文章编号:1000-582X(2007)06-0093-04

切片在组件系统回归测试中的应用

傅鹤岗,李赋欣,曾 刚
(重庆大学 计算机学院,重庆 400030)

摘 要:在组件系统的回归测试中,为了确定修改所影响的部分,需要建立组件系统的依赖模型。组件系统具有高复用性和高复杂性,现有的依赖模型已经不适应描述组件系统,所以提出一种描述组件系统的层次依赖模型。通过对该模型中修改点的向前切片,得到修改所影响的部分,然后遍历该部分得到修改所影响的测试路径,进而有效地选择回归测试用例,提高了回归测试的效率。

关键词:切片;基于组件的系统;回归测试
中图分类号:TP311.5 **文献标志码**:A

组件是软件工程的一个重要研究领域,随着组件应用的不断扩大和作用的日益明显,组件的可靠性显得越来越重要,因此,国内外学者于20世纪90年代后期对组件测试开展了研究[1]。

基于组件的系统会随着需求的变化而不断更新,当系统被修改后,通常要进行回归测试来验证系统的正确性,以保证修改后系统的质量。在通常使用的回归策略中,最简单的策略是重用原来的所有测试用例进行测试,但这样可能导致测试成本过于昂贵,为了减少回归测试的成本,有人提出了回归测试选择技术,即从原有的测试用例集合中选取部分测试用例来进行测试,从而可以大大减少回归测试的时间和人力开销[2-6]。对于回归测试选择技术,首先要考虑如何高效识别软件修改所影响的部分,然后考虑如何重用现存的测试用例和测试套。在这方面,一些学者进行了研究和探索,如Chengying Mao等人在文献[3]中提出通过加强与组件开发者的联系来对组件系统进行回归测试,A. Orso和M. J. Harrold等人在文献[4]提出用Meta数据和方法来进行组件系统的回归测试。

切片技术最初由Weiser提出[5],它具有简化问题和缩小目标范围的特征,后来Horwitz等人提出通过建立系统依赖图来进行切片[7],进而进行测试用例的选择。可见对于组件系统的回归测试,可以通过建立依赖模型,计算向前或者向后切片来判别修改所影响的部分,进而选择和重用测试用例进行测试。但是,组件系统所具有的高复用性和高复杂性使得现有的依赖模型,如Larsen,Liang和Harrold所提出的面向对象依赖模型[8],已经不能够应用到现有的组件系统中。因此,针对基于组件的系统,提出了一种层次依赖描述模型,它通过对方法依赖、组件依赖以及系统依赖的分层描述,可以更清晰的阐述组件系统之间的依赖关系,更好的确定修改部分,进而进行测试用例的选择和重用。

1 依赖模型

一个基于组件的系统主要由一组组件和相关的计算(例如胶合代码)构成[3]。因此,文中的依赖模型扩充了Larsen,Liang和Harrold所提出的面向对象依赖模型[8],通过构造方法依赖图,组件依赖图和系统依赖图来建立组件系统的依赖模型。

1.1 方法依赖图

系统中的每一个方法都有自己的方法依赖图(Method Dependence Graph, MDG)。类似于传统的过程依赖图[9],可以用一个12元组的有向图来表示。MDG = $\{E_n, F_{in}, F_{out}, S, A_{in}, A_{out}, \text{CallSite}, E_{DD}, E_{CD}, E_{Pin}, E_{Pout}, E_{TD}\}$,其中$E_n$是方法的唯一入口点,$F_{in}$是方法的形式输入点集合,$F_{out}$是形式输出点集合,$S$是声明或谓词表达式集合,$A_{in}$是实际输入点集合,$A_{out}$是实际输出点集合,CallSite代表调用方法的顶点,E_{DD}是数据

收稿日期:2007-01-23。
作者简介:傅鹤岗(1950-),男,重庆大学副教授,主要从事软件工程,电子商务方向研究。
(Tel)023-65102495;(E-mail)hgfu@cqu.edu.cn。

依赖边集合,E_{CD}是控制依赖边集合,E_{Pin}和E_{Pout}分别是参数输入边和参数输出边集合,E_{TD}是传输依赖边集合。

对于F_{in}, F_{out},A_{in}, A_{out},E_{TD},E_{Pin},E_{Pout}笔者给出如下定义:

F_{in},方法的形式输入点x集合,x表示该方法的形式参数。

F_{out},方法的形式输出点x集合,x表示可能被方法修改的形式参数。

A_{in},方法的实际输入参数点x集合,x表示该方法的实际参数。

A_{out},方法的实际输出参数点x集合,x表示被方法修改的实际参数。

E_{TD},传输依赖边集合,为了表示上下文依赖关系,当某一实际输入点影响了某一实际输出点时,则从该实际输入点到对应的实际输出点间增加一条传输依赖边。

E_{Pin}, 参数输入边集合,从实际输入点到对应的形式输入点用参数输入边来连接。

E_{Pout}, 参数输出边集合,从形式输出点到对应的实际输出点用参数输出边来连接。

除去以上定义,从每个方法的入口点到方法的形式输入点、形式输出点通过控制依赖边相连;从方法的调用顶点 CallSite 到实际输入点、实际输出点也通过控制依赖边相连;图中剩余的边,则根据顶点间的数据依赖和控制依赖[5]关系由数据依赖边和控制依赖边进行连接。

1.2 组件依赖图

组件依赖图(Component Dependence Graph, CDG)表示了组件内的依赖关系,对于单个组件,可以用一个11元组的有向图来表示。CDG = {Com, I_{entry}, I_{Fin}, I_{Fout}, $I_{CallSite}$, I_{Ain}, I_{Aout}, E_{CD}, E_{Pin}, E_{Pout}, E_{TD}},其中 Com 是该组件唯一入口点,I_{entry}是接口入口点集合,I_{Fin}是接口的形式输入点集合,I_{Fout}是接口的形式输出点集合,$I_{CallSite}$是接口调用点,I_{Ain}是接口的实际输入点集合,I_{Aout}是接口的实际输出点集合,E_{CD}是控制依赖边集合,E_{Pin}是参数输入边集合,E_{Pout}是参数输出边集合,E_{TD}是传输依赖边集合。

其中,I_{Fin}, I_{Fout},I_{Ain}, I_{Aout}, E_{TD},E_{Pin},E_{Pout},与 MDG 中F_{in},F_{out}, A_{in}, A_{out},E_{TD}, E_{Pin},E_{Pout}的定义相似。而从组件的入口点 Com 到各个接口入口点通过控制依赖边连接;从接口入口点到形式输入点、形式输出点通过控制依赖边相连;从调用点$I_{CallSite}$到实际输入点、实际输出点通过控制依赖边相连。

1.3 系统依赖图

对于一个基于组件的系统$P = \{c_1, c_2, \cdots, c_i; m_1, m_2, \cdots, m_j\}$,$c$是组件,$m$是相关的计算,通常认为是一组方法;系统依赖图(System Dependence Graph, SDG)主要描述了它们之间的依赖关系,它可以形式化的用一个2元组表示,SDG = {CDGs, MDGs}。对于所有的附加计算,可以把它们看作一个虚拟的组件,这样在组件和组件之间,组件和虚拟组件之间,可能存在调用关系,则在调用点和被调用点之间增加一条调用边 C (Call edge)。组件系统层次依赖模型可以用图1表示。

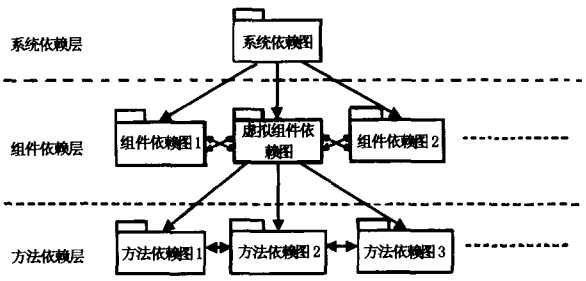

图1 组件系统层次依赖模型

2 对组件系统进行切片

Horwitz 提出了一种2阶段图形可达性算法来对系统依赖图进行切片[3]。其主要思想是:对于切片标准(s, v),第1步是从s点开始沿着各种流边、控制边、调用边和参数输入边(不沿参数输入边)反向遍历,并标记所有到达的顶点和边;第2步从第1步中到达的所有顶点开始沿着流边、控制边、参数输出边(不沿参数输入边和调用边)反向遍历,标记到达的顶点和边,最终的切片由2阶段所到达顶点的集合和这些顶点引入的边的集合构成。

该算法考虑了由参数输出边所引起的不准确现象,使所得到的切片具有比较高的精度,但是该算法是针对过程间切片而提出的,并且是向后切片;为了对提出的组件系统依赖模型进行向前切片,笔者扩展了2阶段图形可达性切片算法,提出了一种适合文中依赖模型的向前切片算法:组件系统向前切片算法。

2.1 切片准则

在切片准则$<p, x>$中,x是变量或者方法调用,p是切片点;如果x是变量,那么它必然在p处定义或者使用,如果x是方法,那么它在p处一定被调用。

2.2 组件系统向前切片算法

第一阶段:从切片点沿着所有边(除了参数输入和调用边)计算可以到达的点;第二阶段:从第一阶段中所到达的点出发,沿着所有边(除了参数输出边)遍历。这样可以避免遍历时随着参数输出边的下降和随后的上升而导致的切片不准确。

算法:向前切片算法 ForwardSlicing(G,(p,x))

Input：基于组件系统依赖图 G 和切片标准(p,x)

Output：切片 slice

begin

n = 一个对应于 G 中 p 的顶点。

从 n 出发沿着所有边(除了参数输入、调用边)遍历，标记遍历所经过的结点。

从第一阶段中所标记的点出发，沿着所有边(除了参数输出边)遍历，标记遍历所经过的结点。

切片 slice 就是这 2 部分的合集。

return slice

end

3　回归测试中测试用例的选择

在文献[2]中，对测试用例的选择是通过对修改前后的程序进行对比，找到所有受影响的测试路径，进而进行测试用例的选择。在程序不太复杂的情况下，具有不错的可行性，但是随着程序复杂度的提高，该方法的复杂度提高的很快，因此在面对复杂的组件系统时可行性较小。

通过对组件系统的层次依赖模型进行切片，可以确定修改所影响的部分，大大缩小了搜索的范围；然后通过对所得切片的遍历，得到受影响的状态序列，进而求得所影响的执行路径，然后复用所有经过这些执行路径的测试用例，达到对修改部分测试的目的。下面是测试用例选择算法。

算法：Getaffectpath(x,p,G,Epath)

Input：基于组件系统依赖图 G，切片标准(p,x)和所有执行路径 Epath

Output：回归测试用例集 Regression suite

begin：

Dependences = Φ；affectpath = Φ

Regression suite = Φ；

slice = ForwardSlicing(G,(p,x))

n = p 点

for 每一个{slice - n}中入度为 0 的顶点

通过深度优先策略求出从顶点 n 到该顶点所有路径，即所有的状态序列。

将该状态序列存入 Dependences 中。

endfor

for 对 Dependences 每一个状态序列

for Epath 中的每一条执行路径

if 这条执行路径包含该状态序列

把这条执行路径插入 affectpath

endif

endfor

endfor

将所有经过这些影响路径 affectpath 的测试用例插入回归测试用例集 Regression suite 中

Return Regression suite

end

4　组件系统回归测试过程及性能分析

提出的组件系统回归测试过程如图 2 所示，主要由 3 个步骤构成，分别是对组件系统进行预处理、对组件系统进行切片、回归测试用例的选择和执行。

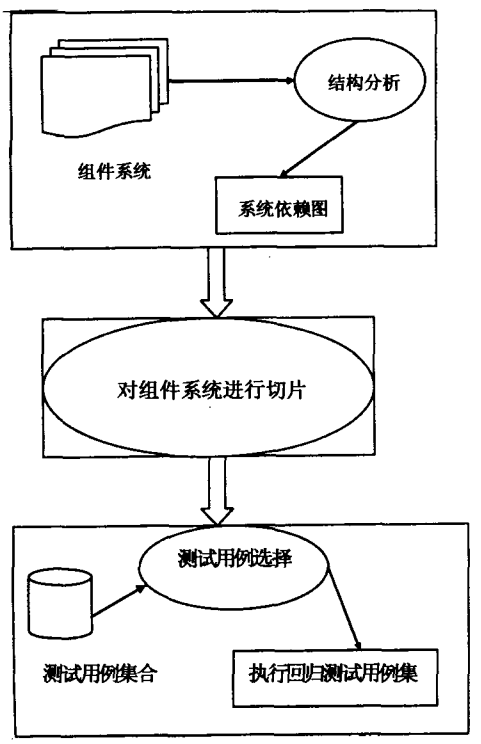

图 2　组件系统回归测试过程

为了验证提出测试策略的有效性，采用文献[4]中的自动售货机例子来进行实验分析。为了能更好的比较，我们分别对自动售货机中第 54 行和第 62 行人为植入错误，记为错误 Ⅰ 和错误 Ⅱ，下面表 1 反映了文献[3]中方法调用图(Method Call Graph，MGG)和文献[4]中控制流图(Control-flow Graph，CFG)及基于规约的选择方法与文中方法在测试效果相同的情况下，选择回归测试用例数量上不同(文献[4]给出了 2 种不同粒度的方法)。

表 1　发现错误所需测试用例数比较

错误	测试方法			
	MCG	CFG	规约法	本文方法
Ⅰ	6	6	75	6
Ⅱ	27	21	27	21

该自动售货机原本有 75 个测试用例，当系统被修

改后,需要进行回归测试,笔者所用的方法和文献[4]中所提到的控制流图方法效果相同,但是该方法需要知道组件 Dispenser 中的源代码,并且需要比较修改前后的控制流图(Control - flow Graph, CFG),时间和空间消耗比较大,这对于只有一个组件的自动售货机系统来说是可行的,但是当系统变复杂的时候,该方法就会出现较大的局限性。

5 结 论

提出了一种依赖模型来描述基于组件的系统,通过对基于组件系统的分层描述,将系统的复杂度分散和压缩到各个层次,清晰地表现了系统内部的依赖关系。当基于组件的系统发生改变时,根据该层次依赖模型,通过对模型进行切片,来确定修改所影响的部分,然后对所得切片进行遍历得到受影响的路径,以此来选择重用的测试用例,方便了基于组件系统的回归测试,提高了回归测试的效率。下一步的研究重点是如何与形式化方法结合,提高此方法的自动化程度。

参考文献:

[1] 齐治昌,谭庆平,宁洪. 软件工程[M]. 北京:高等教育出版社,2001.

[2] 刘凯枫. 回归测试选择技术研究[D]. 湖南:湖南大学,2004.

[3] MAO CHENGYING, LU YANSHEN. Regression testing for component-based software system by enhancing change information: Proceedings of the 12th Asian-Pacific Software Engineering Conference(APSEC'05) 2005[C]. Washington. D. C:IEEE Computer Society,2005:8-15.

[4] ORSO A,HARROLD M J,ROSENBLUM D, ROTHERMEL G. Using component metacontent to support the regression testing of Component - based software[J]. Proc. of IEEE International Conference on Software Maintenance (ICSM'01),IEEE Press,2001:716-725.

[5] WEISER M. Program slicing[J]. IEEE Transactions on Software Engineering, 1984,10(4):352-357.

[6] HORWITZ S,REPS T,BINKLY D. Interprocedural slicing using dependence graphs[J]. ACM Transactions on Programming Languages and Systems 1990:26-60.

[7] LARSEN L,HARROLD M J. Slicing object-oriented software: Proceedings of the 18th International Conference on software engineering Mar 1996[C]. Berlin:[s. n],1996:495-505.

[8] FERRANTE J, OTTENSTEIN K J, WARREN J D. The program dependence graph and its use in optimization[J]. ACM Transactions on Programming Languages and Systems,1987:9(3),319-349.

Application of Slicing in Component-based System Regression Testing

FU He-gang, LI Fu-xin, ZENG gang

(College of Computer Science, Chongqing University, Chongqing 400030, China)

Abstract: In component-based system regression testing, the dependence model of component-based system should be built in order to confirm the part which will be affected by modify. Since high reusability and high complexity, the existing dependence models may not be applicable to component-based systems, a hierarchical dependence model is proposed to describe component-based. Through the forward slicing of modified points in the model, the modified and affected parts can be gotten, then traverse these parts and get affected testing path, thereby regression testing examples can be chosen effectively, the efficiency of regression testing can be improred.

Key words: slicing; component-based system; regression testing

(编辑 吕建斌)