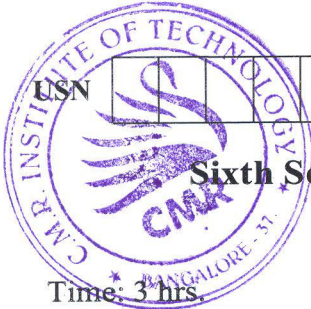


# CBCGS SCHEME



18IS62

Sixth Semester B.E. Degree Examination, June/July 2023

## Software Testing

Time: 3 hrs.

Max. Marks: 100

Note: Answer any FIVE full questions, choosing ONE full question from each module.

### Module-1

- 1 a. Explain program behavior insights from a Venn Diagram for functional testing and structural testing. (10 Marks)
- b. Identify and explain fault taxonomies with example. (10 Marks)

OR

- 2 a. With the flowchart for the traditional triangle problem implementation. (10 Marks)
- b. Analyse and explain the SATM screen. (10 Marks)

### Module-2

- 3 a. Write a program to solve the triangle problem. Derive test cases for program based on boundary value analysis. (10 Marks)
- b. Write a program to solve commission problem. Analyze it from the perspective of equivalence class testing and derive the test cases. (10 Marks)

OR

- 4 a. Write a program to solve the triangle problem. Derive test cases for program based on decision table approach. (10 Marks)
- b. List the assumptions made in fault based testing and explain the mutation analysis with sample program. (10 Marks)

### Module-3

- 5 a. Analyze and explain metric – based testing. (10 Marks)
- b. Explain define/Use testing with example. (10 Marks)

CMRIT LIBRARY  
BANGALORE - 560 037

OR

- 6 a. Describe about scaffolding. Discuss about Generic versus specific scaffolding. (08 Marks)
- b. Define : (12 Marks)
  - i) Test oracles
  - ii) Self – checks
  - iii) Capture
  - iv) Replay.

### Module-4

- 7 a. Explain the basic principles in the frame work for test and analysis. (12 Marks)
- b. List and explain the dependability properties test and analysis actives. (08 Marks)

OR

- 8 a. Explain Software Reliability Engineered Testing (SRET) approach with diagram. (10 Marks)
- b. Identify and explain risk management in quality plan with respect to generic and specific issues. (10 Marks)

18IS62

Module-5

- 9 a. Analyze and explain integration testing strategies. (10 Marks)  
b. What is regression testing? Explain regression test selection technique. (10 Marks)
- OR
- 10 a. Explain Rapid Prototyping Life Cycle with diagram. (10 Marks)  
b. Explain Decomposition – Based Integration. (10 Marks)

\*\*\*\*\*

**VTU Examination – June/July 2023  
Solution**

Sub:	<b>Software Testing</b>				Sub Code:	<b>18IS62</b>	Branch:	<b>ISE</b>
Exam Date:	<b>28/07/2023</b>	Duration:	<b>3 Hrs</b>	Max Marks:	<b>100</b>	Sem	<b>VI</b>	

**Answer any FIVE FULL Questions**

MARKS    CO

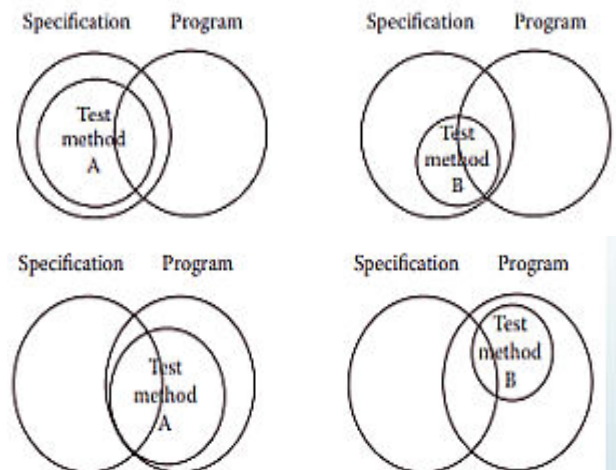
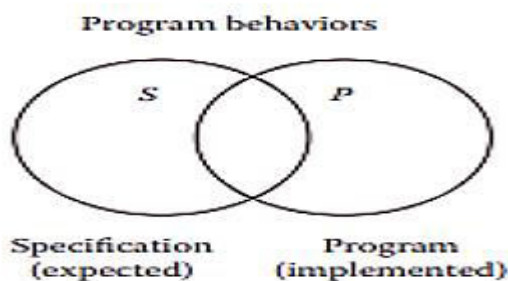
**MODULE-1**

- 1 (a) **Explain program behavior insights from a Venn Diagram for functional testing and structural testing.**

**Solution:**

**Insights from a Venn diagram**

- Testing is fundamentally concerned with behavior, and behavior is orthogonal to the structural view common to software (and system) developers.
- A quick distinction is that the structural view focuses on what it is and the behavioral view considers what it does.
- Consider a universe of program behaviors.
- Given a program and its specification, consider the set  $S$  of specified behaviors and the set  $P$  of programmed behaviors.



[10]    CO1

- (b) **Identify and explain fault taxonomies with example.**

**Solution:**

- Definitions of error and fault hinge on the distinction between process and product: process refers to how we do something, and product is the end result of a process.
- The point at which testing and Software Quality Assurance (SQA) meet is that SQA typically tries to improve the product by improving the process.

[10]    CO1

Table 1.1 Input/Output Faults

Type	Instances
Input	Correct input not accepted
	Incorrect input accepted
	Description wrong or missing
	Parameters wrong or missing
Output	Wrong format
	Wrong result
	Correct result at wrong time (too early, too late)
	Incomplete or missing result
	Spurious result
	Spelling/grammar
	Cosmetic

Table 1.2 Logic Faults

Missing case(s)
Duplicate case(s)
Extreme condition neglected
Misinterpretation
Missing condition
Extraneous condition(s)
Test of wrong variable
Incorrect loop iteration
Wrong operator (e.g., < instead of ≤)

Table 1.3 Computation Faults

Incorrect algorithm
Missing computation
Incorrect operand
Incorrect operation
Parenthesis error
Insufficient precision (round-off, truncation)
Wrong built-in function

Table 1.5 Data Faults

Incorrect initialization
Incorrect storage/access
Wrong flag/index value
Incorrect packing/unpacking
Wrong variable used
Wrong data reference
Scaling or units error
Incorrect data dimension
Incorrect subscript
Incorrect type
Incorrect data scope
Sensor data out of limits
Off by one
Inconsistent data

Table 1.4 Interface Faults

Incorrect interrupt handling
I/O timing
Call to wrong procedure
Call to nonexistent procedure
Parameter mismatch (type, number)
Incompatible types
Superfluous inclusion

OR

2 (a) With the flowchart for the traditional triangle problem implementation.

[10]

CO1

Solution:

Program triangle1 'Fortran-like version

Dim a, b, c, match As INTEGER

Output("Enter 3 integers which are sides of a triangle")

Input(a,b,c)

Output("Side A is ",a)

Output("Side B is ",b)

Output("Side C is ",c)

match = 0

If a = b Then match = match + 1

EndIf

If a = c Then match = match + 2

EndIf

If b = c Then match = match + 3

EndIf

If match = 0

Then If (a+b) <= c

Then Output("NotATriangle")

Else If (b+c) <= a

Then Output("NotATriangle")

Else If (a+c) <= b

Then Output("NotATriangle")

Else Output("Scalene")

EndIf

Else If match=1

Then If (a+c) <= b

Then Output("NotATriangle")

Else Output("Isosceles")

EndIf

Else If match=2

Then If (a+c) <= b

Then Output("NotATriangle")

Else Output("Isosceles")

EndIf

Else If match=3

Then If (b+c) <= a

Then Output("NotATriangle") (18)

Else Output("Isosceles") (19)

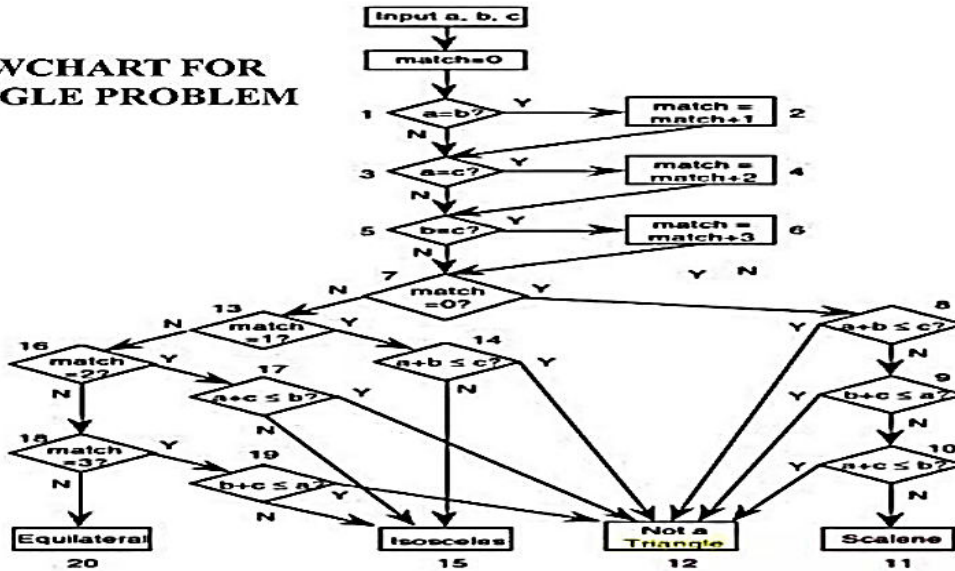
Else Output("Equilateral") (20)

EndIf

EndIf

EndIf

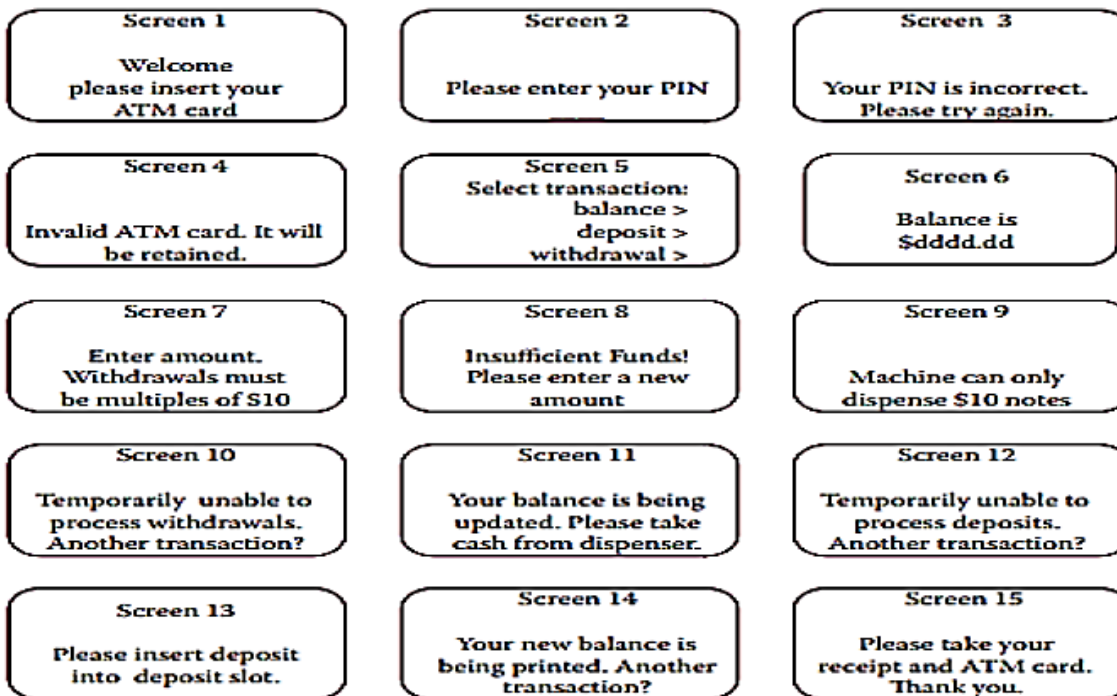
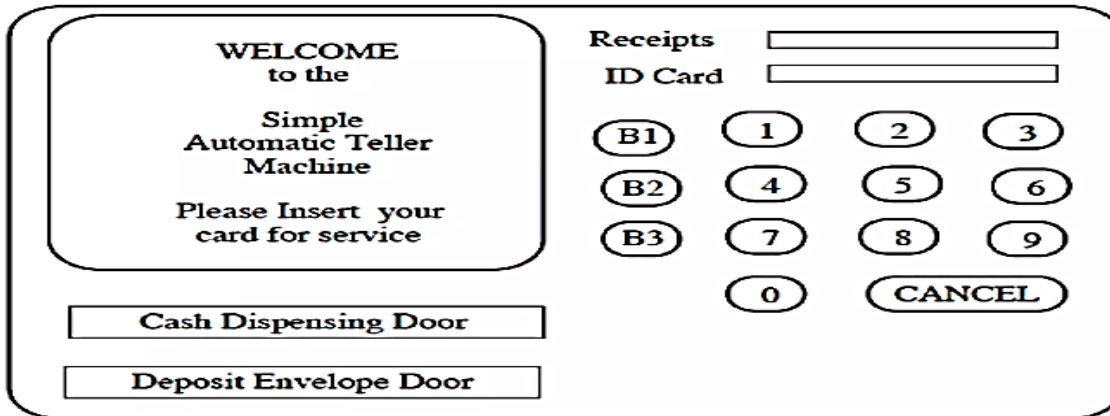
**FLOWCHART FOR TRIANGLE PROBLEM**



(b) Analyze and explain the SATM screen.

Solution:

**THE SATM TERMINAL**



[10]

CO1

**MODULE-2**

3 (a) Write a program to solve the triangle problem. Derive test cases for program based on boundary value analysis.

[10]

CO2

**Solution:**

```
#include<stdio.h>
int main()
{
    int a,b,c,c1,c2,c3;
    char istriangle;
    do
    {
        printf("\nenter 3 integers which are sides of triangle\n");
        scanf("%d%d%d",&a,&b,&c);
        printf("\na=%d\tb=%d\tc=%d",a,b,c);
        c1 = a>=1 && a<=10;
        c2= b>=1 && b<=10;
        c3= c>=1 && c<=10;
        if (!c1)
            printf("\nthe value of a=%d is not the range of permitted value",a);
        if (!c2)
            printf("\nthe value of b=%d is not the range of permitted value",b);
        if (!c3)
            printf("\nthe value of c=%d is not the range of permitted value",c);
    } while(!(c1 && c2 && c3));

    // to check is it a triangle or not
    if( a<b+c && b<a+c && c<a+b )
        istriangle='y';
    else
        istriangle ='n';
    if (istriangle=='y')
        if ((a==b) && (b==c))
            printf("equilateral triangle\n");
        else if ((a!=b) && (a!=c) && (b!=c))
            printf("scalene triangle\n");
        else
            printf("isosceles triangle\n");
    else
        printf("Not a triangle\n");
    return 0;
}
```

**Triangle Problem -Boundary value Test cases for input data**

Case Id	Description	Input Data			Expected Output
		A	b	c	
1	Enter the min value for a , b and c	1	1	1	Should display the message Equilateral triangle
2	Enter the min value for 2 items and min +1 for any one item1	1	1	2	Message should be displayed can't form a Triangle
3	Enter the min value for 2 items and min +1 for any one item1	1	2	1	Message should be displayed can't form a triangle
4	Enter the min value for 2 items and min +1 for any one item1	2	1	1	Message should be displayed can't form a triangle
5	Enter the normal value for 2 items and 1 item is min value	5	5	1	Should display the message Isosceles triangle
6	Enter the normal value for 2 items and 1 item is min value	5	1	5	Should display the message Isosceles triangle
7	Enter the normal value for 2 items and 1 item is min value	1	5	5	Should display the message Isosceles triangle
8	Enter the normal Value for a, b and c	5	5	5	Should display the message Equilateral triangle

(b) Write a program to solve commission problem. Analyze it from the perspective of equivalence class testing and derive the test cases.

[10]

CO2

**Solution:**

```
#include<stdio.h>
int main()
{
    int locks, stocks, barrels, tlocks, tstocks, tbarrels;
    float lprice, sprice, bprice, sales, comm;
    int c1,c2,c3,temp;
    lprice=45.0;
    sprice=30.0;
    bprice=25.0;
    tlocks=0;
    tstocks=0;
    tbarrels=0;
    printf("\nenter the number of locks and to exit the loop enter -1 for locks\n");
    scanf("%d",&locks);
    while(locks!=-1)
    {
        c1=(locks<=0||locks>70);
        printf("enter the number of stocks and barrels\n");
        scanf("%d%d",&stocks,&barrels);
        c2=(stocks<=0||stocks>80);
        c3=(barrels<=0||barrels>90);
        if(c1)
            printf("value of locks not in the range 1..70 ");
        else
        {
            temp=tlocks+locks;
            if(temp>70)
                printf("new total locks =%d not in the range 1..70 so old ",temp);
            else
                tlocks=temp;
        }
        printf("total locks = %d\n",tlocks);
        if(c2)
            printf("value of stocks not in the range 1..80 ");
        else
        {
            temp=tstocks+stocks;
            if(temp>80)
                printf("new total stocks =%d not in the range 1..80 so old ",temp);
            else
                tstocks=temp;
        }
        printf("total stocks=%d\n",tstocks);
        if(c3)
            printf("value of barrels not in the range 1..90 ");
        else
        {
            temp=tbarrels+barrels;
            if(temp>90)
                printf("new total barrels =%d not in the range 1..90 so old ",temp);
            else
                tbarrels=temp;
        }
        printf("total barrel=%d",tbarrels);
        printf("\nenter the number of locks and to exit the loop enter -1 for locks\n");
        scanf("%d",&locks);
    }
    printf("\ntotal locks = %d\ntotal stocks =%d\ntotal barrels =%d\n",tlocks,tstocks,tbarrels);
    sales = lprice*tlocks+sprice*tstocks+bprice*tbarrels;
    printf("\nthe total sales=%f\n",sales);

    if(sales > 0)
    {
        if(sales > 1800.0)
        {
            comm=0.10*1000.0;
            comm=comm+0.15*800;
            comm=comm+0.20*(sales-1800.0);
        }
        else if(sales > 1000)
        {
            comm =0.10*1000;
            comm=comm+0.15*(sales-1000);
        }
        else
            comm=0.10*sales;

        printf("the commission is=%f\n",comm);
    }
    else
        printf("there is no sales\n");
    return 0;
}
```

**Weak Robustness Equivalence Class**

Case Id	Description	Input Data			Expected Output
		Locks	Stocks	Barrels	
WR1	Enter the value locks = -1	-1	40	45	Terminates the input loop and proceed to calculate sales and commission ( if Sales > 0)
WR2	Enter the value less than -1 or equal to zero for locks and other valid inputs	0	40	45	Value of Locks not in the range 1..70
WR3	Enter the value greater than 70 for locks and other valid inputs	71	40	45	Value of Locks not in the range 1..70
WR4	Enter the value less than or equal than 0 for stocks and other valid inputs	35	0	45	Value of stocks not in the range 1..80
WR5	Enter the value greater than 80 for stocks and other valid inputs	35	81	45	Value of stocks not in the range 1..80
WR6	Enter the value less than or equal 0 for barrels and other valid inputs	35	40	0	Value of Barrels not in the range 1..90
WR7	Enter the value greater than 90 for barrels and other valid inputs	35	40	91	Value of Barrels not in the range 1..90

**Strong Robustness Equivalence Class**

Case Id	Description	Input Data			Expected Output
		Locks	Stocks	Barrels	
SR1	Enter the value less than -1 for locks and other valid inputs	-2	40	45	Value of Locks not in the range 1..70
SR2	Enter the value less than or equal than 0 for stocks and other valid inputs	35	-1	45	Value of stocks not in the range 1..80
SR3	Enter the value less than or equal 0 for barrels and other valid inputs	35	40	-2	Value of Barrels not in the range 1..90
SR4	Enter the locks and stocks less than or	-2	-1	45	Value of Locks not in the range 1..70

**OR**

4 (a) **Write a program to solve the triangle problem. Derive test cases for program based on decision table approach.**

**Solution:**

```

#include<stdio.h>
int main()
{
int a ,b ,c;
char istriangle;
printf("enter 3 integers which are sides of triangle\n");
scanf("%d%d%d",&a, &b, &c);
printf("a=%d\t, b=%d\t, c=%d\n", a, b, c);
if( a<b+c && b<a+c && c<a+b )
istriangle='y';
else
istriangle ='n';
if (istriangle=='y')
if ((a==b) && (b==c))
printf("Equilateral triangle\n");
else if ((a!=b) && (a!=c) && (b!=c))
printf("Scalene triangle\n");
else
printf("Isosceles triangle\n");
else
printf("Not a triangle\n");
return 0;
}

```

[10]

CO2



Input data decision Table		R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11
RULES	C1: $a < b + c$	F	T	T	T	T	T	T	T	T	T	T
	C2: $b < a + c$	-	F	T	T	T	T	T	T	T	T	T
	C3: $c < a + b$	-	-	F	T	T	T	T	T	T	T	T
	C4: $a = b$	-	-	-	T	T	T	T	F	F	F	F
	C5: $a = c$	-	-	-	T	T	F	F	T	T	F	F
	C6: $b = c$	-	-	-	T	F	T	F	T	F	T	F
Actions	a1 : Not a triangle	X	X	X								
	a2 : Scalene triangle											X
	a3 : Isosceles triangle							X		X	X	
	a4 : Equilateral triangle				X							
	a5 : Impossible					X	X		X			

Case Id	Description	Input Data			Expected Output
		a	b	c	
1	Enter the value of a, b and c Such that a is not less than sum of two sides	20	5	5	Message should be displayed can't form a triangle
2	Enter the value of a, b and c Such that b is not less than sum of two sides and a is less than sum of other two sides	3	15	11	Message should be displayed can't form a triangle
3	Enter the value of a, b and c Such that c is not less than sum of two sides and a and b is less than sum of other two sides	4	5	20	Message should be displayed can't form a triangle
4	Enter the value a, b and c satisfying precondition and $a=b$ , $b=c$ and $c=a$	5	5	5	Should display the message Equilateral triangle
5	Enter the value a, b and c satisfying precondition and $a=b$ and $b \neq c$	10	10	9	Should display the message Isosceles triangle
6	Enter the value a, b and c satisfying precondition and $a \neq b$ , $b \neq c$ and $c \neq a$	5	6	7	Should display the message Scalene triangle

(b) List the assumptions made in fault based testing and explain the mutation analysis with sample program.

[10]

CO2

Solution:

*Assumptions -*  
 \* competent programmer hypothesis  
 \* coupling effect hypothesis

*Mutation Analysis*

\* Mutation operator  
 \* Mutant  
 \* Valid Mutant  
 \* Invalid mutant

*Sample program*

MODULE-3

5 (a)	<p><b>Analyze and explain metric-based testing.</b> Solution:</p> <p><i>Metric-Based Testing</i></p> <ol style="list-style-type: none"><li><i>1. Statement and Predicates Testing</i></li><li><i>2. DD -path testing</i></li><li><i>3. Dependent Pairs of DD-paths.</i></li><li><i>4. Multiple condition coverage.</i></li><li><i>5. Loop coverage</i></li></ol> <p><i>Explanation</i></p>	[10]	CO3
5 (b)	<p><b>Explain define/Use testing with example.</b> Solution:</p> <p><i>5 definitions.</i></p> <p><i>DEF (v,m)</i></p> <p><i>USE (v,m)</i></p> <p><i>P-USE.</i></p> <p><i>C-USE.</i></p> <p><i>PATH (p) - du-path</i> <i>dc-path</i></p>	[10]	CO3
	<b>OR</b>		
6 (a)	<p><b>Describe about scaffolding. Discuss about Generic versus specific scaffolding.</b> Solution:</p> <p><b>SCAFFOLDING</b></p> <ul style="list-style-type: none"><li>• Code developed to facilitate testing is called scaffolding, by analogy to the temporary structures erected around a building during construction or maintenance.</li><li>• Scaffoldings may include<ul style="list-style-type: none"><li>→ Test drivers (substituting for a main or calling population)</li><li>→ Test harness (substituting for parts of the deployment environment)</li><li>→ Stubs (substituting for functionally called or used by the software under test)</li></ul></li><li>• The purpose of scaffolding is to provide controllability to execute test cases and observability to judge the outcome of test execution.</li></ul>	[08]	CO3

## GENERIC VERSUS SPECIFIC SCAFFOLDING

How general should scaffolding be? To answer

- We could build a driver and stubs for each test case or at least factor out some common code of the driver and test management (e.g., JUnit)
- ... or further factor out some common support code, to drive a large number of test cases from data... or further, generate the data automatically from a more abstract model (e.g., network traffic model)
- Fully generic scaffolding may suffice for small numbers of hand-written test cases
- The simplest form of scaffolding is a driver program that runs a single, specific test case.
- It is worthwhile to write more generic test drivers that essentially interpret test case specifications.
- A large suite of automatically generated test cases and a smaller set of handwritten test cases can share the same underlying generic test scaffolding
- Scaffolding to replace portions of the system is somewhat more demanding and again both generic and application-specific approaches are possible
- A simplest stub – *mock* – can be generated automatically by analysis of the source code
- The balance of quality, scope and cost for a substantial piece of scaffolding software can be used in several projects
- The balance is altered in favour of simplicity and quick construction for the many small pieces of scaffolding that are typically produced during development to support unit and small-scale integration testing
- A question of costs and re-use – Just as for other kinds of software

(b) Define:

i) Test oracles    ii) Self-checks    iii) Capture    iv) Replay.

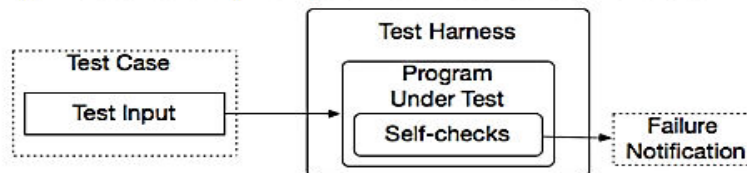
Solution:

### TEST ORACLES

- In practice, the pass/fail criterion is usually imperfect.
- A test oracle may apply a pass/fail criterion that reflects only a part of the actual program specification, or is an approximation, and therefore passes some program executions it ought to fail
- Several partial test oracles may be more cost-effective than one that is more comprehensive
- A test oracle may also give false alarms, failing an execution that is ought to pass.
- False alarms in test execution are highly undesirable.
- The best oracle we can obtain is an oracle that detects deviations from expectation that may or may not be actual failure.

### SELF-CHECKS AS ORACLES

- An oracle can also be written as self checks  
-Often possible to judge correctness without predicting results.
- Typically these self checks are in the form of assertions, but designed to be checked during execution.
- It is generally considered good design practice to make assertions and self checks to be free of side effects on program state.
- Self checks in the form of assertions embedded in program code are useful primarily for checking module and subsystem-level specification rather than all program behaviour.
- Devising the program assertions that correspond in a natural way to specifications poses two main challenges:
  - ▶ Bridging the gap between concrete execution values and abstractions used in specification
  - ▶ Dealing in a reasonable way with quantification over collection of values



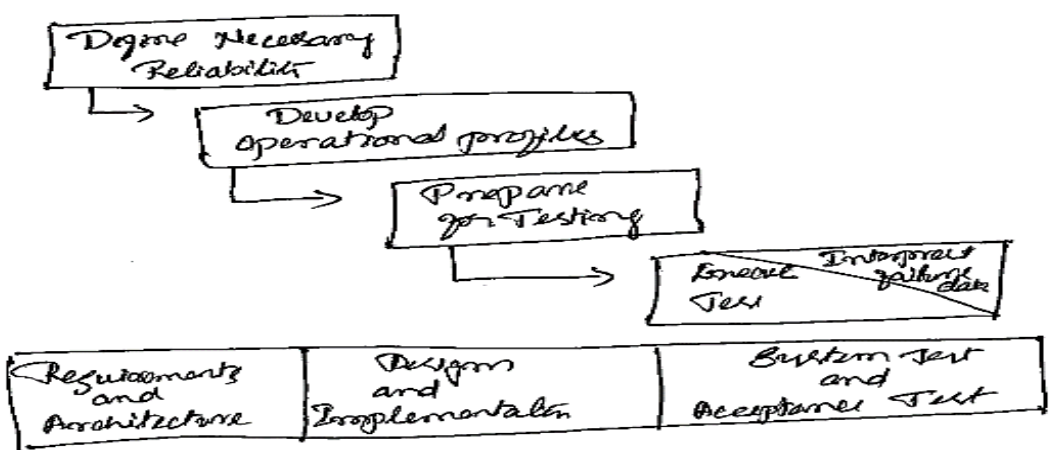
### CAPTURE AND REPLAY

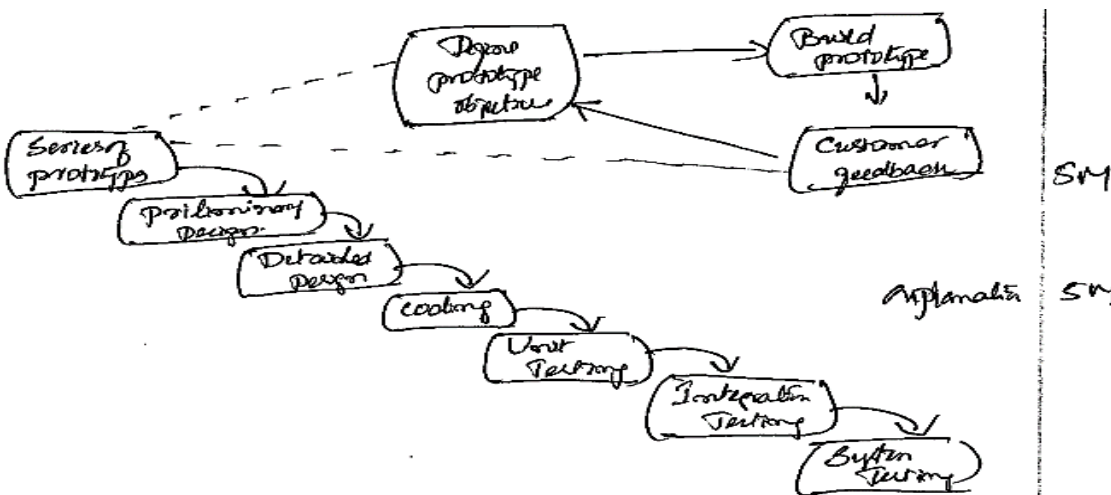
- Sometimes it is difficult to either devise a precise description of expected behaviour or adequately characterize correct behaviour for effective self checks.  
Example: even if we separate testing program functionally from GUI, some testing of the GUI is required.
- If one cannot completely avoid human involvement test case execution, one can at least avoid unnecessary repetition of this cost and opportunity for error.
- The principle is simple:  
The first time such a test case is executed, the oracle function is carried out by a human, and the interaction sequence is captured. Provided the execution was judged (by human tester) to be correct, the captured log now forms an (input, predicted output) pair for subsequent automated testing.
- The savings from automated retesting with a captured log depends on how many build-and-test cycles we can continue to use it, before it is invalidated by some change to the program.
- Mapping from concrete state to an abstract model of interacting sequences is some time possible but is generally quite limited.

[12]

CO3

MODULE-4

7 (a)	<p><b>Explain the basic principles in the frame work for test and analysis.</b>  <b>Solution:</b></p> <ol style="list-style-type: none"> <li>1) Sensitivity</li> <li>2) Redundancy</li> <li>3) Restriction</li> <li>4) Partition</li> <li>5) Visibility</li> <li>6) Feedback</li> </ol> <p align="right">Explanation</p>	[12]	CO4
8 (b)	<p><b>List and explain the dependability properties test and analysis actives.</b>  <b>Solution:</b></p> <ol style="list-style-type: none"> <li>1) correctness</li> <li>2) reliability</li> <li>3) Availability</li> <li>4) MTBF</li> <li>5) Safety</li> <li>6) Hazards</li> <li>7) robustness</li> </ol> <p align="right">explanation</p>	[08]	CO4
<b>OR</b>			
8 (a)	<p><b>Explain Software Reliability Engineered Testing (SRET) approach with diagram.</b>  <b>Solution:</b></p>  <p align="right">explanation</p>	[10]	CO4

(b)	<p>Identify and explain risk management in quality plan with respect to generic and specific issues.</p> <p>Solution:</p> <p><del>Identify</del> Risk generic to product management</p> <p>1) Personnel Risks 2) Technology Risks 3) Schedule Risks</p> <p>Risk specific to quality management</p> <p>1) Development Risks 2) Execution Risks 3) Requirement risks</p> <p style="text-align: center;">- (SM)</p>	[10]	CO4
<b>MODULE-5</b>			
9 (a)	<p>Analyze and explain integration testing strategies.</p> <p>Solution:</p> <p>1) Big bang Testing 2) Structural integration test Strategy 3) Top down &amp; bottom up testing 4) Sandwich or backbone. 5) Thread testing 6) critical module.</p>	[10]	CO5
(b)	<p>What is regression testing? Explain regression test selection technique.</p> <p>Solution:</p> <p>Regression Testing explanation * control gbs graph regression Test. Regression Test Selection Techniques with example-</p>	[10]	CO5
<b>OR</b>			
10 (a)	<p>Explain Rapid Prototyping Life Cycle with diagram.</p> <p>Solution:</p> 	[10]	CO5
(b)	<p>Explain Decomposition – Based integration.</p> <p>Solution:</p> <p>1) Topdown integration with example</p> <p>2) Bottom up integration with example</p>	[10]	CO5