浙江理工大学学报(自然科学版),第 33 卷,第 4 期,2015 年 7 月
Journal of Zhejiang Sci-Tech University (Natural Sciences)
Vol. 33, No. 4, Jul. 2015

文章编号:1673-3851 (2015) 04-0565-05

基于依赖结构的测试用例优先级技术

张　娜,刘阳秋,包晓安,俞成海,许立成,戴燕云
(浙江理工大学信息学院,杭州 310018)

摘　要:优先级技术能够在不减少测试用例规模的情况下,提高测试用例集的性能。为了降低测试工作量,已有的很多文献通常假定测试用例是独立的,但在实际测试中,测试用例之间存在依赖关系,并且只有考虑了依赖关系的测试才能正常运行。针对这个问题,文章将依赖关系引入到优先级技术,提出了一种基于依赖结构的测试用例优先级技术,首先将依赖关系转化为量化指标依赖深度,然后以此指标为权重函数,运用加权的深度优先搜索算法进行测试用例排序。结果表明:与随机排序和基于功能覆盖的优先级技术相比,减少了测试代价,提高了缺陷检测率。

关键词:软件测试;软件可靠性;测试用例优先级;依赖结构;深度优先搜索
中图分类号:TP311.5　　**文献标志码**:A

0　引　言

随着软件复杂性和系统规模的增大,软件的质量保证已经日益引起开发者的重视。而软件测试是保证软件质量的关键元素。然而软件测试花费巨大,如何更有效地利用时间和测试资源来提高测试效率,成为目前软件开发中亟待解决的问题。

为了适应环境变化的需求,软件系统的版本不断更新,这就导致测试用例库中的测试用例不断增多,而测试用例优先级技术是进行测试用例集维护的重要手段。优先级技术按照某种性能指标对测试用例进行排序,可以尽早的达到较高的错误检测率,降低开发成本。目前有关测试用例优先级的很多研究都证明了优先级技术有效地提高了错误检测率。早期的优先级技术使用覆盖信息作为指标,如语句、分支[1-2]、数据流/控制流覆盖[3],应用贪婪算法、遗传算法进行排序。近几年的研究转向其他方法,如基于系统的需求信息[4],但是需求属性通常是估计的,包含很大的主观因素;还有的是基于模型状况,如包晓安等[5]提出了一种改进的、资源约束的受控马尔科夫链(CMC)模型,根据该模型设计的优化测试策略能利用测试过程收集到的数据,对测试策略进行在线调整,从而实现软件的自适应测试。然而,以上的这些有关优先级技术的研究都是基于独立的测试用例,没有考虑测试用例间的功能依赖关系。

依赖关系的研究遍布软件系统开发的各个方面,已有的很多研究利用场景、系统、函数之间的依赖关系,不同程度地提高了软件开发的效率。Johannes等[6]引进依赖关系图来表达场景依赖关系,通过遍历依赖状态图的所有路径来提取测试用例。为了更好地维护可靠性系统,Jungyoon 等[7]根据依赖关系将系统的特征分为本质特征和偶然特征,并用实验证明了通过有效地管理偶然特征,可以减少测试工作量,提高软件可信度。李国徽等[8]提出了一种通过数据实例信息得到元素间的部分函数依赖关系,并利用这种依赖关系来辅助模式匹配。为了更好地划分测试资源,周毓明等[9]提出了一种通过分析类间依赖关系的拓朴结构来定量地刻画类重要程度的

收稿日期:2014－11－07
基金项目:国家自然科学基金项目(61379036);浙江省重大科技专项(2014C01047);浙江省自然科学基金项目(Y13F020175,LY12F02041);浙江理工大学 521 人才培养计划项目;浙江省公益性技术应用研究计划(2012C32004,2014C33102)
作者简介:张　娜(1977－),女,浙江杭州人,副教授,主要从事自适应软件及软件测试等领域的研究。

方法，并根据此度量方法实现了一个 Eclipse 插件工具。本文将依赖结构引入到测试领域，利用测试用例间的依赖关系进行优先级排序。

在实际测试中，测试用例之间是存在依赖关系的，并且只有将这种依赖关系应用到测试排序中才能使测试正确、高效地运行。测试用例间的依赖结构与系统各部分间的交互作用有密切的关系，假设在软件测试周期中越早的测试交互作用多的部分，越有可能提高缺陷检测率[10]，那么，通过给包含更多依赖的测试分配较高的优先级，这样可以增加更早发现错误的可能性。根据以上的分析研究，本文提出了一种基于测试用例间的功能依赖关系的测试用例优先级技术，命名为依赖结构优先级(dependency structure prioritization, DSP)。该技术利用测试用例间存在的依赖关系，以待测测试用例为顶点，依赖关系为路径，构成依赖结构图；定义依赖关系的量化指标，将之作为权重函数，运用加权深度优先搜索算法进行优先级排序。该方法充分考虑到测试用例间的依赖，更接近实际测试，实验结果也证明了该技术的有效性。

1 依赖结构

场景是从用户的角度来描述系统的运行行为，反映系统的期望运行方式，是由一系列的相关活动组成的。场景可以看作是用户需求的内容，完全站在用户的视角来描述用户与系统的交互。而基于场景的测试方法是在场景的基础上进行的测试，通过执行测试场景或与需求以及系统可操作的流程相关的测试用例来验证系统的功能。

场景的测试顺序代表用户与系统之间的交互顺序，而这种交互顺序不能随意改变，交互之间的这种制约关系就是本文表达的依赖关系。如果交互 A 的执行需要交互 B 的执行结果数据，那么我们就说交互 A 依赖于交互 B。

测试用例间的依赖关系可以构成依赖关系图，为了简化起见，本文中的依赖关系图定义为一个有向无环图(DAG)，$G=(V,E)$，其中，节点 V 代表了一组测试用例，而节点间的连接弧 E 代表了测试之间的功能依赖关系。

图 1 为依赖结构的示意图。图 1 中所有的节点都表示测试用例，其中，$T1$ 和 $T2$ 没有父依赖节点，而 $T3$ 到 $T9$ 都依赖于其他节点。例如，$T3$ 依赖于 $T1$。节点之间还存在有直接的和间接的依赖关系：$T8$ 直接依赖于 $T4$，而间接依赖于 $T3$。

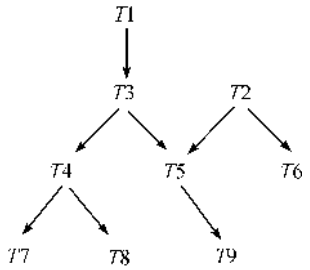

图 1 依赖结构

2 依赖结构优先级算法设计

由于测试用例间依赖关系的存在，为了更好地表达这种依赖关系，将测试过程转化为一张有向连通图：以待测测试用例为节点，依赖关系作为路径，权重函数的设定为依赖关系的量化值，并运用加权的优先深度搜索算法获取测试用例被执行的顺序。该方法优先测试复杂度高的部分，也即潜在缺陷相对较多的部分，从而优化了测试过程，提高了缺陷检测率。

2.1 依赖深度

软件系统各个部分之间的交互和耦合作用导致了软件系统结构的复杂，由于复杂度越高的部分潜在的软件故障存在概率越大，在软件测试的早期优先测试复杂度高的部分有利于尽早地发现更多的缺陷。本文中表达软件复杂性的度量指标就是测试用例间依赖关系，通过给予包含更多依赖关系的测试用例更高的优先级，可以尽快地达到更高的缺陷检测率，从而尽早完成测试任务，提高测试效率。

为了更好地计算测试用例间的依赖关系，定义了依赖结构图中依赖关系的量化图形覆盖指标，即依赖深度。依赖深度表明测试用例直接和间接依赖的测试用例数目，也即依赖关系图中从这个测试用例出发的所有路径中的路径最大值。要解决这个问题，首先要计算出依赖结构图中任意两点间的路径值，然后找出其中的最大值即可。而 Floyd 算法正是解决此问题的经典的动态规划算法，该算法的核心思想是：设 $G=(V,E)$ 为一个有向无环图，其中结点 $u,v,w\in V$，对于每一对结点 u 和 v，看看是否存在一个结点 w 使得从 u 到 w 再到 v 比已知的路径更长，如果是更新它。

算法 1 利用 Floyd 算法来计算每个测试用例的依赖深度。其中 G 为依赖结构图的布尔邻接矩阵，表示测试用例间的直接依赖关系；D 为所求矩阵，表示测试用例间的间接依赖关系。矩阵 D 第 i 行的最大值就是第 i 个测试用例的依赖深度。

算法 1　测试用例的依赖深度计算

$D:=G$//初始化

for $k:=1$ *to* n　//k 为中间节点

　　for $i:=1$ *to* n

　　　for $j:=1$ *to* n

　　　　$D[i,j]:=\max\{D[i,j],D[i,k]+D[k,j]\}$;

return D　//D 为包含最长路径长度的距离矩阵

2.2　测试用例优先级排序

优先测试复杂度高的部分软件的缺陷检测率会提高,为了达到这个目的,测试用例优先级排序要求满足以下两个条件:一,每次总是选择依赖深度值最大的测试用例执行;二,下一个测试用例的选择必须是与前测试用例有依赖关系,即是当前测试用例的邻接点。而满足以上两个条件的搜索算法为加权的深度优先搜索,其权值为依赖深度值。

深度优先搜索的核心思想是沿着顶点的邻点一直搜索下去,直到当前被搜索的顶点不再有未被访问的邻点为止,此时,从当前被搜索的顶点原路返回到在它之前被访问的顶点,并以此顶点作为当前被搜索顶点。继续这样的过程,直到不能执行为止。算法 2 即为采用加权深度优先搜索算法进行测试排序的优先级算法。

算法 2　基于依赖结构的优先级排序

WEIGHTED_DFS(v)　//访问由 v 到达的所有顶点

Visited(v)=1;

$C:=$get_children(G,v)　//邻接于 v 的每个顶点

for $C\neq 0$ do

　　$c\in\{t:C|(\forall t'\in C|w(t)>w(t'))\}$　//选取邻接点中权重最大的节点

　　$T:=$WEIGHTED_DFS(c,G,w,T);

　　$C:=C\backslash\{c\}$　//除去访问过的邻接点

end for

return T　//返回优先级排序后的测试用例集 T

从上面的优先级排序算法可以看出,首先执行图形覆盖值最高的测试用例,其次是依赖于这个测试用例,其中图形覆盖值最高的测试用例。对于依赖深度相同的情况,子节点的选取是任意的。当所有的依赖(包括直接和间接依赖)已经执行,然后执行图形覆盖值第二高的测试用例。以此类推,最终完成所有测试用例排序。

例如,用依赖深度覆盖指标对图 1 的依赖结构进行排序,可以看出测试用例 $T1$ 包含了比 $T2$ 更多的子节点,因此 $T1$ 具有最高的优先级,接下来执行 $T1$ 的直接依赖子节点 $T3$。由于测试用例 $T4$ 有两个子节点,而 $T4$ 的邻节点都只有一个子节点,那么 $T4$ 有第三高的优先级。在 $T1$ 的依赖子节点都执行完之后,$T2$ 将被执行,其次是 $T2$ 的依赖子节点。最终的测试排序顺序为 $T1-T3-T4-T7-T8-T5-T9-T2-T6$。

深度优先搜索算法增加了含有更多交互的部分更早测试的可能性。包含越多交互和依赖关系的部分,系统复杂度越高,越容易引起故障,从而优先执行含有更多依赖的测试用例可以提高缺陷检测率。

3　算法分析

本文针对以下两方面进行讨论:一是将依赖结构引入到优先级技术,利用测试用例间的依赖关系来提高测试有效性;二是将依赖结构量化为可计算的图形覆盖指标依赖深度,并由此提出了一种基于测试用例间依赖结构的优先级技术。为验证本文技术的有效性,设计实验解答以下问题。

问题 1:验证本文的优先级技术比功能覆盖优先级技术有更高的缺陷检测率,花费更少的测试代价。

问题 2:验证本文的优先级算法采用深度优先搜索比采用广度优先搜索要好,并解释其原因。

3.1　实验设计

为回答上述问题,选择了 3 个工业应用系统作为实验评估对象。实验所选类型广泛,既包含有小规模的 GSM 系统,仅有几百行代码,也包含有几万行代码的中规模系统 Bash。第 1 个为 GSM 单元系统,Miller 等对该系统有详细的描述。第 2 个为 Elite 组件系统,它提供自动语音服务,是一个交互式语音应答系统。第 3 个为 Bash (GNU bourne-again shell) 系统,它是一个为 GNU 计划编写的 Unix shell,下载于 SIR 库。其中,依赖关系的提取主要依据测试用例实现的功能和功能相关性,并且有些测试套件中已经包含测试人员预先定义的依赖结构。表 1 列出了这 3 个系统的相关信息。

表 1 测试系统相关信息

评估对象	类型	代码行数	功能数	缺陷数	测试用例数	依赖数目	独立测试数
GSM1	单元	385	14	15	51	65	0
GSM2	单元	975	60	14	51	65	0
Elite1	组件	2 487	54	72	64	64	3
Elite2	单元	2 487	54	64	64	65	3
Bash	系统	59 800	1051	11	1061	461	460

3.2 缺陷检测能力

为了计算经过优先级排序后的测试用例检测软件缺陷的测试效率，以 APFD[12]（average percentage of faults detected）作为度量标准，用以度量采用不同优先级排序技术的缺陷监测能力。

假设测试用例集 T 含有 m 个测试用例，被测软件 P 中有 k 个缺陷，则 APFD 的定义为

$$\mathrm{APFD}=1-\frac{\sum_{i=1}^{k}TF_i}{mk}+\frac{1}{2m} \tag{1}$$

式中，TF_i 表示检测到缺陷 i 的第一个测试用例在 T 中的位置。

为了验证本文技术的有效性，选取了随机排序（random ordering，random）、原序排序（untreated ordering，untreated）、总体功能覆盖优先级（total function coverage，total-fun）、附加功能覆盖优先级（additional function coverage，add-fun）、贪婪排序（greedy prioritization，greedy）、采用广度优先算法排序（breadth-first search，BFS）的依赖关系优先级与本文提出的优先级技术相比较。功能覆盖优先级技术依据测试覆盖的功能数目进行排序，最早由 Elbaum 等[13]提出，选取此技术的原因是由于本文中测试用例依赖关系是从功能测试用例套件中提取，依据的是测试用例间的功能依赖关系。本文基于贪婪排序算法，依据每个程序实体（例如语句、分支或函数）的覆盖能力来进行测试排序，定义依赖结构的图形覆盖值指导测试排序。采用广度优先算法作对比技术是为了验证本文假设的正确性，因为深度优先算法的假设条件是系统交互作用和依赖关系的复杂度越高，缺陷存在的概率越大，优先测试这部分会提高缺陷检测率。

图 2 描述了各种优先级技术在不同规模系统 GSM、Elite、Bash 上缺陷监测能力的变化情况。可以看出，对 Elite1 系统进行的随机排序跟 DSP 相差不多，原因是随机排序在偶尔情况下表现的很好，然而在大部分情况下，DSP 排序要比随机排序、原序排序缺陷检测能力更高。正如所料，BSP 排序结果比随机和原序排序表现要差，这证明了本文的假设，即优先测试复杂度高的部分，缺陷检测率越高。对于功能覆盖优先级技术，本文的 DSP 排序远远优于总体功能覆盖优先级，而相对于附加功能覆盖技术优势不是特别明显。同样的，贪婪技术除了 Bash 系统外，其余的缺陷检测能力表现都没有 DSP 排序好，这是由于相对于几万行代码植入 11 个缺陷而言，缺陷比率相对较少，而且依赖关系的比率也比较少，因此 DSP 技术的缺陷检测效率相对会低一点。

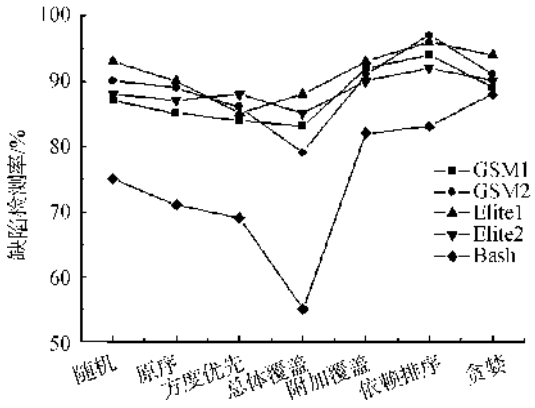

图 2 各种优先级技术缺陷检测能力比较

3.3 检测到所有缺陷消耗的单位测试代价

测试代价表明了软件测试过程中所消耗的资源情况，而如果能够计算出检测到所有缺陷所需的单位测试代价，对于优先级技术的评估具有相当重要的影响。

假设有测试用例集 T，其中包含 n 个测试用例，m 个缺陷，给定一个测试用例执行次序，则检测到所有缺陷时所需的单位测试代价 $TCFD$（test cost per unit of all faults detected）的计算公式为

$$TCFD=\frac{TF_m}{n} \tag{2}$$

式中，TF_m 表示检测到第 m 个缺陷的测试用例在该执行次序中所处次序。

对比优先级技术的选择与上述 3.2 节的一致。

图 3 描述了几种优先级技术检测到所有缺陷消耗的单位测试代价的折线图。由图 3 可以看到，对于相同的系统折线的弯折情况相似，并且所有的系统总体的趋势走向一致。对于随机排序和原序排

序,本文的 DSP 技术优于这两种优先级技术,但是 BFS 技术检测到所有缺陷花费的单位测试代价要比这两种技术要高,这个结果同样证明了本文的假设。DSP 技术的表现比总体功能覆盖优先级好很多,但相对于附加功能覆盖优势不是很明显,总体来说,DSP 技术还是比功能覆盖技术测试代价花费相对要少。对于贪婪排序,不同的系统表现很不一样,相对于本文的 DSP 技术,GSM 系统测试代价更高,与 Elite 系统的测试代价相差不多,比 Bash 系统的优势更明显,这表明了系统规模越大,DSP 技术对于贪婪技术测试代价的优势更显著。

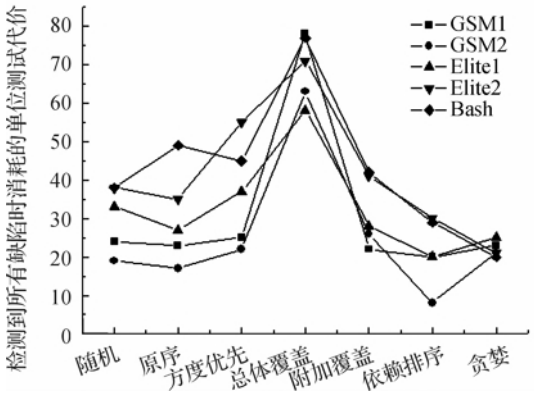

图 3　各种优先级技术单位测试代价比较

综合以上两个指标的情况,本文提出的优先级技术缺陷监测能力要优于功能覆盖优先级和贪婪排序技术,并且能以较低的测试代价检测到所有的缺陷。

4　结束语

测试用例间依赖关系的存在制约了测试执行的合理性和有效性,本文基于这一点提出了一种基于测试用例间依赖关系的优先级排序技术。该技术将依赖关系应用到测试排序问题中,首先将依赖关系转化为图形覆盖值依赖深度并把覆盖值作为权重函数,然后利用加权深度优先搜索算法进行优先级排序。实验结果表明,相比于随机排序和基于功能覆盖的优先级技术,本文的优先级技术能够在较短的时间内检测到更多的缺陷,花费更少的测试代价,表明本文提出的算法具有较好的效果。

针对以上的研究成果,将考虑依据代码覆盖信息来定义测试用例间的依赖关系及研究模型间的相关信息来增加测试用例优先级排序技术,以进一步提高测试的有效性。

参考文献:

[1] Jones J A, Harrold M J. Test-suite reduction and prioritization for modified condition/decision coverage [J]. IEEE Transactions on Software Engineering, 2003, 29(3): 195-209.

[2] 卢德中. 基于 MC/DC 的测试用例优先级算法研究[J]. 科学技术与工程, 2009, 9(14): 4040-4043.

[3] Rajiv G, Mary L S. Priority based data flow testing [C]//Proceedings of the IEEE International Conference on Software Maintenance(ICSM'95), 1995: 348-357.

[4] Krishnamoorthi R, Mary S A S A. Factor oriented requirement coverage based system test case prioritization of new and regression test cases[J]. Information and Software Technology, 2009, 51(4): 799-808.

[5] 包晓安, 姚　澜, 张　娜, 等. 基于受控 Markov 链的软件自适应测试策略[J]. 计算机研究与发展, 2012, 49(6): 1332-1338.

[6] Johannes R, Martin G. Using dependency charts to improve scenario-based testing[C]//Proceedings of the 17th International Conference on Testing Computer Software(TCS2000), 2000: 1-10.

[7] Jungyoon K, Doo H B. An approach to feature based modelling by dependency alignment for the maintenance of the trustworthy system [C]//Proc 28th Ann Int'l Computer Software and Applications Conf, 2004: 416-423.

[8] 李国徽, 杜小坤, 杜建强. 基于部分函数依赖的结构匹配方法[J]. 计算机学报, 2010, 33(2): 241-248.

[9] 周毓明, 徐宝文. 基于依赖结构分析的类重要性度量方法[J]. 东南大学学报: 自然科学版, 2008, 38(3): 380-384.

[10] Gregg R, Sebasrian E, Alexey M, et al. The impact of test suite granularity on the cost-effectiveness of regression testing [C]//Proceedings of the 24th International Conference on Software Engineering. ACM, 2002: 130-140.

[11] Miller T, Strooper P. A case study in model-based testing of specifications and implementations[J]. Software Testing, Verification and Reliability, 2012, 22(1): 33-63.

[12] Sebastian E, Alexey G M, Gregg R. Test case prioritization: a family of empirical studies[J]. IEEE Transactions on Software Engineering, 2002, 28(2): 159-182.

[13] Elbaum S, Malishevsky A G, Rothermel G. Prioritizing test cases for regression testing[C]//Proceedings of the International Symposium on Software Testing and Analysis. ACM Press, 2000: 102-112.

(下转第 578 页)

Speech Endpoint Detection Based on the Empirical Mode Decomposition and Mel Cepstrum Coefficient

CHEN Wei, *XIONG Wei-hua*, *SHI Wei-wei*
(School of Mechanical Engineering & Automation, Zhejiang Sci-Tech University, Hangzhou 310018, China)

Abstract: This paper puts forward speech endpoint detection method based on empirical mode decomposition and Mel cepstrum coefficient. A group of IMF components were gained through decomposing speech signal. Noise signals concentrating at low-layer IMF components were filtered. The surplus IMF components were reconstructed to speech signal. Mel frequency cepstrum coefficient of reconstructed signal was extracted to detect endpoint of speech signal. The results show that this method can well eliminate effects of noise on endpoint detection of speech signal, and correctly detect endpoint of speech signal under different signal-to-noise ratios. This method has certain robustness.

Key words: speech endpoint detection; empirical mode decomposition; intrinsic mode function; Mel frequency cepstrum coefficient; signal-to-noise ratio

(责任编辑: 康 锋)

(上接第569页)

Test Case Prioritization Techniques Based on Dependence Structure

ZHANG Na, *LIU Yang-qiu*, *BAO Xiao-an*, *YU Cheng-hai*, *XU Li-cheng*, *DAI Yan-yun*
(School of Information Science & Technology, Zhejiang Sci-Tech University, Hangzhou 310018, China)

Abstract: The prioritization technique can improve the performance of test case set without reducing the size of the test cases. In order to reduce testing workload, many literatures generally assume that the test case is independent, but in the actual test, there is dependence relationship among test cases. Only the test which considers dependence relationship can run properly. Aiming at this problem, this paper introduce the dependence relationship into prioritization technique and propose a kind of test case prioritization technique based on dependence structure. Firstly, the dependence relationship is transformed to quantitative index dependence depth. Then, this index serves as weighting function, and weighted depth-first search algorithm is applied to sort test cases. The results show that compared with the random sorting and prioritization techniques based on functional coverage, our strategy can reduce testing cost and improve defect detection rate.

Key words: software testing; software reliability; test case prioritization; dependency structure; depth-first search

(责任编辑: 陈和榜)