Journal of Southeast University (English Edition) Vol.20 No.1 Mar. 2004 ISSN 1003—7985

Generating test cases for software configuration testing

Nie Changhai[1,3] Xu Baowen[1,2,3,4]

([1] Department of Computer Science and Engineering, Southeast University, Nanjing 210096, China)
([2] School of Computer Science, National University of Defense Technology, Changsha 410073, China)
([3] Jiangsu Institute of Software Quality, Nanjing 210096, China)
([4] State Key Laboratory of Software Engineering, Wuhan University, Wuhan 430072, China)

Abstract: Software configuration testing is used to test a piece of software with all kinds of hardware to ensure that it can run properly on them. This paper generates test cases for configuration testing with several common methods, such as multiple single-factor experiments, uniform design, and orthogonal experiment design used in other fields. This paper analyzes their merits and improves the orthogonal experiment design method with pairwise testing, and decrease the testing risk caused by incomplete testing with a method of multiple-factors-covering. It also presents a simple factor cover method which can cover all the factors and pairwise combinations to the greatest degree. Finally some comparisons of these methods are made on the aspects of test suite scale, coverage, and usability, etc.

Key words: software testing; configuration testing; test case; combination cover

Configuration testing is an absolutely necessary step in software testing. It tests software operation with all kinds of hardware devices to check whether some bugs exist caused by the software when the software is running on these hardware devices. As hardware devices, which include the computer, may come from different producers, to ensure the software under test can run on all kinds of hardware combinations to the greatest degree, configuration testing must be made.

The workload for configuration testing is usually enormous. Generally the first step of configuration testing is to determine which types of hardware devices should be tested according to the software under testing. The next step is to determine how many different products there are for each type of hardware device and which of those products are popular now or have been popular and then make some separation of these products. Finally we must decrease all the hardware combinations to an appropriate size[1].

This paper proposes to generate test cases for configuration testing with some common methods used in industrial and agricultural production and scientific study, such as multiple single-factor experiments, uniform design, and orthogonal experiment design, etc. and analyzes the merits of these methods and then presents a method of single-factor cover which can not only test all the products of each type of hardware device, but also cover all the pairs to the greatest extent possible.

Received 2003-05-20.
Foundation items: The National Natural Science Foundation of China (No. 60373066); Opening Foundation of State Key Laboratory of Software Engineering in Wuhan University; and Opening Foundation of Jiangsu Key Laboratory of Computer Information Processing Technology in Soochow University.
Biographies: Nie Changhai (1971—), male, graduate; Xu Baowen (corresponding author), male, professor, bwxu@seu.edu.cn.

1 Generating test cases for configuration testing with common methods

Suppose that software under test (SUT) has m classes of hardware devices to be tested through some preprocessing, each class of hardware devices has a_i ($1 \leqslant i \leqslant m$) types of products. The main task of configuration testing is to check whether SUT can run successfully on all kinds of hardware configurations. When the number of $n = a_1 \times a_2 \times \cdots a_m$ is small, we can make exhaustive testing. But it is usually very large and exhaustive testing is not feasible. Hence, a software tester needs to efficiently generate an effective set of test cases as a means of verifying the correct work of SUT. Following we present some methods for test suite generation of configuration testing.

For convenience, we only consider five classes of hardware devices that comprise the configurations in Tab.1.

Tab.1 Hardware devices table for configuration testing

Graphic card	Sound card	Modem	Printer	Mainboard
A1	B1	C1	D1	E1
A2	B2	C2	D2	E2

1.1 Mu1tiple single factor experiment

Since each different combination of hardware devices creates a different configuration, and in Tab.1 each of the five classes has two different products, the table defines a total of $2^5 = 32$ different configurations. Suppose for argument's sake that 32 tests are too many, as each individual test is expensive. Then one alternative will select a typical value for each parameter and then vary one parameter in each test until all the parameter values are covered. Tab.2 shows the resulting test set. It has 6 tests instead of the 32 required for exhaustively testing all the possible parameter combinations. However, although it covers all the individual parameter values, it covers only 30 of the $C_5^2 \times 2^2 = 40$ possible pairwise interactions between the test parameters.

Tab.2 Test suite for multiple single factor experiment

A1	B1	C1	D1	E1
A2	B1	C1	D1	E1
A1	B2	C1	D1	E1
A1	B1	C2	D1	E1
A1	B1	C1	D2	E1
A1	**B1**	**C1**	**D1**	**E2**

1.2 Pairwise testing

Definition 1 Let $\boldsymbol{A}$ be a matrix $\boldsymbol{A} = (a_{ij})_{n\times m}$, $a_{ij} \in T_j (i = 1, 2, \cdots, n; j = 1, 2, \cdots, m)$, such that if any two columns c_i and c_j of $\boldsymbol{A}$ satisfy: all the pairwise combinations of symbols between set T_i and set T_j that appeared the same number of times in the ordered pairs formed by column i and column j, then $\boldsymbol{A}$ is called an orthogonal array; if all the pairwise combinations of symbols between set T_i and set T_j appeared at least once, then $\boldsymbol{A}$ is called a pairwise cover table. The method of experiment design based on an orthogonal array is called an orthogonal experiment design method; if the method of experiment design is based on a pairwise cover table, then it is called a pairwise combination cover method, or pairwise testing in software testing. m is the number of test cases. Every row of $\boldsymbol{A}$ is a piece of a test case[2-6].

By the same method, we can call the matrix $\boldsymbol{A}$ a 3-way cover table, if any 3 columns c_i, c_j and c_k of $\boldsymbol{A}$ satisfy: all the triple-wise combinations of symbols among set T_i, T_j and T_k appear at least one time in the ordered 3-tuples formed by column i, column j and column k. The n-way cover table is defined in the same way as above. The method of experimental design based on the n-way cover table is called the multiple factor combinations cover method.

For example, to SUT we can design a pairwise combination cover table (see Tab. 3) for pairwise testing according to the hardware devices table for configuration testing (see Tab.1).

Tab.3 Pairwise combination cover table

A1	B1	C1	D2	E1
A1	B1	C2	D1	E2
A1	B2	C1	D1	E1
A2	B1	C2	D1	E2
A2	B2	C1	D2	E2
A2	**B2**	**C2**	**D2**	**E1**

1.3 Orthogonal experiment design method

Orthogonal arrays are beautiful and useful. They are essential in statistics and they are used in computer science and cryptography. In statistics, they are primarily used in designing experiments, which simply means that they are immensely important in all areas of human investigation, for example, in medicine, agriculture and manufacturing.

In software testing we can apply the orthogonal experiment design method. When the classes of hardware devices are not many and products for each class are few, configuration testing with orthogonal experiment design method is very practical and reasonable. For the configuration testing of SUT, we can give a test suite by the orthogonal array $L_8(2^5)$ (see Tab.4).

Tab.4 $L_8(2^5)$

A1	B1	C1	D1	E1
A1	B1	C2	D2	E2
A1	B2	C1	D2	E1
A1	B2	C2	D1	E2
A2	B1	C1	D2	E2
A2	B1	C2	D1	E1
A2	B2	C1	D1	E2
A2	B2	C2	D2	E1

The test suite generated by Tab.4 can cover all the pairwise combinations, but its size is larger than the suite for pairwise testing. In software testing, many gains have been obtained through the use of the orthogonal experiment design method[7-11].

1.4 Uniform design method

The uniform design method is presented by Chinese mathematicians Fang and Wang in 1978[12]. This method can cover all the parameter values, synchronously all the points for the experiments

distribute uniformly in the whole experiment space. It has been widely used and has yielded a great harvest. Next we design the test suite for the software configuration testing of SUT using this approach.

For the test requirement of SUT described in Tab. 1, there doesn't exist a proper uniform design table. In this case, we can choose a related uniform design table, like $U_6^*(6^6)$[12], and obtain Tab. 5 for configuration testing with a good degree of uniformity.

Tab.5 Test suite for SUT by uniform design

A1(1)	B1(1)	C1(1)	D2(2)	E2(2)
A1(1)	B2(2)	C2(2)	D1(1)	E1(1)
A1(1)	B2(2)	C1(1)	D2(2)	E1(1)
A2(2)	B1(1)	C2(2)	D1(1)	E2(2)
A2(2)	B1(1)	C1(1)	D2(2)	E2(2)
A2(2)	B2(2)	C2(2)	D1(1)	E1(1)

1.5 Multiple factor combinations cover method

For some important software, configuration testing must be done sufficiently. Though it is scientific and has good quality to test the software by the use of the above methods, there still exists too much risk associated with incomplete testing. To decrease this risk to the greatest extent possible, we can design the test suite for configuration testing by the use of a multiple factor combinations cover method, such as the 3-way combination cover method, 4-way combination cover method, and so on.

For example, we can design a 3-way combination cover table and a 4-way combination cover table according to Tab.1 of SUT. It requires 8 test cases when we use a 3-way combination cover method, and 16 test cases when we use a 4-way combination cover method. The test suite generated by these methods are much more smaller than the suite needed by exhaustive testing (in this example 32 test cases are needed for exhaustive testing). Thus multiple factor combinations cover method can greatly decrease the risk of incomplete testing, and at the same time it can also decrease the cost of software testing.

2 Simple Factor Cover Method

In software configuration testing, there is a large number of hardware device classes and different types of products to be tested. We present a new method for configuration testing called "the simple factor cover method" which can cover all the factors and can cover pairwise combinations to the greatest degree.

The first step of the test cases generation method for the simple factor cover method is to rank the parameters in descending order of the number of values, then generate test cases by this order. For SUT we can suppose $a_1 \geqslant a_2 \geqslant \cdots \geqslant a_m$ without loss of generality. It first generates $a_1 m$-tuples. In each of the m-tuples the first element is one of the a_1 symbols from set T_1. Then place each of a_2 symbols in T_2 in the second position in each of the $a_1 m$-tuples, respectively. If $a_1 > a_2$, there are still $(a_1 - a_2)\,m$-tuples in the $a_1 m$-tuples in which the second position is still empty. Select a symbol from T_2 and fill it in the second position in one of the remaining $(a_1 - a_2)\,m$-tuples so that it can cover the missing pairs to the greatest degree. Repeating the process, until all the empty positions are full, we can get a simple factor cover table.

This method has two good properties: the size of the test suite it generates is $m = \max\limits_{1 \leqslant i \leqslant n}(a_i)$; the test suite can also cover all the values of the parameters, and it can cover the pairwise combination to the greatest degree. So this method is different from the method that only covers all the parameter values[7].

The simple factor cover method requires the smallest test suite of all the experiment design methods. It is very applicable to configuration testing where the number of parameter values and parameters is large, and the number of the values for each parameter is different. This method can be an improvement and complement to the other methods. Tab.6 is the simple factor cover table for the example in Tab.1.

Tab.6 Test suite for SUT by simple factor cover method

A1	B1	C1	D1	E1
A2	B2	C2	D2	E2

3 Comparison of Methods

For the convenience of making some comparison of methods, we can denote multiple single factor experiment as M1, pairwise combination cover method as M2, orthogonal experiment design method as M3, uniform design method as M4, multiple factor combination cover method as M5 and the simple factor cover method as M6. $T(\mathrm{M}i)$ represents the size of the test suite generated by the method Mi; $C(\mathrm{M}i)$ represents the combination coverage of the method Mi.

For the size of each test suite, there is generally such a result: $T(M6) \leqslant T(M4) \leqslant T(M1) \leqslant T(M3) \leqslant T(M2) \leqslant T(M5)$. Correspondingly the result with respect to combination coverage is as follows: $C(M6) \leqslant C(M4) \leqslant C(M1) \leqslant C(M3) \leqslant C(M2) \leqslant C(M5)$.

The uniform design method is based on the uniform design table. Although the construction method has been given, there is usually not a suitable table for application, and only through the use of the related uniform design table to design experiments, for the example in 1.4. As there is no order relation between the values of each parameter in the configuration testing, from this sense, the merits of the uniform design method cannot be embodied sufficiently.

Orthogonal experiment design method is an efficient test cases generation method, but it is dependent on the construction of an orthogonal array. There are still many remaining problems in this area, and in software testing we only need cover each pair of arbitrary two parameters once, and need not cover the pairs the same number of times. In configuration testing, the test suite generated by this method is usually still very large and thus impractical for testing.

The multiple single factor experiment method is a very easy and practical method, but it is not applicable to configuration testing in which there are many parameters and each of them has many parameter values. In these cases, the test suite generated by this method is not of good quality.

The simple factor cover method, pairwise combination cover method and multiple factor combination cover method are of a kind of test generation method, which are suitable for software testing. In fact we can see that simple factor cover method has evolved from uniform design method, and pairwise combination cover method has evolved from orthogonal experiment design method. These two methods are more practical for configuration testing. The multiple factor combination cover method provides a further way to decrease the risk caused by incomplete testing. People can choose a proper n-way cover table according to the importance and testing requirements of the software under testing.

4 Conclusion

Test case selection and design play a key role in software testing, and determine directly the quality, efficiency, and cost of software testing. This paper presents some methods for test suite generation of configuration testing that can be used in various cases according to various requirements. They can also be used to complement each other. These methods are also very useful in other areas of software testing[9-11], such as black box testing based on interface parameters, and web testing, etc. We have developed a series of tools to support these methods for test suite generation that can greatly improve the efficiency of software testing.

References

[1] **Patton Ron. Software testing** [M]. Indianapolis: Sams Publishing, 2001.

[2] Cohen D M, Dalal S R, Fredman M L, et al. The AETG system: an approach to testing based on combinatorial design [J]. *IEEE Trans on Software Engineering,* 1997, **23**(7): 437-444.

[3] Cohen D M, Dalal S R, Parelius J, et al. The combinatorial design approach to automatic test generation [J]. *IEEE Software*, **1996, 13**(5): 83-87.

[4] Lei Y, Tai K C. In_parameter_oder: a test generation strategy for pairwise testing [R]. Technical Report TR-2001-03. Raleigh, North Carolina: Department of Computer Science, North Carolina State University, 2001.

[5] Tai K C, Lei Y. A test generation strategy for pairwise testing [J].*IEEE Trans on Software Engineering*, 2002, **28**(1): 109-111.

[6] Kobayashi N, Tsuchiya T, Kikuno T. A new method for constructing pairwise covering designs for software testing [J]. *Information Processing Letters*, **2002, 81**(2): 85-91.

[7] Heller E. Using design of experiment structures to generate test cases [A]. In: *Proc* 12*th Int'l Conf Testing Computer Software, ACM* [C]. New York, 1995. 33-41.

[8] Mandl R. Orthogonal Latin squares: an application of experimental design to compiler testing [J]. *Communications of the ACM*, **1985, 28**(10): 1054-1058.

[9] Brownlie R, Prowse J, Phadke M. Robust testing of AT&T PMX/StarMail using OATS [J]. *AT&T Technical Journal*, **1992, 71**(3): 41-47.

[10] Cohen D M, Fredman M L. New techniques for designing qualitatively independent systems [J]. *J Combin Designs*, **1998, 6**(6): 411-416.

[11] Williams A W, Probert R L. A practical strategy for testing pairwise coverage of network interfaces [A]. In: *Proc 7th International Symp Software Reliability Engineer* [C]. 1997. 246-254.

[12] Fang K T, Wang Y. *Number-theoretic methods in statistics* [M]. London: Chapman & Hall, 1993.

软件配置测试的测试数据生成

聂长海[1,3] 徐宝文[1,2,3,4]

([1] 东南大学计算机科学与工程系, 南京 210096)
([2] 国防科技大学计算机学院, 长沙 410073)
([3] 江苏省软件质量研究所, 南京 210096)
([4] 武汉大学软件工程国家重点实验室, 南京 210096)

摘要: 本文提出利用多次单因素试验、正交实验设计和均匀设计等常用方法为软件的配置测试产生测试数据, 以确保待测软件可以在不同的配置上正确运行, 并分析比较了这些方法的优缺点.在此基础上提出使用两两组合覆盖方法来改善正交试验设计方法的不足、使用多因素组合覆盖方法来降低因不完全测试而带来的风险, 并提出一种单因素覆盖方法, 该方法可以产生最少的测试数据, 在覆盖各个系统因素的同时, 尽可能多地覆盖各个因素间的两两组合.最后, 对这些测试数据生成方法从测试数据集的规模、组合覆盖率、可用性等方面进行了比较.

关键词: 软件测试; 配置测试; 测试用例; 组合覆盖

中图分类号: TP311