BCS Oxfordshire

Thursday 6th October 2005

State transition testing

A practical workshop in the use of a software testing technique

WORKBOOK

Your name:	

Workshop presented by: Peter Quentin

QBIT
www.qbit.co.uk

contents

Introduction

the problem...

very large or infinite
number of test scenarios
+
finite amount of time
=
impossible to test everything

Most software systems that are subject to testing have an impossibly large number of operational scenarios to test. For example despite the fact that a device such as a mobile phone, has a finite number of functions, when it is in use these functions can be combined in an infinite number of ways.

If there is an infinite number of tests to be run a tester needs to find some way to reduce this number to a finite and manageable amount of work.

One solution is to prioritise all the tests that the tester is able to think of and then execute the most important ones. Testing stops when either time runs out or the tester believes the risk of failure once the system goes live is now sufficiently low.

Another solution is to use a software testing technique.

the solution...

Software test techniques exist to reduce the number of tests to be run whilst still providing sufficient coverage of the system under test

Software test techniques exist to reduce the number of tests to be run whilst still providing sufficient coverage of the system under test.

Another advantage of software testing techniques is that by following a systematic process creating tests becomes more efficient and there is less likelihood of making mistakes or missing important tests.

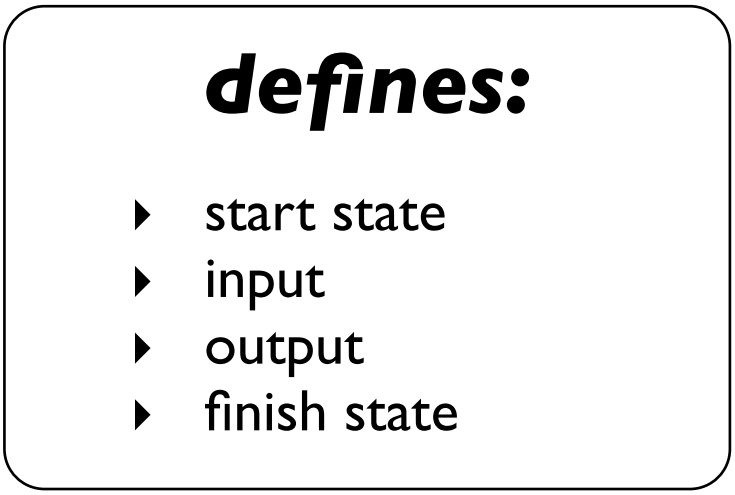
In practice both methods are combined, ie tests are created by using a software testing technique, the coverage level is determined according to the perceived risk of failure in live and then tests are prioritised so that the most important tests are executed first.

State transition testing

One such software testing technique is state transition testing.

State transition testing models the states that a system should exist in. It also models each state transition and for each state transition it defines the start state, the input that causes the state transition, the output of the state transition and the final state in which the system exists.

This information is derived from the specification of the system under test and not from reading or executing the code.

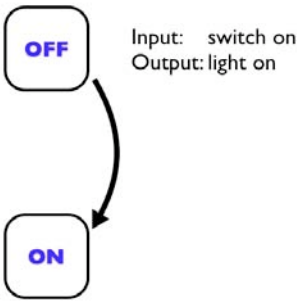


In the simplest implementation of state transition testing a test is defined for each state transition. The coverage that is achieved by this testing is called 0-switch or branch coverage. Better coverage is achieved by testing each state transition pair (sequence of two state transitions). This is called 1-switch or simply switch coverage. By creating a testing tree, as is explained below, tests are created to exercise sequences of state transitions, achieving these different levels of coverage, that when put together will reflect the operational scenarios.

Start

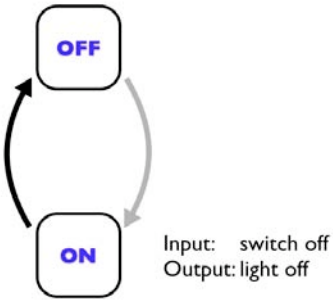
The initial state is off.

Step 1



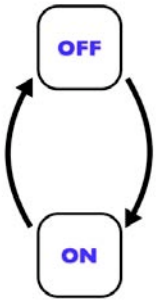
By giving the system an input, in this case switch it on, the system switches from the state of 'off' to the state of 'on'. In the process there is a measurable output, in this case the light comes on.

Step 2



In the state of 'on' the system can be given another input, it can be switched off, and the system returns to the state of off. Again there is a measurable output, the light is switched off.

The tests:

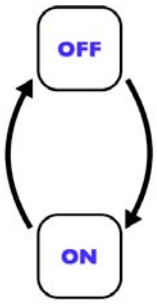


TEST 1	TEST 2
start state: **off**	start state: **on**
input: **switch on**	input: **switch off**
output: **light on**	output: **light off**
finish state: **on**	finish state: **off**

Two tests can therefore be defined:

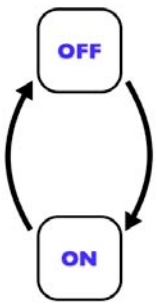
Test 1 - 'off' to 'on'
Test 2 - 'on' to 'off'

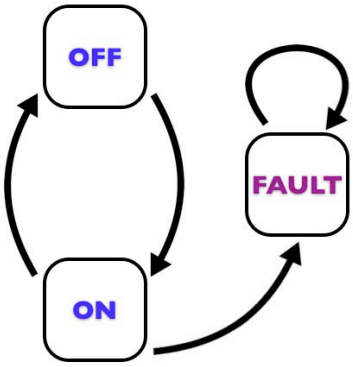
But what if...



TEST 1	TEST 2
start state: **off**	start state: **on**
input: **switch on**	input: **switch off**
output: **light on**	output: **light off**
finish state: **on**	finish state: **off**

Once the two tests have been executed, is there any way of knowing that if test 1 was repeated the result would be the same?

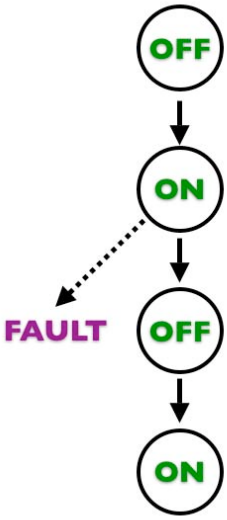




What actually happened is that during the second test the system developed a fault. This meant that it never entered the state of off, instead it entered a new state. This state has been called 'fault'.

The trouble is that the measured output entering the state of 'fault' appears to be the same as entering the state of 'off'.

Hence the fault was not detected until some time later.



A testing tree can be drawn to represent a better, single test that would have detected this fault.

TEST

	STEP 1	STEP 2	STEP 3
START STATE	OFF	ON	OFF
INPUT	SWITCH ON	SWITCH OFF	SWITCH ON
OUTPUT	LIGHT ON	LIGHT OFF	LIGHT ON
FINISH STATE	ON	OFF	ON

This test would have three steps exercising three state transitions.

This single but more complex test provides better coverage.

Example

electronic clock

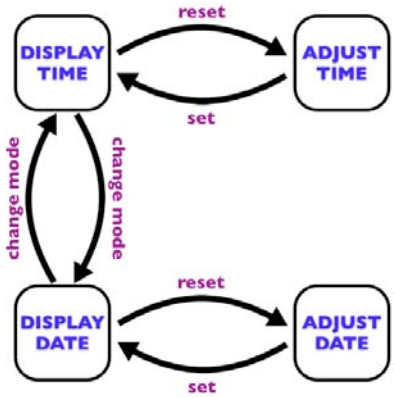
A simple electronic clock has four modes, *display time*, *change time*, *display date* and *change date*

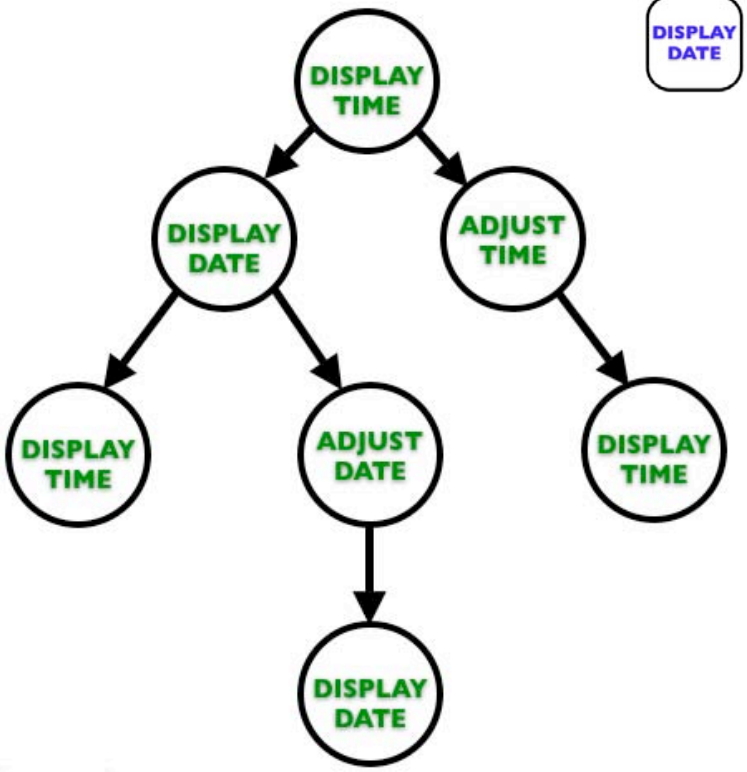
The *change mode* button switches between display time and display date

The *reset* button switches from *display time* to *adjust time* or *display date* to *adjust date*

The *set* button returns from *adjust time* to *display time* or *adjust date* to *display date*

0-switch or branch coverage





TEST 1

	STEP 1	STEP 2
START STATE	DISPLAY TIME	DISPLAY DATE
INPUT	CHANGE MODE	CHANGE MODE
OUTPUT	DISPLAY DATE	DISPLAY TIME
FINISH STATE	DISPLAY DATE	DISPLAY TIME

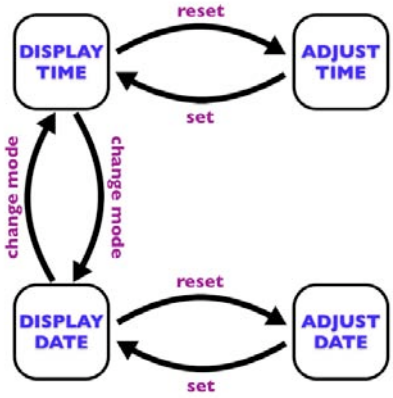
TEST 2

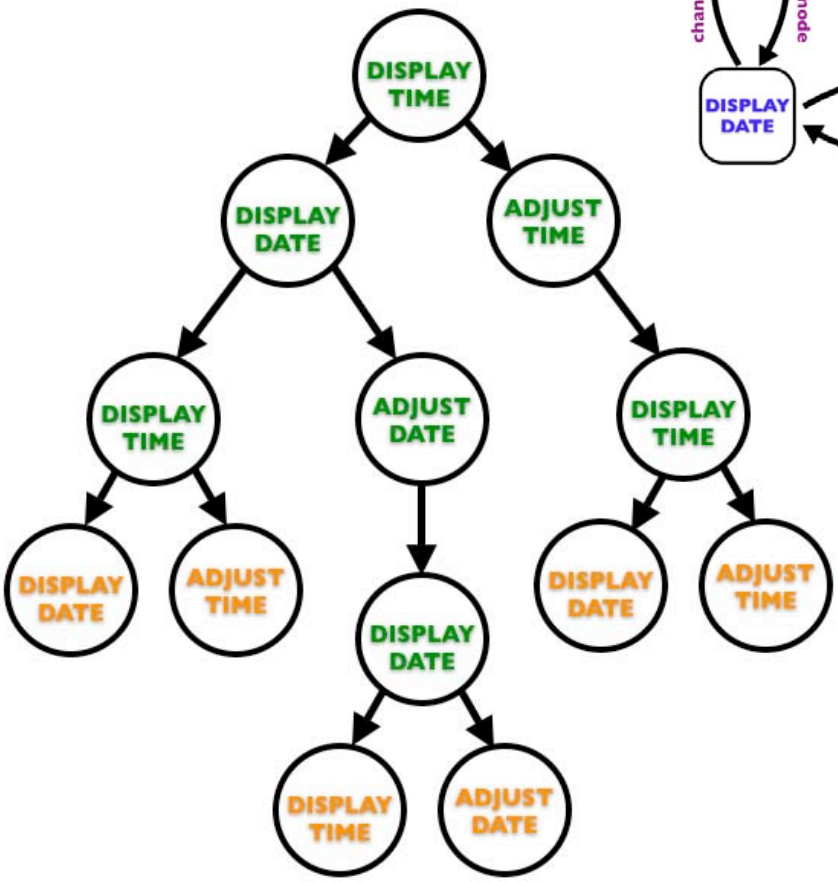
	STEP 1	STEP 2	STEP 3
START STATE	DISPLAY TIME	DISPLAY DATE	ADJUST DATE
INPUT	CHANGE MODE	RESET	SET
OUTPUT	DISPLAY DATE	ADJUST DATE	DISPLAY DATE
FINISH STATE	DISPLAY DATE	ADJUST DATE	DISPLAY DATE

TEST 3

	STEP 1	STEP 2
START STATE	DISPLAY TIME	ADJUST TIME
INPUT	RESET	SET
OUTPUT	ADJUST TIME	DISPLAY TIME
FINISH STATE	ADJUST TIME	DISPLAY TIME

1-switch or switch coverage





TEST 1

	STEP 1	STEP 2	STEP 3
START STATE	DISPLAY TIME	DISPLAY DATE	DISPLAY TIME
INPUT	CHANGE MODE	CHANGE MODE	CHANGE MODE
OUTPUT	DISPLAY DATE	DISPLAY TIME	DISPLAY DATE
FINISH STATE	DISPLAY DATE	DISPLAY TIME	DISPLAY DATE

TEST 2

	STEP 1	STEP 2	STEP 3
START STATE	DISPLAY TIME	DISPLAY DATE	DISPLAY TIME
INPUT	CHANGE MODE	CHANGE MODE	RESET
OUTPUT	DISPLAY DATE	DISPLAY TIME	ADJUST TIME
FINISH STATE	DISPLAY DATE	DISPLAY TIME	ADJUST TIME

TEST 3

	STEP 1	STEP 2	STEP 3	STEP 4
START STATE	DISPLAY TIME	DISPLAY DATE	ADJUST DATE	DISPLAY DATE
INPUT	CHANGE MODE	RESET	SET	CHANGE MODE
OUTPUT	DISPLAY DATE	ADJUST DATE	DISPLAY DATE	DISPLAY TIME
FINISH STATE	DISPLAY DATE	ADJUST DATE	DISPLAY DATE	DISPLAY TIME

TEST 4

	STEP 1	STEP 2	STEP 3	STEP 4
START STATE	DISPLAY TIME	DISPLAY DATE	ADJUST DATE	DISPLAY DATE
INPUT	CHANGE MODE	RESET	SET	RESET
OUTPUT	DISPLAY DATE	ADJUST DATE	DISPLAY DATE	ADJUST DATE
FINISH STATE	DISPLAY DATE	ADJUST DATE	DISPLAY DATE	ADJUST DATE

TEST 5

	STEP 1	STEP 2	STEP 3
START STATE	DISPLAY TIME	ADJUST TIME	DISPLAY TIME
INPUT	RESET	SET	CHANGE MODE
OUTPUT	ADJUST TIME	DISPLAY TIME	DISPLAY DATE
FINISH STATE	ADJUST TIME	DISPLAY TIME	DISPLAY DATE

TEST 6

	STEP 1	STEP 2	STEP 3
START STATE	DISPLAY TIME	ADJUST TIME	DISPLAY TIME
INPUT	RESET	SET	RESET
OUTPUT	ADJUST TIME	DISPLAY TIME	ADJUST TIME
FINISH STATE	ADJUST TIME	DISPLAY TIME	ADJUST TIME

n-switch or boundary interior

- execute each loop n times
- at least twice
- tests each loop irrespective of the start/end point
- 6 loops to be tested in this case

Exercises

for each exercise...

- Draw a state transition diagram
  - mark the state transitions
  - define input and output for each state transition
- Determine the level of coverage
- Draw a testing tree
- Define the tests

electric toothbrush

A two-speed electric toothbrush is operated by pressing its one button. The first press of the button turns the toothbrush from off to speed one, the second press of the button turns it to speed two

When the button is pressed for a third time the electric toothbrush is turned off

electric toothbrush exercise continued...

tape cassette player

A tape player has three operations: play, fast forward and fast play.

Play and fast forward are activated using the play and fast forward buttons respectively. These operations can be cancelled using the stop button.

When in play mode, the fast forward can be used to fast play.

When in fast play mode, the fast forward button can be pressed again to enter fast forward or the stop button can be used to return to play.

When in fast forward the play button can be pressed to enter play mode directly.

tape cassette player exercise continued...

Conclusions...

There are many useful techniques available and state transition testing is only one of many. State transition testing allows a finite number of tests to be defined from an infinite number of test scenarios and a level of coverage can still be achieved.

The process for state transition testing is:

- draw state transition diagram
- determine start state, input, output and finish state
- determine coverage level to be achieved
- draw testing tree
- define tests

Additional notes

Contact details

Peter Quentin

peter@qbit.co.uk

QBIT
1-9 Memel Street
London
EC1Y 0UT

+44 (0) 20 7014 8950
www.qbit.co.uk

References

BS 7925-2:1998 Software testing. Software component testing

Testing Software Design Modeled by Finite-State Machines - Tsun S. Chow
IEEE Transactions on Software Engineering, Vol. SE-4, No. 3, May 197