Graph Theory Techniques in Model-Based S/W Testing

Index

1. What is S/W testing?
2. What is modeling ?
3. What's Model-Based Testing?
4. Example : Switch Cover
5. How "Hard" is a software to test?
6. Metric : McCabe's Cyclomatic Complexity
7. Is CC is a right metric?
8. Proposing Some Addition to CC

What is S/W testing?

Testing is process of executing S/W system to ensure that it works according to its specifications of design.

Let us see importance and depth of S/W testing?

Ariane 5 rocket was a Project of European Space Agency

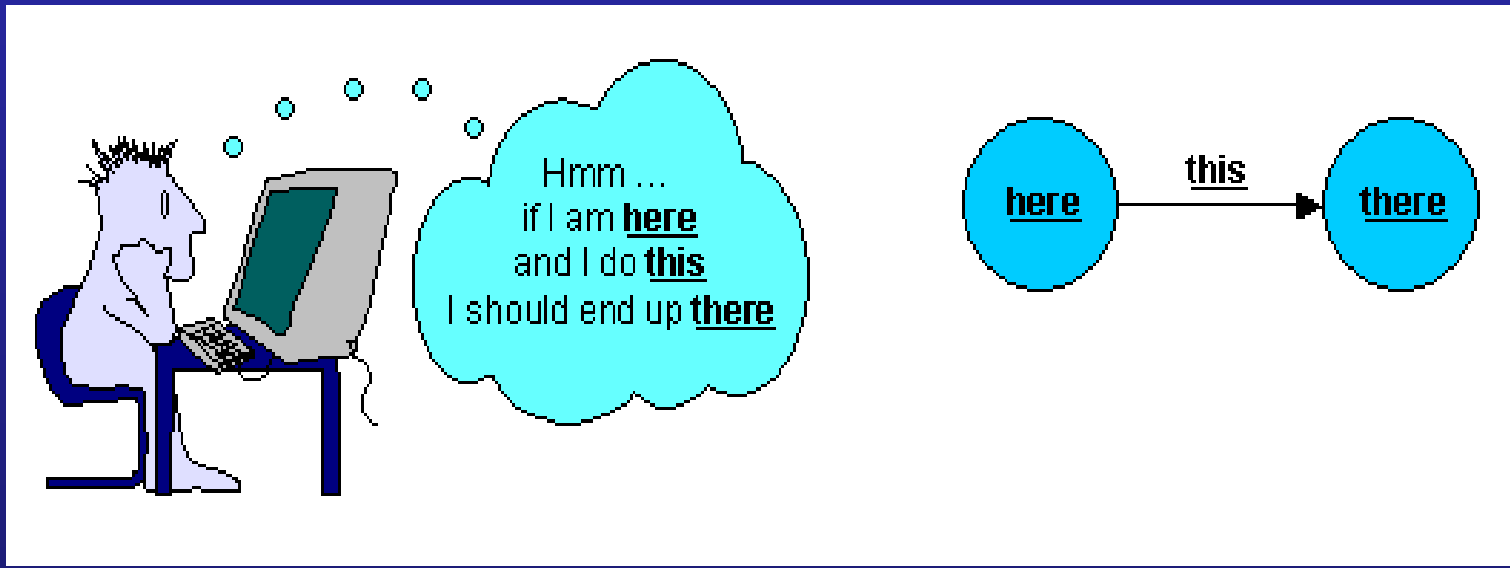
- A decade Hard work of 100s of eminent scientists
- development costing $7 billion
- cargo were valued at $500 million

Ariane 5 – June 4, 1996

- It turned out that the cause of the failure was a software error.
  Specifically a 64 bit floating point number relating to the horizontal velocity of the rocket with respect to the platform was converted to a 16 bit signed integer.The number was larger than 32,767, the largest integer storeable in a 16 bit signed integer, and thus the conversion failed.

What's Modeling?

Modeling is a way of representing the behaviour of a system. Models are simpler than the system they describe, and they help us understand and predict the system's behaviour.



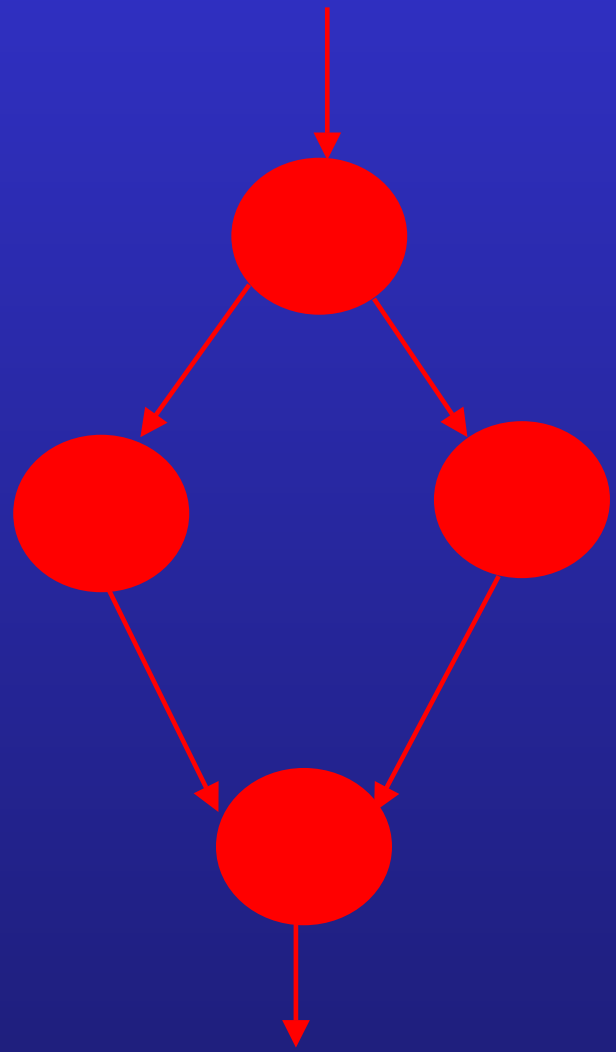
Models & S/W behaviour

Abstract

Models are a method of representing software behaviour. Graph theory is an area of mathematics that can help us use this model information to test applications in many different ways. This paper describes several graph theory techniques, where they came from, and how they can be used to improve software testing.

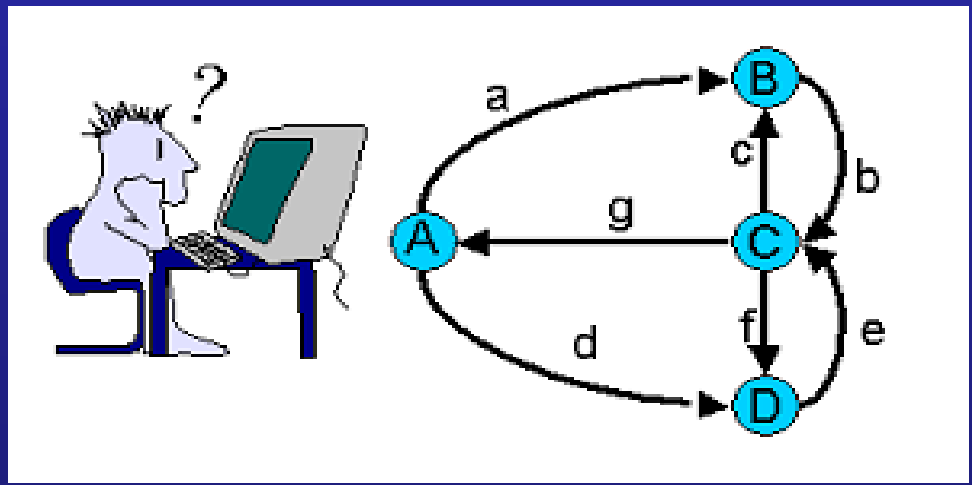
S/W to Graph

```
If (sal<=10000)
    {
     sal=1.05*sal;
     countA++;
    }
Else
    {
     countB++;
   }
```



A Testing Problem:

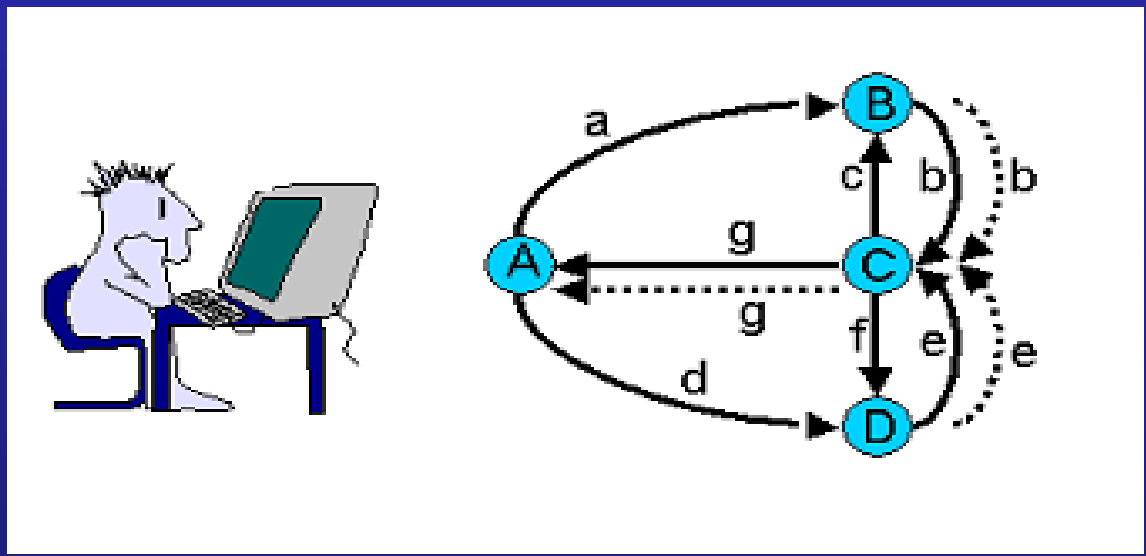
Suppose you are a tester and you have a behavioral model of some software you would like to test. And suppose further that the model looks like the left hand digraph in Figure , where the nodes represent the states of the application and the arcs are the actions you can perform



But how can you do it in an efficient manner?

Since you already have the behavioural model available to you as a graph, you can apply the appropriate graph algorithm to generate an efficient graph traversal automatically. In the example in Figure , our tester could test all available actions as follows:

a b c b f e g d e g



Switch Cover

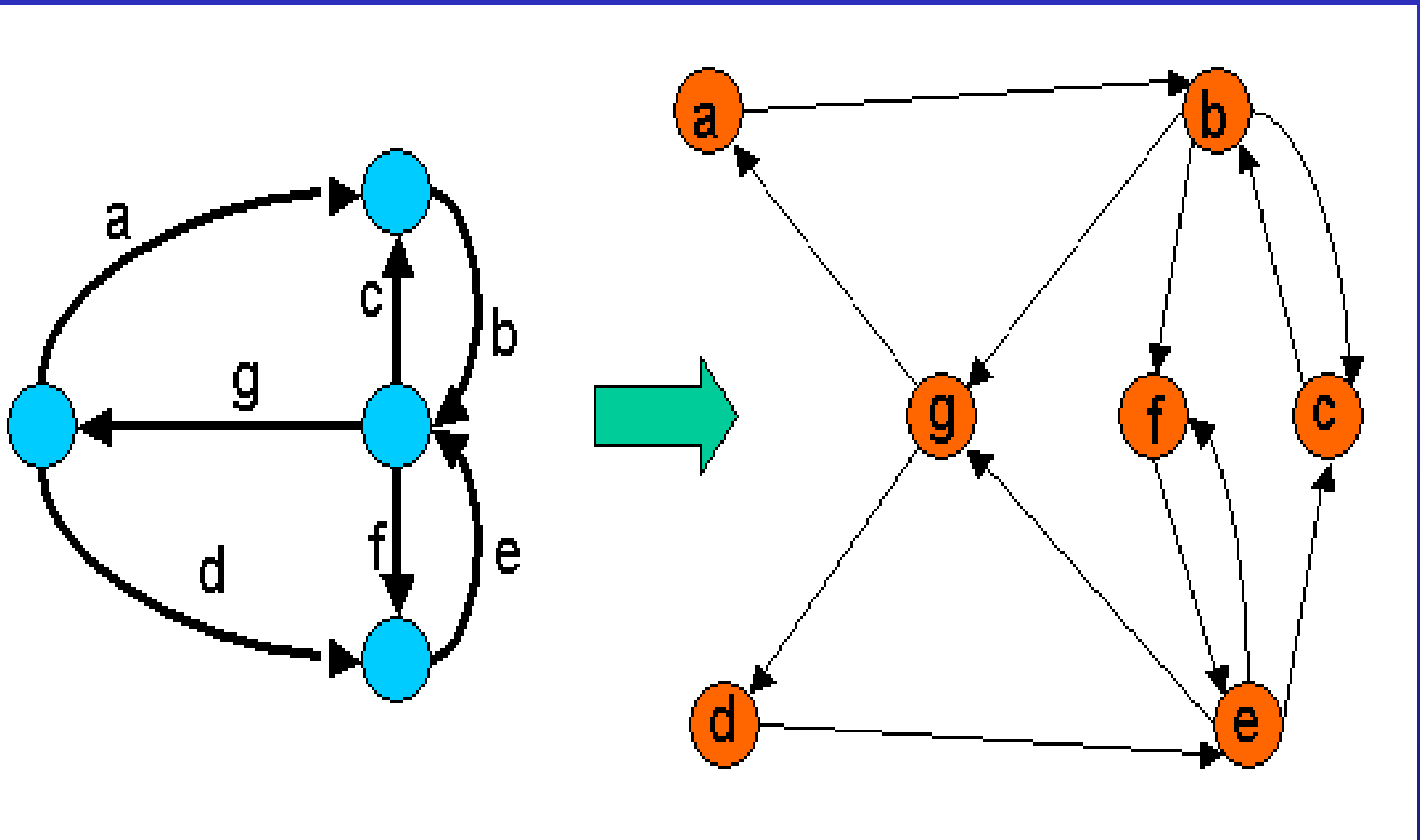
The approach of testing combinations of actions is called "switch cover" in finite state machine testing

The de Bruijn sequence for combinations Can be obtained as follows:

1. Create a dual graph of the original graph (i.e., a graph in which the arcs of the original graph are converted to nodes)
2. Everywhere in the original graph that arc 1 is an incoming to a node and arc 2 is outgoing from the same node, create an arc in the dual graph from node 1 to node 2.

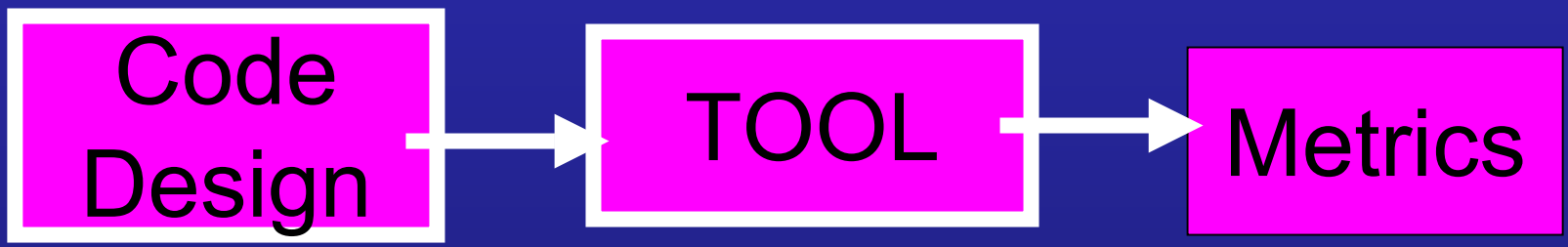
3. Eulerize the dual graph (by duplicating arcs to balance node polarities)
4. Traverse the Eulerized graph, test path is noting the names of the nodes that you pass.

Ex. Dual of Graph



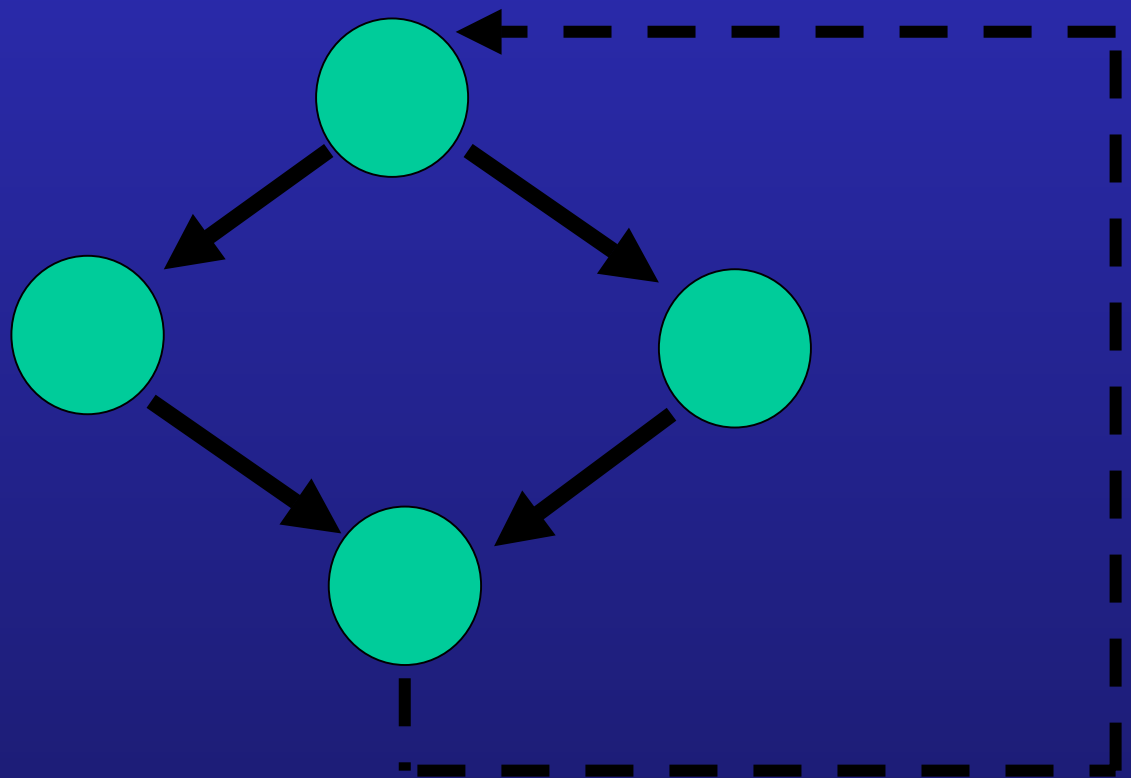
Switch cover : a b c b f e c b g d e f e g

How "Hard" is a software to test?
The metric showing complexity of a software. The tool converts a design to a metric on the basis of Which we can compare two software.



McCabe's Cyclomatic Complexity

- Based on Graph Theory
- Assume each program is a strongly connected graph



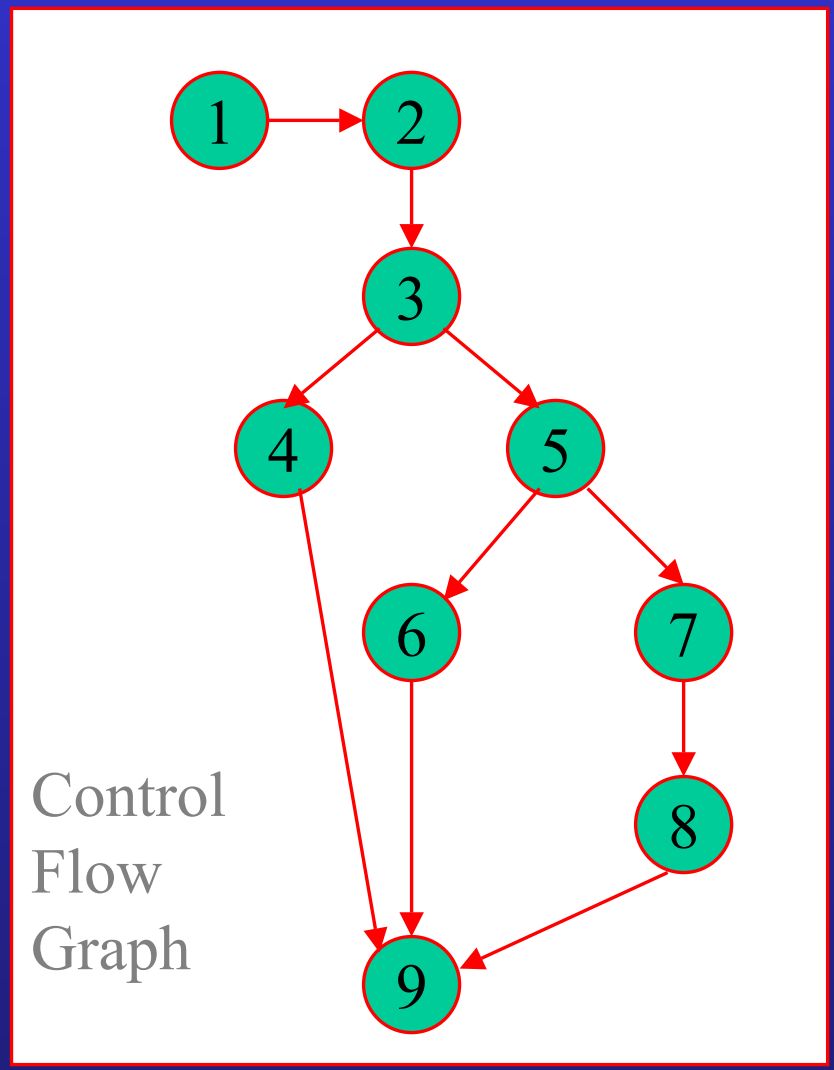
The Formulae:

CC = Edges – Nodes + 2

CC = Decisions + 1

For the graph shown

CC=3



Q: Is this a right way to calculate Complexity and Make Test Scenarios for Software?

Ans: It is simple so seems to be good from point of view of understanding but what I personally feel that it fails in following situations shown in the following slides:

(A)Different complexity still mapped to same complexity level.

(B) Data Structure and operation on them are not considered.

(C) Data Reference Mechanisms are not considered.

(A)The statements are equally complex according to CC but actually they are not:

- If (X==1)
- If (C)
- If (++(C==D++))
- If (C&&(C||D))

(B) It doesn't consider the complexity in the understanding of the Data structures and operations on them.

according to CC all data structures are equally complex.

For example:

```
Emp a
{
Int id;
Char name[100];
Float salary;
};
```

Is as complex as an integer.

"If the above proposed notion of testing was considered then Ariane might have not been crashed."

References

Apfelbaum, L. (1997) "Model-Based Testing", **Proceedings of Software Quality Week 1997**

Beizer, B. (1990) **Software Testing Techniques,** 2nd Edition

Beizer, B. (1995) **Black Box Testing**

Thanking You

What's Model-Based Testing?

In recent years, there has been a growing movement in software testing to use the information contained in explicit models of software behavior to make it simpler and cheaper to do testing. (Beizer 1995, Apfelbaum 1997)

Model-based testing is a black-box technique that offers many advantages over traditional testing:

- Constructing the behavioural models can begin early in the development cycle.
- Modeling exposes ambiguities in the specification and design of the software.
- The model embodies behavioural information that can be re-used in future testing, even when the specifications change.
- The model is easier to update than a suite of individual tests.

What's Wrong with Traditional Software Testing?

Traditional software testing consists of the tester studying the software system and then writing and executing individual test scenarios that exercise the system. These scenarios are individually crafted and then can be executed either manually or by some form of capture/playback test tool.