*2012 International Conference on Computer Technology and Science (ICCTS 2012)*
***IPCSIT vol. 47 (2012) © (2012) IACSIT Press, Singapore***
*DOI: 10.7763/IPCSIT.2012.V47.15*

# An Effective Coverage Strategy for System Function Testing

Du Qingfeng and Huang Jing[+]

School of Software Engineering, Tongji University, Shanghai, China

**Abstract.** Software testing is an important quality activity. System level testing (system testing) is the final checkpoint before delivering for acceptance testing. However, until now, there are no effective coverage strategies for system function testing. In this paper, aim is to put forward an effective function coverage strategy for system testing. Concrete method is in terms of system level input and output, use case diagram, system transition diagram and system entity relationship diagram to analyze and design test cases. The merit of the coverage strategy is that test cases can cover system functions from different angles. Experimental example shows the effectiveness of our proposed system function coverage.

**Keywords:** system testing, system functions, coverage strategy

## 1. Introduction

System testing focuses on system function and other non-function system features[1]. The goal of system testing is to validate whether developed system meets system requirements. After system testing, if there are still some defects, especially major defects in system, then acceptance testing will not be passed. So before the acceptance testing, how to put into effect system testing is very critical.

System testing is the integration of hardware, shelf software, application software, network and so on. It is required to carry out in real environment. The environment of system testing may include[2]: (1) PCs and mainframes for specified configuration. (2)Different types of peripherals, such as printer, ATM, barcode reader, sensor and so on. (3) Specified operation system, DBMS, middleware and application software. (4) Required network. (5) Necessary testing tools. (6) Other computer system that interacts with the computer system.

In this paper, we propose an effective coverage strategy for system function testing but not including non-function features. The coverage strategy is based on the analysis of input and output at system level, use case diagram, system transition diagram and system entity relationship diagram. After analyzing, some experimental example will be given to verify the coverage strategy.

## 2. Coverage Strategy Analysis

### 2.1. Coverage strategy based on input and output at system level

Whatever computer system has input and output at system level.， system can interact with different users, devices or other systems through input/output. Usually, the input to a computer system may include:

1) To input by users directly, such as by screen and keypad. Then analyze all possible input types. The coverage indicator is that all these types must be covered more than once.

2) To input by different hard devices except for screen and keypad, such as bankcard, sensor and so on. Analyzing all possible device types and different types of input information for the same device, the coverage indicator is that all possible device types and different types of input information must be covered more than once at least.

[+] Huang Jing
*E-mail address*: kabu926@gmail.com

3) To input by wireless signals. Analyzing all possible wireless signal types, the coverage indicator is that all signal types must be covered more than once at least.

4) To input by cable signals. Analyzing all possible cable signal types, the coverage indicator is that all signal types must be covered more than once at least.

5) To input by reading external file. Analyzing all possible file types, the coverage indicator is that all file types must be covered more than once at least and the size of file must be considered when reading.

6) To input by Bluetooth. Analyzing all possible signal types, the coverage indicator is that all signal types must be covered more than once at least.

7) To input by infrared ray. Analyzing all possible infrared ray signal types, the coverage indicator is that all signal types must be covered more than once at least.

8) To input by other computer systems. Analyzing all possible interfaces types when communicating, the coverage indicator is that all interface types must be covered more than once at least.

The output may include:

1) To output from screen. Analyzing all possible output type, the coverage indicator is that all possible output types must be covered more than once at least.

2) To output from different hard devices except for screen and keypad, such as conveyor, robot and so on. Analyzing all possible device types and different types of output information for the same device, the coverage indicator is that all possible device types and different types of output information must be covered more than once at least.

3) To output by transmitting wireless signals. Analyzing all possible wireless signal types, the coverage indicator is that all signal types must be covered more than once at least.

4) To output by cable signals. Analyzing all possible cable signal types, the coverage indicator is that all signal types must be covered more than once at least.

5) To output by specified format file. Analyzing all possible file format types, the coverage indicator is that all file format types must be covered more than once at least and the size of file must be considered when output.

6) To output by printer. Analyzing all possible printing types, the coverage indicator is that all printing types must be covered more than once at least.

7) To output by Bluetooth, Analyzing all possible signal types, the coverage indicator is that all signal types must be covered more than once at least.

8) To output by infrared ray. Analyzing all possible infrared ray signal types, the coverage indicator is that all signal types must be covered more than once at least.

9) To output to other computer system. Analyzing all possible interfaces types, the coverage indicator is that all output interface types must be covered more than once at least.

## 2.2. Coverage strategy based on use case diagram

**1.** Definition of basic flow and alternative flow

Use case diagram represents different level abstraction of system functions. Use case diagram provides a description of how a system is used by different actors. Use case diagram achieves the following objectives[4]:

(1) To define the functional requirements for system.

(2) To define the scenarios of usage that is agreed by the end-user and the software engineering team.

(3) To provide a clear and unambiguous description of how the end-user and the system interact with each other.

At system level, every use case represents a high cohesion function, i.e. function that cannot be decomposed in use case diagram.

**Definition 1. (basic flow)** In system level, basic flow is a normal transaction flow and there is no exceptional event in a basic flow. Basic flows are designed from use case diagram and should cover all normal transaction flows.

**Definition 2. (alternative flow)** In system level, alternative flow is abnormal transaction flow and is an exceptional event that is not presented in use case diagram. We need analyze alternative flows from different aspects, such as use case diagram, system requirements, system environments, different devices and so on.

**Definition 3. (system scenario)** In system level, scenario is comprised by a basic flow or several different basic flows, or a basic flow and an alternative flow, or several basic flows and an alternative flow,

or several basic flows and several alternative flows and so on. System scenario should be feasible and reasonable when understood from system business logic.

Actually, in most systems there are lots of basic flows and alternative flows. According to the idea of scenario and theory of permutation and combination, the amount of scenario is infinite, such as in a scenario the same alternative flow may appear many times and the number of times is not limited. The fundamental coverage indicator is that the set of designed scenarios must cover all basic flows and alternative flows once at least.

**2.** Experimental example

Figure 1 is a use case diagram of bank system. In the figure 1, there are five actors. According to the definition above, we may identify 21 basic flows, such as actor Electronic Funds Transfer, it includes four basic flows: (1) Electronic Funds Transfer makes withdrawal. (2) Electronic Funds Transfer makes deposit 1000 dollars. (3) Electronic Funds Transfer makes deposit 10000 dollars. (4) Electronic Funds Transfer makes deposit with an invalid account number. The analysis of basic flows to the other four actors is similar.

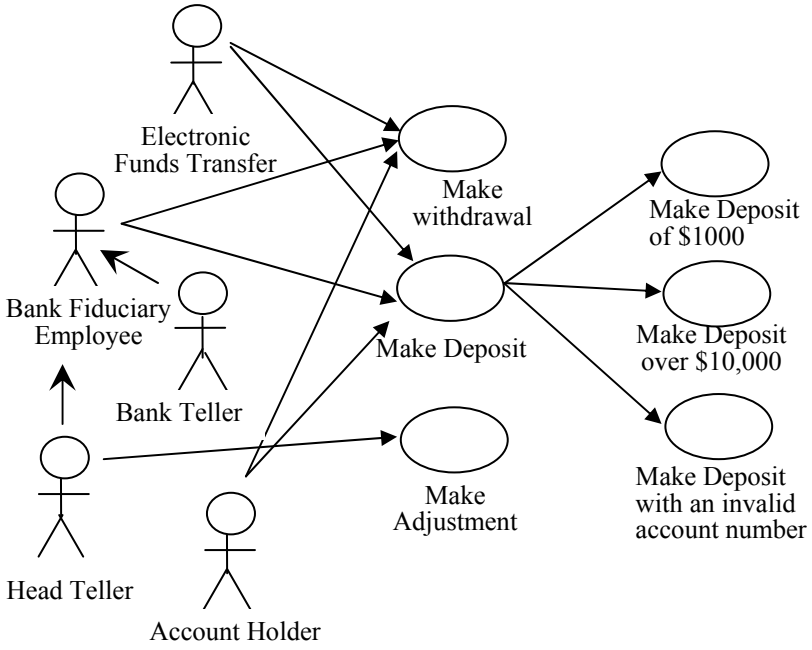


Fig. 1: use case diagram of a bank system

From the example above, alternative flows for electronic funds transfer may include: (1)When withdrawing, cash is not enough. (2) When depositing, cash is not the multiples of 100. (3)When depositing, cash includes counterfeit money. (4) Password is error. (5)Device of deposit is fault. (6) When withdrawing or depositing, network is interrupted. (7) When withdrawing or depositing, press “cancel” key. (8) Recording data prompts error, and system enters safe mode. (9) Receipt prints failure, and so on. The analysis to the other four actors is similar.

When basic flows and alternative flows are identified for all actors, the next step is to design scenarios to cover all basic flows and alterative flows once at least. As the amount of scenarios may be infinite, the set of scenarios is not unique.

The following is three examples of scenarios for electronic funds transfer:

(a) Withdrawing (1000dollars) →succeed→exit.

(b) Depositing(first,1000dollars)→succeed→Depositing (second,500dollar)→succeed→exit.

(c) Withdrawing(first,1000dollars)→succeed→Withdrawing(second,10000dollars)→succeed→ Withdrawing(third,10000dollars)→Cash is not enough in box→exit.

After designing scenarios, test cases should be designed to cover all these scenarios.

## 2.3. Coverage strategy based on system transaction flow thread(STFT)

1. Manuscript Understanding system transition diagram.

System transition diagram is an abstraction of system behavioral requirement and provides a description of how the system responses to system events. In the diagram, nodes represent states and edges represent transitions.

2. Definition of STFT

Definition of STFT is based on system transition diagram. The definition is as follows:

STFT is a path that is comprised by different state nodes and edges in system transition diagram, and in this path, it is necessary that there is at least a state node is used twice or more. By the definition, we know that a STFT may be very long, even infinite in theory, as a state node may be used unlimited times in a STFT.

3. Designing STFTs and corresponding test cases

Firstly, we analyze probability of branch to every node in system transition diagram according to practical businesses logic[3]. As for a node, the sum of all branch probability is 1. Secondly, STFTs will be identified as much as possible. The third, a probability boundary value must be specified for STFTs and the probability of a STFT is the product of its all branch probability in the STFT. Calculate probability of all these STFTs. All of the STFTs, whose probability is larger than or equal to the specified probability boundary value must be covered by test cases.

Generally, when specifying the probability boundary value for STFTs, several factors must be considered, such as importance of businesses, 80-20 rule, coverage indicator, allowed testing time, testing cost and so on. The value may be adjusted according to the testing requirements.

4. Experimental example

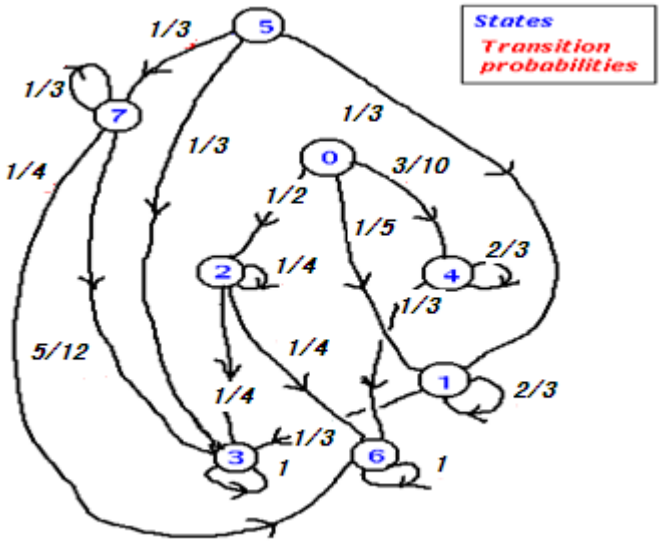


Fig. 2: a system transition diagram

Figure 2 is a system transition diagram. In the system transition diagram, nodes represent system states, arrows (or edges) represent transitions and all the fractional numbers represent probability of node branch. In the Figure 2, STFTs should be identified as much as possible and then calculate the probability of all STFTs. From the example we can see, if the probability boundary value is 0.02 and all these STFTs whose probability is larger than or equal to 0.02 must be identified and these STFTs should be covered by test cases. Table 1 is some examples of STFTs:

Table 1: some examples of STFT

| STFTs(number represents node) | STFTs probability | Remarks |
|---|---|---|
| 0→2→2→3→3 | (1/2)*(1/4)*(1/4)*1=0.03125 | Accepted test case |
| 0→1→1→3→3 | (1/5)*(2/3)*(1/3)*1=0.04444 | Accepted test case |
| 5→7→7→6→6 | (1/3)*(1/3)*(1/4)*1=0.02778 | Accepted test case |
| 5→7→7→7→3→3→3 | (1/3)*(1/3)*(1/3)*(5/12)*1*1=0.015432 | Not accepted test case |
| 0→4→4→6→6 | (3/10)*(2/3)*(1/3)*1=0.06667 | Accepted test case |

## 2.4. Coverage strategy based on system entity relationship diagram

1. Understanding system entity relationship diagram

System entity relationship diagram is an abstraction of system data, and it depicts the relationship among different entities. These relationships include 1:1, 1:m, and m:n. System entity relationship diagram must be reviewed.

2. Designing test cases for covering all relationships in system entity relationship diagram

Among these relationships, cardinality and modality are the basis of designing. The coverage indicator is that test cases should cover all relationships among entities in system entity relationship diagram.

3. Experimental example

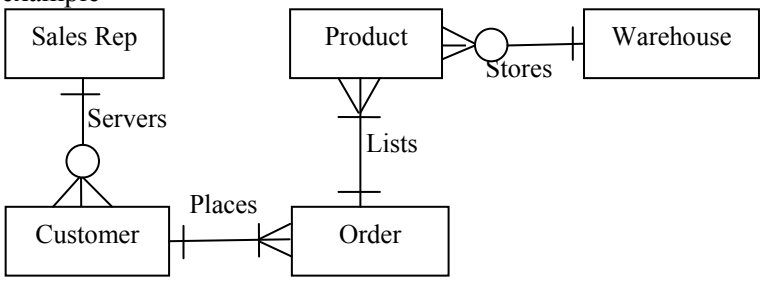


Fig. 3: an example of system entity relationship diagram

Figure 3 is an example of system entity relationship diagram. The figure shows five entities and their relationships. According to the relationships in the figure, we can design some test cases to cover all these relationships:

(1) There is relationship of 1: m or 1:0 between Sales Rep and Customer. Test cases should satisfy: Sales Rep servers zero Customer or servers one Customer or more than one Customer.

(2) There is relationship of 1:m or 1:1 between Customer and Order. Test cases should satisfy: Customer places an Order or places more than one Order.

(3) There is relationship of 1: m or 1:1 between Order and Product. Test cases should satisfy: Order lists one Product or lists more than one Product.

(4) There is relationship of 1: m or 1:0 between Warehouse and Product. Test cases should satisfy: Warehouse stores zero Products or stores one Product or more than one Product.

# 3. Conclusion

In this paper, we discuss an effective coverage strategy for system function testing. As a computer system functions can be presented by system inputs, system outputs, system use case diagram, system transition diagram and system entity relationship diagram, it is reasonable for us to analyze coverage strategy for system function testing from these different angles. Experimental shows the coverage strategy is systematic and effective.

# 4. References

[1] Levin MS, Last M, Design of test inputs and their sequences in multi-function system testing, Applied Intelligence, vol, 25-1: 1 ,p: 107-126,AUG 2006

[2] Paul C. Jorgensen, Software Testing ,A Craftsman's Approach (Second Edition) ,2003.

[3] Ng P, Fung RYK, Application of Formal Concept Analysis in Model-Based Testing ,Communications in Computer and Information Science v 30, p 110-123, 2009.

[4] Bandyopadhyay A, Ghosh S, Test Input Generation using UML Sequence and State Machines Models, SECOND INTERNATIONAL CONFERENCE ON SOFTWARE TESTING, VERIFICATION, AND VALIDATION, PROCEEDINGS, p 121-130, 2009.