基于 UML 顺序图的回归测试用例生成研究

文 瑾
（昆明学院信息技术学院 昆明 650214）

摘 要 首先在分析 UML 模型可测试性的基础上，解析顺序图，得到场景测试树，再遍历该树，得到场景的输入、预期输出、约束条件以及场景环境，这 4 部分组成了测试用例。再通过比较 UML 设计修改前后两个版本的顺序图，找出其中的改变信息，并在测试用例和变化信息间建立映射关系，最后从原测试用例组中挑选出与映射关系有关的测试用例。

关键词 软件测试；UML 顺序图；面向对象；回归测试用例

Generation of Regression Test Cases Using UML Sequence Diagram

WEN Jin
(*Kunming University, College of Information Technique, Kunming* 650214, *China*)

Abstract Beginning with analyzing the testability of UML model, a scenario tree can be created by parsing it. The input, respected output, constraint and scenario context are generated through traversing the scenario tree, and the four parts constitute the test case. We can find the changed information before and after revision by comparing versions of the sequence diagram. Finally, a mapping relation between changed information and test cases is generated. Using this map relation we can select test cases affected by revision.

Keywords software test; UML sequence diagram; object-oriented; regression testing cases

1 引 言

回归测试是软件维护活动中一个必不可少的过程，它用来确保软件修改后的正确性和可靠性，提高软件的质量。在面向对象的程序中，由于使用了大量类的继承，程序修改的扩散效应更为明显，减少回归测试的耗费尤为重要。为了降低这部分开支，研究人员针对测试用例的选择和使用提出各种各样方法，其中包括基于修改等信息从原有的测试用例集合中选取部分测试用例来进行回归测试[1]；使用一些典型算法生成测试用例[2,3]；用 UML 状态图和类图生成测试用例[4,5]；以及其他一些面向对象程序的回归测试[6,7]。

但是，目前面向对象的回归测试用例生成的研究大多集中在基于程序源代码、数据流分析方法、确定性测试方法、或者基于软件的规约测试技术，其中采用 UML 状态图是规约测试的典型代表之一。上述方法存在一定的局限性，如确定性测试方法使得理想的序列同步执行，但动态绑定令相同的同步信息可能导致不同的绑定结果，会产生冲突；基于状态的方法在并发单元数量增加时，技术的难度也会增加。UML 顺序图描述了对象之间传递的交互消息在时间上的依赖关系，也是一种重要的设计测试用例方法，它开辟了测试领域的新方向。本文将论述把 UML 顺序图描述的依赖关系引入到面向对象程序软件的回归测试中的合理性和可用性，并从 UML 顺序图生成回归测试用例，以便进一步提高回归测试的效率。

2 回归测试策略

回归测试同其他测试的主要区别在于有无可供复用的测试用例集。对于所开发的一个软件项目，测

基金项目：云南省教育厅科学研究基金(09Y0348)。
作者简介：文瑾，副教授，主要研究方向为软件测试和算法，E-mail：wencomputer@hotmail.com。

试组成员会将所开发的测试用例保存到“测试用例库”中，并对其进行维护和管理。当软件需要进行回归测试时，就可以根据所选择的回归测试策略，从测试用例库中提取合适的测试用例并对其进行优先排序后组成回归测试包，随后运行回归测试包来实现回归测试。

如何有效地复用测试用例因成为回归测试主要的研究方向[7]，Rothermel 等人在深入研究了测试用例的选择和执行问题后，提出了一个典型的回归测试过程：

假设 P 是一个过程或程序，P' 是 P 修改后版本，T 是 P 的测试用例集。

(1)选择 $T' \subseteq T$，其中 T' 是程序 P' 用来测试的用例集；

(2)利用 T' 测试程序 P'，以确认程序 P' 用 T' 测试的正确性；

(3)如果需要，创建 T''，其中 T'' 为程序 P' 新增的部分测试用例；

(4)利用 T'' 测试程序 P'，以确认程序 P' 用 T'' 测试的正确性；

(5)根据 T、T' 和 T'' 产生新的测试用例集 T'''。

第一步涉及回归测试用例选择方法，它选择 T 中的一个子集 T' 用于测试修改后的程序 P'；第二步和第四步是为了有效执行测试用例和验证测试结果；第三步处理覆盖率问题；第五步处理测试库维护问题，更新和存储新的测试用例集。以上每一步都涉及一些重要问题，本文只考虑回归测试用例的选择方法。

3 UML 顺序图的可测试性

UML 顺序图以时间顺序描述了执行特定用例时对象之间动态的交互关系，即触发哪些交互以及交互以何种次序发生[8]。它的长处是以简单有效的方式传达交互内事件的次序，在 UML2.0 顺序图中，事件发生是交互的基本语义单元，交互被认为是事件的有序集合，不再是消息的有序集合[9]。为了便于分析顺序图中的交互，对顺序图进行简单的形式化定义[10]。

定义 1：顺序图 SD 是一个 3 元组，记 SD=(O，M，E)，其中，$O=\{o_i \mid i=1, 2, \cdots, m\}$指 SD 所含对象的有限集合；$M=\{m_j \mid j=1, 2, \cdots, n\}$指 SD 所含消息的有限集合；E 指事件的有限集合，含全部发送事件和接收事件。消息 m 的发送用(m，s)表示，接收用(m，r)表示。

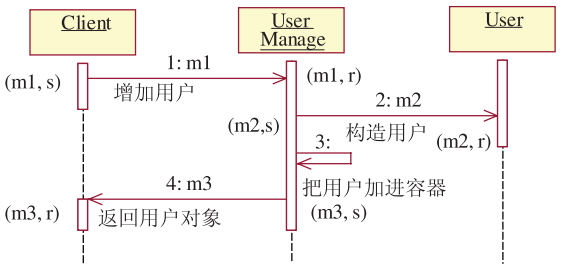

图 1 简单顺序图

在图 1 所示的顺序图中：

O={Client, User Manager, User}；

M={ml, m2, m3}；

E2{(m1, s)，(ml, r)，(m2, s)，(m2, r)，(m3, s)，(m3, r)}；

UML 顺序图描述了消息序列之间的前后关系，即消息的发送事件和接收事件之间的时间顺序。

定义 2：用“ϵ”表示事件之间的顺序关系，并且满足以下三个性质：

(1)因果性：对任意 $m_j \in M$，有$(m_j, s) \in (m_j, r)$，即消息必须先发送后接收。

(2)可控性：在一个对象的生命线上，假设 el 事件出现在发送事件 e2 的上方，则 el∈e2；也就是说，若在发送事件的生命线上方，有其他事件，那么这些事件都先于这个发送事件发生，这反映了对象有机会选择发送消息的时机。

(3)队列性：在一个对象的生命线上，如果有二个接收事件 e1 和 e2，并且 e1 在 e2 上方，同时它们对应的发送事件也在同一个生命线，则 el<<e2；即如果对象 o_1 向对象 o_2 发送消息，先发送的消息先接收。

但“<<”不是一个全序关系，因为一个顺序图，允许多个事件序列，但不是所有的事件序列都有效。有效的事件序列定义如下：

定义 3：简单顺序图 SD=(O , M , E)中，事件序列 E=(e1, e2, …, em)为有效事件序列，仅满足以下 2 个条件：

(1)所有事件在事件序列E中均出现，并只出现一次。即{el, e2, …, em}=E，若 i≠j，有 ei≠ej；

(2)任意二个事件 ei, ej∈E，若 ei<<ej，则在事件序列 E 中，ei 位于 ej 前面。

一个行为的特定动作顺序，即一个用例的实例可以用场景表示。顺序图中，对象之间的消息流可以用场景来定义。

定义 4：简单顺序图 SD=(O , M , E)中，定义场景为消息序列 M=(ml, m2, …, mn)，且满足如下 2个

条件：

(1)所有消息在消息序列 M 中均出现，并只出现一次，即{m1, m2, …, mn}=M，若i≠j，有mi≠mj；

(2)任意二个消息 mi、mj ∈ M，若(mi , s)<<(mj, s)发生，则在消息序列 M 中，mi 位于 mj 前面。

根据场景的定义得知，找到事件的序列，就可确定该事件序列的场景。

基于 UML 的顺序图回归测试用例的生成，主要是比较 UML 设计修改前后两个版本的顺序图，找出其中的变化信息，并将这些变化信息映射到回归测试用例上。对顺序图来说，修改后顺序图可能变化情况是：添加了新的消息、删除了旧的消息以及消息的顺序的改变。下面再定义一个规则用来确定在顺序图中的不同变化类型：

定义 5：添加的顺序图的集合，删除的顺序图的集合，定义如下：$S^a=S^2—S^1$，$S^d=S^1—S^2$，其中：S^a 表示添加的顺序图，S^d 表示删除的顺序图，S^1 表示顺序图的初始版本，S^2 表示顺序图的修改版本。

4　测试用例的生成算法

在基于前面定义基础上，下面给出的测试用例生成算法步骤[10]是以 UML 顺序图的场景为基础。

步骤 1：将提取的状态信息加到顺序图中；即检查顺序图的中每个对象，若存在状态图，从中找出所有初始状态，若有多个则用 OR 相连，然后以“对象名.in(状态名)”的格式将其添加到生命线的最上方。最后，用对象约束语言 OCL 为其添加约束条件。

步骤 2：创建顺序图的场景测试树；通过对顺序图中所有有效事件进行遍历，能够得到所有的场景，从而获得场景测试树。

步骤 3：创建测试用例[11]；(1)遍历场景测试树，用 and 逻辑符将各个方法调用时的约束以及场景的输入和预期输出相连。(2)确定场景的环境条件。即先确定顺序图中所有的测试单元(test unit)，对每个测试单元，从带 OCL 约束条件的顺序图中推出其环境条件，称其为范畴(category)；根据一些约束信息将每个范畴划分成多个选择(choice)，然后将与每个场景相关的选择组合在一起，就构成了该场景的环境条件。最后将方法调用序列、输入、预期输出、环境条件组合在一起就构成了测试用例。

5　回归测试

假定每个用例的名字在用例图中是唯一的，通过对软件修改前后的二个顺序图进行比较，找出添加、删除和改变的消息，以获取有关 UML 的设计变更。将这些变更的信息映射到测试用例上，作为测试用例生成的依据[11]。由于本文只考虑 UML 中的顺序图，通过比较修改前后二个版本的 UML 顺序图，找出其中变化的顺序图就能找到与该顺序图对应的用例。通过变更了的顺序图的集合 S^c，根据每个场景分别对应一个顺序图和需求之间的可溯性，就找到了该顺序图对应的场景，再由场景到其对应的测试用例，这些测试用例就组成了回归测试用例。

6　实验分析

实验选用自动取款机取款测试用例作为测试对象，重点检测回归测试用例的使用情况。根据第 4 节介绍的“测试用例的生成算法”，用户利用自动取款机取款的全部流程：插卡、输入密码、选择金额、取款、取卡等操作，测试用例设计为：

(1)插入正确银行卡，提示输入密码；

(2)密码正确，进入主界面，提示用户选择；

(3)密码不正确，提示密码错误，重新输入；

(4)输入金额<余额，提示用户金额不足，重新输入或取卡；

(5)输入金额为 150，提示用户取款金额不符合规范，重新输入或退出；

(6)输入正确金额，输出钱款；

(7)用户未按时取款，自动收回钱款；

(8)用户未按时取卡，自动吞卡；

(9)用户按时取卡，返回到主页面。

以上一个测试目的仅对应一个测试用例，此实验共有 9 个测试用例。把原始测试用例按场景测试树分为 5 类，即 5 个测试子集，即：T1 为(1)中的测试用例，T2 为(2)、(3)中的测试用例，T3 为(4)、(5)、(6)中的测试用例，T4 为(7)中的测试用例，T5 为(8)、(9)中的测试用例。

在实验中，我们设定第(5)项测试用例所对应的对象被删除了，需要进行回归测试。通过比较两个版本的顺序图和场景测试树，找到场景测试树 T3 发生了变化，所以测试时只需用 T3 中的测试用例(4)(6)

进行测试。在以上的实验中，我们只使用 2 个测试用例就完成了回归测试，由此节省近 80%的测试用例，提高了测试效率。

7 结束语

本文通过对 UML 顺序图进行形式化描述，给出基于 UML 顺序图的场景测试用例生成算法的完整方法和步骤，并通过对顺序图进行比较，找出添加、删除和改变的消息，以获取有关 UML 的设计变更。并将这些变化信息映射到场景测试树上，作为回归测试用例生成的依据。但该方法仍有一些问题需要继续研究，如 UML 顺序图语义的完善以及测试覆盖准则中对不同类型信息的处理机制等。今后的研究方向需实现测试的驱动模块，使测试用例具有可执行性，设计一个基于顺序图自动生成回归测试用例的原型系统。

参 考 文 献

[1] Kayes M I. Test case prioritization for regression testing based on fault dependency [C] // the 3rd International Conference on Electronics Computer Technology (ICECT), 2011: 48–52.

[2] 远俊红, 柳青, 李洋. 遗传算法在基于 UML 活动图生成测试用例中的应用 [J]. 云南大学学报(自然科学版), 2009, 31(S1): 67-70.

[3] 游亮, 卢炎生. 测试用例集启发式约简算法分析与评价 [J]. 计算机科学, 2011, l2: 147-150.

[4] Sabahat N, Farooq Q A, Malik Z I. Analyzing impact of change in uml sequence diagrams on state machine based regression testing [J]. Software Engineering, 2010.

[5] 李杨, 张春海, 张美玲. 基于 UML 的回归测试软件测试方法的研究与应用 [J]. 计算机技术与发展, 2010, 9: 235-238.

[6] Rothermel G, Harrold M J, Dedhia J. Regression test se1ection for C++ software [J]. Journal of Software Testing, Verification and Reliability, 2000, 10(6): 77-l09.

[7] Rothermel G, Untch R H, Chu C Y, et al. Prioritizing test cases for regression testing [J]. IEEE Transactions on Software Engineering, 2001: 929–948.

[8] Miles R, Hamilton K. UML2.0 学习指南 [M]. 汪青青译. 北京: 清华大学出版社, 2007.

[9] Hammal Y. Branching Time semantics for UML 2.0 sequence diagrams [C] // International Federation for Information Processing, 2006, LNCS 4229: 259-274.

[10] 李志强, 邵培南, 来辉. 基于 UML 顺序图的测试用例生成 [J]. 计算机工程, 2010, 36(22): 58-60.

[11] 逄瑞娟. 基于 UML 顺序图的场景测试用例生成研究 [D]. 青岛大学, 2007.