

基于“Good”测试套件的 GUI 回归测试优化

魏冬梅, 洪 玫, 李 波

(四川大学 计算机学院, 四川 成都 610064)

摘 要: 回归测试在软件测试过程中是非常重要的, 同时也是非常费时费力的。为了适应软件测试的需求, 提高回归测试的效率, 降低测试成本, 针对目前 GUI 回归测试的困难, 依据 Atif M. Memon 等人提出的新的回归测试方法, 根据一个“Good”的测试套件对测试成本和“fault-detection effectiveness”的影响, 突出一个“Good”的测试套件的特点, 给出一个新的回归测试方法。该方法根据测试套件的特点, 有针对性地选择合适的测试用例来构建回归测试套件。这样不仅有针对性, 同时也优化了测试套件, 提高了测试效率。

关键词: 回归测试; “Good”测试套件; 覆盖标准; GUI

中图分类号: TP206⁺.1

文献标识码: A

文章编号: 1673-629X(2008)07-0001-04

A Regression Test Optimization Based on “Good” GUI Test Suite

WEI Dong-mei, HONG Mei, LI Bo

(School of Computer, Sichuan University, Chengdu 610064, China)

Abstract: The regression test is very important and time-consuming in software testing process. For the need of software testing, enhancing the effectiveness and reducing the cost of the testing, as for the current difficulty of GUI regression testing, in terms of a “Good” test suite’s effect on test cost and “fault-detection effectiveness”, focusing on the character of a “Good” GUI’s test suite, proposes a new optimizing method of regression testing to structure test suite. According to the characteristic of a test suite, chose suited test case to build the test suite. So this method was not only pertinence, but also optimized the test suite, enhance the effectiveness of regression testing.

Key words: regression testing; “Good” test suite; coverage criteria; GUI

0 引 言

GUIs (Graphical User Interfaces) 已经变成一种与软件系统交互的普遍方式。这种方式通过接收来自用户或系统产生的事件作为输入, 通过消息或方法的调用作为与内部应用程序的交互, 经过处理事件产生图形化的输出; 为用户提供了软件的层次化、图形化、可视化的操作前端, 使得软件更加易于使用。现在 GUI 开发人员致力于大量的代码来实现它们, 这使得“GUI 几乎占据了一个应用程序 60% 的代码量^[1]”。考虑到 GUIs 应用的重要性, GUIs 的正确性测试是不容忽视的。

目前的 GUI 测试技术是不完善的, 而且需要大量的手工操作。现在运用的比较多的技术是所谓的“记录/回放”技术 (record/playback), 著名的测试软件 win-

runner 就是采用了这种技术对 GUI 程序进行回归测试, 这种技术先将测试者的标准操作过程以简单的脚本形式记录下来, 测试时自动回放所有这些操作以检验是否会引发新的问题, 这个过程需要大量的手工操作。经验表明, 通过这种工具生成一个只有 50 个事件的测试用例就需要花费 20~30 分钟的时间, 这也是这种技术不能满足 GUI 测试的原因。

虽然回归测试是传统软件测试和维护的重要环节, 占据了整个软件开发成本的 1/3^[2], 而且大约 30% 左右的错误 (Bug) 是通过回归测试发现的^[3], 但是对于 GUI 的软件测试, 传统的回归测试仍束手无策。一个 GUI 的输入和输出依赖于图形元素的布局, 布局的变化——按钮的摆放位置、菜单的组织结构都会引起 GUI 状态的变化, 使得旧的测试用例不能使用。而且 GUIs 软件产品的开发过程通常是渐进式或迭代式的, 这需要在开发过程中不断测试代码改变部分对未改变部分的影响, 即不断进行回归测试。

随着 GUIs 在现在软件系统中的广泛应用, GUIs 给整个软件的回归测试带来了更大的挑战。即使简单地修改事件代码而不对原有的 GUI 或操作模式做修

收稿日期: 2007-10-19

基金项目: 国家“863”计划基金项目 (2006AA12A104)

作者简介: 魏冬梅 (1980-), 女, 硕士研究生, 研究方向为软件工程; 洪玫, 教授, 硕士生导师, 研究方向为数据库及其应用、计算机网络、软件工程、实时系统等。

改也将产生问题,如果没有进行详细的回归测试,那么引入新错误将是不可避免的。因此,在 GUI 回归测试过程中,非常有必要通过选择正确的回归测试策略来改进回归测试的效率和有效性。可以通过两个途径来提高回归测试效率,即回归测试自动化和测试用例优化。

回归测试自动化就是测试工具能够自动选择测试用例进行回归测试;测试用例优化,就是在测试套件中选择或构造合理的测试用例进行回归测试,所选择或构造的测试用例集覆盖度尽可能地大,而所花费的代价尽可能地少。

文中针对 GUI 测试过程中频繁的回归测试,在构造一个“Good”测试套件过程中,参考文献[4]Atif M. Memon 等人针对一个“Good”的测试套件的特点对测试成本和故障检测效率影响的研究,利用这些特点对测试用例进行优化,进一步选择或构建更有效的测试套件对软件进行测试,来提高整个回归测试的效率。

1 GUI 相关概念

定义 1: 一个 GUI 由对象集合 $O = \{o_1, o_2, \dots, o_m\}$ (例如, label, form, button, text) 及其属性集合 $P = \{p_1, p_2, \dots, p_m\}$ (例如, font, caption) 组成。

定义 2: GUI 的状态 S 是 GUI 包含的所有对象的属性的集合。其中 S_I 定义为有效的初始状态集, 当 GUI 第一次被调用时, 状态为 S_i , 其中 $S_i \in S_I$ 。

定义 3: 函数表达式 $S_j = e(S_i)$ 表示在状态 S_i 执行事件 e 时得到的状态 S_j , 其中事件 e 作为一个序列发生作用。

定义 4: 一个合法的事件序列由 $\{e_1, e_2, \dots, e_n\}$ 组成, 其中事件 e_{i+1} ($0 < i < n$) 在 e_i 之后执行。

定义 5: GUI 测试用例 T 是一个二元组 $\langle S_0, e_1, e_2, \dots, e_n \rangle$, S_0 是测试用例 T 的初始状态, $\{e_1, e_2, \dots, e_n\}$ 是一个合法的事件序列, n 是这个测试用例的长度。

定义 6: 如果测试用例 T_n 中指定的状态 S_0 是不可达的, 或者 $\{e_1, e_2, \dots, e_n\}$ 是不合法的, 那么 T_n 是不可用的测试用例。

定义 7: 如果一个 GUI 修改后测试用例 T_n 仍可以使用, 那么 T_n 是可用的测试用例。

定义 8: 不可用的测试用例 $T_p \langle S_0, e_1, e_2, \dots, e_n \rangle$, 如果其中 GUI 初始状态 S_0 是可达的, 或者可以把事件序列 $\{e_1, e_2, \dots, e_n\}$ 修改为合法的事件序列, 那么 T_p 是可修改的测试用例; 否则 T_p 是不可修改的测试用例。

2 GUI 测试套件的影响

文献[5]说明了一个测试套件的三个重要因素(测试套件的大小, 事件的组织形式, 测试用例的长度)对测试成本和故障检测效率可能的影响。在每个实验中改变一个因素, 保持另外两个因素不变。实验结果表明:

(1) 套件中测试用例长度和事件的组织形式没有变化, 测试套件的大小改善了它的故障检测能力。不同的是, 生成一个较大的测试套件需要更多的时间, 同样执行也需要更多的时间。就是说, 时间和测试套件的大小是成比例的。

(2) 尽管测试用例的长度对故障检测数量没有明显的影响, 但是分析表明存在一些故障只能被一些长的测试用例检测到, 而短的测试用例不能检测。就是说测试用例的长度影响检测的故障的种类。

(3) 如果某个事件与内部代码的一个功能单元有交互的话, 那么不含这个事件的测试套件肯定就不能检测到在功能单元代码中的故障; 但是一些不与内部代码交互的事件对某个功能单元代码中的故障也就没什么影响了。

3 基于“Good”GUI 测试套件的优化方法

针对 GUIs 越来越广泛的应用, 频繁的回归测试使得 GUI 测试更加困难, 而且测试自动化工具也只能起到辅助的作用, 最根本的还是对测试用例的组织和选择, 文中选择通过测试用例的优化来提高 GUIs 回归测试的效率。

在一个有效的软件测试过程中, 设计一个测试套件被广泛认为是最基本的一步^[5], 而一个“Good”的测试套件的构建最根本的还是许许多多测试用例的组织和创建, 所以设计者需要根据测试套件的特点来合理组织测试用例。这部分主要参考文献[2]给出的回归测试方法如图 1、图 2 所示, 把一个“Good”的测试套件的特点作为整个测试套件设计和组建的指导原则, 把以事件为依据的两种类型覆盖标准——intra-component coverage criteria 和 inter-component coverage criteria^[4]来测试事件的充分性, 作为检验事件序列测试充分性的标准, 权衡一个“Good”的测试套件的影响和事件序列测试充分性之间的关系, 合理组织和创建事件序列来设计或修改测试用例, 达到优化的目的。

针对图 3 中用椭圆标注的部分, 提出了一种新的优化方法, 由三部分组成:

(1) Test case checker: 参考文献[2], 这部分把原来的测试套件分成三部分: a. 可用的测试用例; b. 可修改的不可用的测试用例; c. 不可修改的不可用的测试用

例。

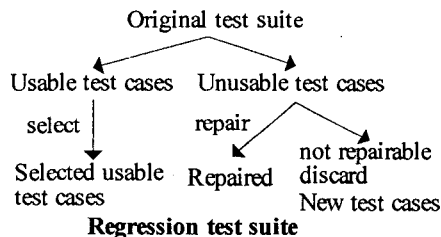


图 1 回归测试方法

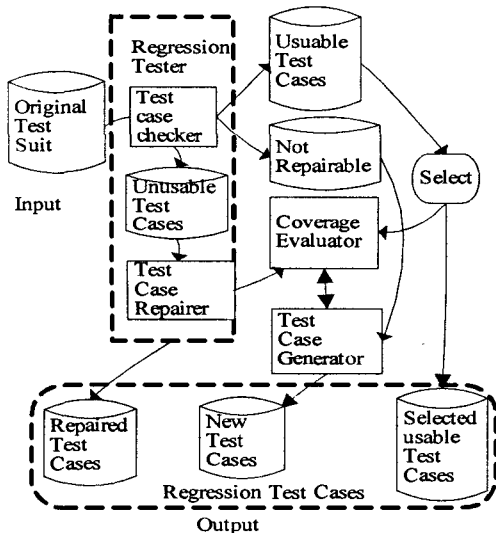


图 2 回归测试各组件及组件之间的交互

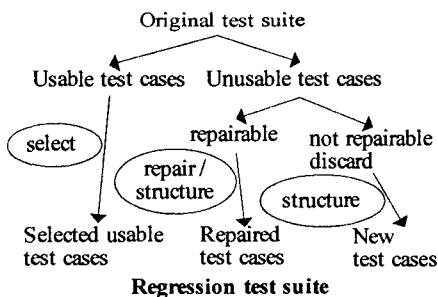


图 3 改进了的回归测试方法

(2) Test case repairer: 参考文献[2], 这部分通过添加或删除事件来调整事件序列, 修改不可用的测试用例。

(3) Test case optimizer: 这部分根据文献[5]一个“Good”测试套件对测试成本和错误检测效率的影响, 合理组织事件序列, 合理设计事件的长度和大小, 在此期间需要按照文献[4]以事件为依据的覆盖标准来检验测试用例对 GUI 测试的充分性, 从而设计或修改测试用例。

此优化方法说明了构建一个“Good”测试套件需要满足第三部分的要求, 不管是可用的测试用例还是不可用的测试用例, 在设计和修改的时候都需要考虑测试用例的有效性和充分性, 使得在最短的时间内最大程度地发现故障, 来提高回归测试的效率。方法中

各组件及它们之间的交互如图 4 所示。

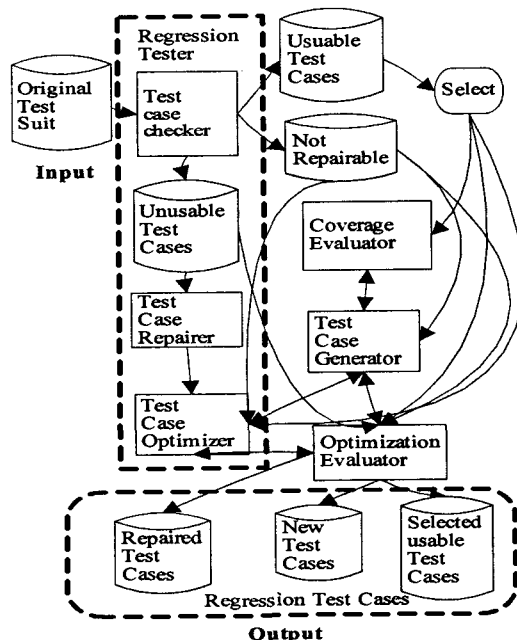


图 4 优化的回归测试方法各组件及组件之间的交互

文中提出的优化方法步骤如下:

第一步 对修改后的 GUIs 的状态进行分析, 根据修改前后整个 GUIs 的 event-flow-graph(EFG)事件流图^[4]的差异, 明确 event-flow-graph(EFG)中需要测试的点和路径。

第二步 利用 Test case checker 对原来的测试套件中的所有测试用例分类: 可用的测试用例 T_u 和不可用的测试用例 T_n 。

第三步 针对第一步中的需求在可用的测试用例中选择测试用例进一步测试。

第四步 根据第一步中的需求, 针对不可用的测试用例 T_n 利用 Test case repairer 进行分类: 可修改的测试用例 T_p 和不可修改的测试用例 T_p' 。同时针对可修改的测试用例 T_p , 修改或者删除或者添加事件改变事件序列来修改测试用例, 使得可以进一步使用。把不可修改的测试用例 T_p' 丢弃掉, 利用 Test case generator 进一步设计新的测试用例。

第五步 对第二、三、四中得到的 T_u 、 T_p 和 T_p' 进行评价和优化, 直至得到一个“Good”测试用例。

其中测试套件的构造过程与图 2 的不同之处体现在:

(1) 在图 4 中增加了一个 Test Case Optimizer 和 Optimization Evaluator。前者针对从 Test case checker 和 Test case repairer 得到的三种测试用例进行优化; 后者对优化后得到的测试用例进行分析和评价。

(2) 在可用的测试用例中, 选择合适的测试用例。图 4 中的 select 不同于图 2 中的 select。这里针对一个

“Good”的测试用例对测试用例进一步选择。比如根据不同的故障类型选择合适长度和大小测试用例等。

(3)在不可用的但可修改的测试用例中,通过添加或删除事件来修改或组建事件序列,修改测试用例。图 3 中的 repaired/structure 是针对不可用的测试用例,修改事件所涉及的代码,或者是修改事件序列,或者是创建合适的事件来修改测试用例。在这个过程中根据以事件为依据的覆盖标准^[6]——intra - component coverage criteria 和 inter - component coverage criteria 来测试事件的充分性,进一步合理组织测试用例,并对得到的测试用例通过 Optimization Evaluator 进行分析和评价。

(4)构建新的测试用例。同样图 4 中重新构建新的测试用例时,根据 GUI 的修改后的状态变化、用户需求、软件设计规范和结构,需要综合考虑(2)和(3)测试用例的交互空间和覆盖范围来设计新的测试用例。在创建过程中始终按照以事件为依据的覆盖标准来选择事件序列,组织测试用例。

(5)其中 Coverage Evaluator 和 Optimization Evaluator 针对“Good”的测试套件的构建过程进行充分性和有效性分析和评价。前者针对测试用例的测试充分性进行分析和评价,后者针对测试用例的测试有效性进行分析和评价。在整个回归测试过程中,二者分别依据测试用例的覆盖标准和“Good”的测试套件的特点对测试用例进行构造和优化,而且二者是相辅相成的。

在上述 5 个不同的过程中,始终把把一个“Good”测试套件对测试成本和故障检测效率的影响作为基本的指导原则,合理组织事件序列,构建合适的测试用例,使得设计的测试套件能够在预算内最大可能地发现软件的缺陷和错误,提高整个软件的测试效率。虽然一个“Good”测试套件的影响因素不止三个,但是同样可以根据对测试成本和故障检测效率的影响合理设计测试套件。

以上说明可以看出此优化方法的优点主要表现在以下几点:

- 1)提高了故障检测的效率;
- 2)缩短了整个软件测试的周期;
- 3)降低了整个软件回归测试的成本;
- 4)提高了回归测试的充分性和有效性。

同样在不同程度上也带来了回归测试的难度,主要表现在以下几点:

a. 增加了测试人员分析和构建充分的和有效的测试用例的难度;

b. 增加了测试套件构造的复杂性;

c. 由 a. 和 b. 可知,此方法对测试人员的测试能力要求更高。

比较上述优缺点,可以看出,此方法虽然增加了回归测试的难度和复杂性,但是在整个软件的测试过程中,考虑到频繁的回归测试带来的更加巨大的困难,从总体上来看,还是提高了整个软件的测试效率。因为在此方法中,基于一个“Good”测试套件的构建,使得回归测试套件对故障检测的充分性和有效性得以提高,缩短了整个软件的测试过程,所以可以说此方法降低了回归测试的成本,提高了回归测试的效率。

4 结束语

根据 Atif M. Memon 等人用一个“Good”的测试套件对测试成本和故障检测效率影响的研究探讨了回归测试优化方法。就是在 Atif M. Memon 等人提出的回归测试方法的基础上,针对测试套件的特点,根据以事件为依据的覆盖标准组织合理的事件序列,进一步组建和设计测试用例达到优化的目的。该想法还处在理论分析阶段,还不成熟,不能很好地应用,所以需要进一步通过实验进行进一步的研究。

参考文献:

- [1] Memon A M. GUI Testing: Pitfalls and process[J]. Software Technologies, 2002(8): 87 - 88.
- [2] Memon A M, Sofia M L. Regression Testing of GUIs[C]// Proceedings of the 9th ACM/IEEE European Software Engineering Conference (ESEC) and 11th ACM SIGSOFT International Symposium on the Foundations of Software Engineering (FSE - 11). Helsinki, Finland: [s. n.], 2003: 118 - 127.
- [3] Maedche A, Motik B, Stojanovic I. Managing Multiple and Distributed Ontologies in the Semantic web[J]. VI DB J, 2003, 12(4): 286 - 302.
- [5] Memon A M, Pollack M E, Sofia M L. Hierarchical GUI test case generation using automated planning[J]. IEEE Trans on Software Engineering, 2001, 27(2): 144 - 155.
- [4] Xie Qing, Memon A M. Studying the Characteristics of a Good GUI Test Suite[C]//proceedings of the 17th IEEE International Symposium on Software Reliability Engineering (ISSRE 2006). Raleigh, NC, USA: [s. n.], 2006: 159 - 168.
- [6] Memon A M, Sofia M L, Pollack M E. Coverage Criteria for GUI Testing[C]//Proceedings of the 8th European Software Engineering Conference (ESEC) and the 9th ACM SIGSOFT International Symposium on the Foundations of Software Engineering. Austria: Vienna University of Technology, 2001: 256 - 267.