ST Assignment No.2

Q1. Describe Black Box Testing & white box testing.

Ans.

- ***Black Box Testing***

1. This approach tests all possible combinations of end-user actions.

2. Black box testing assumes no knowledge of code and is intended to simulate the end-user experience.

3. The tester can use sample applications to integrate and test the application blockfor black box testing.

4. Planning for black box testing immediately after the requirements and the functional specifications are available.

5. Black box testing should be conducted in a test environment close to the target environment.

- ***White box testing***

1. This is also known as glass box, clear box, and open box testing.

2. In white box testing, test cases are created by looking at the code to detect any potential failure scenarios.

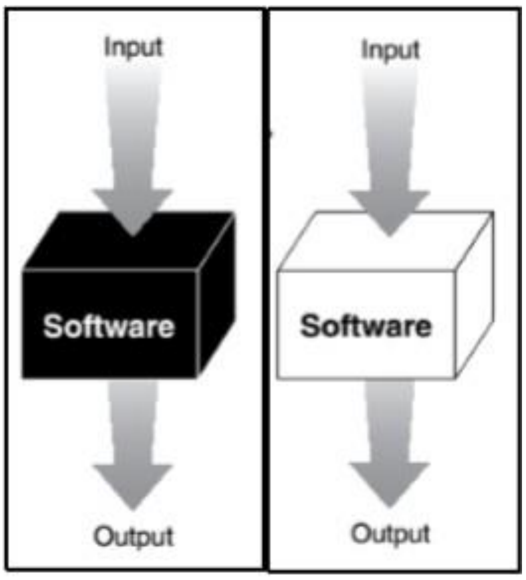
3. The suitable input data for testing various APIs and the special code paths that need to be tested by analysing the source code for the application block.

4. Therefore, the test plans need to be updated before starting white box testing and only after a stable build of the code is available.

5. White box testing assumes that the tester can take a look at the code for the application block and create test cases that look for any potential failure scenarios.

6. During white box testing, analyze the code of the application block and prepare test cases for testing the functionality to ensure that the class is behaving in accordance with the specifications and testing for robustness.

7. A failure of a white box test may result in a change that requires all black box testing to be repeated and white box testing paths to be reviewed and possibly changed.



Black box testing White Box Testing

Q2. Explain static testing and dynamic testing.

Ans.

1. *Dynamic testing is testing that is performed when the system is running.*
2. *The basic requirement is to review test plans.*
3. *Recommend tests based on the hazard analyses, safety standards and checklists, previous accident and incidents, operator task analyses etc.*
4. *Specify the conditions under which the test will be conducted.*
5. *Review the test results for any safety-related problems that were missed in the analysis or in any other testing.*
6. *Ensure that the testing feedback is integrated into the safety reviews and analyses that will be used in design modifications*
7. *Static testing is performed when the system is not running.*
8. *Static testing works with peer review, and mostly referred to as pen and pencil run.*

Q3. Explain the concept of data testing in dynamic black box testing.

Ans.

1. *The simplest view of software is to divide its world into two parts: the data and the program.*
2. *The data is the keyboard input, mouse clicks, disk files, printouts, and so on.*

3. The program is the executable flow, transitions, logic, and computations.

4. When software testing is performed on the data, the user information is checked and the data is tabulated with the expected results.
Examples of data would be:

•The words you type into a word processor

•The numbers entered into a spreadsheet

•The number of shots you have remaining in your space game

•The picture printed by your photo software

•The backup files stored on your floppy disk
•The data being sent by your modem over the phone lines

Q4. Explain the concept of boundary conditions and sub boundary conditions in dynamic black box testing.

Ans.

a) ***Boundary Condition***

1. Boundary conditions are special because programming, by its nature, is susceptible to problems at its edges.

2. The boundary conditions are defined as the initial and the final data ranges of the variables declared.

3. If an operation is performed on a range of numbers, odds are the programmer gotit right for the vast majority of the numbers in the middle, but maybe made a mistakeat the edges.

4. The edges are the minimum and the maximum values for that identifier.

For example

1. #include<stdio.h>

2. void main()

3. {

4. int i , fact=1, n;

5. printf("enter the number ");

6. scanf("%d",&n); 7.

for(i =1 ;i <=n;i++)

*8. fact = fact * i;*

9. printf (“the factorial of a number is ▫”%d”, fact);

10. }

The boundary condition in the above example is for the integer variable.

b) Sub-Boundary Conditions

1. *They’re the ones defined in the specification or evident when using the software.*
2. *Some boundaries, though, that are internal to the software aren’t necessarily apparent to an end user but still need to be checked by the software tester.*
3. *These are known as sub-boundary conditions or internal boundary conditions.*
4. *In the given example the sub boundary condition is the value of factorial*

For example

```
1. #include<stdio.h>
2. void
main()3. {
4. int i , fact=1, n;
5. printf(“enter the number “);
6. scanf(“%d”,&n);
7. for(i =1 ;i
<=n;i++)
8. fact = fact * i;
9. printf (“the factorial of a number is ▫”%d”, fact);
10. }
```

Q5. Explain the concept of state testing in dynamic black box testing.

Ans.

1. *The data gets tested on the numbers, words, inputs, and outputs of the software.*
2. *The product or the software should be tested for the program’s logic flow through*

its various states.

3. A software state is a condition or mode that the software is currently in.

4. Consider Figures 2.2, which illustrates the software mode of the paint software in airbrush mode.

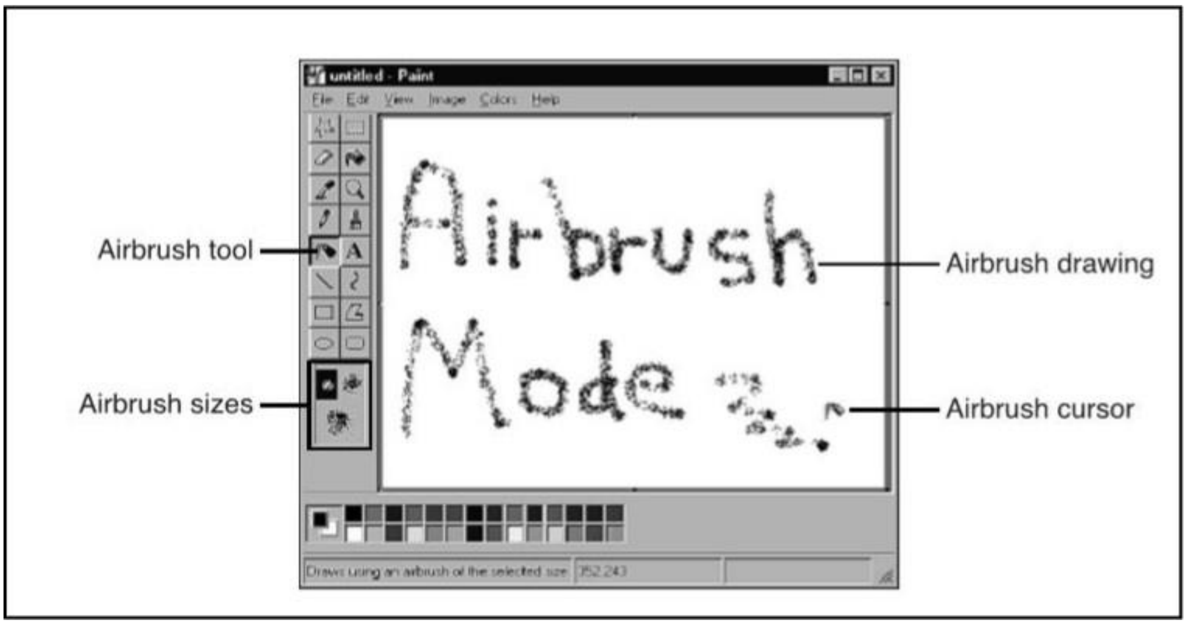


Figure 2.2
a)

Testing Software's Logic Flow

1. Testing the software's states and logic flow has the same problems. It's usually possible to visit

2. Check for the all possible states for test to pass condition

3. The complexity of the software, especially due to the richness of today's user interfaces, provides so many choices and options that the number of paths grows exponentially.

Q6. Define in short: Repetition testing , stress, load.

Ans.

1. *The software fails under are repetition, stress, and load.*
2. *These tests target state handling problems where the programmer didn't consider what might happen in the worst-case scenarios.*
3. *Repetition testing involves doing the same operation over and over.*
4. *This could be as simple as starting up and shutting down the program over*

and over.

5. It could also mean repeatedly saving and loading data or repeatedly selecting the same operation.

6. A bug can be encountered after only a couple repetitions or it might take thousands of attempts to reveal a problem.

7. The repetition testing is done to look for memory leaks.

8. A common software problem happens when computer memory is allocated to perform a certain operation but isn't completely freed when the operation completes.

9. The result is that eventually the program uses up memory that it depends on to work reliably.

10. Stress testing is running the software under less-than-ideal conditions such as low memory, low disk space, slow CPUs, slow modems.

11. The software should be tested for external resources and dependencies it has on them.

12. Stress testing is simply limiting them to their bare minimum.

Q7. Explain concept of creating state transition map.
Ans.

1. There are various techniques for state transition diagrams.

2. Figure 2.3 shows two examples. One uses boxes and arrows and the other uses circles (bubbles) and arrows.

3. The sate transition diagram should be easily understandable and should be able to explain clearly about the various stages the software passes through.

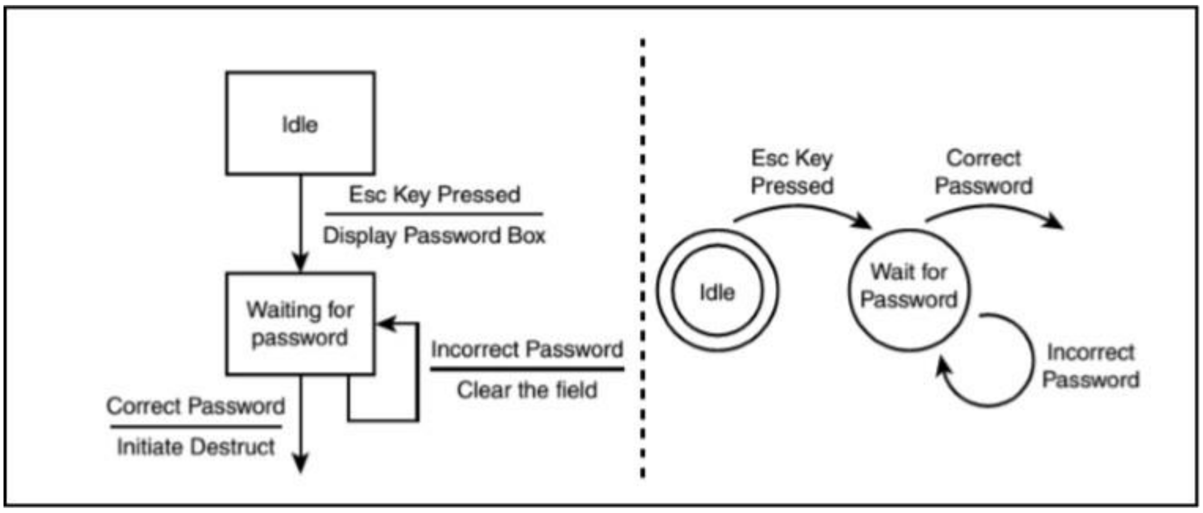


Figure 2.3

A state transition map should show the following items:

1. *Each unique state that the software can be in.*
2. *A good rule of thumb is that if the state is unsure whether something is a separate state, it probably is.*
3. *Always collapse it into another state if you find out later that it isn't.*
4. *The input or condition that takes it from one state to the next.*
5. *This might be a key press, a menu selection, a sensor input, a telephone ring, and so on.*
6. *A state can't be exited without some reason. The specific reason is what you're looking for here.*
7. *Set conditions and produced output when a state is entered or exited.*
8. *This would include a menu and buttons being displayed, a flag being set, a printout occurring, a calculation being performed, and so on.*
9. *It's anything and everything that happens on the transition from one state to the next.*