Journal of Southeast University (English Edition) Vol. 26, No. 1, pp. 36 - 42 Mar. 2010 ISSN 1003—7985

Coverage criteria and test requirement reduction for component-based web application

Gu Jingxian[1] Xu Lei[2] Xu Baowen[2]

([1]School of Computer Science and Engineering, Southeast University, Nanjing 210096, China)
([2]State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China)

Abstract: In order to analyze and test the component-based web application and decide when to stop the testing process, the concept of coverage criteria and test requirement reduction approach are proposed. First, four adequacy criteria are defined and subsumption relationships among them are proved. Then, a translation algorithm is presented to transfer the test model into a web application decision-to-decision graph (WADDGraph) which is used to reduce testing requirements. Finally, different sets of test requirements can be generated from WADDGraph by analyzing subsumption and equivalence relationships among edges based on different coverage criteria, and testers can select different test requirements according to different testing environments. The case study indicates that coverage criteria follow linear subsumption relationships in real web applications. Test requirements can be reduced more than 55% on average based on different coverage criteria and the size of test requirements increases with the increase in the complexity of the coverage criteria.

Key words: web application; coverage criteria; test requirement reduction

Component-based web application is a multi-tier architecture containing a variety of components, dynamic pages and static pages. The most popular architecture is three-tier, which contains three types of server components, i. e., the web server component, the application server component and the database server component. The web server component is composed of web pages such as ASP, JSP and HTML. It receives HTTP requests and returns HTTP responses. The application server component contains logical components which process business logics and complete user requests. The database server component acts as the data warehouse of component-based web applications, including large-scale databases, files, XML documents and so on.

In Ref. [1], we analyzed component-based web applications from the view of white-box testing. We constructed three kinds of dependency graphs: PCDG, CCDG and MC-DG, and proposed an extended MM-Path approach to generate testing paths by extending MM-Path testing in software integration testing.

This paper focuses on when to stop the testing process; i. e., how many extended MM-Paths are required? The reason why we consider such a question is as follows.

First, web application testing activity is a time- and labor-consuming process. Test requirements can be used as a well-accepted measure for selecting test cases, reducing test suites and deciding when to stop testing. In this paper, extended MM-Paths based on different coverage criteria can be regarded as test requirements. So if we can answer how many extended MM-Paths are required, the costs for testing the component-based web application will be reduced.

Secondly, the selection of test cases should guarantee that each test requirement is satisfied by at least one test case. Thus, test requirement reduction can help to reduce the number of test cases and avoid redundant test cases.

So, we first propose four kinds of coverage criteria, and subsumption relationships among them are analyzed. Then based on the two of the coverage criteria, we propose an algorithm to transfer our test model in previous work to a decision-to-decision graph, WADDGraph, and generate a spanning set of component-based web applications from WADDGraph to find the smallest test requirements.

1 Related Work

Current researches on structure testing for web application mainly focus on the construction of the static model and automated testing[2-5]. However, only a few of them have considered test criteria.

Ricca and Tonella [6] proposed a UML-based model for web application and described several static coverage criteria derived from traditional program-based criteria. However, the control flow within server pages is not considered in the UML model.

Sampath et al. [7] defined a set of novel coverage criteria called dynamic coverage criteria based on URL requests. However, these criteria cannot judge the adequacy of a single test suite with respect to how it satisfies a criterion and they can be used only when there are vast amounts of user sessions in web application.

Cai et al. [8] described a kind of test criteria based on page coverage sequences only navigated by web application. They also considered the test criteria based on page coverage sequences made by interactions in web application. To avoid ambiguity of natural language, these coverage criteria are depicted using Z formal language.

Test reduction includes test requirement reduction and test case reduction. Many researches have been conducted in

Received 2009-09-14.
Biographies: Gu Jingxian (1985—), female, graduate; Xu Lei (corresponding author), female, doctor, associate professor, xlei@nju.edu.cn.
Foundation items: The National Natural Science Foundation of China (No. 90818027, 60873050), the National High Technology Research and Development Program of China (863 Program) (No. 2009AA01Z147), Opening Foundation of State Key Laboratory Software Engineering in Wuhan University (No. SKLSE20080717), Opening Foundation of State Key Laboratory for Novel Software Technology in Nanjing University (No. ZZKT2008F12), the Key Laboratory Foundation of Shanghai Municipal Science and Technology Commission (No. 09DZ2272600).
Citation: Gu Jingxian, Xu Lei, Xu Baowen. Coverage criteria and test requirement reduction for component-based web application[J]. Journal of Southeast University (English Edition), 2010, 26(1): 36 - 42.

software testing[9-13]. To the best of our knowledge, however, test requirement reduction for web application has never been proposed, and we will do some discussion of that in this paper.

2 Coverage Criteria of Component-Based Web Applications

An extended MM-Path can be formalized as

$$\text{emmp} = \text{ep} \cdot (\text{mp} \cdot \text{ep})^{*}$$

where "·" is the connection between the execution path and the message path; "*" means that the sub-path expression (mp·ep) does not occur or occurs at least once. If all the extended MM-Paths (denoted as emmps) generated by an extended MM-Path algorithm are taken as test requirements and are covered by a test suite T, we say these test requirements are satisfied by T. However, if all the execution paths or all the message paths in these extended MM-Paths are covered by T, the emmps generated by the extended MM-Path algorithm will lead one execution path or one message path to be covered many times, which will cause a high cost of testing activity. Thus more sophisticated coverage criteria should be defined so as to reduce a high testing cost.

2.1 Four coverage criteria

Definition 1 (P, T, TC) is a triple where 1) $P = \langle \text{webpage1}, \text{webpage2} \rangle$, where webpage1 and webpage2 $\in$ a set of web pages of component-based web applications, and the relationship between them is webpage1 $\xrightarrow{\text{operation}}$ webpage2, operation = {page redirection, form submission}; 2) TC $\in$ {ALL_EMMP, ALL_S_EMMP, ALL_EP, ALL_MP} is a coverage criteria set; 3) T is a test suite satisfying TC of P.

We presently define four coverage criteria for component-based web application. MCDG_P in the following definitions represents the MCDG model of P, and EMMP(t) means extended MM-Paths set covered by the test suite T.

1) ALL_EMMP ALL_EMMP coverage criterion requires every extended MM-Path to be covered by T.

A triple (P, T, TC) satisfies TC = ALL_EMMP iff $\forall$ emmp $\in \text{MCDG}_P$, $\exists$ at least one test case $t \in T$ such that emmp $\in$ EMMP(t).

2) ALL_S_EMMP ALL_S_EMMP coverage criterion requires every simple extended MM-Path to be covered by test case T. "simple" means a restriction of ALL_EMMP by limiting loop iterations to zero and once in an extended MM-Path. Extended MM-Paths generated by the extended MM-Path algorithm satisfy this kind of coverage criterion.

A triple (P, T, TC) satisfies TC = ALL_S_EMMP iff $\forall$ emmp $\in \text{MCDG}_P$, $\exists$ at least one test case $t \in T$ such that emmp $\in$ EMMP(t) and no loop is iterated more than once.

3) ALL_EP ALL_EP coverage criterion requires every execution path in MCDG to be covered by T.

A triple (P, T, TC) satisfies TC = ALL_EP iff $\forall$ ep $\in \text{MCDG}_P$, $\exists$ at least one test case $t \in T$ such that ep $\in$ EMMP(t).

4) ALL_MP ALL_MP coverage criterion requires every message path in MCDG to be covered by the test case T.

A triple (P, T, TC) satisfies TC = ALL_MP iff $\forall$ mp $\in \text{MCDG}_P$, $\exists$ at least one test case $t \in T$ such that mp $\in$ EMMP(t).

Fig. 1 gives an MCDG model example which contains Update.jsp and UpdateResult.jsp. Two functions updateItem() and ItemExist() in a logical component CartBean.java will be called by the two pages. The corresponding source code is presented in Ref. [1].

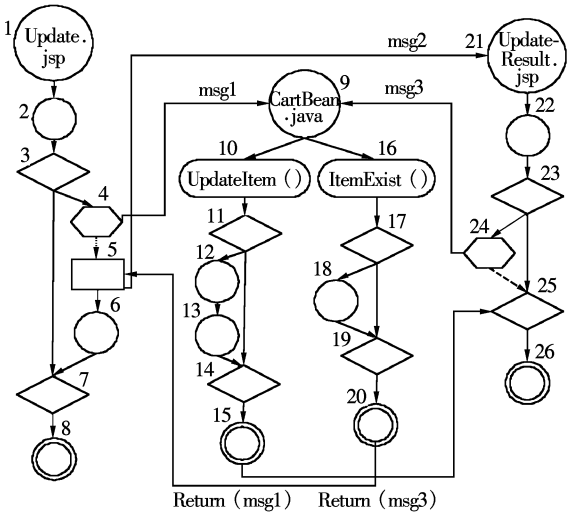

Fig. 1 MCDG model

Nodes in the MCDG model are labeled with id, and the type attribute of each node represents control information. If the type attribute is redirect, action or call, the node also has a condition attribute representing the page to be redirected or the component to be called. Detailed information is listed in Tab. 1. Edges are divided into three kinds. Real lines labeled with msg$i$ or return(msg$i$) (i = 1, 2, …) are message edges. Real lines with no label are control flow edges. Dashed lines are control flow edges in PCDG and CCDG, but do not belong to MCDG.

Tab. 1 Node information in MCDG model

Id	Type	Condition	Id	Type	Condition
1	classDef		14	branchfinal	
2	sequence		15	final	
3	branch		16	methodDef	
4	call	ItemExist()	17	branch	
5	action	Update-Result.jsp	18	sequence	
6	sequence		19	branchfinal	
7	branchfinal		20	final	
8	final		21	classDef	
9	classDef		22	sequence	
10	methodDef		23	branch	
11	branch		24	call	updateItem()
12	sequence		25	branchfinal	
13	sequence		26	final	

The extended MM-Path algorithm generates nine extended MM-Paths based on the MCDG model, which satisfies ALL_S_EMMP coverage criterion:

1) ⟨1, 2, 3, 7, 8⟩;

2) ⟨1, 2, 3, 4, msg1, 9, 16, 17, 18, 19, 20, return(msg1), 5, msg2, 21, 22, 23, 24, msg3, 9, 10, 11, 12, 13, 14, 15, return(msg3), 25, 26⟩;

3) ⟨1, 2, 3, 4, msg1, 9, 16, 17, 18, 19, 20, return

(msg1), 5, msg2, 21, 22, 23, 25, 26〉;

4)〈1, 2, 3, 4, msg1, 9, 16, 17, 18, 19, 20, return (msg1), 5, msg2, 21, 22, 23, 24, msg3, 9, 10, 11, 14, 15, return(msg3), 25, 26〉;

5)〈1, 2, 3, 4, msg1, 9, 16, 17, 19, 20, return (msg1), 5, msg2, 21, 22, 23, 24, msg3, 9, 10, 11, 12, 13, 14, 15, return(msg3), 25, 26〉;

6)〈1, 2, 3, 4, msg1, 9, 16, 17, 19, 20, return (msg1), 5, msg2, 21, 22, 23, 24, msg3, 9, 10, 11, 14, 15, return(msg3), 25, 26〉;

7)〈1, 2, 3, 4, msg1, 9, 16, 17, 19, 20, return (msg1), 5, msg2, 21, 22, 23, 25, 26〉;

8)〈1, 2, 3, 4, msg1, 9, 16, 17, 18, 19, 20, return (msg1), 5, 6, 7, 8〉;

9)〈1, 2, 3, 4, msg1, 9, 16, 17, 19, 20, return (msg1), 5, 6, 7, 8〉.

The same set of the extended MM-Paths can be given satisfying ALL_EMMP, for no loop existing in the source code. The ALL_EP criterion requires 11 execution paths to be covered by test cases: 〈1, 2, 3, 4〉, 〈1, 2, 3, 7, 8〉, 〈5〉, 〈5, 6, 7, 8〉, 〈9, 10, 11, 12, 13, 14, 15〉, 〈9, 10, 11, 14, 15〉, 〈9, 16, 17, 19, 20〉, 〈9, 16, 17, 18, 19, 20〉, 〈21, 22, 23, 24〉, 〈21, 22, 23, 25, 26〉, and 〈25, 26〉. The ALL_MP criterion requires five message paths to be covered by test cases: msg1, return(msg1), msg2, msg3, and return(msg3).

2.2 Subsumption relationships among coverage criteria

Four coverage criteria have subsumption relationships among them, which can be represented by a mathematical symbol ⊇. A test coverage criterion TC_A ⊇ another criterion TC_B iff a triple (P, T, TC_A) can also satisfy another triple (P, T, TC_B) for the same P and T. So the subsumption relationships among four coverage criteria are ALL_EMMP ⊇ ALL_S_EMMP ⊇ ALL_EP ⊇ ALL_MP.

Theorem 1 ALL_EMMP ⊇ ALL_S_EMMP

Proof According to the description of ALL_EMMP, a triple (P, T, TC) satisfies TC = ALL_EMMP for a page pair P iff there exists a test suite $T_1 \subseteq T$ covering a set of extended MM-Paths that loops in every emmp iterate zero time and once, and a test suite $T_2 \subseteq T$ covering a set of extended MM-Paths that loops in every emmp iterate more than once, where $T = T_1 \cup T_2$.

On the other hand, a triple(P, T', TC′) satisfies TC′ = ALL_S_EMMP for the same page pair P iff there exists a test suite T' covering a set of extended MM-Paths, where loop iterations in every emmp are limited to zero and one. So T' and T_1 are equivalent on covering all the simple extended MM-Paths and triple(P, T_1, TC′) satisfies TC = ALL_S_EMMP.

In sum, the test suite T that satisfies ALL_EMMP coverage criterion can also satisfy ALL_S_EMMP, i.e., ALL_EMMP ⊇ ALL_S_EMMP.

Theorem 2 ALL_S_EMMP ⊇ ALL_EP

Proof The ALL_S_EMMP coverage criterion is satisfied by the extended MM-Path algorithm, which traverses every execution path ep in webpage1 and connects all the possible message paths and execution paths in terms of the type attribute of the last node of ep.

According to the definitions about the execution path in Ref. [1], the type attribute of the last node of ep can point to the first node of other execution paths, or the type attribute of the last node of other execution paths can point to ep. Thus, $\forall ep \in MCDG_P$ is not isolated and can be included in extended MM-Paths by the extended MM-Path algorithm. If a test suite T can satisfy ALL_S_EMMP, it will also cover all the execution paths. So ALL_S_EMMP ⊇ ALL_EP is held.

Theorem 3 ALL_EP ⊇ ALL_MP

Proof Based on the formalization emmp = ep·(mp·ep)*, each mp must connect an ep, which means test suite T that covers all the execution paths(ALL_EP) also covers all the message paths(ALL_MP). So ALL_EP ⊇ ALL_MP.

Each criterion defined above has advantages and disadvantages. The test capability of the ALL_EMMP criterion is the strongest. But it is useless in practice because it is impossible to find such a test suite for the infinite loops. The ALL_S_EMMP criterion is practical, but it may produce infeasible paths which confuse testers. Additionally, many execution paths and message paths will be covered repeatedly to increase testing costs. The ALL_EP criterion and the ALL_MP criterion are easy to be satisfied. But their test capabilities are weaker.

3 Test Requirement Reduction for Component-Based Web Application

Now we return to the question: how many extended MM-Paths are required? Obviously, a set of test paths satisfying ALL_S_EMMP is unsuitable for execution paths and message paths covered too many times and infeasible paths. How about reducing the number of extended MM-Paths but satisfying the ALL_EP criterion or the ALL_MP criterion? That sounds great. Then how to generate extended MM-Paths satisfying the ALL_EP criterion or the ALL_MP criterion?

In an MCDG model, we can find subsumption and equivalence relationships among eps and mps. For example, according to the form of ep·(mp·ep)*, a test case passing by a mp must pass by an ep and thus the mp and the ep are equivalent. So if all the subsumption and equivalence relationships are found based on one criterion, we can reduce test requirements into a smaller size.

3.1 Transferring algorithm

The MCDG model contains subsumption and equivalence relationships among eps and mps, but it cannot be directly used because: 1) An MCDG model may have multi-exits, while a DDGraph used for generating a spanning set to reduce test requirements in traditional software testing requires a unique entry and a unique exit; 2) A DDGraph is a digraph that a node is associated with a decision and an edge is associated with a program segment. But nodes and edges in an MCDG are all associated with decisions and program segments. Algorithm 1 gives an algorithm to transfer an MCDG to a new digraph called WADDGraph.

Definition 2 (WADDGraph) A WADDGraph is a graph $G_{\text{waddgraph}} = (V_{\text{waddgraph}}, E_{\text{waddgraph}})$ with two virtual edges e_0 and e_k representing the unique entry and the unique exit. $\forall e = (\text{TAIL}(e), \text{HEAD}(e)) \in E_{\text{waddraph}}$, reached by e_0 and reaching e_k, represents an execution path or a message path in the MCDG model; $\forall v \in V_{\text{waddraph}}$, except for $\text{HEAD}(e_0)$ and $\text{TAIL}(e_k)$, represents a branch or a junction of an execution path; $(\text{indegree}(v) + \text{outdegree}(v)) > 2$, while $\text{indegree}(\text{TAIL}(e_0)) = 0$ and $\text{outdegree}(\text{TAIL}(e_0)) = 1$, or $\text{indegree}(\text{HEAD}(e_k)) = 1$ and $\text{outdegree}(\text{HEAD}(e_k)) = 0$.

Algorithm 1 TransferMCDG(EP, EP′, $G_{\text{waddgraph}}$)
ConstructWADDGraph(EP, EP′, $G_{\text{waddgraph}}$) {
TAIL(e_0) = Ø; HEAD(e_0) = s_0; TAIL(e_k) = s_k; HEAD(e_k) = Ø;
while(EP ! = Ø) {
EP = EP − ep; $E_{\text{waddraph}} = E_{\text{waddraph}} \cup$ ep;
ConstructSubsequence(ep, EP, $G_{\text{waddgraph}}$);
}}
ConstructSubsequence(ep, EP, $G_{\text{waddgraph}}$) {
Accumulate other execution paths(oep) in EP′ in which the last node of oep is identical with the predecessor of the first node of ep.
if(the accumulated number is zero) TAIL(ep) = s_0;
else if(the accumulated number is more than one) {
Create a node s_i($i = 1, 2, \ldots$); TAIL(ep) = s_i;
$V_{\text{waddraph}} = V_{\text{waddraph}} \cup s_i$;
foreach(oep in E_{waddraph}) HEAD(oep) = s_i;
}
Accumulate other execution paths(oep′) in EP′ in which the first node of oep′ is identical with the subsequence of the last node of ep.
if(the accumulated number is zero) HEAD(ep) = s_k;
else if(the accumulated number is one) {
EP = EP − oep′; TAIL(oep′) = ep; HEAD(ep) = oep′;
$E_{\text{waddraph}} = E_{\text{waddraph}} \cup$ oep′;
ConstructSubsequence(oep′, EP, $G_{\text{WADDGraph}}$);
}else{
Create a node s_j($j = 1, 2, \ldots$); TAIL(ep) = s_j;
$V_{\text{waddraph}} = V_{\text{waddraph}} \cup s_j$
foreach(oep′) {
EP = EP − oep′; TAIL(oep′) = s_j; $E_{\text{waddraph}} = E_{\text{waddraph}} \cup$ oep′;
ConstructSubsequence(oep′, EP, $G_{\text{WADDGraph}}$);
}}}

Algorithm 1 accepts EP, EP′ and $G_{\text{waddgraph}}$ as input. EP is a set of execution paths in MCDG and EP′ is initialized by EP. $G_{\text{waddgraph}}$ is a digraph that stores nodes and edges transferred from the MCDG model. Initially, it has two virtual edges e_0 and e_k, and two nodes s_0 and s_k.

Algorithm 1 first traverses all the execution paths in EP. It selects one execution path ep, adds it to $E_{\text{waddgraph}}$ and removes it from EP. Next, ConstructSubsequence will be called to construct the predecessor and the successor of ep. Algorithm 1 accumulates oep numbers in EP′ pointing to ep. If the number is zero, ep is the first execution path in the MCDG model. Otherwise, ep is the successor of oep. So the algorithm creates a node s_i to connect every oep with ep and ConstructSubsequence will be called recursively. Then algorithm 1 accumulates oep′ numbers in EP′ pointed from ep. If the number is zero, ep is the last execution path in the MCDG model. If the number is one, oep′ is the unique successor of ep and will be added to $G_{\text{waddgraph}}$. If the number is more than one, oep′ is the successor of ep. So the algorithm creates a node s_j to connect ep with every oep′ and ConstructSubsequence will be called recursively.

After processing all the eps in EP, algorithm 1 outputs $G_{\text{waddgraph}}$. The left part of Fig. 2 shows the WADDGraph transferred from the MCDG model of Fig. 1. It is composed of $V_{\text{waddgraph}}$ and $E_{\text{waddgraph}}$. $V_{\text{waddgraph}}$ is a set of nodes from s_0 to s_7, while $E_{\text{waddgraph}}$ is a set of 11 execution paths. The corresponding relationship between edges of $E_{\text{waddgraph}}$ and eps of the MCDG is shown in Tab. 2.

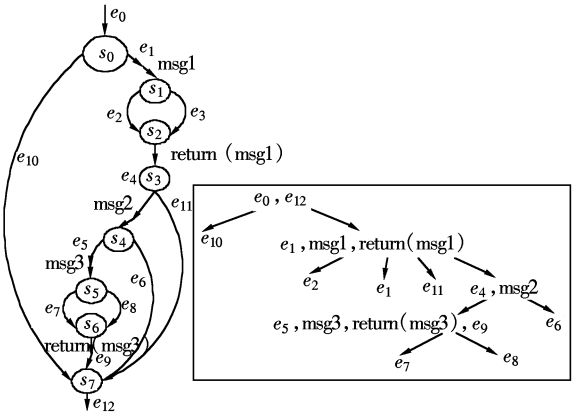

Fig. 2 WADDGraph and its edges relations

Tab. 2 Relationship between $E_{\text{waddgraph}}$ and E_{MCDG}

$e \in E_{\text{waddgraph}}$	$e' \in E_{\text{MCDG}}$
e_0	Unique virtual edge of $G_{\text{waddgraph}}$
e_1	⟨1, 2, 3, 4⟩
e_2	⟨9, 16, 17, 18, 19, 20⟩
e_3	⟨9, 16, 17, 19, 20⟩
e_4	⟨5⟩
e_5	⟨21, 22, 23, 24⟩
e_6	⟨21, 22, 23, 25, 26⟩
e_7	⟨9, 10, 11, 12, 13, 14, 15⟩
e_8	⟨9, 10, 11, 14, 15⟩
e_9	⟨25, 26⟩
e_{10}	⟨1, 2, 3, 7, 8⟩
e_{11}	⟨5, 6, 7, 8⟩
e_{12}	Unique virtual edge of $G_{\text{waddgraph}}$

Suppose that the number of EP is e. Function ConstructSubsequence is called recursively to process each ep in EP. So the execution number of this function is e. When a recursive call executes, algorithm 1 will do two accumulation operations to calculate the number of other execution paths associated with the current ep. So function ConstructSubsequence requires $O(e)$ time. To sum up, the time complexity of algorithm 1 is $O(e^2)$.

3.2 Spanning set generation and test requirement reduction

Next, we can analyze subsumption and equivalence relationships among edges in $G_{\text{waddgraph}}$ and obtain a spanning set according to the approach in Refs. [9 − 10]. For example, e_1 is an edge of $G_{\text{waddgraph}}$. No matter which edge, e_2 or e_3,

that the control flow chooses to execute, it always passes by msg1 and return(msg1), and exits from e_{12}. So e_1, msg1 and return (msg1) are equivalent. When the control flow passes node s_5, e_7 or e_8 can be chosen to execute. So the relationship between msg3 and e_7 or between msg3 and e_8 is subsumption. Detail relationships are shown in the right part in Fig. 2.

e_{10}, e_2, e_3, e_{11}, e_6, e_7, and e_8 in the right part in Fig. 2 compose a spanning set U. If all the edges in U are covered by a test suite, other edges in the WADDGraph will also be covered, which satisfies the ALL _ EP criterion. Therefore, nine extended MM-Paths listed in section 2. 1 can be reduced into five based on the ALL _ EP criterion. That is: the first path which covers e_{10} in U; the second path which covers e_2, e_7 in U; the sixth path which covers e_3, e_8 in U; the seventh path which covers e_3, e_6 in U and the ninth path which covers e_3, e_{11} in U.

For the ALL _ MP coverage criterion, only one extended MM-Path which covers e_7 or e_8 is required. Because the path covering e_7 or e_8 will also cover all the message paths by the relationships shown in Fig. 2. The second path, the fourth path and the sixth path in the previous nine paths cover all the message paths, so any one can be selected to satisfy the ALL _ MP criterion.

4 Case Study

In this section, test requirements and their reductions will be examined based on different coverage criteria.

CWAT is a tool which we have developed to support analysis and testing of JSP-based web applications. It statically analyzes web application programs. The main functions of this tool are constructing test models proposed in Ref. [1], generating extended MM-Paths, generating different sets of test requirements based on different coverage criteria, and reducing test requirements. Next we will use this tool to examine whether the proposed expectations in the following are right.

We expect the trend in ALL _ S _ EMMP, ALL _ EP and ALL _ MP to follow linear subsumption relationships in real web applications, i. e., ALL _ S _ EMMP ⊇ ALL _ EP ⊇ ALL _ MP. We also expect the approaches proposed in this paper can help to reduce test requirements in real web applications, and the more complex coverage criterion will lead to smaller reduction ratios and stronger test capabilities. Here "complex" means stronger subsumption capabilities proved in section 2.

4. 1 Subject web application

An e-commerce website called online CD shop(http: // www. cdshop09. com/), which has typical three-tier architecture, is used as a case study. This website has 26 pages, 14 components and 13 servlets, which allows common users to visit and allows administrators to manage this website. The characteristics of 25 dynamic pages are shown in Tab. 3.

4. 2 Subsumption relationships among different criteria

Subsumption relationships among three cover criteria for 25 dynamic pages are shown in Tab. 3 generated by CWAT. Columns 2 and 3 represent the number of functions to be called and the number of pages to be redirected by each page. Column 4 refers to emmp numbers and ep numbers of all the emmps based on the ALL _ S _ EMMP criterion. Column 5 refers to ep numbers and mp numbers of all the eps based on the ALL _ EP criterion. Column 6 refers to mp numbers based on ALL _ MP.

Tab. 3 Subject program characteristics and subsumption relationships among three criteria

Pages	Function	Redirection	emmp/mp	ep/mp	mp
userdetail _ jsp	13	1	3/33	33/30	30
ShowUserPoint _ jsp	3	1	3/15	15/12	12
usermodify _ jsp	29	2	8/135	135/129	129
showSaleStatistics _ jsp	1	1	4/6	6/2	2
shopcar _ jsp	20	2	130/645	645/545	545
orderdetail _ jsp	40	2	26/115	115/107	107
logout _ jsp	0	0	1/2	2/1	1
regsuccess _ jsp	2	0	2/6	6/4	4
admcd _ jsp	39	3	47/209	209/193	193
login _ jsp	33	2	16/94	94/86	86
cddetail _ jsp	16	1	3/49	49/46	46
admaddcd _ jsp	34	2	55/349	349/326	326
admlogin _ jsp	26	1	16/82	82/74	74
admshowSaleStatistics _ jsp	1	0	3/5	5/2	2
reg _ jsp	29	1	5/110	110/106	106
order _ jsp	8	0	5/30	30/25	25
cdmodify _ jsp	43	2	109/210	210/198	198
admlogout _ jsp	0	0	1/2	2/1	1
detail _ jsp	13	1	3/45	45/42	42
admorder _ jsp	38	2	125/199	199/184	184
admaddcdclass _ jsp	15	2	535/409	409/336	336
buy _ jsp	13	2	9/38	38/31	31
info _ jsp	7	1	5/24	24/19	19
admuser _ jsp	20	2	23/115	115/104	104
cdlist _ jsp	25	1	7/65	65/60	60

It can be inferred from Tab. 3 that the number of eps in column 4 equals the number of eps in column 5, and the number of mps in column 5 equals the number of mps in column 6. That is to say, the extended MM-Paths generated based on ALL_S_EMMP covers all the execution paths. And the execution paths generated based on ALL_EP cover all the message paths. If a test suite covers all the extended MM-Paths based on ALL_S_EMMP, it also satisfies ALL_EP. If a test suite covers all the execution paths based on ALL_EP, it also satisfies ALL_MP. So ALL_S_EMMP $\supseteq$ ALL_EP $\supseteq$ ALL_MP.

4.3 Test requirement reduction

Fig. 3 shows test requirement reduction on different criteria for 14 dynamic pages. The *x*-axis represents the dynamic pages of the web application. The *y*-axis shows the number of test requirements, i. e., the number of the extended MM-Paths. As is mentioned earlier, the ALL_EMMP criterion cannot be satisfied for infinite loops, so it is not discussed.

In order to compare the reduction effectiveness on different pages, $R_{\text{reduction}} = \dfrac{N_{\text{before}} - N_{\text{after}}}{N_{\text{before}}} \times 100\%$ is used to express reduction ratios, where N_{before} means emmp numbers before reduction, while N_{after} means emmp numbers based on ALL_EP or ALL_MP. By calcuating, the average reduction ratios on ALL_EP and ALL_MP are 54.45% and 71.11%, respectively, where the maximum ratios are 89.0% (cdmodify_jsp, $R_{\text{reduction}} = (109 - 12)/109$) and 92.8% (admoder_jsp, $R_{\text{reduction}} = (125 - 9)/125$), respectively, and the minimum ratios are 20.0% (reg_jsp, $R_{\text{reduction}} = (5 - 4)/5$) and 33.3% (buy_jsp, $R_{\text{reduction}} = (9 - 6)/6$), respectively. So, as expected, reduction approaches do help to reduce test requirements into a smaller size.

Moreover, all white bars are the shortest and the gray bars are the longest. This means that with an increase in the complex of coverage criterion, such as ALL_EP $\supseteq$ ALL_MP, the number of test requirements increases. In addition, the ALL_MP criterion simply requires that all the methods in components are covered by test cases, while the ALL_EP criterion requires all the executable statements to be covered by test cases. So the test requirement on ALL_EP represents a stronger test capability than that on ALL_MP.

There remain 11 dynamic pages those reduction requirements cannot be reduced, except for 14 dynamic pages in Fig. 3. We find that only two pages' test requirements (order_jsp and info_jsp) are reduced into a smaller size based on ALL_MP, while others have no test requirement reduction. It does not mean the approach in this paper is not effective, but indicates that more path numbers will lead to more test requirements. With an increased number of test requirements, the effectiveness of test requirement reduction will be more obvious.

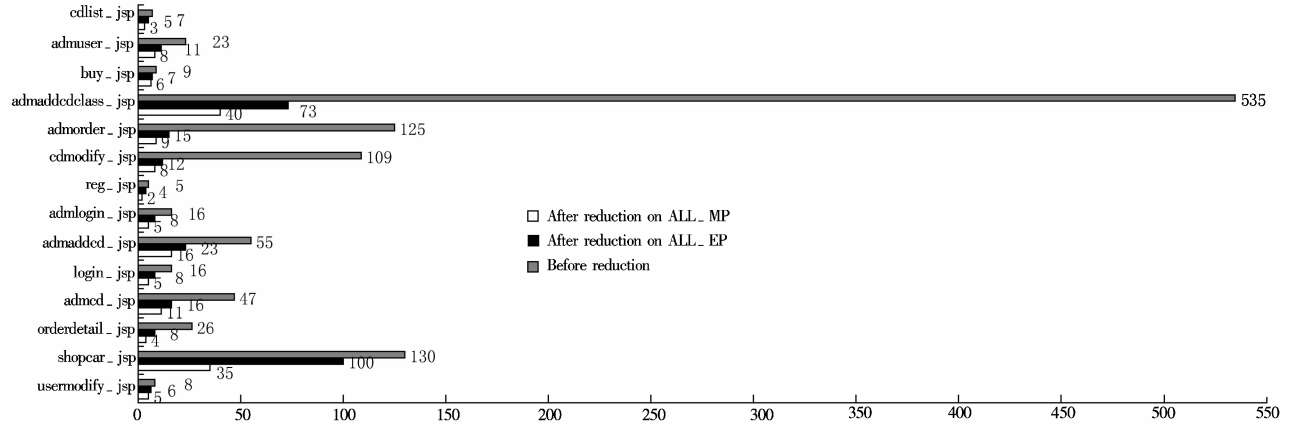

Fig. 3 Reduction in test requirements on different criteria

5 Conclusion

This paper defines four coverage criteria for testing component-based web application and proves the subsumption relationships among them. Based on the three coverage criteria, reduction approaches are proposed and their applicability to the problem of test requirement reduction is evaluated. Our results indicate that the three coverage criteria follow linear subsumption relationships in a real web application. Our results also indicate that reduction approaches can obviously reduce test requirements, and requirements generated based on more complex coverage criteria can be reduced into a larger size, but express stronger test capability. Testers can choose applicable test requirements to generate and replay test cases according to testing resources or other testing contexts. In the future, we plan to apply test requirement reduction methods to larger or more complex web applications and investigate additional uses. We also plan to investigate the spanning set and find whether it can make testing cheaper while not producing a loss in the fault-detection effectiveness.

References

[1] Gu Jingxian, Xu Lei, Xu Baowen, et al. An extended MM-Path approach to component-based web application testing [C]//*Proceedings of the 12th International Workshop on Future Trends of Distributed Computing Systems*. Kunming, China, 2008: 144 - 150.

[2] Liu C H, Kung D C, Hsia P, et al. Structural testing of web applications [C]//*Proceedings of the ISSRE*. San Jose, CA, USA, 2000: 84 - 96.

[3] Xu Lei, Xu Baowen, Chen Zhenqiang, et al. Regression testing for web applications based on slicing [C]//*Proceedings of the 27th Annual International Computer Software*

and Applications Conference. Dallas, Texas, USA, 2003: 652 - 656.
[4] Xu Lei, Xu Baowen. Testing forms in web applications automatically[J]. *Wuhan University Journal of Natural Sciences*, 2006, **11**(3): 561 - 566.
[5] Qian Zhongsheng, Miao Huaikou, Zeng Hongwei. A practical web testing model for web application testing[C]// *Proceedings of Signal-Image Technologies and Internet-Based System*. Shanghai, China, 2007: 434 - 441.
[6] Ricca F, Tonella P. Analysis and testing of web applications[C]//*Proceedings of the* 23*rd International Conference on Software Engineering*. Toronto, Canada, 2001: 25 - 34.
[7] Sampath S, Gibson E, Sprenkle S, et a1. Coverage criteria for testing web applications[R]. Newark, DE, USA: Computer and Information Sciences of University of Delaware, 2005.
[8] Cai Lizhi, Tong Weiqin, Yang Genxing. The web application test based on page coverage criteria[J]. *Journal of Donghua University*: *English Edition*, 2008, **25**(3): 291 - 296.
[9] Bertolino A, Marre M. Automatic generation of path covers based on the control flow analysis of computer programs [J]. *Journal of IEEE Trans Software Eng*, 1994, **20**(12): 885 - 899.
[10] Marre M, Bertolino A, Using spanning sets for coverage testing[J]. *IEEE Transactions on Software Engineering*, 2003, **29**(11): 974 - 984.
[11] Chung C G, Lee J G. An enhanced zero-one optimal path set selection method[J]. *Journal of Systems and Software*, 1997, **39**(2): 145 - 164.
[12] Tallam S, Gupta N. A concept analysis inspired greedy algorithm for test suite minimization[J]. *Journal of ACM SIGSOFT Software Engineering Notes*, 2006, **31**(1): 35 - 42.
[13] Zhang Xiaofang, Xu Baowen, Nie Changhai, et al. Test suite optimization based on testing requirements reduction [J]. *Journal of Electronics and Computer Science*, 2005, **7**(2): 9 - 15.

基于组件 Web 应用程序的覆盖率准则和测试需求约简

顾静娴[1] 许 蕾[2] 徐宝文[2]

([1] 东南大学计算机科学与工程学院, 南京 210096)
([2] 南京大学计算机软件新技术国家重点实验室, 南京 210093)

摘要: 为了更好地分析测试基于组件的 Web 应用, 并抉择何时结束测试过程, 提出了覆盖率准则的概念和测试需求约简的方法. 首先,定义了 4 种覆盖率准则, 并证明它们之间的包含关系. 然后,使用一种转换算法将 Web 应用测试模型转换成一种可以约简测试需求集合的 Web 应用决策-决策图(WADD 图). 最后, 基于各种覆盖率准则, 并通过分析图中各条边的等价和包含关系, 约简测试需求集合. 测试人员可以根据不同的测试环境需求选择不同测试需求集合. 案例分析表明, 在实际的 Web 应用中所定义的覆盖率准则确实存在线性包含关系. 基于不同的测试覆盖率准则, 测试需求集合平均可约简 55% 以上. 随着覆盖率准则复杂度的提高, 测试需求的规模也随之增大.

关键词: Web 应用; 覆盖率准则; 测试需求约简
中图分类号: TP391