DOI:10.15923/j.cnki.cn22-1382/t.2017.2.14

回归测试用例设计策略

王荣丽， 侯秀萍*

(长春工业大学 计算机科学与工程学院，吉林 长春 130012)

摘 要：通过 DAG 图将部分测试用例重新整合，形成回归测试的测试用例，对修改的功能及其相关功能进行测试，还可以测试程序的业务流程。

关键词：测试用例；依赖关系；DAG 图

中图分类号：TP 311 **文献标志码**：A **文章编号**：1674-1374(2017)02-0179-05

Regression test case design strategy

WANG Rongli， HOU Xiuping*

(School of Computer Science and Engineering, Changchun University of Technology, Changchun 130012, China)

Abstract: With DAG diagram, test cases are reunited to form the cases for regression test. The modification and related functions can be tested, so other business diagram of the test program.

Key words: test cases; dependency; DAG diagram.

0 引 言

在软件开发过程中，软件测试越来越受到重视，测试能够帮助开发人员开发出更健壮、易用性更强、适应性更好的程序。为了更快地执行测试，有很多学者研究了云测试[1-3]，将测试任务分发至云平台，并行执行测试。针对回归测试，有效地执行测试用例则是测试过程中最重要的任务，如何设计高效的回归测试用例是文中主要研究的问题。提出一种将测试任务重新整理的办法，设计出回归测试的测试用例，用来定位功能性错误的位置。

功能测试主要分为功能模块测试和业务流程测试。各个功能模块之间可能存在依赖关系，一般白盒测试中设计测试用例时，会采用插装的方式来屏蔽掉这种依赖关系。但黑盒测试如何处理这种依赖关系是需要解决的问题。

测试用例的执行策略直接影响到测试的进度和结果，测试任务间的依赖关系也制约着测试用例的执行顺序。有很多学者研究过测试用例的执行顺序问题。载如昕[4]等对测试用例的优先级进行调整来提高测试的错误检测率，但没有对有依赖关系的测试用例进行处理；陈翔[5]等对回归测试中的测试用例优先排序技术进行了总结，通过

收稿日期：2016-12-25

基金项目：吉林省科技厅科技成果转化项目(20130303010GX)

作者简介：王荣丽(1991—)，女，汉族，吉林长春人，长春工业大学硕士研究生，主要从事软件测试方向研究，E-mail：wangrongli1217@163.com. * 通讯作者：侯秀萍(1964—)，女，汉族，吉林长春人，长春工业大学教授，硕士，主要从事软件测试方向研究，E-mail：houxiuping@ccut.edu.cn.

实例分析阐述了用例优先级对测试的影响；苏小红[6]等实现了结合测试用例约简与联合依赖概率建模对软件进行错误定位的方法；张艳梅[7]分别针对类单元测试中的路径覆盖测试数据生成问题和集成测试中类测试顺序的确定问题研究了类间依赖关系。

将依赖关系整理清楚，有利于对测试进行更好的规划。测试任务之间的依赖关系可以通过DAG图来表示，深度遍历DAG图并设计修改测试用例，文中将有依赖关系的测试用例进行整合，形成适合回归测试[8]的新用例，这些用例间将不存在依赖关系。在云测试环境下，使得回归测试用例可以并行执行。

1 相关概念

1)测试任务。是用户用来管理测试的基本单位，其中包括测试所需环境的要求，要执行的相关测试用例等。可以包含一个功能模块的测试用例，也可以包含多个模块的测试用例。

2)测试用例(Test Case)[9]。能有效地发现软件缺陷的最小测试执行单元，一个测试任务可能对应一个或多个测试用例。可以将一个测试任务的测试用例集形式化表示为：

$$\text{Tests}=\{T_1,T_2,\cdots,T_i,\cdots\}$$

$$T_i=\{\text{Target},\text{ID},\text{Input},\text{Resullts}\}$$

式中：Tests——测试用例集；

T_i——其中一个测试用例；

Target——测试的目标项；

ID——测试用例的编号；

Input——描述、输入、操作；

Results——预期结果。

3)测试任务之间的依赖关系。对于一个测试任务，有可能与其他测试任务之间有依赖关系，即一个测试任务的执行可能影响到另一个测试任务的执行。一个测试任务的输出可能成为另外一个测试任务的输入。

4)测试任务DAG图。是有向无环图$G=\{V,E\}$，$V=\{v_1,v_2,\cdots,v_n\}$，用来表示测试任务集合。E是边集合，记$E=\{e_1,e_2,\cdots,e_n\}$，其中，$e_i=<V_i,V_j>(i,j\in n$且$i\neq j)$，表示测试任务之间存在依赖关系。

DAG图如图1所示。

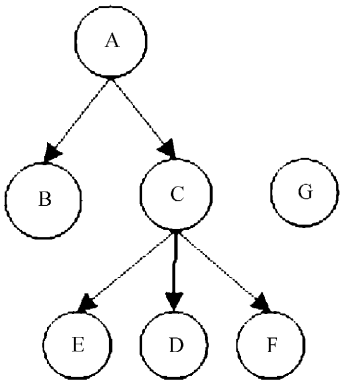

图1 DAG图

图中每个顶点代表一个测试任务，B任务依赖于A任务，G是个孤立的节点，与其他任务都不存在依赖关系。

5)所有的测试任务可以在一张DAG图中表示，从图中一个节点出发，能到达另一个节点，称这两个节点之间存在路径。存在路径的任务之间一定存在直接或间接依赖关系。v_i与v_j的路径$p=v_iv_{i+1},\cdots,v_{j-1}v_j$，其中，$v_i$与$v_{i+1}$之间存在边，$v_{j-1}$与$v_j$之间存在边。图1中，A与B、C、D、E、F均存在路径，G是一个孤立的节点，与其他节点均不存在路径。

2 测试用例整合策略

通常功能测试中，总会存在功能之间的依赖关系，即有些功能测试的前提条件是其他功能成功通过测试。如在ATM机上做取款操作，前提是账户里有存款金额且大于要取款的金额，那么取款功能就是依赖于存款功能的。在回归测试中，主要对修改过的部分进行测试。文中通过功能关系抽象出DAG图，将存在依赖关系的功能测试用例重新整合，形成部分新的测试用例。对修改功能代码的测试用例整合策略按照步骤执行如下：

1)对于修改过的测试任务v_i，根据功能需求，抽象出与v_i相关任务的DAG图。

2)如果修改的测试任务v_i是孤立节点，则用其原有测试用例$\text{Tests}v_i=\{Tv_{i1},Tv_{i2},\cdots,Tv_{ii},\cdots\}$对其进行测试；如果不是孤立节点，不改变其原有测试用例，然后从原有$\text{Tests}v_i$中选择Tv_{ii}，其中Tv_{ii}满足可以正确执行并可以跳转到以v_i为初始节点路径中的下一节点v_j。

3)深度优先遍历该DAG图，每遍历一个节

点，都做如下操作：

如果这个节点 v_j 是一条路径中最后一个节点，则遍历后不对测试用例做修改；

如果不是一条路径中最后一个节点，则对路径中下一节点 v_j 的测试用例 $\text{Testsv}_j=\{Tv_{j1},Tv_{j2},\cdots,Tv_{ji},\cdots\}$ 与 2）中选中的 Tv_{ii} 按照 4）中方法整合。

4）结合方法为：原测试用例 $Tv_{ii}=\{\text{Target},\text{ID},\text{Input},\text{Resullts}\}$，将 Testsv_j 中所有测试用例分别与 Tv_{ii} 合并，形成新的测试用例集 $N\text{Tests}v_j=\{NTv_{j1},NTv_{j2},\cdots,NTv_{ji},\cdots\}$。将 Tv_{ii} 分别与 Tv_{ji} 合并，形成新的测试用例 $NTv_{ji}=\{\text{Target},\text{ID},\text{Input},\text{Resullts}\}$，其中，$\text{Target}=Tv_{ji}.\text{Target}$，$\text{ID}=Tv_{ji}.\text{ID}$，$\text{Input}=T_{ii}.\text{Input}+Tv_{ji}.\text{Input}$，$\text{Results}=Tv_{ji}.\text{Results}$。

5）从 $N\text{Tests}v_j$ 中选出 NTv_{ji}，其中，NTv_{ji} 满足可以正确执行并可以跳转到遍历路径中下一节点。

6）反复执行以上过程，将新用例与路径中下一节点用例结合，直到遍历到所有节点并形成对应的新测试用例。

新的测试用例专门测试该路径中最后一个节点任务，并且可以测试这条路径是否有 BUG，即程序的流程是否正确执行，这样的测试用例有利于缺陷定位。

3　实例分析

以购物网站页面的部分功能测试用例为例进行整合。如果对登录页面进行了源码修改，那么与其有依赖关系的功能都应该进行回归测试，为了测试功能及业务流程是否畅通，将测试用例整合。用界面原型工具画出其业务跳转流程，如图 2 所示。

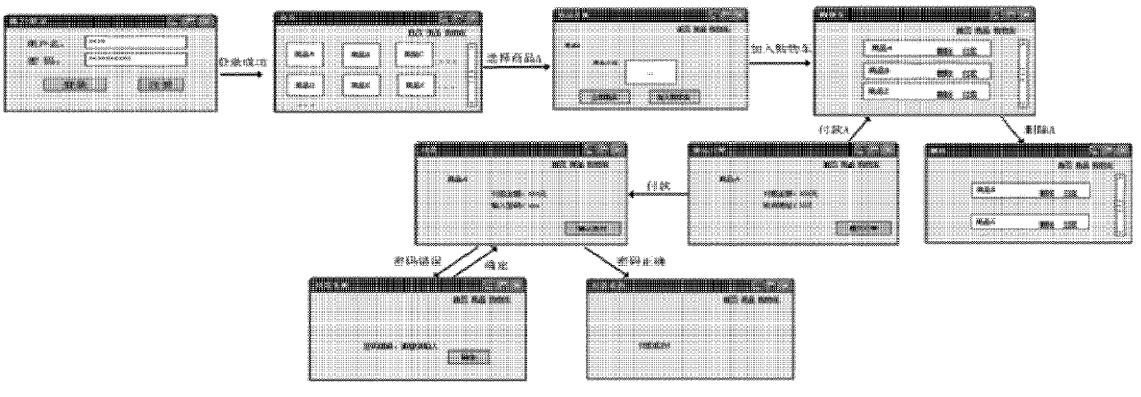

图 2　购物流程界面图

每个页面都是要进行测试的一个功能点，对应 DAG 图中的一个节点，即一个测试任务。每个测试任务都有其对应的局部测试用例，如改动一部分功能的源码，将与其有路径的功能点的测试用例做整合。

图 2 对应的 DAG 图如图 3 所示。

为了方便表示，将节点名称分别用字母表示，对照表见表 1。

原始测试用例见表 2。

最后的删除、付款成功、付款失败界面是为了展示最终结果的，所以没有设置它们的测试用例。

由于是对登录功能进行修改，所以 A 的测试用例 T_1、T_2、T_3 不做改变，从中选择可以正确执行到下一个路径节点的用例 T_3，深度遍历 DAG 图，遍历到节点 B，将 T_3 与 B 的测试用例 T_4、T_5 按照 4）中方法结合。得到新的测试用例 $T_4=\{\text{B},4,\text{Zhangsan 123 点击登录 点击商品 A，跳转到商品 A}\}$，$T_5=\{\text{B},5,\text{Zhangsan 123 点击登录 点击商品 B，跳转到商品 B}\}$。

从新的 T_4、T_5 中选择一个可正确执行到下一路径节点的用例 T_4，继续遍历 DAG 图到节点 C，将新用例 T_4 和 C 的用例 T_6 结合，得到新的测试用例 $T_6=\{\text{C},6,\text{Zhangsan 123 点击登录 点击商品 A 点击 A 加入购物车，跳转到购物车}\}$。

以此类推，得到用例 $T_7=\{\text{D},7,\text{Zhangsan 123 点击登录 点击商品 A 点击 A 加入购物车 点击 A 付款，跳转到订单界面}\}$，$T_8=\{\text{D},8,\text{Zhangsan 123 点击登录 点击商品 A 点击 A 加入购物车 点击 A 删除，跳转到删除界面}\}$，$T_9=\{\text{E},9,\text{Zhangsan 123 点击登录 点击商品 A 点击 A 加入购物车 点击 A}$

付款 点击确定,跳转到付款界面},T_{10}={F,10,Zhangsan 123 点击登录 点击商品 A 点击 A 加入购物车 点击 A 付款 点击确定 123,跳转到支付成功界面},T_{11}={F,11,Zhangsan 123 点击登录 点击商品 A 点击 A 加入购物车 点击 A 付款 点击确定 空,跳转到支付失败界面}。

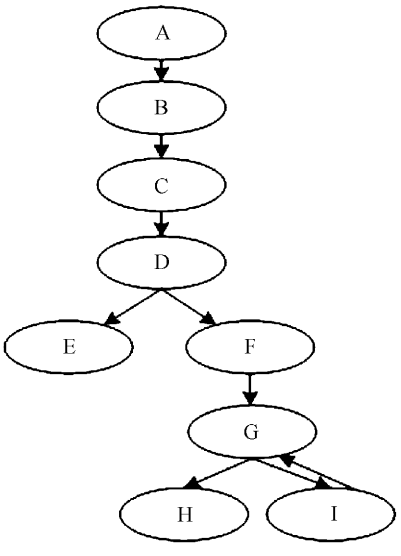

图 3 购物流程 DAG 图

这样使有依赖关系的功能成为一个整体,各个测试用例可以看成独立的任务。整合好的回归测试用例不仅可以测试其对应的功能点,同时也解决了依赖关系的问题,还可以测试该路径的逻辑是否因代码的修改而失效,使测试更具针对性。

表 1 节点对照表

字符	节点名称
A	登录
B	首页
C	商品详情
D	购物车
E	删除
F	提交订单
G	付款
H	付款成功
I	付款失败

表 2 原始测试用例

测试项	编号	描述/输入/操作	预期结果
登录	1	空 123 点击登录	请输入用户名
	2	Zhangsan 空 点击登录	请输入密码
	3	Zhangsan 123 点击登录	登录成功,跳转到首页
首页	4	点击商品 A	跳转商品 A
	5	点击商品 B	跳转商品 B
商品	6	点击加入购物车	跳转到购物车
购物车	7	点击付款	跳转到付款界面
	8	点击删除	跳转到删除界面
订单	9	点击确定	跳转到付款界面
付款	10	123	跳转到支付成功界面
	11	空	跳转到支付失败界面

4 结 语

通过对 DAG 图的分析,将有依赖关系的功能串联起来,重新整合了测试用例,不同测试用例针对不同功能,有利于快速定位到错误的功能模块,而且能够测试流程是否顺利执行。这些新的测试用例直接不存在依赖关系,可以发布到云平台上进行并行测试,该策略也对丰富软件测试理论起到了相应作用。

参考文献:

[1] 李乔.云测试研究现状综述[J].计算机应用研究,

2012,29(12):4401-4406.

[2] 曹咏春.云测试综述[J].研究与开发,2011,10:25-29.

[3] Lian Yu, Weik Tsail, Xiangji Chen, et al. Testing as a service over cloud [C]//Proceedings of Fifth IEEE International Symposium on Service Oriented System Engineering.2010:181-188.

[4] 载如昕,顾春华.用于黑盒测试的测试用例优先级改进算法[J].计算机工程与设计,2010,31(20):4343-4346.

[5] 陈翔,陈继红.回归测试中的测试用例优先排序技术述评[J].软件学报,2013,24(8):1695-1712.

[6] 苏小红,龚丹丹,王甜甜,等.结合用例约简与联合依赖概率建模的错误定位[J].软件学报,2014,25(7):1492-1504.

[7] 张艳梅.基于依赖性分析的面向对象程序测试技术研究[D].北京:中国矿业大学,2012.

[8] 高灿.基于抽象语法树的修改影响分析方法[J].长春工业大学学报:自然科学版,2012,33(4):387-390.

[9] 朱少民.软件测试方法和技术[M].北京:清华大学出版社,2014.