Software Test Case Automated Generation Algorithm with Extended EDPN Model

Jinlong Tao and Lirong Chen
Beijing Computer Technology and Application Institute, Beijing, China
Email: taojinlong@126.com

Abstract—To improve the sufficiency for software testing and the performance of testing algorithms, an improved event-driven Petri network model using combination method is proposed, abbreviated as OEDPN model. Then it is applied to OATS method to extend the implementation of OATS. On the basis of OEDPN model, the marked associate recursive method of state combination on category is presented to solve problems of combined conflict. It is also for test case explosion generated by redundant test cases and hard extension of OATS method. Meanwhile, the generation methods on interactive test cases of extended OATS are also presented by research on generation test cases.

Index Terms—Test Case, Orthogonal Array, OATS, OEDPN

I. INTRODUCTION

The developing technology of object-oriented software develops so fast that it acquires more and more extensive implementation. Among them there are object-oriented analysis, design technology and object-oriented programming design language, etc [1, 2, 3, 4, 5, 6]. In contrast, the research on object-oriented software test technology is relatively weak. Object-oriented tests generate test cases usually automatically or semi-automatically, according to inner structure of program and formal specification. The generation methods of corresponding test cases are divided into generation methods based on external behavior and internal behavior. Based on the abstract data type, Doong and Frankl [7] introduced an algebraic specification describing language LOBAS, which is similar to object-oriented language grammar. It put special stress on testing information series which acts on tested object to prove whether it can make this object to be in a good state, by analyzing the software requirement and function specification. Then a set of testing tool ASTOOT is designed. It puts emphasis on method interaction, but it can not distinguish inheritance and polymorphism well. Duaie etc, in [8] discussed stage division of object-oriented software test and test content of each stage. They thought about the congener inheritance test and presented a kind of technology to give incremental test according to the inheritance relationship of hierarchical features. It worked by directly sub-category test through reusing, incrementing or updating test information of parent class. Barbey came up with an object-oriented software testing method based on formal specification CO-OPN/2 to test refinement of context. ADT test theory [9] is adopted to study the generation of class hierarchy testing cases. It analyzed relevant problems of sub-class. Test method of state machine presented by Wei [10] is based on finite state machine theory to realize and achieve object oriented state machine. It introduced specific sub-state value and general sub-state value. The objected state is only made up of a group of sub-states. Equivalent sets are separated in state numbers to reduce test sets. Mc Gregor, etc, [11] came up with “N-way switch” principle and test selection algorithm to obtain object behavior model with W-method, according to tested class analysis model or design model. They adopted more abstract modes to describe state and they corrected many shortcomings of Turner’s method. Saeed [12] applied reverse engineering to obtain corresponding state figure according to source code. By the functional test of state machine. It can converge all the states transformation. Meanwhile, the data flow coverage principle was used to test class data member in order to ensure that they can satisfy the coverage of data flow. The object-oriented software and information transmit mechanism added the amount of classes and class states. They also make interaction between classes to become more complex. So traditional software test technology is not completely applicable. McGregor came up with Orthogonal Array Test System method based on object-oriented class interaction test. But this method has the problem caused by the blind combination of class states.

We have introduced hierarchical division and testing methods of the object-oriented software test. A class test model based on EDPN model is put forward in this paper. The case of “automatic coffee machine” is referred to describe the coverage rate of test case automated generation. All the states transformation are covered at least once by this method, which ensures the adequacy of test. For the interoperability Test, we summarize the interactive relationship among the object-oriented systematic classes and propose a novel method for interaction based on EDPN model. This method is implemented in OATs and forms a software test case automated generation algorithm. It helps to extend the OATs method and is proved by experiments to have solved the combined conflict problem with case explosion caused by over testing cases.
II. SOFTWARE TEST BASED ON EDPN MODEL

A. EDPN Model

EDPN [13] is a multi-directed graph \((P, D, S, In, Out)\) including three node sets \(P, D, S\), two input and output function sets which are \(In\) and \(Out\). Among them, \(P\) is a port affair set which can be understood as input and output of sending information transference and behavior. \(D\) is a data position set which can be understood as the state of EFSM model. \(S\) is a transferring set which can be interpreted as behavior. In is an ordered duality set of \((P \cup D) \times S\) and \(Out\) is an ordered duality set of \(S \times (P \cup D)\). Transformation of input and output in \(S\) is defined by \(In\) and \(Out\) to port affair position \(P\) and data position \(D\). On several transferring combination and transferring series, during test, \(S\) can always construct input and output test series through transformation of input and output.

![Figure 1. EDPN graph.](image)

Class test model EDPN must satisfy the following conditions [14-16]:

- Uniqueness of data location. In EDPN, when one affair is allowed to occur and one transformation is triggered, it should not have the same output location. Otherwise, it is difficult to guarantee predicted effect based on protocol;
- Marked. As long as each input location of one transference in EDPN can have at least one marker, this transference can be allowed to occur and to be triggered;
- Determined. One mark in EDPN is determined so that EDPN exists in transferring allowance occurrence and trigger or it exists in stable state during performance.

B. Software Test Process

Test cases used in EDPN systematic structure can be constructed through product usage cases and can also be constructed through test models. However, it must satisfy the standards of structure design model [17]:

1. Completeness: It defines enough interface sets to provide all needed functional services for application program. The relationship between interfaces can allow control and realize needed data flow on application figure description of applying cases.

2. Accuracy: The structure satisfies its restricted conditions with using appropriate structure style and detailed illustrating interaction between interfaces.

3. Consistency: each application in the system can only be dealt with in one interface set.

Test cases are basically defined in interface layer between subsystems. For example, in figure 2, the basic interface of state graph on automatic coffee machine is between one model and one view. The model is made up by object COFFEE and object MONEY while the view is presented by object CVM.

In figure 2, rounded rectangle denotes object, ellipse denotes the object state, and rectangle denotes object behavior. Test cases can be designed according to this figure, as shown in figure 3.

III. AUTOMATIC GENERATION ALGORITHM OF CLASS INTERACTION TEST CASES

A. Combinations Problem in Class Interaction Test

McGregor presents three common standards of object-oriented class test completeness, which are coverage based on state, restriction and code. He also points out the coverage that is based on state to test how much state transformation in state transformation graph can be systematically covered. If test series are not more than one state transformation, the test performed on class is not sufficient. Conversely, if all state transformation in class interaction can be covered at least one time, it cannot be state that test series is sufficient. Therefore, on class interaction test, the coverage of state transformation at least one time is the necessary condition on class interaction test.

<table>
<thead>
<tr>
<th>No.</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

It is supposed that there are three classes of \(A\), \(B\) and \(C\) with and three states each. Three numbers of 1, 2 and 3 can represent three levels. If class \(A\) sends information to class \(B\) and class \(B\) sends information to class \(B\) all state combination will be tested during interaction. The total combination state is: three states of \(A\) multiply three states of \(B\) multiply three states of \(C = 27\). If combination of matching is adopted, that is, a given level can only appear two times, and there are totally 9 combination statuses.

During class interaction test, all class states must cover at least one time. Meanwhile, one given state level only appears two times in each state combination. During testing, in order to avoid omission of state coverage and generating unnecessary test redundancy, we can specify three classes as one group with successive relevance from up to down and from left to right, to seek the class interaction path.
In figure 4(a), when A2 (the second state in class A) is combined with B3, C2 should be selected to perform combination. That is, it should be a combination path \(<2 \rightarrow 3 \rightarrow 2\>\). However, when A3 is combined with B3, state 3 in class C is not combined, the combination path at this time is \(<3 \rightarrow 3 \rightarrow 3\>\), which is contradicted to one given state rank with only two times hypothesis in each state combination. The solution method is to perform state interaction between two interactive paths on class C, as figure 4(b) shows, and is to change in two paths of \(<2 \rightarrow 3 \rightarrow 3\>\) and \(<3 \rightarrow 3 \rightarrow 2\>\). That is, two pairs of combination of 8 and 9 in table 1.

McGregor believes that one standard array can be larger than one problem but it cannot be smaller than that problem. The main reason is if one testing hierarchical number will occur in the future, there are more sub classes which will possibly add to one receiving class. Using larger matrix can be more convenient for extension. However, when the hierarchy of one class further increases, it needs to change original orthogonal array and it once selects a new as well as larger matrix to test, which leads to restart of the whole interactive test. This will further increase complexity and blindness of the test.

B. Test Data Based on Orthogonal Array Test System

OATS method [18] is a special sampling method which is by means of defining a group of matching combination on interactive objects. It makes efforts to restrict rapid increase of combination numbers on test configuration. OATS uses a kind of balancing design: The appearing times of one hierarchy in one factor can be completely the same to those of other hierarchies. On class interactive test, most errors appear between each pair of objects rather than appear among several objects. Figure 5 extends OATS through an example and optimizes OATS method. This method can be formed by interaction among class A sender, class C, and class P receiver and each class has a relevant state conversion graph whose realization detail is not important.
In figure 5, there are two hierarchies in A. There are 2 states in class A and 3 states in class B. There is one hierarchy with 2 states in classes P. There are 3 members in classes C with 2 states in class A and 3 states in class D and in class E. The hierarchy and the state in each class can be respectively expressed by corresponding integral value, to describe the threshold value and state value in arrays. For example, if array value in the first row is 1, representing class A. The corresponding array value 3 in the second row can be taken as 1 or 2, to represent that class A is in state 1 or in state 2. The class P in the third class only has one hierarchy so three array values all represent class P, etc. The minimum standard array which is coincided with this problem is $L18(2^3 \times 3^3)$, as is shown by table II. It indicates this array enumerates 1 factor containing 2 classes and 7 factors containing 3 classes. Each line in the table indicates one specific test case. Through ranking number of each line in the array decoding and restoring each independent list of each factor, the orthogonal array will be explained and restored to test case. During explanation, the array value of each state row will rely on its matching class threshold. For example, the 10 line in $L18$ will be interpreted as the number 10 test case through one case of class p in state 3 transmitting to one case of class E in state 2. One case of class B in state 1 will be sent to information. Since it cannot be used, the last two array values of each line can be ignored.

![Figure 5. A case for OATS application.](image)

**TABLE II. STANDARD ORTHOGONAL ARRAYS L18(2^3 × 3^3) OF COMMON SELECTION CASE**

<table>
<thead>
<tr>
<th>Test cases</th>
<th>A</th>
<th>Amax</th>
<th>P</th>
<th>Pmax</th>
<th>C</th>
<th>Cmax</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>11</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>16</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>17</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>18</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

IV. USING COMBINATION EDPN MODEL TO OPTIMIZE ORTHOGONAL ARRAY TEST SYSTEM

According to EDPN model definition from above part, we can perform extension to give following definitions:

OEDPN model is a multi-directed graph (EDPN, E). EDPN individually depicts each classes. E is a directed edge set of connecting several EDPN. It also satisfies that if the directed edge $e = (u, v) \in E$, then $u, v \in P \cup D \cup S$. Here, $P$, $D$ and $S$ are node sets of EDPN, which is shown by figure 6.

![Figure 6. OEDPN model for class A, B, C.](image)

Class interactive testing OEDPN model must satisfy the following conditions:

1. Uniqueness of data location. When one affair in OEDPN is allowed to occur and one transference is triggered, it should not exist in the same output location. Otherwise, predicted result based on state is difficult to be guaranteed;
2. State can be transferred. As long as one transformation receives information from other objects or attribute value change, this transformation can be occurred and triggered;
3. Marked. Each node in node set in OEDPN $P \cup D \cup S$ can be marked and they can be marked by all nodes in precursor EDPN and subsequent EDPN. For example, in figure 7, each data location of class B in EDPN can be marked by all the data location in class A EDPN and class C EDPN. P and N denote indicated precursor data location set and subsequent data location set. For instance, the precursor data location set and subsequent data location set of "1", "2", "3" in B are all $P = N = \{1, 2, 3\}$.

Corresponding OEDPN model of figure 6 is provided, which is shown by figure 7 and one class hierarchy and corresponding status numbers can be expressed respectively by different EDPN. Meanwhile, dotted line nodes in all state EDPN indicate state reached by the only dotted node of corresponding class EDPN. We define all nodes in OEDPN model as state nodes and integral value in all nodes are all called "state class" or "array value" for easy discussing.

It is supposed that there are $n(n \geq 3)$ classes $C_1, C_2, \ldots, C_n$ and corresponding class members are individually $m_1, m_2, \ldots, m_n$. Each class in classes is interacted in pairs and each state node covers at least one time. There are at least $TCNum = \max \{m_{i} \times m_{i+1}\}$ $(1 \leq i \leq n)$ test cases. If all corresponding status nodes of $\max(m_1, m_2, \ldots, m_n)$ in classes can be covered and only covered once, it only needs $TCNum$ test cases.
The key problem in class interactive test is to seek for interactive path. One interactive path can be set as one line of orthogonal array and node value in the path is orthogonal array formed by array value. Our discussing interactive path refers that \( n \) classes of interaction in pairs gradually get one and only one state node to form a directed path. Since our matrix is nonstandard, it is also called the extended orthogonal array (EOA for short). The method of generation test cases based on extended orthogonal array is called extended orthogonal array test system.

We come up with generation interaction path of marked relevance recurrence based on combination EDPN model. Assume \( C_{i+1}, C_i, \ldots, C_1(1 < i < n) \) is randomly adjacent three classes and \( m_{i+1} \), \( m_i \), \( m_{i-1} \) are individually corresponding state node numbers. Since each state node in combination EDPN can be marked, positive integer \( k \) can be used to mark this node \((1 \leq k \leq m_i)\) towards one state node in \( C_i \). Moreover, all state ranks of precursor node set and subsequent node interacted with \( k \). They are used to mark \( k \) and individually form precursor node set \( \text{PrevSet} \) containing \( m_{i+1} \) factors of \( k \). The subsequent node set \( \text{NextSet} \) contains \( m_{i+1} \) factors. In-degree and out-degree of \( k \) are respectively \( m_{i+1} \) and \( m_{i+1} \). Through marking \( k \), it can be ensured that each directed edge can cover at least one time in relevance to \( k \). During generating interactive path with combination of EDPN, each adjacent three EDPN can be set as one group to generate all interactive paths with three state nodes, which are expressed by entity-triple \((j, k, l)\) with \( j \in \text{PrevSet}, l \in \text{NextSet} \). Then, each EDPN is recurded forward and the newly generated entity-triple with 3 nodes can be connected with previous entity-triple. Recurrence will be repeated till all EDPN are handled. Therefore, the newly generated interactive paths will not be restricted by class numbers with state node of each class, which can be extended flexibly and conveniently. In order to ensure that all state nodes and directed edge in OEDPN can be covered, we still specify that one given state rank only appears two times in each entity-triple, the searching association is specified from up to down. It can prevent test case explosion caused by blind state combination To a large extent. The following programs describe the marked association recurrence algorithm with C++ pseudo-code.

```
GeneratTriplet(Triolet & T, int m_1, int m_2, int m_3) {
    //Generate 3 classes interaction triple. T is generated triple set
    for each StateNode k ∈ EDPN;{
        // EDPN_k is EDPN of the second classes, 1≤k≤m_2
        Add marks to k with all PrevNode j ∈ PrevNode and NextNode l ∈ NextSet
        //Using all PrevNodes and NextNodes to mark, 1≤j≤m_1, 1≤l≤m_3
        For each j ∈ PrevSet or each l ∈ NextSet{
            If (PrevSet=φand NextSet=φ) {
                If (j=k) {
                    Find the minimal l ∈ NextSet and l≤K;
                    If (only exist one l ∈ NextSet ) select the l;
                } // In-degree or out-degree of k is 1
                Else
                    Find the minimal l ∈ NextSet and (j=1 v k=1);
                    T=T∪(j, k, l);
                }
            Delete j from PrevSet and l from NextSet;
            //Delete mark from PrevNode and NextNode set
            If (j=k==l) exchange l-1 of Triplet (j-1, k, l-1) and 1 of Triplet (j, k, l)
            If (PrevSet=φand NextSet=φ) {
                //The case of m_2=1
                If (j=k) T=T∪(j, k, l);
                Else if (k=m_1) T=T∪(j, k, l);
                Else T=T∪(j, k, l);
            } //Considering the special case that there is not the same in (j, k, l)
            If (PrevSet=φand NextSet=φ) {
                //The case of m_3=1
                If (l=k) T=T∪(l, k, l);
                Else if (k=m_1) T=T∪(l, k, l);
                Else T=T∪(l, k, l);
            } //Considering the special case that there is not the same in (j, k, l)
        }
        // for each j
        // for each k
        //GeneratTriplet()
    }
}
```

With OATS method and MARM method given by McGregor, we optimize OATS method to come up with interactive test cases of the generation class with extended OATS method, based on combination EDPN. It is described as:

All factors and each factor hierarchy are determined. According to UML class graph, all factors, corresponding hierarchy and state numbers of each factor are determined. The class hierarchy and its corresponding state number tables is set up as table III.

Corresponding OEDPN model is established according to analysis at first step. MARM method is applied to solve interactive path. The extended orthogonal array table is also established. Function `GenerationInteractPath()` is transferred to solve...
interactive path $P_1 \sim P_6$ of common examples shown by figure 7 and the established orthogonal array table is shown by table IV.

TABLE III. LEVEL OF CLASSIFIED AND THEIR CORRESPONDING STATE NUMBER TABLE

<table>
<thead>
<tr>
<th>Classes</th>
<th>Class</th>
<th>Level</th>
<th>State number</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>P</td>
<td>C</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>E</td>
<td>D</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

Test cases are constructed based on mapping and lines in tables. Ranking number of each line in the array can be decoded and restored to each individual array of each factor so that orthogonal array can be explained to restore test cases.

V. CONCLUSION

On the basis of summarizing software test case generation and relevant technology research states, this paper has studied object-oriented automatic generation technology of software test cases in-depth. Based on interactive relationship of object-oriented systematic classes, for interactive test, we also propose interactive test method based on combination EDPN model and applies this model to OATS method to extend OATS method. The next job is to perform research on automatic generation technology of software test cases, to design and realize corresponding software test platform in order to lay foundation on automatic generation technology direction and application of software test cases.

REFERENCES


