## Contents

1. ABSTRACT .............................................................................................................................. 3
2. INTRODUCTION .................................................................................................................. 3
3. TEST PROCESS ..................................................................................................................... 5
   3.1 Basic parts of State Transition Diagram ........................................................................... 5
       3.1.1 How do we distinguish a state, an event, and an action? ........................................ 5
   3.2 State Transition Testing Model ....................................................................................... 6
       3.2.1 State Transition Diagrams (STDs) ............................................................................ 6
       3.2.2 State Transition Tree ............................................................................................... 8
           3.2.2.1 Coverage Criteria: ........................................................................................... 9
       3.2.3 State Transition Table ............................................................................................. 10
4. CASE STUDY ........................................................................................................................... 11
5. COMPARISON OF STATE TRANSITION TESTING WITH OTHER TESTING TECHNIQUES .... 15
   5.1 State Transition Diagrams v/s UML Activity Diagrams .................................................... 15
   5.2 State Transition Diagrams v/s Flow Charts ..................................................................... 16
6. SUMMARY .............................................................................................................................. 16
1 ABSTRACT

Software testing is the process of analyzing software to find the difference between required and existing condition. Software testing is performed throughout the development cycle of software and it is also performed to build quality software. For this purpose two basic testing approaches are used, they are White box testing and black box testing.

**White Box Testing** (also known as Clear Box Testing, Open Box Testing, Glass Box Testing, Transparent Box Testing, Code-Based Testing or Structural Testing) is a software testing method in which the internal structure/design/implementation of the item being tested is known to the tester. The tester chooses inputs to exercise paths through the code and determines the appropriate outputs. Programming know-how and the implementation knowledge is essential. White box testing is testing beyond the user interface and into the nitty-gritty of a system.

**Black-box testing**, also called behavioral testing, focuses on the functional requirements of the software. That is, black-box testing enables the software engineer to derive sets of input conditions that will fully exercise all functional requirements for a program.

There are various techniques in developing test cases for black box testing.

One such technique is explained in this White Paper.

2 INTRODUCTION

The Test Technique, State Transition Testing is helpful where you need to test different system transitions. It is a dynamic testing technique, which is used when the system is defined in terms of a finite number of states and the transitions between the states is governed by the rules of the system.

Any system where you get a different output for the same input, depending on what has happened before, is a finite state system. A finite state system is often shown as a state diagram.

As an example for the Finite State System, let us consider the ATM machine. The States are:

Start, Wait for Pin, 1st Try, 2nd Try, 3rd Try, Eat card and Access to Account.

Moving to one of these states depends on the Input given, here in this case, if the Pin is valid or not.
State Transition Testing is a black-box test design technique in which test cases are designed to execute valid and invalid state transitions.

This technique is useful when we have sequences of events that occur and conditions that apply to those events.
3 TEST PROCESS

3.1 Basic parts of State Transition Diagram

![State Transition Diagram](image)

FIGURE 2

- The **states** that the software may occupy (Eg: open/closed or funded/insufficient funds);
- The **transitions** from one state to another;
- The **events** that cause a transition (closing a file or withdrawing money);
- The **actions** that result from a transition (Eg: an error message or being given your cash).

3.1.1 How do we distinguish a state, an event, and an action?

**State**

- Persists until something external happens, usually triggering a transition
- A state can persist for an indefinite period

**Event**

- Occurs, either instantly or in a limited, finite period
- It is the something that happens
- The external occurrence that triggers the transition
Action

- The response of the system during the transition
- An action, like an event, is either instantaneous or requires a limited, finite period

3.2 State Transition Testing Model

The testing model typically has 3 entities

- State Transition Diagrams
- State Transition Trees
- State Transition Table

The above entities are explained with an example below.

3.2.1 State Transition Diagrams (STDs)

Software name – Manage_display_changes

The software responds to input requests to change display mode for a time display device.

The display mode can be set to one of the four values:

- Two corresponding to displaying either the time or date.
- The other two when altering either the time or the date.

<table>
<thead>
<tr>
<th>4 possible input requests</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change mode</td>
<td>Display date</td>
</tr>
<tr>
<td>Reset</td>
<td>Display Time</td>
</tr>
<tr>
<td>Time set</td>
<td>Alter Date</td>
</tr>
<tr>
<td>Date set</td>
<td>Alter Time</td>
</tr>
</tbody>
</table>
Different States:

- Display Time (S1),
- Change Time (S3),
- Display Date (S2) and
- Change Date (S4).

Explanation of the STD:

1: State Transition between S1 and S3: Let us start from the State S1 (Display Time). When there is an event, ‘Reset’, the machine transitions into the State S3 (Changing Time). If there is another event, ‘Time Set’ when the machine is in the State S3, then the state of the machine is changed to S1 (Display Time) again.

2: State Transition between S1 and S2: Consider the machine is in State S1 (Displaying Time) again. When there is an event, ‘Change Mode’, the transition happens to State S2 (Displaying Date) and the action is performed, ‘Display Date’. When machine in the State S2 (Displaying Date), undergoes an event, ‘change mode’ again, the transition happens to State S1.

3: State Transition between S2 and S4: Consider the machine is in State S2 (Displaying Date). When the event, ‘reset’ occurs, then the transition happen to the State S4 (Changing Date).
Again, if there is an event, ‘Date Set’, on the State S4 (Changing Date), the transition of the machine happens to State 2 (Displaying Date).

### 3.2.2 State Transition Tree

Once we have the state transition diagram, we can draw a State Transition Tree for this scenario.

![State Transition Tree Diagram](image)

**FIGURE 4**

In the above tree, we have taken S1 as the First node and the whole tree is drawn based on the diagram.

The tree is drawn up to level 1: So that each node is visited at least once and hence covers all the nodes.
This can be further extended based on the testing requirements. If the tester needs to revisit the nodes to check the consistency, then, the tree can be further extended to cover the same structure beyond once.

### 3.2.2.1 Coverage Criteria:
Various coverage criteria apply for state-based testing:

1. Visiting every state and traverse every transition
2. At least one test covers every row in a state transition table
3. At least one test covers each transition sequence of N or less length.

The N can be 1, 2, 3, 4, or higher. This is also called as "N-1 switch coverage".

Coverage depends on the size of N:
E.g.: If we cover all transitions of length one and two, then N-1 switch coverage means 1-switch coverage

Example for ‘N-1’Switch Coverage:

Consider the Below State Transition Diagram

![State Transition Diagram](image)

**FIGURE 5**
3.2.3 State Transition Table

The main part of the technique is to deduce to a table after drawing the STD. This is the table, where we can derive the test cases for the business process/scenario.

The table connects beginning states, events, and conditions with resulting states and actions.

Procedure for Deriving the Test Cases:
Setting a rule for where a test procedure or test step must start and where it may or must end

- E.g., a test step may start in an initial state and may only end in a final state
- The initial and final states can be the same
- Sequences of states and transitions that pass through the initial state more than once can be allowed

State Transition Table for the Software: Manage_display_changes (Based on STD in FIGURE 3)
CASE STUDY

The Case Study is constructed on a Banking System.

The scenario considered here is Disbursement Scenario & Payment Distribution Item Scenario

**Disbursement Scenario**

The statues of the Disbursement order is considered as different States for the State Transition Diagram

**Steps:**

1> Create Account with any LOAN product.
2> Create Disbursement order:
   a> If you will save the order it should be in ENTERED status.
   b> If you will delete the saved order it should be in DELETED status.
   c> If you will activate the order it should be in ACTIVATED status.
   d> If you will reverse the order it should be in REVERSED status.
   e> If you will close the account it should be in DEACTIVATED status.

**The State Transition Diagram for this Scenario is as below:**
State Transition Table for Test Cases:

<table>
<thead>
<tr>
<th>Test Case</th>
<th>TC1</th>
<th>TC2</th>
<th>TC3</th>
<th>TC4</th>
<th>TC5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start State</td>
<td>A</td>
<td>B</td>
<td>A</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>Input</td>
<td>Save</td>
<td>Delete</td>
<td>Activate</td>
<td>Reverse</td>
<td>Closure</td>
</tr>
<tr>
<td>Output</td>
<td>Entered</td>
<td>Deleted</td>
<td>Activated</td>
<td>Reversed</td>
<td>De-activated</td>
</tr>
<tr>
<td>Finish State</td>
<td>B</td>
<td>D</td>
<td>C</td>
<td>E</td>
<td>F</td>
</tr>
</tbody>
</table>

**Payment Distribution Item Scenario**

The statues of the Payment order is considered as different States for the State Transition Diagram
Steps:

1> Create Account with any LOAN product.
2> Replicate it to FICAx system.
3> Create Disbursement order.
4> Do the initial billing and initial settlement as day end task.
5> Check the first billing date.

Bank Posting Date: - First billing date

1> Do the first billing as day end task.

Bank Posting Date: - Next working day

1> Check the billing items in ACCOUNT MANAGEMENT system.
2> Check the open items in OPEN ITEM MANAGEMENT system (FICAx system).
3> Check the due date of repayment.

Bank Posting date: - Due date of repayment

1> Create payment item which is equal to sum of open items.
2> Check the payment distribution item status; it should be in NEW status.
3> Do the payment distribution run.
4> Check the payment distribution item status; it should be in WAITING status.
5> Check the open items in FICAx system, it should be cleared.
6> Do the FPOITR run in FICAx system.
7> Check the payment distribution item status; it should be in DONE status.
8> Do the settlement run.

Special Scenario for payment distribution item:-

1> If above created payment item goes to returned status, then the status of payment distribution item will change i.e. RETURNED.
2> If above created payment item goes to reversed status, then the status of payment distribution item will change i.e. REVERSED.
3> If any customer does not paid the actual amount i.e. sum of open items in FICAx system then on payment distribution run, the status of payment distribution item will change from NEW to POSTPROCESSING. When we process the payment distribution item from post control office then it goes to WAITING status.
4> In case of certain BUSINESS OPERATION the payment distribution item status changes from NEW to POSTPROCESSING and on processing of that payment distribution item it directly goes to DONE status as FPOITR triggered automatically.
State Transition Diagram for this Scenario is as below:

FIGURE 7
State Transition Table for Test Cases:

<table>
<thead>
<tr>
<th>Test Case</th>
<th>TC1</th>
<th>TC2</th>
<th>TC3</th>
<th>TC4</th>
<th>TC5</th>
<th>TC6</th>
<th>TC7</th>
<th>TC8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start State</td>
<td>A</td>
<td>A</td>
<td>B</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>D</td>
<td>Pay Item Reversed-PD Run</td>
</tr>
<tr>
<td>Input</td>
<td>Create Payment Item</td>
<td>PD Run</td>
<td>PD Run</td>
<td>FPOITR – Not Registered</td>
<td>FPOITR-Registered Automatically</td>
<td>FPOITR Run</td>
<td>Pay Item Returned-PD Run</td>
<td></td>
</tr>
<tr>
<td>Output</td>
<td>New Post Processing</td>
<td>Waiting</td>
<td>Waiting</td>
<td>Done</td>
<td>Done</td>
<td>Returned</td>
<td>Reversed</td>
<td></td>
</tr>
<tr>
<td>Finish State</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>C</td>
<td>D</td>
<td>D</td>
<td>E</td>
<td>F</td>
</tr>
</tbody>
</table>

**TABLE 4**

5 COMPARISON OF STATE TRANSITION TESTING WITH OTHER TESTING TECHNIQUES

5.1 State Transition Diagrams v/s UML Activity Diagrams

The basis of the State Transition Testing Technique is the State Transition Diagram.

Often people get confused between the State Transition Diagrams and the UML Activity Diagrams. But the intension of both these diagrams is different.

**Activity diagrams (In UML)** describe activities and **State charts** describe states. So those models are orthogonal - you might imagine there is an activity between two states (something that occurs during the transition) and you might say there is a state between two activities.

Activity diagram is used to document the logic of a single operation/method, a single use case or the flow of logic of a business process. It is equivalent to flowchart and data flow diagram from structured development.

The state diagram depict (show)the state of objects as their attributes change from state to the other state. State chart modeling is used to show the sequence of states that an object goes through, the cause the transition from one state to other and the action that result from a state change.
In short, in UML, the focus is more on the activities where as in State Transition diagrams; the focus is more on states.

5.2 State Transition Diagrams v/s Flow Charts

Both types of diagrams have the concept of ‘flow’ but state machine diagrams emphasize state. 'Flow' in a state diagram is always from state to state (or out of the diagram.) State machine diagrams describe a (typically) closed system composed of discrete states. The 'state' in this case determines how a system behaves to stimulus or events. The system described by a state diagram (a finite state machine) can behave completely differently to the same events depending on which state it is in. Because of this, the system has a kind of ‘memory’ due to how it moves from state to state.

On the other hand, a flowchart, emphasizes procedure. And, unlike a state machine diagram, there isn't a concept of state. The 'flow' of a flowchart is of a process. A flowchart shows the steps and actions to achieve a goal.

Note that the term ‘system’ is used here but terms 'subsystem' or 'entity' can also be used. State machine diagrams and flowcharts don't apply only at the system level. They can describe smaller pieces of an overall system.

6 SUMMARY

- State-Transition diagrams are excellent tools to capture certain system requirements, namely those that describe states and their associated transitions. These diagrams then can be used to direct our testing efforts by identifying the states, events, and transitions that should be tested. Together, these define how a system interacts with the outside world, the events it processes, and the valid and invalid order of these events.

- The advantage of a state-transition table is that it lists all possible state-transition combinations, not just the valid ones.

- When testing critical, high-risk systems such as avionics or medical devices, testing every state-transition pair may be required, including those that are not valid.

- Creating a state-transition table often unearths combinations that were not identified, documented, or dealt with in the requirements. It is highly beneficial to discover these defects.
- Using a state-transition table can help detect defects in implementation that enable invalid paths from one state to another.

- The generally recommended level of testing using state-transition diagrams is to create a set of test cases such that all transitions are exercised at least once under test.