赵　勇,张智强,严　俊,等.软件兼容性测试的故障定位分析[J].计算机科学与探索,2013,7(5):405-411.

ISSN 1673-9418　CODEN JKYTA8
Journal of Frontiers of Computer Science and Technology
1673-9418/2013/07(05)-0405-07
doi: 10.3778/j.issn.1673-9418.1211020

E-mail: fcst@vip.163.com
http://www.ceaj.org
Tel: +86-10-51616056

软件兼容性测试的故障定位分析*

赵　勇[1,2+],张智强[1,2],严　俊[3],张　健[1,2]

1. 中国科学院 软件研究所 计算机科学国家重点实验室,北京 100190
2. 中国科学院大学 信息工程学院,北京 100190
3. 中国科学院 软件研究所 软件工程技术研究开发中心,北京 100190

Analysis of Fault Location in Software Compatibility Testing*

ZHAO Yong[1,2+], ZHANG Zhiqiang[1,2], YAN Jun[3], ZHANG Jian[1,2]

1. State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China
2. College of Information Engineering, University of Chinese Academy of Sciences, Beijing 100190, China
3. Technology Center of Software Engineering, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China
+ Corresponding author: E-mail: zhaoy@ios.ac.cn

ZHAO Yong, ZHANG Zhiqiang, YAN Jun, et al. Analysis of fault location in software compatibility testing. Journal of Frontiers of Computer Science and Technology, 2013, 7(5): 405-411.

Abstract: This paper proposes a new model for fault location about software compatibility testing based on combinatorial testing in order to lower cost, improve accuracy and simplify process during test. Based this model, this paper proposes a method to analyze fault location in compatibility testing. First of all, the model is built for softwares which need to do compatibility testing. After that, the tool Cascade is used to generate test cases for this model which are executed and get results. Finally, these test cases and execution results are input as the tool of Facil which is used for fault location about combinatorial testing. Facil will precisely find software set which causes the compatibility fault. Experiments show that the process of this method is simple and the result is effective and reliable. To some extent, it lowers cost, improves accuracy and simplifies process during test.

Key words: compatibility testing; combinatorial testing; fault location

摘　要:为了有效降低软件兼容性测试的测试成本,提高测试精度和简化测试过程,设计了一种基于组合测试的建模方案。基于该建模方案,提出了一种软件兼容性测试的故障定位分析方法。该方法首先针对需要进行

* The National Natural Science Foundation of China under Grant No. 60903049 (国家自然科学基金).
Received 2012-10, Accepted 2012-12.

兼容性测试的软件集建立组合测试模型;然后用组合测试用例生成工具Cascade对该模型生成测试用例集,并执行这些测试用例集得到结果;最后将这些测试用例及其执行结果作为组合测试错误定位工具Facil的输入,从而精确定位出发生兼容性错误的软件组合。实验表明,该方法步骤简捷,结果有效、可靠,在一定程度上降低了测试成本,提高了测试精度,简化了测试过程。

关键词:兼容性测试;组合测试;错误定位

文献标志码:A **中图分类号:**TP306.2

1 引言

软件兼容性测试是测试软件在特定的软硬件系统上和不同的应用软件之间能否协调工作的过程,是软件开发中不可缺少的一个步骤。随着软件开发复杂性的增加,以及不同平台下的应用软件和实现方法的多样化,软件运行过程中不可预见的情况越来越多,软件之间不兼容的情况也变得越来越常见。一个软件无论功能多强大或运行多高效,当和其他已有常用软件存在冲突时,其应用价值必将大打折扣。因此,在软件开发后期执行有效、全面的兼容性测试是非常有必要的。而基于软件产品及其系统平台的复杂多样化所造成的软件兼容性测试代价越来越大,寻找一个实用、高效的软件兼容性测试故障分析方法成为一个亟待解决的问题。

本文首先提出了一种针对软件兼容性测试的组合测试建模方法,对发生软件兼容性错误的软件建立组合测试模型;其次,将该模型作为自动化组合测试用例生成工具Cascade[1]的输入,生成相关的测试用例,执行每条测试用例并记录其执行状态;最后,将每条测试用例以及执行状态作为组合测试错误定位工具Facil[2]的输入,从而精确判断发生软件兼容性错误的软件组合。

2 软件兼容性测试和组合测试

2.1 软件兼容性测试

软件兼容性测试包括:操作系统/平台的兼容性、应用软件的兼容性、浏览器之间的兼容性、数据库之间的兼容性和操作系统语言、传输协议、代理服务器、防火墙、自身产品集成等兼容性测试。目前市场上有很多不同的操作系统类型,例如Windows、Unix、Linux、Mac OS等,而每一系列的操作系统又有很多的版本,例如Windows就有Windows 98/2000/XP/Vista等版本。除了针对特定用户的应用软件之外,其他应用软件的开发应考虑对以上各种系统的兼容性。当系统中已安装某些相似用途的软件或可能和当前软件有交互时,应测试这些软件之间的兼容性,检测它们同时运行是否会造成其他应用软件发生错误,或软件本身某些功能不能正常运行。例如,某些软件的运行可能需要浏览器和数据库的支持,如果不能兼容一些主流的浏览器和数据库软件,会造成软件核心功能的丢失。此时需要对该软件和浏览器、数据库的兼容性进行测试。此外,对操作系统语言的支持,以及对传输协议等的测试,也是不容忽视的问题。

目前,国际上从事软件兼容性测试技术研究和工具开发的机构还比较少。其中Oracle公司、IBM公司等只是针对自己公司的某个产品系列进行软件兼容性测试,而专门从事软件兼容性测试的公司迄今所知只有美国的Spike Source公司。

国内这方面的研究还比较少,并且缺乏有效的兼容性测试方法和相关测试标准。

考虑到若对上述所有兼容性因素及其相互之间的组合作用进行穷尽测试,其测试代价相当巨大,可以将待测的软件集建立组合测试模型,然后将组合测试相关工具应用在建立好的模型中。组合测试方法对软件兼容性因素间相互组合关系进行针对性覆盖,在保证测试效果的同时,大大降低了测试代价。

2.2 组合测试背景及相关原理

软件可以看做一个复杂的逻辑系统,其正常运行可能受到多因素的影响,这些因素可能是系统的配置、内部事件、外部输入等,除单个因素之外,上述各个因素之间的相互作用也可能对软件的正常运行产生影响。因此,在测试时不仅要考虑对软件产生

影响的可能因素,而且对存在于这些因素之间的相互作用,即因素之间的组合也要进行充分的测试。

为尽可能少地使用测试用例来有效地检测这些因素,以及它们之间的相互作用对系统产生的影响,从而在保证测试用例集错误检测能力的基础上,尽可能地降低测试成本,提出了基于组合的软件测试,称为组合测试[3-7]。

例如,对于一个具有 k 个参数的待测系统SUT(software under test),这些参数有 $v_1, v_2, \cdots, v_k$ 个取值,完全测试这个系统需要 $v_1 \times v_2 \times \cdots \times v_k$ 个测试用例。对于一般的被测系统而言,这个组合数是一个很庞大的数字。此时,可以使用组合测试方法生成测试用例,对于任意 t(t是一个很小的正整数,一般是2或者3)个参数,这 t 个参数的所有可能取值组合至少被一个测试用例覆盖,这样生成的测试用例数量较前者大幅度减少,并且错误检测能力比前者有所提高。这里的 t 被称为覆盖强度[8-10]。因为根据Kuhn和Reilly分析Mozilla浏览器的错误报告记录,发现超过70%的错误是由某两个参数的相互作用引起的[11-14],超过90%的错误是由3个以内的参数相互作用而触发的。

3　组合测试建模方法及相关工具

3.1　组合测试建模方法

下面通过实例来说明怎样根据一个软件或者软件集建立相关的组合测试模型。

在使用Foxit Pdf Reader(版本号V4.3.1.1208)的过程中,发现该软件的版本升级模块时常出现崩溃现象,从而导致整个软件不可用,同时版本升级模块需要网络连接,因此怀疑Foxit Pdf Reader的版本升级模块与系统中其他软件相关模块或系统的网络连接间存在兼容性错误。本文列出了此时系统中所有运行的软件,如表1所示。其中有三类软件和Windows7提供的系统配置,这三类软件分别是浏览器、编辑软件和编程软件。

Table 1　Software configurations involved in system

表1　系统中涉及的软件类型及配置

浏览器	编辑软件	编程软件	系统配置
FireFox	Foxit Pdf Reader	CodeBlocks	本机网络连接
IE	Word 2010	VS 2008	管理员权限
	NotePad		

对系统中涉及的所有软件建立组合测试模型,如表2所示。每个软件对应一个进程名,每个进程分别有两个状态:Open(进程打开状态)、Close(进程关闭状态)。系统配置有两个参数,分别为网络连接和管理员权限。网络连接为Open代表网络打开,为Close代表网络关闭;管理员权限为Open代表系统以管理员权限运行,为Close代表系统以非管理员权限运行。在此组合测试模型中,将每个进程作为一个模型参数,而Open和Close可以作为该参数的两个取值。目的是用组合测试相关工具生成该模型的测试

Table 2　Model of combinatorial testing

表2　组合测试模型

序号	进程名	软件列表(版本号)	状态1	状态2
P1	firefox.exe	FireFox(V12.06)	Open	Close
P2	notepad++.exe	NotePad(V5.9.8)	Open	Close
P3	codeblocks.exe	CodeBlocks(V10.05)	Open	Close
P4	WINWORD.exe	Microsoft Word 2010(V14.0.6112.5000)	Open	Close
P5	lexplorer.exe	IE 9(V9.0.8112.16412)	Open	Close
P6	devenv.exe	Visual Studio 2008(V9.0.21022.8)	Open	Close
P7	Foxit reader.exe	Foxit Pdf Reader(V4.3.1.1208)	Open	Close
P8	系统配置	Internet	Open	Close
P9	系统配置	Administrator	Open	Close

用例,从而检测该模型不同参数不同取值之间的交互是否引起该软件集的兼容性错误。

Internet连接状态和管理员权限是操作系统提供的一种系统服务,并且它的状态可以手工地改变,因此在表2中没有用具体的进程体现出。

本实验中需要一直打开Foxit Pdf Reader作为实验对象,因此实际实验中Foxit Pdf Reader的进程状态P7无需进行参数调节,可以将其移出进程列表。

在用组合测试建模方法对软件或者软件集建立模型之前,要调研该软件或者软件集是否适合用组合测试的观点进行测试。如果可以,还要看这个软件或者软件集是否能抽象出各个参数,各个参数是否都有取值,各个参数之间是否存在约束关系,以及各个参数之间的默认交互强度等多种因素。因此,针对不同的软件或者软件集,是否适合用该方法建立对应的模型要调研清楚,然后再选择不同的测试方法。在本文中,软件兼容性测试可以将软件集看做建模的对象,每个参数就是各个软件,而每个参数也有其相应的取值,并且兼容性测试的特点也适合用组合测试的建模方法。

3.2 组合测试介绍

3.2.1 组合测试用例生成工具Cascade

Cascade是作者所在项目组开发的组合测试用例生成工具,需要特定软件模型作为输入文件。输入文件包括四个段,分别是参数段、覆盖强度段、种子组合段和约束段。其中参数段和覆盖强度段是必须具有的,其他两个段可以根据具体的软件模型自由选择。对用户已经建立的组合测试模型和指定的覆盖强度,Cascade可以自动化逐条生成测试用例,并且能保证在测试用例条数尽可能少的情况下元组的覆盖率尽可能高。针对表2所示的组合测试模型,其输入文件有两个段,分别是参数段和覆盖强度段。

3.2.2 组合测试错误定位工具Facil

Facil[2]是作者所在项目组开发的组合测试错误定位工具。对于一个特定的软件模型,由Cascade生成测试用例后,在该软件上执行这些测试用例并记录执行结果,执行成功的测试用例标记为Pass,执行失败的测试用例标记为Fail。在使用测试用例对被测系统进行测试后,某些测试用例会通过,而有些测试用例会失败。由于组合测试假设错误是由参数之间的组合引起(把能够引起错误的参数组合称为错误模式),如果能定位到这些参数组合,那么就可以帮助开发人员更快地找到错误的内在原因。但仅仅从测试用例的运行结果,很难判断出究竟是哪些参数组合引起了错误。更极端的情况,在某些情况下,仅仅只有一条被报告上来的失败测试用例,显然无从得知错误模式究竟是哪些。

组合测试错误定位工具Facil就是针对这样的问题进行定位。该工具以一条失败的错误模式为输入,通过设计并运行一些附加的测试用例,来逐步缩小与错误相关的参数范围,直到定位出引起该条测试用例出错的错误模式为止。该工具实现了FIC(faulty interaction characterization)、FIC_BS(faulty interaction characterization based on binary search)、DDMin(delta debugging minimization)、OFOT(adaptive interaction fault location)四个算法[2],用户可以根据需要选择任意算法进行错误定位。

4 实验设计与分析

4.1 实验设计步骤

下面用表2中的实际例子来说明如何用组合测试建模方法所建立的模型以及工具集Cascade和Facil,来定位系统中存在的软件兼容性错误。本文的实验环境如表3所示。

Table 3 System configuration

表3 系统配置情况

操作系统	CPU类型	内存大小/GB
Windows 7	Intel Core i7-2600	4

步骤1 根据表2的组合测试模型建立Cascade的输入文件。该输入文件包含两个段,分别是参数段和覆盖强度段。参数段中的每个参数代表一个进程或者系统配置,参数的名字用 $P_i(i=1,2,\cdots,8)$ 表示,每个参数有两个取值,即Open和Close。覆盖强度段的取值为2,代表最后生成的测试用例集要覆盖8个参数中任意2个参数的所有取值组合。这

Table 4　Test cases generated by Cascade and execution results

表4　Cascade生成的测试用例集及测试情况

	P1	P2	P3	P4	P5	P6	P7	P8	P9	Result
Case 1	Close	Close	Close	Close	Close	Close	Close	Close	Close	Pass
Case 2	Close	Open	Open	Open	Open	Open	Open	Open	Open	Fail
Case 3	Open	Open	Open	Close	Open	Open	Close	Close	Close	Pass
Case 4	Open	Close	Close	Open	Close	Close	Open	Open	Open	Fail
Case 5	Close	Close	Close	Close	Open	Open	Open	Close	Open	Fail
Case 6	Close	Open	Open	Open	Close	Close	Close	Open	Close	Pass
Case 7	Close	Close	Open	Open	Open	Close	Open	Close	Close	Pass
Case 8	Close	Open	Close	Close	Close	Open	Close	Open	Open	Pass

里覆盖强度取值为2的原因分别是:组合测试模型的相关经验以及软件兼容性错误一般是两个软件不兼容造成的。

步骤2 表4为Cascade生成的8条测试用例,最后一列为手动执行每条测试用例的结果。在执行每条测试用例时,如果对应列为进程,Open代表将该进程开启,Close代表关闭;如果对应列是系统配置,按照3.1节描述的情况执行。最后的执行结果为第1、3、6、7、8条测试用例执行成功,第2、4、5条测试用例执行失败。其中Fail表示Foxit Pdf Reader崩溃,Pass表示该软件未出现崩溃现象。

步骤3 将崩溃的进程状态信息和表4的测试用例集作为组合测试错误定位工具Facil[2]的输入,选择执行该工具中所集成的某一故障定位算法。运行该工具后,最终定位出网络的连接状态和Foxit Pdf Reader更新模块之间不兼容引发的Foxit Pdf Reader更新模块的崩溃。

4.2　实验结果有效性分析

在本实验中,列出了3类共7款软件和相关系统配置,这仅仅是本次实验所处的主机环境。虽然针对不同的主机环境可以有不同的软件列表,但是实验的步骤和目的是一样的,都是用一定的方法来找出不兼容的软件列表。因此,从实验的目的而言,本实验的步骤是可行的,结果是有效的。

4.3　实验结果验证

为了验证结果的正确性,即在Windows7平台下,Foxit Pdf Reader的更新模块与网络连接状态的不兼容,将本机中除了Foxit Pdf Reader外的所有软件全部卸载,并且将本机的网络连接断开,Foxit Pdf Reader更新模块没有出现崩溃;将本机的网络连接开启,Foxit Pdf Reader更新模块出现崩溃现象。上述过程充分说明,Foxit Pdf Reader崩溃正是Foxit Pdf Reader更新模块和网络连接状态不兼容造成的。

5　相关工作说明及对比

在本文实验中,对要进行兼容性测试的软件集建立组合测试模型,用组合测试用例生成工具Cascade对该模型生成相关测试用例,以及用错误定位工具Facil定位出发生兼容性错误的软件集合。本文所涉及的是软件兼容性测试中的定位问题,即给定一个出现错误的软件配置集,通过本文方法可以找出不兼容的软件集合。但如果实际运行中出现了不可预见的错误,只要能够给出并重现该错误配置,那么仍然可以利用本文方法精确定位。另外对于不可预见的兼容性测试,不仅要从软件规格说明书和测试目标等要素出发制定具体的测试需求,还要考虑规格说明书以外的诸多要素,不能仅仅考虑软件规格说明书中的模块功能说明。

目前,国内南京大学的聂长海老师[15]项目组进行过相关软件兼容性测试故障的研究。

6　结束语

本文首先提出了一种针对软件兼容性测试的组合测试建模方法,该方法能有效地对软件集构造合

适的测试模型。将该模型作为组合测试用例生成工具Cascade的输入,可以生成数量尽可能少,覆盖率尽可能高的测试用例集。在被测软件集上运行测试用例集得到相应的测试结果,将测试用例集以及执行结果作为组合测试错误定位工具Facil的输入,从而精确定位引起兼容性错误的软件。该方法过程简单,易于实现,并且能体现软件兼容性测试本质要求,在一定程度上降低了测试成本,简化了测试过程。

References:

[1] Institute of Software, Chinese Academy of Sciences. General software about nonclassical combinatorial testing: China[P]. 2012.

[2] Zhang Zhiqiang, Zhang Jian. Characterizing failure-causing parameter interactions by adaptive testing[C]//Proceedings of the 2011 International Symposium on Software Testing and Analysis (ISSTA ' 11). New York, NY, USA: ACM, 2011: 331-341.

[3] Yan Jun, Zhang Jian. Combinatorial testing: principles and methods[J]. Journal of Software, 2009, 20(6): 1393-1405.

[4] Yan Jun, Zhang Jian. A backtracking search tool for constructing combinatorial test suites[J]. Journal of Systems and Software, 2008, 81(10): 1681-1693.

[5] Ma Feifei, Zhang Jian. Finding orthogonal arrays using satisfiability checkers and symmetry breaking constraints[C]//Proceedings of the 10th Pacific Rim International Conference on Artificial Intelligence (PRICAI '08). Berlin, Heidelberg: Springer-Verlag, 2008: 247-259.

[6] Kuhn D R, Reilly M J. An investigation of the applicability of design of experiments to software testing[C]//Proceedings of the 27th Annual NASA Goddard Software Engineering Workshop (SEW '02). Washington, DC, USA: IEEE Computer Society, 2002: 91-95.

[7] Cohen M B, Gibbons P B, Mugridge W B, et al. Constructing test suites for interaction testing[C]//Proceedings of the 25th International Conference on Software Engineering (ICSE '03). Washington, DC, USA: IEEE Computer Society, 2003: 38-48.

[8] Lei Yu, Tai K C. In-parameter-order: a test generation strategy for pairwise testing[C]//Proceedings of the 3rd IEEE International Symposium on High-Assurance Systems Engineering (HASE '98). Washington, DC, USA: IEEE Computer Society, 1998: 254-261.

[9] Dalal S R, Jain A, Karunanithi N, et al. Model-based testing in practice[C]//Proceedings of the 21st International Conference on Software Engineering (ICSE '99). New York, NY, USA: ACM, 1999: 285-294.

[10] Cohen D M, Dalal S R, Fredman M L, et al. The AETG system: an approach to testing based on combinatorial design[J]. IEEE Transactions on Software Engineering, 1997, 23(7): 437-443.

[11] Colbourn C, McClary D. Locating and detecting arrays for interaction faults[J]. Journal of Combinatorial Optimization, 2008, 15(1): 17-48.

[12] Kamsties E, Lott C. An empirical evaluation of three defect-detection techniques[C]//Proceedings of the 5th European Software Engineering Conference. London, UK: Springer-Verlag, 1995: 362-383.

[13] Wang Ziyuan, Xu Baowen, Chen Lin, et al. Adaptive interaction fault location based on combinatorial testing[C]//Proceedings of the 10th International Conference on Quality Software (QSIC '10)). Washington, DC, USA: IEEE Computer Society, 2010: 495-502.

[14] Xu Baowen, Nie Changhai, Shi Liang, et al. A software failure debugging method based on combinatorial design approach for testing[J]. Chinese Journal of Computers, 2006, 29(1): 132-138.

[15] Li Jie, Nie Changhai, Liang Yalan. Fault diagnosis in software compatibility testing[C]//Proceedings of the 7th China Test Conference, 2012: 295-300.

附中文参考文献:

[1] 中国科学院软件研究所. 通用的非经典组合测试用例生成软件: 中国[P]. 2012.

[3] 严俊, 张健. 组合测试原理与方法[J]. 软件学报, 2009, 20(6): 1393-1405.

[14] 徐宝文, 聂长海, 史亮, 等. 一种基于组合测试的软件故障调试方法[J]. 计算机学报, 2006, 29(1): 132-138.

[15] 李杰, 聂长海, 梁亚澜. 软件兼容性测试的故障定位[C]//第7届中国测试学术会议, 2012: 295-300.

ZHAO Yong was born in 1986. He is a master candidate at Institute of Software, Chinese Academy of Sciences. His research interest is software testing.
赵勇(1986—),男,河南洛阳人,中国科学院软件研究所计算机科学国家重点实验室硕士研究生,主要研究领域为软件测试。

ZHANG Zhiqiang was born in 1990. He is a Ph.D. candidate at Institute of Software, Chinese Academy of Sciences, and the student member of CCF. His research interests include combinatorial testing and fault location based on combinatorial testing.
张智强(1990—),男,河南新乡人,中国科学院软件研究所计算机科学国家重点实验室博士研究生,CCF学生会员,主要研究领域为组合测试,错误定位。

YAN Jun was born in 1980. He received his Ph.D. degree in computer software and theory from Institute of Software, Chinese Academy of Sciences in 2007. Now he is an associate researcher at Institute of Software, Chinese Academy of Sciences, and the member of CCF. His research interests include program analysis and software testing.
严俊(1980—),男,湖北黄石人,2007年于中国科学院软件研究所计算机软件与理论专业获得博士学位,现为中国科学院软件研究所软件工程研究开发中心副研究员,CCF会员,主要研究领域为程序分析,软件测试。发表论文10余篇,参与国家自然科学基金项目多项。

ZHANG Jian was born in 1969. He received his Ph.D. degree in computer software and theory from Institute of Software, Chinese Academy of Sciences in 1994. Now he is a professor and Ph.D. supervisor at Institute of Software, Chinese Academy of Sciences, and the senior member of CCF. His research interests include automated reasoning, constraint solving and software testing.
张健(1969—),男,安徽人,1994年于中国科学院软件研究所计算机软件与理论专业获得博士学位,现为中国科学院软件研究所计算机科学国家重点实验室研究员、博士生导师,CCF高级会员,主要研究领域为自动推理,约束求解,软件测试。发表论文40余篇,参与国家自然科学基金项目多项。