又称为执行轨迹。一个测试用例在程序上的动态切片（Dynamic Slice）是指其执行切片中对程序输出有影响的语句集，所以动态切片是执行切片的子集。为了提高测试用例选择的精度，Agrawal 等人额外提出两种切片准则：相关切片（Relevant Slice）和近似相关切片（Approximated Relevant Slice）。一个测试用例在程序上的相关切片，包括该测试用例的动态切片和可能会影响程序输出的谓词语句集。近似相关切片则在程序分析时更为保守，它包括动态切片和执行切片中的所有谓词语句集。使用上述切片准则，可以对测试用例和被测程序进行预处理，即每个测试用例与相应的切片 sl 对应。当被测程序被修改后，那些在切片 sl 中包含了代码修改的测试用例需要被选择出来并重新执行。但上述方法同样假设代码修改不会影响原有程序的控制流或数据流。当满足该假设时，通过切片技术可以减少需要分析的语句，从而提高分析效率。

与 Gupta 等人[35,36]和 Agrawal 等人[37]的研究工作不同，Bates 和 Horwitz 将被测程序建模为程序依赖图，并提出一种基于切片的 RTS 方法[38]。该方法分两个阶段。第一阶段从 T 中识别出在 P' 中所有可复用测试用例。他们首先提出等价执行模式（Equivalent Execution Pattern）并定义如下。

定义 4（等价执行模式） P 中的语句 S 和 P' 中的语句 S' 具有等价执行模式，当且仅当：(1)对于在 P 和 P' 上均可以正常执行结束的任何测试用例，语句 S 和 S' 的执行次数保持一致。(2)对于在 P 上可以正常执行结束、在 P' 上不可以正常执行结束的测试用例，语句 S 的执行次数不少于语句 S'。(3)对于在 P' 上可以正常执行结束、在 P 上不可以正常执行结束的测试用例，语句 S' 的执行次数不少于语句 S。

随后他们采用切片技术，将 P 和 P' 中的语句划分为不同的执行类（Execution Classes），同一执行类中的所有语句具有等价执行模式。最后通过识别出同时包含 P 和 P' 中语句的执行类来辅助识别出所有可复用测试用例。

第二阶段主要从可复用测试用例中选择出部分测试用例用于 P' 的测试。选择的标准是测试用例可能会覆盖 P' 中受代码修改影响的语句，若 P' 中的语句 S' 受到代码修改影响，当且仅当：(1)在 P 中没有对应语句 S；(2)S' 的行为与 P 中对应语句 S 的行为不等价，若两个语句的 PDG 切片同构，则这两个语句行为等价。

Binkley 进一步提出基于系统依赖图的切片方法[39,40]，将 Bates 和 Horwitz 的过程内选择技术扩展为过程间选择技术。他们通过引入共同执行模式（Common Execution Pattern）来描述过程间的调用模式。

3.1.5 *防火墙法*

Leung 和 White 在集成测试阶段引入并随后实现了防火墙法[41-43]。其核心思想是在受代码修改影响且需要重新测

试的模块周围构建一道防火墙。他们将模块划分为如下3类：(1)未改动模块：模块M未作任何修改并记为NoCh(M)。(2)仅代码修改模块：模块M修改后规约保持一致，仅内部代码发生修改，记为CodeCh(M)。(3)规约修改模块：模块M的规约已经被修改并记为SpecCh(M)。

假设模块A调用模块B，则两个模块间的可能状态组合有9种，在回归测试中将模块A和模块B进行集成时，可以忽略组合NoCh(A)∧NoCh(B)。如果模块A和B在代码级别或规约级别上同时发生修改，则集成测试这两个模块的测试用例均需要同时执行，这种情况占据了4种组合；剩下的4种组合属于改动的模块调用未改动的模块，或者相反，这种情况构成了集成测试的边界(即防火墙)。

根据上述分析，当一个模块被修改时，任何与修改后模块集成测试相关的测试用例均需要被选择出来。所以所有可覆盖代码修改的测试用例将被选择出来，但也可能会选择出一些无关测试用例。除此之外，Leung和White在他们的研究工作中发现集成测试时的测试效果并不理想，其根源在于一些可检测出缺陷的测试用例并不在已有测试用例集T中。所以回归测试中的风险一部分是由于已有测试用例集的质量较低。

目前防火墙法已经成功应用于面向对象程序[44-46]、图形用户界面程序[47]和COTS组件[48-51]的测试工作中。

3.1.6 其他基于源代码的RTS方法

除了上述线性规划法、数据流分析法、图遍历法、程序切片法和防火墙法外，研究人员也提出了一些其他基于源代码的RTS方法，现分别介绍如下。

符号执行(Symbolic Execution)通过使用抽象符号表示程序中的变量取值来模拟程序的运行。Yau和Kishmot结合输入域划分和符号执行来选择和执行测试用例[52]。首先对代码及规约通过静态分析完成对输入域的等价类划分。随后，移除失效测试用例并设计新的测试用例，以确保每个等价类至少被其中一个测试用例覆盖到，当获知代码修改信息后，可以从对应的控制流图中确定可到达代码修改的边集。当符号执行这些测试用例时，选择覆盖代码修改的测试用例并用于程序P'的测试。该方法从理论上讲非常有效，但在应用推广上受限于符号执行法本身的高复杂度，所以难以适用于大规模软件；同时该方法仅支持代码添加和代码修改操作，不支持代码删除操作。

Benedusi等人提出基于路径分析法(Path Analysis)的RTS方法[53]。该方法将P'中的多条路径(其路径用代数表达式进行表示)作为输入；随后处理这些表达式，并获得多个无循环典型路径；然后通过对P和P'中的典型路径进行对比，将P'的路径划分为new、modified、cancelled或unmodified类型；最后分析测试用例在P上覆盖的典型路径信息，并选择出所有覆盖modified类型路径的测试用例。但该方法在选择时仅考虑覆盖modified类型路径而忽略new和cancelled类型路径，这并不合理，所以这类方法也不具有安全性，同时计算开销也较大。

Laski和Szermer提出簇(Cluster)识别法[54]来对被测程序的CFG进行分析，从中识别出具有单一入口和出口的子图，并将这些子图称为簇。给定修改前程序P和修改后程序P'，该方法假设：(1)P中的每个簇封装了某些代码修改；(2)P中的每个簇在P'中都有唯一的簇对应；(3)若将图中的每个簇视为单一节点，则两个图中对应节点存在1-1映射关系。

基于上述假设，采用一系列操作，如节点折叠和节点移除，可以对修改前后程序的CFG进行简化。在该过程中，若P对应的CFG中某一节点在P'中不存在对应节点，则将该节点标记为MOD状态。最终这些MOD节点被封装入一个或多个簇内。如其他RTS技术一样，该方法也需要搜集测试用例在P上的执行信息，但在测试用例选择时仅需选择覆盖了上述簇的测试用例。这类方法虽然可以选择出所有可覆盖代码修改的测试用例，且支持任何代码修改类型，但因修改代码仅占簇包含代码的一部分，所以可能会选择出未覆盖代码修改的测试用例。可以说该方法通过牺牲选择精度来确保其安全性。

Chen等人提出基于修改的方法，他们引入测试框架TestTube，该框架采用基于代码修改的方法进行测试用例选择[55]。TestTube将被测程序P划分为程序实体，通过搜集测试用例的执行信息，可以建立测试用例和程序实体间的关联关系。TestTube同样将P'划分为程序实体，并从P中识别出包含代码修改的程序实体，所有覆盖这些程序实体的测试用例需要被选择出来。基于修改的方法与基于图遍历法的共同处在于：对修改前后代码进行分析并识别出修改模块，并从T中选择出覆盖修改模块的测试用例来，所以该方法也是安全的RTS方法。但TestTube关注的程序实体不仅可以是函数，还可以是变量、数据类型和预处理宏等。但其不足在于不能处理指针运算。TestTube为了可以处理变量和数据类型这类程序实体，假设程序内所有变量的创建和操作均可从源代码分析中进行推断，但该假设仅适用于没有指针和类型强制运算的程序。

Vokolos和Frankl提出文本差异法，该方法基于修改前后程序的文本差异来执行测试用例的选择。具体来说，借助unix系统的diff工具在源程序级别上识别出代码修改[56]。

陈振宇等人提出聚类测试用例选择法[57,58]。他们根据测试用例在P上的执行轨迹采用聚类算法进行分类，并假设同一聚类内的测试用例在P'上也具有相似的执行行为。他们首先尝试采用一种动态测试用例聚类采样策略从每一个聚类中选择典型测试用例[57]，随后引入半监督聚类学习方法，该方法需要预先分析已有测试结果以得到测试用例间的两两约束关系(即Must-link和Cannot-link关系)。实证研究表明该方法可以有效提高测试用例的选择效果[58]。

上述所有研究工作一般将RTS问题视为单目标优化问题，Yoo和Harman将帕累托效率(Pareto Efficiency)引入到RTS问题，他们的方法同时考虑了多个优化目标，例如代码覆盖率、历史缺陷检测信息和测试用例执行开销，并可以通过计算获得一组非主导的且等价最优的测试用例子集[59]。

3.1.7 基于源代码的典型RTS技术比较

本节对基于源代码的典型RTS技术进行系统比较，主要比较因素包括方法类型、程序建模方法、代码修改分析方式、RTS技术是否安全、实现评测程序的编程语言和RTS技术有效性的评估方式，最终结果如表1所列。

表1 基于源代码的典型RTS技术比较

文献	方法类型	建模方法	代码修改分析	安全性	编程语言	评估方式
Fischer 等人[4,8]	线性规划法	源代码	可达矩阵	否	Fortran	案例研究
Hartmann 等人[9,10]	线性规划法	源代码	可达矩阵	否	C	案例研究
Harrold 等人[11,13]	数据流分析法	流图	数据流分析	否	Fortran	案例研究
Rothermel 等人[15]	图遍历法	控制依赖图	控制依赖分析	是	C	案例研究
Rothermel 等人[16]	图遍历法	程序依赖图和系统依赖图	控制依赖和数据依赖分析	是	C	案例研究
Rothermel 等人[17,18]	图遍历法	控制流图	控制依赖分析	是	C	实证研究
Rothermel 等人[21]	图遍历法	过程间控制流图	控制依赖分析	是	C++	实证研究
Harrold 等人[22]	图遍历法	Java 类间图	控制依赖分析	是	Java	实证研究
Gupta 等人[35,36]	程序切片法	控制流图	数据依赖分析	否	Fortran	案例研究
Agrawal 等人[37]	程序切片法	源代码	程序切片	否	Fortran	案例研究
Bates 等人[38]	程序切片法	程序依赖图	程序切片	否	Fortran	案例研究
Binkley 等人[39,40]	程序切片法	系统依赖图	程序切片	否	C	案例研究
Leung 等人[41-43]	防火墙法	源代码	防火墙策略	是	C	案例研究
Yau 等人[52]	符号执行法	控制流图	控制依赖分析	否	Fortran	案例研究
Benedusi 等人[53]	路径分析法	源程序	路径对比	否	Fortran	案例研究
Laski 等人[54]	簇识别法	控制流图	基于簇的分析	是	Fortran	案例研究
Chen 等人[55]	基于修改的方法	源代码	代码对比	是	C	案例研究
Vokolos 等人[56]	文本差异法	源代码	Diff 工具	否	C	案例研究
陈振宇等人[57,58]	聚类分析法	源代码	无	否	C	实证研究
Yoo 等人[59]	多目标优化法	源代码	无	否	C	实证研究

3.2 基于模型的RTS技术

近些年来，基于模型的测试逐渐受到学术界和工业界的关注。Briand 等人在软件设计阶段，提出一种基于 UML 模型的 RTS 方法[60,61]。他们假设在设计模型和测试用例间存在追踪关系，则可以通过对 UML 设计模型进行代码修改影响分析来选择面向代码级别的测试用例。具体来说，他们对 UML 模型中的修改类型进行了分析，并按照 Leung 和 White 的建议[2]进行划分，包括 Obsolete、Retestable 和 Reusable 类型。他们实现了一个针对 UML 模型的自动影响分析工具并通过实证研究进行了分析。结果虽然表明方法的有效性与系统特征密切相关，但值得注意的是，将该方法应用于一些大规模系统时能取得良好效果。除此之外，Dent 等人、Pilskalns 等人和 Farooq 等人也分别对基于 UML 的 RTS 技术展开了研究[62-64]。Le Traon 等人、Wu 和 Offutt 则在 UML 模型基础上对一般回归测试问题进行了研究[65,66]。Cartaxo 等人基于标记迁移系统(Labeled Transition Systems)，提出基于相似函数的 RTS 方法[67]。Hemmati 等人基于 UML 状态机，对 Cartaxo 等人提出的相似函数进行扩展，借助状态机中迁移上的触发(Triggers)和防范(Guards)来构造相似函数，同时采用遗传算法来进一步提高测试用例的选择效果[68]。随后他们对该方法为什么，以及在何种条件下可以生效进行了更为深入的探讨[69]。

Muccini 等人同样也在软件设计阶段对 RTS 技术进行了研究，但他们主要关注的是软件体系结构[70,71]。

4 评测数据集

从表1可以看出，为了评估 RTS 技术的有效性，早期的研究工作一般采用案例研究(Case Study)的方式。但随着 Rothermel 和 Harrold 等人在 RTS 问题上的深入研究，在1997年之后研究人员逐渐采用实证研究(Empirical Study)的方式，本节将实证研究中常用的评测程序分为两类(C/C++和 Java)并依次对其中的程序对象特征进行简介。目前大部分程序都可以从内布拉斯加大学林肯分校的 SIR 库中去下载[72]。

4.1 C/C++程序

在实证研究的早期，研究人员经常使用西门子套件(Siemens Suite)中的程序作为标准评测数据[17-19]。这些程序最早用于研究控制流和数据流准则对缺陷检测能力的影响，该套件包含7个小规模 C 语言程序(程序名为 printtok、printtok2、replace、schedule、schedule2、tcas 和 totinfo)。这些程序的代码规模一般介于200行到500行之间。程序中的缺陷注入由西门子研究人员完成并确保与实际缺陷保持一致。但西门子套件中包含的7个程序规模普遍较小，不能有效代表大规模程序中的缺陷特征。在后续的实证研究中，研究人员也逐渐采用了一些大规模程序[57-59]。其中 Space 程序是欧洲航天局开发的针对数组定义语言的解释器程序。该程序包含6000多行可执行代码、1万多个测试用例和38个版本。Flex 程序是词法分析器生成器，在输入词法规则后可以生成一个词法分析器，该程序包含1万多行可执行代码、500多个测试用例和5个版本。

4.2 Java程序

随着 Java 编程语言的广泛应用和 JUnit 测试框架的日益成熟，研究人员开始考虑使用 Java 程序进行有效性评估。Harrold 等人在他们的实证研究中考虑了4个 Java 程序[22]，其中 Siena 是一个基于互联网的事件通知系统，包含185个方法、7个程序版本和138个测试用例。JEdit 是一个基于 Java的文本编辑器，包含多个功能，例如语法突出、正则表达式查找和替换、多粘贴板和宏记录，该程序包含3495个方法、11个程序版本和189个测试用例。Jmeter 是 Web 应用压力测试工具，包含109个方法、5个程序版本和50个测试用例。RegExp 是一个正则表达式解析库，包含168个方法、10个程序版本和66个测试用例。

Orso 等人为了评估他们的 RTS 技术，采用了3个规模更大的 Java 程序[24]，其中 Jaba 是 Java 字节码的分析工具，包含70k行代码、5个程序版本和707个测试用例。Daikon 是动态不变式检测工具，包含167k行代码、5个程序版本和200

个测试用例。Jboss是Java应用服务器，包含532k行代码、5个程序版本和639个测试用例。

5 评测指标

Rothermel和Harrold提出了分析和比较不同RTS技术的统一框架，他们采用的评测指标包括查全率、准确率、执行效率和可扩展性[5,6]。

假设RTS技术M选择出的测试用例构成集合T'，其中可检测出代码修改的测试用例构成集合T_f'，T中所有可以检测出代码修改的测试用例构成集合T_f。

则查全率R的计算公式为：

$$R=\frac{|T_f'|}{|T_f|}\times 100\% \quad (1)$$

准确率P的计算公式为：

$$P=\frac{|T_f'|}{|T'|}\times 100\% \quad (2)$$

假设测试用例集T中包含100个可检测代码修改的测试用例，采用RTS技术M选择出80个测试用例，其中可检测出代码修改的测试用例是40个，则该测试用例选择技术的查全率是40%，查准率是50%。对于任意修改前程序P、修改后程序P'和已有测试用例集T，若RTS技术M的查全率是100%，则称RTS技术M具有安全性。

对RTS执行效率的度量主要从时间开销和存储开销两个角度出发。其中时间开销主要包括RTS技术的执行时间，若执行时间小于运行$T-T'$测试用例所需的时间，则回归测试中采用该RTS技术具有经济有效性。而存储开销主要包括方法执行时需要存储的测试用例执行历史信息和程序分析信息。

RTS技术的可扩展性主要考察RTS技术是否适用于大型软件产品、是否适用于各种编程语言、是否适用于复杂的代码修改，以及是否适用于实际的测试应用场景等。

已有的研究表明，不同的RTS技术在执行效率和准确率上存在折衷。例如Orso等人[24]在分析大规模软件时，发现准确率高的RTS技术分析一般程序（需要分析到语句或分支）的开销较大，而程序分析开销较小（仅需分析到类或方法）的RTS技术则准确率较低。

6 RTS问题未来面临的挑战

RTS问题作为回归测试中的一个研究热点，在未来仍存在很多研究点值得国内外学者关注。在这一节对这些研究点进行尝试性探讨：

(1)充分利用目前代码修改影响分析领域的最新研究成果，精确高效地识别出与代码修改存在依赖关系的其他程序实体，从而提高RTS技术的选择精度并降低其执行开销。

(2)已有研究表明已有测试用例集的质量和测试用例调度策略对修改后程序的测试具有重要影响及作用。为了对代码修改进行充分测试，除了考虑本文综述的RTS技术，测试人员还需要综合考虑测试用例集扩充技术、缩减技术和优先级技术，从而在企业实际回归测试流程中提出高效可行的解决方案，以提高回归测试的效果和效率。

(3)缺陷定位(Fault Localization)是一种高效自动的软件调试技术并在近些年得到国内外学者的关注，已有研究成果表明测试用例的维护（例如测试用例集缩减和测试用例优先级排序等）将对缺陷定位效果产生重要影响。随着RTS技术的深入研究，进一步探讨旨在提高缺陷定位效果的RTS技术值得研究人员关注。

(4)将RTS的已有研究成果与企业的实际回归测试流程进行结合。已有研究方法一般采用对照试验(Controlled Experiment)方式进行有效性评估，所得结论不一定适用于来自企业的大型软件产品。其次已有工具也未充分考虑到与软件企业目前已有的开发、测试和调试流程相结合。若要将学术界的研究成功应用到企业界，则需要进一步提高测试工具的自动化程度，改善工具的用户界面并保证工具具有一定的鲁棒性。对上述挑战提出有效解决方案将能有效提高研究成果的应用价值并大幅度提高企业内的测试和调试效率。

结束语 RTS问题是回归测试研究中的一个研究热点。本文对该研究问题的背景、已有研究成果、评测数据集和评测指标进行了系统的总结和分析。目前该研究领域仍很活跃，本文也对未来的可能研究方向进行了一些尝试性的探讨。相信针对这些问题提出的有效解决方案将能提高软件企业内部的回归测试效率并大幅度降低软件开发和维护的费用。

参考文献

[1] Beizer B. Software Testing Techniques [M]. New York: Van Nostrand Reinhold, 1990

[2] Leung H, White L. Insights into Regression Testing [C]// Proceedings of the International Conference on Software Maintenance. 1989: 60-69

[3] Harrold M J, Orso A. Retesting Software during Development and Maintenance [C] // Proceedings of Frontiers of Software Maintenance. 2008: 99-108

[4] Fischer K. A Test Case Selection Method for the Validation of Software Maintenance Modifications [C]// Proceedings of International Computer Software and Applications Conference. 1977: 421-426

[5] Rothermel G, Harrold M J. A Framework for Evaluating Regression Test Selection Techniques [C]// Proceedings of the International Conference on Software Engineering. 1994: 201-210

[6] Rothermel G, Harrold M J. Analyzing Regression Test Selection Techniques [J]. IEEE Transactions on Software Engineering, 1996, 22(8): 529-551

[7] Yoo S, Harman M. Regression Testing Minimization, Selection and Prioritization: a Survey [J]. Software Testing, Verification & Reliability, 2012, 22(2): 67-120

[8] Fischer K, Raji F, Chruscicki A. A Methodology for Retesting Modified Software [C]// Proceedings of the National Telecommunications Conference. 1981: 1-6

[9] Hartmann J, Robson D J. Revalidation during the Software Maintenance Phase [C] // Proceedings of the International Conference on Software Maintenance. 1989: 70-80

[10] Hartmann J, Robson D J. Techniques for Selective Revalidation [J]. IEEE Software, 1990, 7(1): 31-36

[11] Harrold M J, Soffa ML. An Incremental Approach to Unit Testing during Maintenance [C]// Proceedings of the International Conference on Software Maintenance. 1988: 362-367

[12] Taha A B, Thebaut S M, Liu S S. An Approach to Software

Fault Localization and Revalidation based on Incremental Data Flow Analysis [C]// Proceedings of the International Computer Software and Applications Conference. 1989:527-534

[13] Harrold M J, Soffa M L. Interprocedual Data Flow Testing [C]// Proceedings of the Symposium on Software Testing, Analysis, and Verification. 1989:158-167

[14] Fisher M, Jin D, Rothermel G, et al. Test Reuse in the Spreadsheet Paradigm [C]// Proceedings of the International Symposium on Software Reliability Engineering. 2002:257-268

[15] Rothermel G, Harrold M J. A Safe, Efficient Algorithm for Regression Test Selection [C]// Proceedings of International Conference on Software Maintenance. 1993:358-367

[16] Rothermel G, Harrold M J. Selecting Tests and Identifying Test Coverage Requirements for Modified Software [C]// Proceedings of International Symposium on Software Testing and Analysis. 1994:169-184

[17] Rothermel G, Harrold M J. A Safe, Efficient Regression Test Selection Technique [J]. ACM Transactions on Software Engineering and Methodology, 1997, 6(2):173-210

[18] Rothermel G, Harrold M J. Experience with Regression Test Selection [J]. Empirical Software Engineering: Empirical Software Engineering, 1997, 2(2):178-188

[19] Rothermel G, Harrold M J. Empirical Studies of a Safe Regression Test Selection Technique [J]. IEEE Transactions on Software Engineering, 1998, 24(6):401-419

[20] Ball T. On the Limit of Control Flow Analysis for Regression Test Selection [C]// Proceedings of the International Symposium on Software Testing and Analysis. 1998:134-142

[21] Rothermel G, Harrold M J, Dedhia J. Regression Test Selection for C++ Software [J]. Software Testing, Verification and Reliability, 2000, 10(2):77-109

[22] Harrold M J, Jones J A, Li T, et al. Regression Test Selection for Java Software [C]// Proceedings of the Conference on Object-Oriented Programming, Systems, Languages, and Applications. 2001:312-326

[23] Beydeda S, Gruhn V. Integrating White- and Black-box Techniques for Class-level Regression Testing [C]// Proceedings of the International Computer Software and Applications Conference. 2001:357-362

[24] Orso A, Shi N, Harrold M J. Scaling Regression Testing to Large Software Systems [C]// Proceedings of the International Symposium on Foundations of Software Engineering. 2004:241-251

[25] Xu G, Rountev A. Regression Test Selection for AspectJ Software [C]// Proceedings of the International Conference on Software Engineering. 2007:65-74

[26] Zhao J, Xie T, Li N. Towards Regression Test Selection for Aspect-oriented Programs [C]// Proceedings of the Workshop on Testing Aspect-Oriented Programs. 2006:21-26

[27] Orso A, Harrold M J, Rosenblum D S, et al. Using Component Metadata to Support the Regression Testing of Component-based Software [C]// Proceedings of the International Conference on Software Maintenance. 2001

[28] Orso A, Do H, Rothermel G, et al. Using Component Metadata to Regression Test Component-based Software: Research articles [J]. Software Testing, Verification, and Reliability, 2007, 17(2):61-94

[29] Lin F, Ruth M, Tu S. Applying Safe Regression Test Selection Techniques to JAVA Web Services [C]// Proceedings of the International Conference on Next Generation Web Services. 2006:133-142

[30] Ruth M, Tu S. A Safe Regression Test Selection Technique for Web Services [C]// Proceedings of the International Conference on Internet and Web Applications and Services. 2007:47-47

[31] Ruth M, Tu S. Concurrency Issues in Automating RTS for Web Services [C]// Proceedings of the International Conference on Web Services. 2007:1142-1143

[32] Ruth M, Oh S, Loup A, et al. Towards Automatic Regression Test Selection for Web Services [C]// Proceedings of the International Computer Software and Applications Conference. 2007:729-736

[33] Tarhini A, Fouchal H, Mansour N. Regression Testing Web Services-based Applications [C]// Proceedings of International Conference on Computer Systems and Applications. 2006:163-170

[34] Li BX, Qiu D, Leung H, et al. Automatic Test Case Selection for Regression Testing of Composite Service based on Extensible BPEL Flow Graph [J]. Journal of Systems and Software, 2012, 85(6):1300-1324

[35] Gupta R, Harrold M J, Soffa M L. An Approach to Regression Testing using Slicing [C]// Proceedings of the International Confe-rence on Software Maintenance. 1992:299-308

[36] Gupta R, Harrold M J, Soffa M L. Program Slicing-Based Regression Testing Techniques [J]. Software: Testing, Verification and Reliability, 1996, 6(2):83-111

[37] Agrawal H, Horgan J R, Krauser E W, et al. Incremental Regression Testing [C]// Proceedings of the International Conference on Software Maintenance. 1993:348-357

[38] Bates S, Horwitz S. Incremental Program Testing using Program Dependence Graphs [C]// Proceedings of the Symposium on Principles of Programming Languages. 1993:384-396

[39] Binkley D. Reducing the Cost of Regression Testing by Semantics Guided Test Case Selection [C]// Proceedings of the International Conference on Software Maintenance. 1995:251-260

[40] Binkley D. Semantics Guided Regression Test Cost Reduction [J]. IEEE Transactions on Software Engineering, 1997, 23(8):498-516

[41] Leung H K N, White L. A Study of Integration Testing and Software Regression at the Integration Level [C]// Proceedings of the International Conference on Software Maintenance. 1990:290-301

[42] White L J, Leung H K N. A Firewall Concept for both Control-flow and Data-flow in Regression Integration Testing [C]// Proceedings of International Conference on Software Maintenance. 1992:262-271

[43] White L J, Narayanswamy V, Friedman T, et al. Test Manager: A Regression Testing Tool [C]// Proceedings of International Conference on Software Maintenance. 1993:338-347

[44] Kung D C, Gao J, Hsia P, et al. Class Firewall, Test Order, and Regression Testing of Object-oriented Programs [J]. Journal of

Object-Oriented Programming, 1995, 8(2): 51-65

[45] White L, Robinson B. Industrial Real-time Regression Testing and Analysis using Firewalls [C]// Proceedings of the International Conference on Software Maintenance. 2004: 18-27

[46] White L, Jaber K, Robinson B, et al. Extended Firewall for Regression Testing: an Experience Report [J]. Journal of Software Maintenance and Evolution, 2008, 20(6): 419-433

[47] White L, Almezen H, Sastry S. Firewall Regression Testing of GUI Sequences and their Interactions [C]// Proceedings of the International Conference on Software Maintenance. 2003: 398-409

[48] Zheng J, Robinson B, Williams L, et al. A Lightweight Process for Change Identification and Regression Test Selection in Using COTS Components [C]// Proceedings of the International Conference on Commercial off-the-Shelf (COTS)-Based Software Systems. 2006: 137-146

[49] Zheng J, Robinson B, Williams L, et al. Applying Regression Test Selection for COTS-based Applications [C]// Proceedings of the International Conference on Software Engineering. 2006: 512-522

[50] Zheng J, Williams L, Robinson B, Pallino: Automation to Support Regression Test Selection for COTS-based Applications [C]// Proceedings of the International Conference on Automated Software Engineering. 2007: 224-233

[51] Zheng J, Williams L, Robinson B, et al. Regression Test Selection for Black-box Dynamic Link Library Components [C]// Proceedings of the International Workshop on Incorporating COTS Software into Software Systems: Tools and Techniques. 2007: 9-14

[52] Yau S S, Kishimoto Z. A Method for Revalidating Modified Programs in the Maintenance Phase [C]// Proceedings of International Computer Software and Applications Conference. 1987: 272-277

[53] Benedusi P, Cmitile A, De Carlini U. Post-maintenance Testing based on Path Change Analysis [C]// Proceedings of the International Conference on Software Maintenance. 1988: 352-361

[54] Laski J, Szermer W. Identification of Program Modifications and its Applications in Software Maintenance [C]// Proceedings of the International Conference on Software Maintenance. 1992: 282-290

[55] Chen Y F, Rosenblum D, Vo K P. Testtube: A System for Selective Regression Testing [C]// Proceedings of the International Conference on Software Engineering. 1994: 211-220

[56] Vokolos F, Frankl P. Empirical Evaluation of the Textual Differencing Regression Testing Technique [C]// Proceedings of the International Conference on Software Maintenance. 1998: 44-53

[57] Yan S L, Chen Z Y, Zhao Z H, et al. A Dynamic Test Cluster Sampling Strategy by Leveraging Execution Spectra Information [C]// Proceedings of the International Conference on Software Testing, Verification and Validation. 2010: 147-154

[58] Chen S Y, Chen Z Y, Zhao Z H, et al. Using Semi-supervised Clustering to Improve Regression Test Selection Techniques [C]// Proceedings of the International Conference on Software Testing, Verification and Validation. 2011: 1-10

[59] Yoo S, Harman M. Pareto Efficient Multi-objective Test Case Selection [C]// Proceedings of the International Symposium on Software Testing and Analysis. 2007: 140-150

[60] Briand L C, Labiche Y, Buist K, et al. Automating Impact Analysis and Regression Test Selection based on UML Designs [C]// Proceedings of the International Conference on Software Maintenance. 2002: 252-261

[61] Briand L C, Labiche Y, He S. Automating Regression Test Selection based on UML Designs [J]. Journal of Information and Software Technology, 2009, 51(1): 16-30

[62] Deng D, Sheu P Y, Wang T. Model-based Testing and Maintenance [C] // Proceedings of the International Symposium on Multimedia Software Engineering. 2004: 278-285

[63] Pilskalns O, Uyan G, Andrews A. Regression Testing UML Designs [C]// Proceedings of the International Conference on Software Maintenance. 2006: 254-264

[64] Farooq Q, Iqbal M Z Z, Malik Z I, et al. An Approach for Selective State Machine based Regression Testing [C]// Proceedings of the International Workshop on Advances in Model-based Testing. 2007: 44-52

[65] Le Traon Y, Jeron T, Jezequel J M, et al. Efficient Object-oriented Integration and Regression Testing [J]. IEEE Transactions on Reliability, 2000, 49(1): 12-25

[66] Wu Y, Offutt J. Maintaining Evolving Component-based Software with UML [C]// Proceedings of the European Conference on Software Maintenance and Reengineering. 2003: 133-142

[67] Cartaxo E G, Machado P L, Neto F O. On the Use of a Similarity Function for Test Case Selection in the Context of Model-based testing [J]. Software Testing, Verification & Reliability, 2011, 21(2): 75-100

[68] Hemmati H, Briand L, Arcuri A, et al. An Enhanced Test Case Selection Approach for Model-based Testing: an Industrial Case Study [C] // Proceedings of the International Symposium on Foundations of Software Engineering. 2010: 267-276

[69] Hemmati H, Arcuri A, Briand L. Empirical Investigation of the Effects of Test Suite Properties on Similarity-Based Test Case Selection [C]// Proceedings of the International Conference on Software Testing, Verification and Validation. 2011: 327-336

[70] Muccini H, Dias M, Richardson D J. Reasoning about Software Architecture-based Regression Testing through a Case Study [C]// Proceedings of the International Computer Software and Applications Conference. 2005: 189-195

[71] Muccini H, Dias M, Richardson D J. Software-architecture based Regression Testing [J]. Journal of Systems and Software, 2006, 79(10): 1379-1396

[72] Do H, Elbaum S, Rothermel G. Supporting Controlled Experimentation with Testing Techniques: An Infrastructure and its Potential Impact [J]. Empirical Software Engineering, 2005, 10(4): 405-435