

# Method for test case selection and execution of web application regression testing

Cao Xi<sup>1</sup> Xu Lei<sup>1,2</sup>

(<sup>1</sup>School of Computer Science and Engineering, Southeast University, Nanjing 211189, China)

(<sup>2</sup>Jiangsu Institute of Software Quality, Nanjing 210096, China)

**Abstract:** In order to improve the efficiency of regression testing in web application, the control flow graph and the greedy algorithm are adopted. This paper considers a web page as a basic unit and introduces a test case selection method for web application regression testing based on the control flow graph. This method is safe enough to the test case selection. On the base of features of request sequence in web application, the minimization technique and the priority of test cases are taken into consideration in the process of execution of test cases in regression testing for web application. The improved greedy algorithm is also raised resulting in optimization of execution of test cases. The experiments indicate that the number of test cases which need to be retested is reduced, and the efficiency of execution of test cases is also improved.

**Key words:** regression testing; web application; test case selection; control flow graph; optimizing execution

With the rapid development of information techniques, the application based on the web is gaining great popularity around the world. It has a fast pace and much variation in the development of web application. Thus, regression testing plays an especially important role in web application. However, with the expanding test case sets in regression testing and the old test cases, the cost of regression testing becomes increasingly large<sup>[1]</sup>, especially the time which consumes nearly half of the cost of the software maintenance<sup>[2]</sup>. Usually, there is a certain time limit in regression testing, and it is impractical for us to execute the overall test cases each time. Hence, we need to find some test cases more effectively to ensure that we can discover some errors. When we execute these test cases, a certain sequence needs to be followed so as to consider time limitations.

## 1 Background

### 1.1 Selection technique of test case

The researches about regression testing now mainly focus on how to reuse these test cases effectively. Rothermel et al. pointed out that the process of typical regression testing can be depicted in five steps<sup>[3]</sup>. The current methods based on code analysis are common. Gupta et al.<sup>[4]</sup> analyzed the mod-

ification of the definition-usage in test codes using a data flow technique, and an algorithm about choosing which test cases to execute was given. Agrawal et al.<sup>[5]</sup> reused test cases selectively with the use of slicing techniques. Rothermel<sup>[6]</sup> introduced a safe selection method of the test cases. First, the control flow graph of each program is constructed. Then, according to the modified node, some risk edge can be found using the algorithm in the control flow graph. Finally, test cases based on the risk edge are chosen.

At present, there are also some researches concerning regression testing of web applications. Ricca et al.<sup>[7]</sup> considered every page as a piece of program segment and used traditional program slicing methods to process. Xu et al.<sup>[8]</sup> also adopted the slicing technique to simplify web application content. First, the authors analyzed all kinds of possible changes and their related impact on web application and discussed them from two aspects of direct dependence and indirect dependence, respectively. Then based on the web application regression testing method of the slicing technique, the indirect dependence among data were emphatically studied.

### 1.2 Minimization and priority technique

We can solve test case selection problems through regression testing selection techniques. However, the chosen test case sets are not discussed about how to use them efficiently. The minimization technique and the priority technique are two major kinds of techniques concerning how to use these test cases efficiently. Wong et al.<sup>[9]</sup> pointed out that when test cases of regression testing were executed, the minimization technique could be chosen to pick up part of the test cases to be preferentially executed. They also introduced the idea that the selected test cases could be ranked in Ref. [9] according to certain rules, and, based on this sequence, the test case could be selected and executed in order to improve the rate and efficiency of the error detection in regression testing. Kim et al.<sup>[10]</sup> considered that the priority technique could also be applied in regression testing restricted by resources and time in favor of detecting regression errors involving limited time. The process of the priority technique can be depicted in four steps. And how to determine the selective probability of test cases is the prime problem in the priority technique.

### 1.3 The current drawbacks of research

At present, the research on web application regression testing concerns the selection of test cases. When considering the data dependence, the control dependence or the slicing technique, there are high costs in analyzing dependency relationships between pages. For the optimization of the execu-

Received 2008-04-15.

**Biographies:** Cao Xi (1983—), male, graduate; Xu Lei (corresponding author), female, doctor, associate professor, xlei@seu.edu.cn.

**Foundation items:** The National Natural Science Foundation of China (No. 60503020, 60503033, 60703086), Opening Foundation of Jiangsu Key Laboratory of Computer Information Processing Technology in Soochow University (No. KJS0714).

**Citation:** Cao Xi, Xu Lei. Method for test case selection and execution of web application regression testing[J]. Journal of Southeast University (English Edition), 2008, 24(3): 325 – 329.

tion case, the research also focuses on the traditional software and does not take the characteristics of web application into account. Due to the changeability in web applications, it is urgent for us to find a more effective regression testing program. Therefore, we focus on test case selection and the optimization of the test case execution in web application regression testing.

## 2 Test Case Selection Technique in Regression Testing

Test case selection in regression testing is usually based on the analysis of the code, while the relationships among pages in web applications are complex. The analysis cost is tremendous if it is based on the analysis of the code. We study the strategy of test case selection in regression testing based on the control flow graph (see Fig. 1).

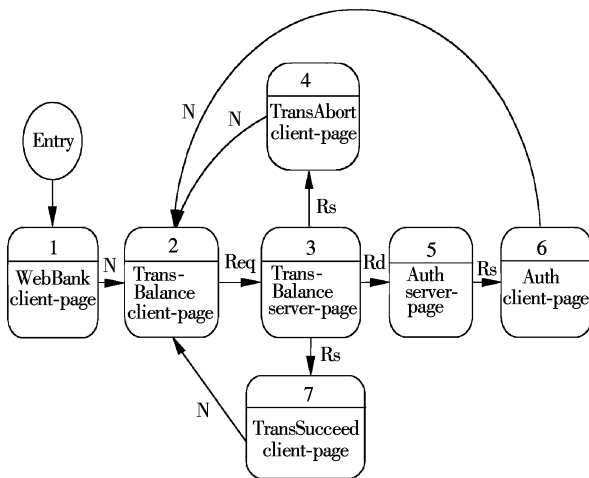


Fig. 1 Control flow graph of web application

### 2.1 Application of control flow graph in web testing

Kung et al. proposed an object-oriented model WTM to support the testing of web applications in Ref. [11]. This model introduced object, behavior, and structure to represent web application and defined the following relationships among pages: Request represented the HTTP request relationships between the client page and the server page; Response represented the HTTP response relationships between the client page and the server page; Navigation represented the targeted relationships between two client pages; Redirect represented redirection relationships between two server pages.

The selection technique of regression testing requires the analysis of the total cost of the test case selection and the execution of the selected test cases, and the total cost should be less than that of re-executing all test cases. Because of the complicated structure of the web application, the efficiency of the regression testing will be affected if analyzing the code. In addition, testers need to be very familiar with the code. Therefore, we take the page as the basic unit, and construct the control flow graph of the web application which results in reduced costs of the analysis. According to the four kinds of relationships mentioned above, the control flow graph can be constructed. In Fig. 1, the rectangular node represents a client page or a server page, the round node represents an entry node, and they can be connected with the rela-

tionships of Request, Response, Navigation and Redirect.

### 2.2 Test case selection method of regression testing based on the control flow graph

Rothermel et al. [6] introduced a kind of test case selection method based on the control flow graph, which was a safe test case selection technique. We adopt this method in the web application. First of all, we can construct the control flow graph  $G$  of the unmodified web application and the control flow graph  $G'$  of the modified web application, respectively. Then, we make a comparison among the corresponding nodes of  $G$  and  $G'$ , so as to find the first node of different paths. During the comparison, mark the compared node in  $G$ , comparing the node  $N$  in the original control flow graph with the node  $N'$  in the modified control flow graph resulting in  $N$  marked  $N'$ -visited. Later when this node is met again, we can avoid duplicated comparisons via marking. Finally, we choose the test cases corresponding to the nodes which need to be retested. The details of the process are shown in algorithm 1.

#### Algorithm 1 SelectTests( $P, P', T$ )

Input:  $P, P', T$  //  $P$  and  $P'$  are the unmodified and modified web applications,  $T$  is the test case set of  $P$ ;

Output:  $T'$  // subset of  $T$ , as the test case set of  $P'$  regression testing.

SelectTests( $P, P', T$ )

```

{
     $T' = \emptyset$ ;
    construct  $G$  and  $G'$  for  $P$  and  $P'$ , with entry nodes  $E$  and  $E'$ ;
    Compare( $E, E'$ );
    return  $T'$ ;
}
Compare( $N, N'$ ) // input  $N$  and  $N'$  as nodes of  $G$  and  $G'$ , respectively
{
    mark  $N, N'$ -visited
    for each successor  $C$  of node  $N$ 
    {
        Flag = false;
        for each successor  $C'$  of node  $N'$ 
        if  $C$  is not marked " $C'$ -visited"
            if ( $C = C'$ )
            {
                Compare( $C, C'$ );
                Flag = true;
            }
        If (Flag = false)
             $T' = T' \cup \text{TestOnEdge}((N, C))$ ;
    }
    // add the test case corresponding edge( $N, C$ ) to  $T'$ 
}

```

## 3 Optimization of the Execution of Test Case

There may be more test cases chosen based on the control flow graph due to the safety of the test case selection technique. In the process of regression testing, we cannot execute all the selected test cases because of the limitation of resources. In this case, we need to choose a group of test cases

that can meet certain test standards from all the selected test cases and execute these test cases preferentially. The other test cases can be executed when there are still resources remaining.

### 3.1 Request sequence length and execution time

Each test case is considered as an ordered request sequence that is characteristic of the web application. The complexity of each request and the combination of requests are different and they relate to the current status, so their execution time is difficult to estimate. According to Bernoulli's law of large numbers, we suppose that  $n_a$  is number of occurrences of event  $A$  in  $n$ -Bernoulli, and the probability of  $A$  appearing in each trial is  $p$ ; that is,  $P(A) = p$ . Then for any positive number  $\varepsilon$ , there is a formula  $\lim_{n \rightarrow \infty} p \left\{ \left| \frac{n_a}{n} - p \right| < \varepsilon \right\} = 1$ . We know that the probability of each type of request and the combination of requests is certain when the sequence length is long enough. We assume that the total number of requests is  $N$ , and the probability of each request (the same request can be considered as different requests in different states) is  $p_1, p_2, \dots, p_n$ , and the corresponding execution time is  $t_1, t_2, \dots, t_n$ . So the total execution time  $T = Np_1t_1 + Np_2t_2 + \dots + Np_nt_n$ , and the average execution time of each request  $t = T/N = (Np_1t_1 + Np_2t_2 + \dots + Np_nt_n)/N = p_1t_1 + p_2t_2 + \dots + p_nt_n$ .

When the length of the request sequence is long enough, then the longer the length of the request sequence is, and the longer the time of execution will be on the basis of the certain average execution time of each request. So a reduction of the total execution sequence length can be taken into consideration to improve the efficiency of web application regression testing. In the testing, we cannot estimate the execution time spent by each test case before the test cases are executed. Even in regression testing, because of the lack of information, we may not know the execution time of each test case. Moreover, in regression testing, the original test cases are always beyond the coverage requirement, so we need to add new test cases whose execution costs are unknown. In such cases, we can conveniently estimate the execution costs of the test cases according to the lengths of the request sequences using the characteristic of web application test cases.

## 3.2 Optimization of the test case execution

### 3.2.1 Minimization of the test case

The greedy algorithm can be used to make the minimization of the test case. We assume a requirement set  $R = \{r_1, r_2, \dots, r_n\}$  and a test case set  $R = \{r_1, r_2, \dots, r_n\}$ . First, choose test cases that can cover most of the requirements from the test case set. Then, delete these test cases from the test case set, and add the deleted test cases into the test case selection set  $S$ , and delete the requirements that meet the needs from the requirement set. Again, choose the test cases that can cover most of the requirements in the renewed requirement set from the renewed test case set. Repeat the above process until all the needs are covered or the time resources are exhausted.

### 3.2.2 Improved greedy algorithm

As mentioned above, in the regression testing of web applications, each chosen test case is a string of the ordered request sequences, so we can choose the shortest length of overall request sequences instead of choosing the fewest test cases, thus improving the efficiency of the regression testing. Testers are sensitive to the length of a request sequence, especially when the testing is executed manually. There is a considerable period of retention time during each operation. The reduction in the lengths of request sequences means a reduction in the number of operations. Therefore, retention time is reduced, resulting in improved test efficiency.

Now, we improve the greedy algorithm. Suppose a requirement set  $R = \{r_1, r_2, \dots, r_n\}$ , a test case set  $T = \{t_1, t_2, \dots, t_n\}$ , and its corresponding request sequence length set  $L = \{l_1, l_2, \dots, l_n\}$ . In the improved greedy algorithm, first, we need to calculate the ratio of the length of each test case and the number of requirements covered by each test case. Then, we obtain the set  $C$ ,  $C = \{c_1, c_2, \dots, c_n\}$ . The test case with the minimum value of  $C$  is picked up. Delete this test case from test case set and add it to the test case selection set  $S$ . Delete the requirements that meet the needs from the requirement set. Calculate each test case's value of  $C$  based on the renewed test case set and the renewed requirement set, and choose the test case with the minimum value of  $C$ . Repeat the above process until all the needs are covered or time resources are exhausted. The details are shown in algorithm 2.

#### Algorithm 2 Greedy algorithm

Input: Requirement set  $R = \{r_1, r_2, \dots, r_n\}$ ; test case set  $T = \{t_1, t_2, \dots, t_n\}$ ; test case request sequence length set  $L = \{l_1, l_2, \dots, l_n\}$ ;

Output: Test case selection set  $S$ .

Greedy algorithm ( $R, T, L$ )

```
{
  S = ∅;
  while (R ≠ ∅)
  {
    A = {a1, a2, ..., an}; //the set of the number of requirements covered by each test case in T
    C = {c1 = l1/a1, c2 = l2/a2, ..., cn = ln/an};
    ci = min(c1, c2, ..., cn);
    S = S ∪ {ti}
    R = R - {the requirement covered by ti}
    T = T - {ti}
  }
}
```

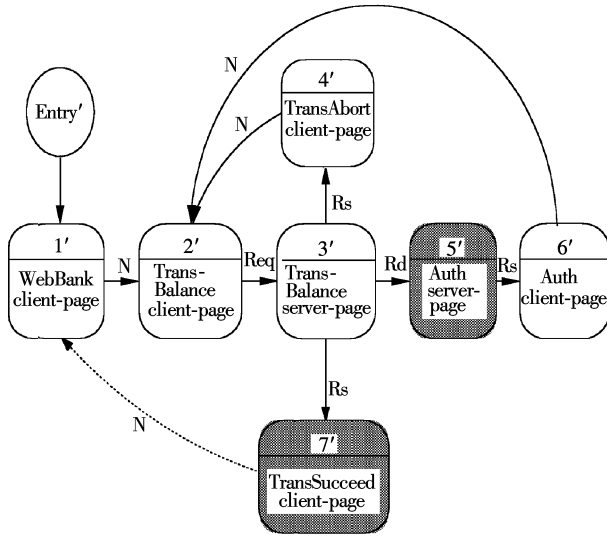
## 4 Experiments

In order to illustrate the approach mentioned above, we use examples to explain the test case selection technique and the optimization of the execution technique.

### 4.1 The selection of test case

Fig. 1 describes an online banking transaction system. The unmodified control flow graph of the web application is shown in Fig. 1. After we modify the web application, we obtain the corresponding control flow graph shown in Fig. 2. The grey part and the dotted line show the modifications.

The original test case and its execution route are shown in Tab. 1 and Tab. 2.



**Fig. 2** Modified control flow graph of web application

**Tab. 1** Execution path of test case

Test case	Execution path
$t_1$	(E, 1) (1, 2) (2, 3) (3, 4) (4, 2)
$t_2$	(E, 1) (1, 2) (2, 3) (3, 5) (5, 6) (6, 2)
$t_3$	(E, 1) (1, 2) (2, 3) (3, 7) (7, 2)

**Tab. 2** Test case set covering each edge

Path	Test case	Path	Test case
(E, 1)	$t_1 t_2 t_3$	(5, 6)	$t_2$
(1, 2)	$t_1 t_2 t_3$	(3, 7)	$t_3$
(2, 3)	$t_1 t_2 t_3$	(4, 2)	$t_1$
(3, 4)	$t_1$	(7, 2)	$t_3$
(3, 5)	$t_2$	(6, 2)	$t_2$

According to algorithm 1, we start from the entry point for comparison. The results of this comparison between node 1 and node 1', node 2 and node 2' are the same. And the same results occur between node 3 and node 3'. There are three branches occurring from node 3. Each branch is compared. First, we compare node 4 with the successor node of node 3'. Node 4 is the same as node 4', so we make a comparison between node 2 and node 2'. Node 2 is marked 2'-visited, so we stop the comparison of this branch and the second branch. We compare node 5 with the successor node of node 3'; due to the modified node 5, there is no node that is the same as node 5 in the successor nodes of 3'. Therefore, we choose the edge of (3, 5) in the original control flow graph  $C$  and its corresponding test case  $t_2$ . So we put the second branch at an end. We make the third comparison, the last comparison. First, we compare node 7 with the successor node of node 3'. Due to the modified node 7, there is no node that is the same as node 7 in the successor nodes of node 3'. So we choose the edge of (3, 7) and its corresponding test case  $t_3$ . Finally, the test case set  $\{t_2, t_3\}$  is chosen. It can be seen that from the original test case set, we just choose part of the test case to do the regression testing via

this method, resulting in the improved efficiency of the regression testing.

## 4.2 Optimization of test case execution

The above example illustrates the test case selection technique, and the following is the example of the optimization of the execution technique. In Tab. 3,  $r$  represents the requirements that need to be covered, and  $t$  represents the test case. If test case  $t_j$  has the requirement  $r_j$ , then there will be an X in the corresponding cell. First, we use the greedy algorithm to choose the test case without giving consideration to the request sequence length. Because  $t_1$  and  $t_3$  cover most of the requirements, we take them into consideration first and we select  $t_1$  randomly. Renew the requirement set, and there are  $r_2, r_4, r_7, r_8$  left unmet in the requirement set. Then we choose  $t_3$ , and we finally choose  $t_5$ . All requirements are met. We obtain the decrement set  $S = \{t_1, t_3, t_5\}$  finally.

**Tab. 3** Relationships between test case and requirement

Test case	$r_1$	$r_2$	$r_3$	$r_4$	$r_5$	$r_6$	$r_7$	$r_8$
$t_1$	X		X		X	X		
$t_2$			X					X
$t_3$			X	X			X	X
$t_4$					X		X	X
$t_5$	X	X						
$t_6$				X		X		
$t_7$							X	X

When the request sequence length is considered, we suppose the length from  $t_1$  to  $t_7$  is 10, 2, 3, 3, 1, 5, 2, respectively. We use the improved greedy algorithm to choose the test case. The average cost of each requirement covered is 2.5, 1, 0.75, 1, 0.5, 2.5, 1, respectively. We choose  $t_5$  due to its minimum value of  $C$ . Since  $t_5$  has met the needs of  $r_1$  and  $r_2$ , we delete  $r_1$  and  $r_2$ . Therefore, the requirement set converts to  $\{r_3, r_4, r_5, r_6, r_7, r_8\}$  and the test case set converts to  $\{t_1, t_2, t_3, t_4, t_6, t_7\}$ . After recalculating, the average cost of the remaining requirement covered by the remaining test cases are 3.3, 1, 0.75, 1, 2.5, 1, respectively. The final choice is the test case  $t_5, t_3, t_4, t_6$ , and the total length is 12, while the total length of test case  $t_1, t_3, t_5$  is 14 by using the original greedy algorithm. The time saved by using the improved greedy algorithm is about 14%.

After the use of the minimization technique, if there is still some time left, test cases can be executed according to the priority from high to low. We can preferentially select the test case with the minimum value of  $C$  to execute. In the above cases,  $t_5, t_3, t_4, t_6$  are executed according to the minimum selection of test cases, so the remaining test cases are  $t_1, t_2, t_7$ . Their corresponding request sequence length are 10, 2, 2 with values of  $C$  2.5, 1, 1, respectively. So the execution sequence is  $t_2, t_7, t_1$ . Thus, more requirements can be covered in shorter time, and result in more errors being found than before.

## 5 Conclusion

Regression testing has a fast pace and much variation in the development of web application. Thus, regression testing

plays an especially important role in web application. This paper first introduces a test case selection method of web application regression testing based on the control flow graph; and on the basis of the features of request sequences, the minimization technique is used in the execution of test cases in the regression testing of web applications and the priority of test cases is taken into consideration resulting in reduced regression testing costs. In the future, we will continue to study test case selection techniques and the optimization of execution techniques in web regression testing, so as to improve the efficiency of the selection algorithm and the effectiveness of the optimization of execution.

## References

- [1] Graves T L, Harrold M J, Kim J-M, et al. An empirical study of regression test selection techniques [J]. *ACM Transactions on Software Engineering and Methodology*, 2001, **10**(2): 184 – 208.
- [2] Rothermel G, Untch R H, Chu C, et al. Prioritizing test cases for regression testing [J]. *IEEE Trans Software Engineering*, 2001, **27**(10): 929 – 948.
- [3] Rothermel G, Harrold M J. Analyzing regression test selection techniques [J]. *IEEE Trans Software Engineering*, 1996, **22**(8): 529 – 551.
- [4] Gupta R, Harrold M J, Soffa M L. An approach to regression testing using slicing [C]//*Proc of International Conference on Software Maintenance (ICSM'92)*. Orlando, 1992: 299 – 308.
- [5] Agrawal H, Horgan J R, Krauser E W. Incremental regression testing [C]//*Proc of International Conference on Software Maintenance( ICSM'93)*. Montreal, 1993: 348 – 357.
- [6] Rothermel G, Harrold M J. A safe, efficient regression test selection technique [J]. *ACM Transactions on Software Engineering and Methodology*, 1997, **6**(2): 173 – 210.
- [7] Ricca F, Tonella P. Web applications slicing [C]//*Proc of International Conference on Software Maintenance (ICSM'01)*. Florence, 2001: 148 – 157.
- [8] Xu Lei, Xu Baowen, Chen Zhenqiang, et al. Regression testing for web applications based on slicing [C]//*Proc of Computer Software and Applications Conference*. Dallas, Texas, USA, 2003: 652 – 656.
- [9] Wong W E, Horgan J R, London S, et al. A study of effective regression testing in practice [C]//*Proc of the 8th IEEE International Symposium on Software Reliability Engineering*. Albuquerque, 1997: 264 – 274.
- [10] Kim Jung-Min, Porter Adam. A history-based test prioritization technique for regression testing in resource constrained environments [C]//*Proc of the 24th International Conference on Software Engineering*. New York, 2002: 119 – 129.
- [11] Kung D C, Liu C H, Hsia P. An object-oriented web test model for testing web applications [C]//*Proc of the First Asia-Pacific Conference on Quality Software*. Hong Kong, 2000: 111 – 120.

# 一种 web 应用回归测试的测试用例选择执行方法

曹 曦<sup>1</sup>    许 蕾<sup>1,2</sup>

(<sup>1</sup> 东南大学计算机科学与工程学院, 南京 211189)

(<sup>2</sup> 江苏省软件质量研究所, 南京 210096)

**摘要:** 为了提高 web 应用回归测试的效率, 采用了控制流图和贪心算法. 以页面为基本单位, 通过构造 web 应用的控制流图, 提出了一种基于控制流图的 web 应用回归测试的测试用例选择方法, 该方法是一种安全的测试用例选择方法. 在 web 应用回归测试的测试用例执行中, 根据 web 应用中请求序列的特点, 采用了最小化技术并考虑测试用例的优先级, 提出了一种改进的贪心算法对测试执行进行了优化. 实验结果表明, 该方法有效地减少了需要重测的用例数并且提高了测试执行的效率.

**关键词:** 回归测试; web 应用; 测试用例选择; 控制流图; 优化执行

**中图分类号:** TP311. 5