A Test Case Suite Generation Framework of Scenario Testing

Ting Li [1,3]

1) Shanghai Development Center of Computer Software Technology
Shanghai, China
lt@ssc.stn.sh.cn

Zhenyu Liu [2]

2) Shanghai Key Laboratory of Computer Software Testing and Evaluating
Shanghai, China
{lzy, jiangx}@ssc.stn.sh.cn

Xu Jiang [2]

3) Shanghai Software Industry Association
Shanghai, China
lt@softline.sh.cn

Abstract—**This paper studies the software scenario testing, which is commonly used in black-box testing. In the paper, the workflow model based on task-driven, which is very common in scenario testing, is analyzed. According to test business model in scenario testing, the model is designed to corresponding test case suite. The test case suite that conforms to the scenario test can be obtained through test case generation and test item design. In the last part of the paper, framework of test case suite design is given to illustrate the effectiveness of the method.**

Keywords-test case; software test; scenario testing; test suite.

I. Introduction

Software testing is the main activity of software quality. The goal of software testing is to validate whether software is good or conform to the initial requirement. However, software engineers always consider that software testing should be terminated under certain conditions. The adequacy of software testing is an important factor according to testing purpose. It is generally agreed that when software testing reaches expected test purpose, the software testing activities could be terminated. Thus, the quality and overhead cost of software testing can be considered fully and controlled effectively.

Nowadays, business processes become complicated by the development of technology and information. At present, many factors led to the e-business more complicated, such as business logical become complexity, component-based development widely accepted and complete workflow processes scattered in various business components. The component-based software led to process the data flow and control flow the more tightly and more complicated. Therefore, the scenario testing and verification has become important increasingly before system runtime. The scenario testing can be regarded as an independent test, which becomes an important part of black box testing. On the one hand, many business processes adopted e-business management. The traditional paper-based business model was replaced. On the other hand, uncertainty of system requirement brings risks to scenario testing. So, it is necessary to consider scenario testing gradually.

Software testing is the critical activity in the software engineering. However, some research shows the design of test cases will cost much time during software testing. While many business workflow management systems have emerged in recent years, few of them provide any consideration for business workflow verification.

The research work on business scenario testing and validation is less. As for the actual development of the application system, the every operation in business is designed and developed well. However, business processes testing and requirements verification are very important and necessary for software quality. The scenario testing satisfies software requirement through the test design and test execution.

Scenario testing is to judge software logic correction according to data behavior in finite test data and is to analyze results with all the possible input test data. It is generally considered that test design can evaluate software testing quality and select the test data. Scenario-based testing focuses on what the end-user does, not what system does. The flow path and boundary conditions are used to design test data corresponding to business scenario. Therefore, the main purpose of scenario testing is to find business flow interaction defects as possible. The test results also help to record results of software runtime during test execution.

Based on the existing test purpose of scenario-testing, in this article, we'll focus our research efforts on test case design of scenario testing. In the first section, we introduce the business scenario testing and its importance. And then, in the second section, we give out a business scenario model. In the third part, a test case suite model is proposed. The generation method of design test case is introduced in the fourth section, according to the requirements of business scenario which conforms to test case suite model. The fifth part gives related works and the conclusion is drawn with a corresponding discussion in the final section.

II. Business Scenario Model

Firstly, we give the typical business scenario model, which supports business model. The business model consists of three elements: business workflow, business scenario and basic operation.

A. Concepts

The business workflow indicates the typical business to accomplish the basic business process. Indeed, the basic workflow always consists of different scenario in the business, and any scenario corresponding to the business workflow. Therefore, we give the definition of the business scenario model.

Firstly, notations are introduced (see Table 1).

TABLE I. NOTATIONS

Symbol	Implication
BW	Business Workflow
SC	Scenario
OP	Operation
I	Input Set
O	Output Set
D	Data, Test Data
U	User
R	Role
S	State
PS	Previous State
SS	Success State
TR	Test Result
TC	Test Case
TCS	Test Case Suite

Definition 1: (Business Workflow) Assumes the business workflow, there are some business models for the fully business operations.

BW= {I, O, SC, D}, in BW, I am input set, O is output set, SC is related the scenarios, D is the test data corresponding to specific scenarios.

Definition 2: (Scenario) As for business process, there are scenarios in the business workflow. Every scenario could represent the possible workflow path in business flow.

SC= {I, O, OP, R, U, D, S, RE}, in SC, I and O indicate input and output separately, OP is the operation related to the current scenario, U and R are abbreviated of user and role separately. User and Role are the execute member of the operation. S is the state which describes the workflow.

Definition 3: (Operation) In any scenario of business workflow, operation is the basic element which accomplishes the specific function or function collection.

OP= {PS, SS, UR, D}, in the OP, PS and SS are previous state and successive state, which indicate the workflow state before and after the operation.

Definition 4: (Role) There are three kinds of roles, one is the workflow initiator (Sender), other two are workflow receiver and workflow informed person the workflow-based business model.

Definition 5: (User) Assumes the business process, there are multiple operators, operator is a member of role for specific operation.

For the two operations are carried out by the two users in workflow, there are seven typical models for operation. In these seven models, one model is concurrency or parallel, and the other models are different orders of successive operations. For the two operations and two users, two operations implement the following executable sequence (see Table 2).

Now, many collaborative business software systems are followed these two models: concurrently and end-start. Other models would exist in some special collaborative software system.

TABLE II. MODELS OF TWO OPERATIONS

Model	Description	Notation
Concurrent	Starts at the same time	A||B
Switch	Any one starts	A|B
Start Start	One start and another state	ss(A, B)
End Start	One finish then another state	es(A, B)
End End	One finish then another could finish	ee(A, B)
Start End	One start the another could finish	se(A, B)
Loop	Repeat when finish	[A]

Based on the workflow definition, four typical business models: sequence, parallel, switch and loop.

B. Model

- Sequence structure is used to define some activities in sequence execute order. In Fig. 1, where OPa, OPb are two independent tasks, OPb is defined as S2 and the causal state is S3. The sequence model could be denoted as es(OPa,OPb).

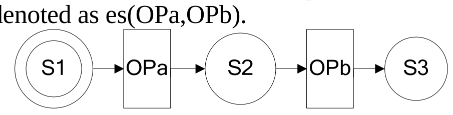

Figure 1. Sequence Structure

- Parallel structure used to define the order is not strictly executable. Every task in branch is not running at the same time, it needs to use two basic workflows: 'branch' and 'connection'. Fig. 2, where OPa, after implementation by S1 directly transferred to the S2 and S4, then the two tasks OPb, OPc can be executed separately, therefore is a parallel relationship, and then performed the OPb, OPc, after its implementation by the OPd. Fig. 2 could be denoted as es(OPa, (OPb || OPc), OPd).

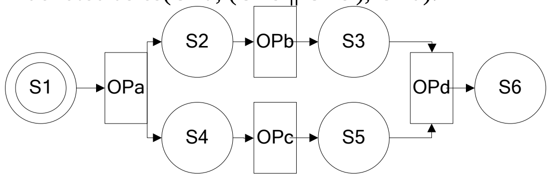

Figure 2. Parallel Structure

- Switch Structure is similar to the parallel structure, but the condition is selected according to state of S2/S3, rather than parallel structures, which OPb and OPc is performed at the same time. This model could be denoted as es(OPa, (OPb|OPc), OPd).

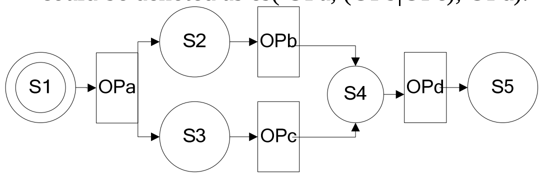

Figure 3. Branch/Select Structure

- Loop Structure. This structure is used to define that repeat the implement is needed in many tasks. The loop structure shown in Fig. 4 is modeled as es(OPa, [es(OPb,OPc)]). Here, operation OPb is executed repeatedly in loop structure.

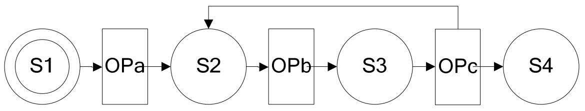

Figure 4. Loop Structure

III. Test Case Suite Model

During software testing, a standard test script template is provided to facilitate execute test case and collect test results. The standard template not only is the standardized collection process of quality elements, but also can assist comparative analysis of different test results.

The test script is the basic element for test execution. Therefore, test script design is accounting for much time on test design. It is necessary to find generic test cases in order to reduce the cost of the test case design for the further test case design and test execution. For some same operation in business, the reuse test script can accelerate the test case design. The process of automatically generating test cases can also improve the efficiency of test design [1]. As for testing activities is high cost work, test design is accounting for the workload and ability of test designer, the test case reuse method is adopted to improve the design efficiency. The reuse test case can reduce the design costs if used existing test case which have been used. The test case reuse technique can improve the test efficiency and reduce test cost [2].

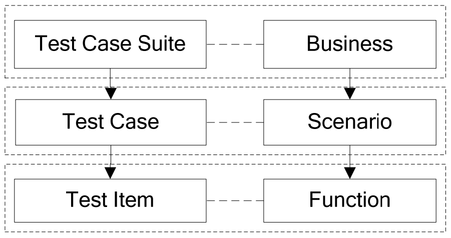

Figure 5. Test Case Suite Level Model

The test case suite is top level in the test suite model. The test case suite, also named test suite, is the collection of test case. The test case suite is corresponding to the business model. The test case is the middle level in Fig. 5. The every test case in test case suite is corresponding to the specific scenario in the business model. In other words, the test case could accomplish one of the business flows. The business flow always consists of many possible workflow paths. For example, as for the bank transaction workflow, the same bank transaction in different bank could design two scenarios in the transaction business workflow. The example demonstrates the relationship between the business and scenario. There are more scenarios than the examples in the actual test requirement. Therefore, the different methods of transaction could design for different test case in the test suite of business.

The test case is related to specific scenario in business workflow. In the business scenario, many tasks constitute the integrated scenario. The operation is executed by user with authority role. The relation between two operations in the specific scenario should be modeled, which introduced in section 2. The operations could be converted into test items in test suite model. In the test suite model, the operation is the basic element. During software testing, test item is the basic elements during testing execution.

Definition 6: (Test Case Suite) Test case suite consists of test purpose and related test cases. Test case consists of related scenario, test method, expected test result and test item collection. BNF is shown as Table 3.

TABLE III. BNF Of Test Case Suite

```
<TestCaseSuite>::=<BusinessRule><TestPurpose><User><Role>
            <TestType><Data>{<TestCase>}
<BusinessRule>::== /*refer to the business model */
<TestPurpose>::= /* test goal for the business*/
<TestType>::= function | performance | security | others
<User>::=/*  user info for the operator */
<Role>::={<User>} /* usergroup */
<TestCase>::=<Scenario><Method>{<TestItem>}<TestResult>
< Scenario >::= /* test case state and its function when start */
<Method>::= manual | automated
<TestResult>::=  /* the expected test result for test case */
<TestItem>::=<TestInput><TestOutput><TestData><TestOracle>
<TestInput>::=   /* operation procedure and input information*/
<TestData>::= /* test data collection refers to the input */
<TestOracle>::= /*expected the result based on input and data*/
<TestOutput>::= /* software output information*/
```

Definition 7: (Test item) Test item is an individual test step in the test case model, including the test input, test data, test oracle and test output.

Test case consists of many test items. The test items belong to the relevant test case. The test item is the part of the test case and every test item is run during the test execution. The different test scenario can add different test items to fulfill the test case. Therefore, scenario testing can be reconstructed according to the description of the test case. Test item is a fundamental element to construct test case.

IV. Generation Method

The test case suite should be designed according to the business model. The business model and workflow is complicated in that the possible business flow path should be considered completed for the validation and verification.

A. Generation Framework

The paper gives an integrated framework. Fig. 6 gives the test case suite generation framework. The framework helps test designer to design test case suite, test case and test item. The business model and test purpose are input items. The important activities in test design are analyzing path and related operation. The analyze path is to get the possible path according to the business model. The all possible paths are to help generate related test case. The purpose of analyzing operation is to design the test item. The test case, test item, and their test data constitute the test case suite, which is needed for test execution.

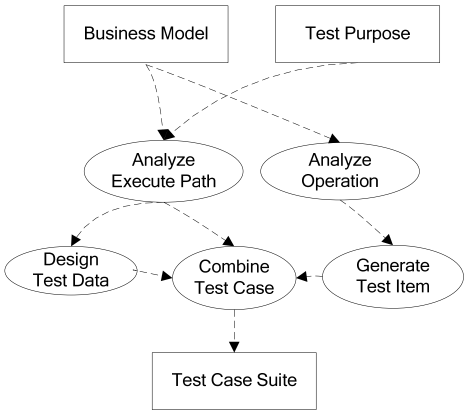

Figure 6. The framework of test case suite generation

- Analyze Execute Path

The test scenarios need to be analyzed all possible execution paths. According to the model presented earlier and test purpose, the designer should analyze the paths to accomplish the specific business. Test cases need to be considered that the coverage of business processes should conform to test purpose. The typical logic coverage consists of statement coverage, decision coverage, condition coverage, decision/condition coverage.

Test requirements consist of various logical coverage, such as path coverage, decision coverage, condition coverage and condition/decision coverage, etc. The different coverage requirements impact on test data design. The decision coverage is to determine the true and false values in one statement.

Here, we consider the situation, including two conditions, the one condition (C1) is x<0 or y>5 and the other condition (C2) is x>2 and y>3. The x value and y value constitute collection 1 and collection 2, the detail value and condition as below. The collection 2 satisfies the condition/decision coverage.

Collection 1(C1)

x	y	x<0	y>5	x>2	y>3	C1	C2
-1	3	T	F	F	F	T	F
2	6	F	T	F	T	T	F
3	2	F	F	T	F	T	T

Collection 2(C2)

x	y	x<0	y>5	x>2	y>3	C1	C2
-1	3	T	F	F	F	T	F
2	4	F	F	F	T	T	F
3	2	F	F	T	F	T	T

The data size should be further considered. The more data size, the more test execution time and cost. That is to say, the minimum number of test cases improved test efficiency. The design test data will affect the size of test case. Although different test data could achieve the same test purpose eventually, the less test data will reduce the test execution time.

It can be seen, the value y in condition 2 is not meet is true of the condition y> 5, so a new value is needed to design. If consider further optimization, the collection can be two data sets, which reduce one data set compare to original data sets. The final collection is:

x	y	x<0	y>5	x>2	y>3	C1	C2
-1	3	T	F	F	F	T	F
3	4	F	T	T	T	F	T

- Analyze Operation

Operation analysis will get the basic operation based on business model, including test items and test cases. Each test item is corresponding to a basic operation. The basic operations maintain consistency with the test data.

- Combine Test Case

Each operation will be designed to one test item. Test case consists of test procedures and related test data. Test procedures are sequence collection of test items, which sequence corresponds to execution path that analyzed during test design. The execution path is transformed into a series of branches that determine the condition and coverage. The user role information is also to be considered.

Through a set of test cases, test designer can be combined into a test case suite based on business models and their test purpose.

B. Algorithm

Here, we develop the algorithm for generating the test case suite, test cases and related test data. The algorithm analyzes the business rule and transverse all possible workflow paths. The every execution path is corresponding to the test case. The all test cases are composed into an integrated test case suite, that is to say, the test case suite consists of all possible execution paths. Indeed, some execution paths may be invalid due to the contradiction between test data and business logical. The some execution paths will be reduced according to test data.

```
INPUT: SC,OP,TR
OUTPUT: TCS
Begin
  OP'= {};
  EP = {};
  TD = {};
  COND = {};
  Foreach sc In SC Do
  Begin
     OP' = OP' U getOper(SC);
     EP = getAllPath(SC);
     TD = createTestData(OP');
     Foreach ep In EP Do
       TD' = TD' U  getTestData(ep, TR);
     TD' = ProcessUnsed(TD);
  End
    Foreach op in OP Do
      TC =TC U Generation (OP);
  TCS = Combine(TC,TD')
End
```

C. Example

Here, we give the actual example to demonstrate the experimental result. The business process consists of four operations, Oa-Od; four conditions, C1-C4. Therefore, the maximum path is 2*2*2*2=16, if consider the possibility of execution, the all possible paths are {OaC1C2Ob, OaC1C2Oc, OaC1C3Oc, OaC1C3C4Oc, OaC1C3C4Od}.

The Oa-Od is needed to design the test item corresponding to an operation.

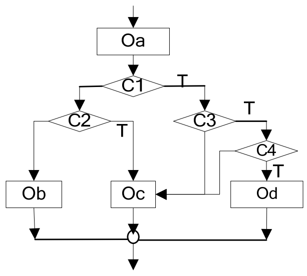

Figure 7. An actual business process

The second step is to design the test data for possible path. If test requirement is condition/decision coverage, some condition coverage should be deleted or redesigned due to the contraction with test data. In table 4, the 8th test data are unreachable for condition (x>1 and y>10000) true. The condition is false due to y=6000 in No.8. And then, we could design the test data for execution path. So the final test data in No.8 should be x=2, y=20000, z=4.

TABLE IV. CONDITIONS AND TEST DATA

No.	Condition (T)	Test Data			
		x	y	z	T/F
1	x>0 or y>5000	0	3000	0	T
2		2	12000	-2	F
3	y<2000 or z<3		1000	2	T
4			4000	3	F
5	x>1 and y>10000	1	5000		T
6		2	15000		F
7	y/(x+z)>10000	1	20000	1	T
8		2	6000	2	F

V. RELATED WORKS

Software testing, as is mentioned by Zhu [3], can be divided into white-box testing and black-box testing, according to whether or not it involves with code. The white-box testing includes testing based on code and testing based on standard. Well, the black-box testing includes hybrid test, which is based on standard. Moreover, FSM (finite state automata) is used to generate test data for lots of models [4-6]. Also, many researches are aim at generating testing path of EFSM (extended finite state automata). As for EFSM testing, testing coverage will involve many aspects, for example, state coverage, transition coverage, path coverage, and so on. These coverages are used to generate FTP (feasible transition path), which is required by test cases [7] [8].

Weyuker's axiomatic system proposed basic properties of software test adequacy criteria. By using axioms set, it could assess the adequacy criteria for the testing. In the evaluation of axiom, Weyuker's criteria for testing are compared with test adequacy of the criteria [2]. The anti-composition axiom in [2] is not positive. This axiom points out, even though test case suite T is adequate for each component in testing program P, T in the case of P, it may not necessarily be adequate. This axiom shows, as for the tested program, the adequacy of testing is impossible to be fulfilled. Although the axiom indicates an intuitive concept in software testing, it cannot enhance the confidence of test engineer with certain test case. So, the test criterion, which is described by the axiom, has a kind of a negative character.

VI. CONCLUSION AND FUTURE WORK

The paper gives the method of generating the complete test case suite according to the business model for the scenario test. The framework is fulfilling the demand of test requirement and supporting the design test case and test suite effectively in scenario testing.

Future work of this research includes deeply research further improve the efficiency and correctness of the test case suite and give more extension of choosing better test cases from the alternative test case which generate thought reuse technique. The related work and the newly exploration of reusing technique are still ongoing for software testing.

ACKNOWLEDGMENT

The work is supported by National Torch Program under Grant No. 2009GH510068 and Shanghai STCSM Program under Grant No.10DZ2291800. The authors would like to thank without knowing the name for their helpful suggestions.

REFERENCES

[1] W. K. Leow, S. C. Khoo, and Y. Sun, "Automated generation of test programs from closed specifications of classes and test cases", Proceedings of the International Conference on Software Engineering, 2004, pp. 96-105.

[2] E.J. Weyuker, Axiomatizing software test data adequacy. IEEE Trans. on Software Engineering, 1986, vol. 12(12), pp. 1128-1138.

[3] H. Zhu and Z. Jin, Software Quality Assurance and Testing. Science Press, Beijing, 1997, pp. 142-147.

[4] Z. Liu, G. Yang, and T Li, "A Component-based Reuse Technique of Software Test Cases, Proceedings of the 3rd World Congress for Software Quality", Munich, Germany, 2005, vol. 1, pp. 26-30.

[5] A. A. Andrews, J. Offutt, and R. T. Alexander, Testing Web Applications by Modeling with FSMs. Software and Systems Modeling, 2005, vol. 4(3), pp. 326-345.

[6] R. Lai, "A survey of communication protocol testing," Journal of Systems and Software, 2002, vol. 62, pp. 21-46.

[7] D. Lee and M. Yannakakis, "Principles and methods of testing finite state machines-a survey," Proceedings of the IEEE, 1996, vol. 84, pp. 1090-1123.

[8] A. Kaliji, R. M. Hierons, and S. Swift, Generating Feasible Transition Paths for Testing from an Extended Finite State Machine(EFSM) In proceedings of 2nd International Conference on Software Testing Verification and Validation, Denver, USA, 2009, pp. 230-239.

A. Y. Duale and M. U. Uyar, "A method enabling feasible conformance test sequence generation for EFSM models," IEEE Transactions on Computers, 2004, vol. 53, pp. 614-627.