Vrije Universiteit, Department of Computer Science

Examination paper for **Software Testing**

15 June 2007 12:00-14:45

Solutions

This is a closed book written exam.

No printed material or electronic devices are admitted for use during the exam.

The answers have to be given in English or Dutch.

Both homework and exam are compulsory and graded on a 1 to 10 scale.

The exam grade is calculated as (Q1+Q2+…+Q5+10)/10.

The final grade is calculated as 0.4*homework + 0.6*exam

A pass is given if both components as well as the final grade are >= 5.5.

	Q1	**Q2**	**Q3**	**Q4**	**Q5**					**ΣQi**	**Maximum credits= (ΣQi+10)/10**
a)	**3**	**5**		**3**							
b)	**3**	**5**		**2**							
c)	**3**	**5**		**10**							
d)	**3**	**5**		**5**							
e)	**3**			**10**							
Total	**15**	**20**	**15**	**30**	**10**					**90**	**10**

Q1 [15p] Define the following terms and in each case give an example.

a) The bug life cycle [3p]
b) On/Off point in domain testing [3p]
c) Equivalent mutant [3p]
d) Software reliability models [3p]
e) Definition-clear path [3p]

a) The bug life cycle is a standardized sequence of states a bug follows during the software testing process. A bug (or fault) is first discovered by the tester and it gets the status OPEN. The bug is assigned to the programmer. When the programmer fixes the bug, the bug gets in the RESOLVED state. The tester conforms that the bug is fixed and the bug gets the status CLOSED.
b) Domain testing deals with domains and their borders. An ON point is a point on the border, an OFF point is not on the border. Where exactly the OFF point is related to the domain depends on the type of the border. The rule is COOOOI (closed off is outside, open off is inside). If the border is closed, the OFF point is outside the domain, if the border is open, the OFF point is inside the domain. For example, for a domain defined as y< 4. The border is open, an OFF point is inside the domain and can be y = 3. An ON point is y=4.
c) An equivalent mutant is a mutant that has the same functionality as the original program. For example: x=x+2 and its mutant x*2 cannot be distinguished semantically and offer the same functionality.
d) A software reliability model is a representation of the random process through which software reliability is characterized as a function of time and properties of the software product or its development process. There are more than 200 SRM, for example, Rayleigh model, exponential model, reliability growth model.
e) Definition-clear path = a path in the data flow graph between a definition and use of a variable without any other definition occurrence of the same variable

Q2 [20p]

a) Define verification and validation and explain their place in the testing process V-model. [5p]

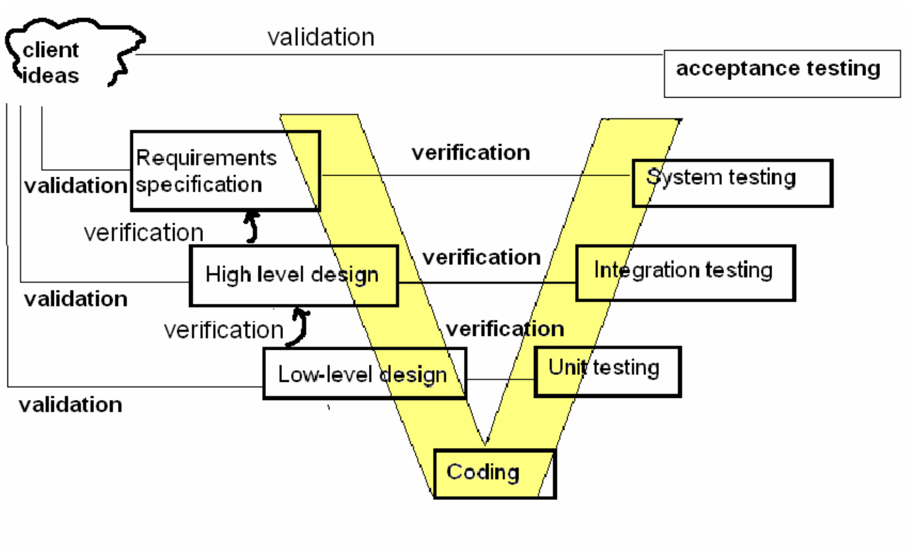
b) Define statement and decision coverage test adequacy criteria. Give an example of fault that cannot be detected by 100% statement coverage testing. [5 p]

c) Explain shortly the state transition diagram testing method. Give an example of a state transition diagram and show its recommended test cases. [5 p]

d) Describe the inspection process and the resulting documents. [5p]

a) Verification=evaluates a component or system to determine whether the products of a given development phase satisfy the conditions imposed at the start of each phase. Answers the question: *Are we building the system right?*

b) *Validation=process of evaluating a system or component during or at the end of development process to determine whether it satisfies specified requirements (IEEE/ANSI). Answers the question: Are we building the right system?*



Q3. Black-box testing [15p]

Consider the following scenario used by a marketing company to test its products on the market:

`The customers are treated according to 3 characteristics: Gender (male/female), City Dweller (Y/N) and Age group: A (young-under 30), B (middle-aged, between 30 and 60) and C (old, over 60.`

`The company has 4 products (W, X, Y and Z) to test on the market. Product W will appeal to female city dwellers. Product X will appeal to young females. Product Y will appeal to male middle-aged shoppers who do not live in cities. Product Z will appeal to all clients except older females.`

Suggest a black-box procedure to test the software module used to decide which products should be evaluated by each customer. Argument your choice and generate the test cases.

This is a case where many business rules are described. The most adequate black-box testing technique in this case is decision table testing. The conditions are the customer characteristics: gender, city dweller and age group and the actions are the products to be evaluated.
The maximum nr. of rules and thus of test cases: 2x2x3 = 12

Each column in the decision table is a test case. For example:

Test case 1: Inputs: Female, city dweller, young Output: Product W, Product X and product Z.

Pairwise testing can also be used n order to reduce the number of test cases.an option.

The decision table.

	1	2	3	4	5	6	7	8	9	10	11	12
Conditions												
Gender	F	M	F	M	F	M	F	M	F	M	F	M
City Dweller	Y	Y	N	N	Y	Y	N	N	Y	Y	N	N
Age	A	A	A	A	B	B	B	B	C	C	C	C
Actions												
Product W	x				x				x			
Product X	x		x									
Product Y								x				
Product Z	X	x	x	x	x	x	x	x	x	x		x

Q4. White-Box testing [30p]

Consider this Java implementation of the triangle problem.

```
 public class Triangle {

      static int INVALID_TRIANGLE = 1 ;
      static int SCALENE_TRIANGLE = 2;
      static int ISOSCELES_TRIANGLE = 3 ;
      static int EQUILATERAL_TRIANGLE = 4 ;

[A] public static int get_Type(int a, int b, int c)

{
      int type;
[B]   if (a>b)
[C]         {int t = a; a = b; b = t;}
[D]   if (a>c)
[E]         {int t = a; a = c;c = t;}
[F]   if (b>c)
[G]         {int t = b;b = c;c = t;}

[H]   if (a+b <= c)
[I]        type= INVALID_TRIANGLE;
      else {
[J]         type = SCALENE_TRIANGLE;
[K]         if (a ==b && b==c)
[L]               type = EQUILATERAL_TRIANGLE;
[M]         else if (a==b || b==c)
[N]               type = ISOSCELES_TRIANGLE;
            }

[O]         return (type);
}


}
```

a) [3p] Draw a control flow-graph for the above module
b) [2p] Determine the McCabe complexity
c) [10p] Prepare suitable test cases using the basis-path testing technique.

d) [5p] Implement these test cases in a unit test for JUnit frameworks.

e) [10p] Generate 3 mutants of this Java class. Test these mutants with the test suite from section c). Which of the mutants will stay alive? What can you say about the quality of your test suite?

McCabe complexity C = E-N+2 = 20-15+2 = 7.

Basis path testing needs 7 independent paths.

Choose as basis path the most common path

Take care that some paths are not feasible. Like A-B-D-E-F-H-J-K-M-O a < b, a>c swap a with c, b<c.

Test case	Path	Inputs	Expected outputs
1	A-B-D-F-H-J-K-M-O	a <=b, a <=c, b <=c a= 3 b=8 c=10	scalene
2	A-B-C-D-F-H-J-K-M-O	a>b , swap a and b, a <= c, b<=c a = 8, b=3, c=10	scalene
3	A-B-C-D-E-F-G-H-J-K-M-O	a< b, a > c, swap a and c, b > c , swap b and c a=7 b=10 c = 5	Scalene
4	A-B-D-F-G-H-J-K-M-O	a<b, a<c, b>c swap b and c a=5 b=10 c=7	Scalene
5.	A-B-D-F-H-I-O	a <=b, a <=c, b <=c a= 3 b=4 c=100	Invalid

6	A-B-D-F-H-J-K-L-O	a=b a=c b=c
		a=b=c=5
			Equilateral

7	A-B-D-F-H-J-K-M-N-O	a=b, a<=c, b<=c
		a=5, b=5, c=7
			isosceles

Junit tests:

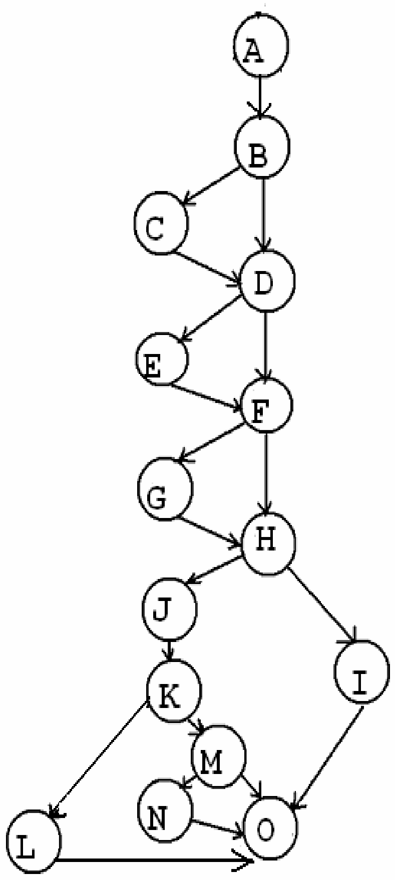
```
package triangle;

import junit.framework.TestCase;

public class TriangleTest extends TestCase {
      static int INVALID_TRIANGLE = 1 ;
      static int SCALENE_TRIANGLE = 2;
      static int ISOSCELES_TRIANGLE = 3 ;
      static int EQUILATERAL_TRIANGLE = 4 ;

      public void testType () {
      assertEquals(SCALENE_TRIANGLE, Triangle.get_Type (3,8,10));
//Test case 1
      assertEquals(SCALENE_TRIANGLE,Triangle.get_Type(8,3,10));
// Test case 2
            assertEquals(SCALENE_TRIANGLE, Triangle.get_Type (7,10,5));
// Test case 3
            assertEquals(SCALENE_TRIANGLE,Triangle.get_Type(5,10,7));
// Test case 4
            assertEquals(INVALID_TRIANGLE,Triangle.get_Type(3,4,100));
// Test case 5
            assertEquals(EQUILATERAL_TRIANGLE, Triangle.get_Type (5,5,5));
// Test case 6
            assertEquals(ISOSCELES_TRIANGLE,Triangle.get_Type(5,5,7));
// Test case 7

            }
}
```


A
B
C
D
E
F
G
H
J
I
K
M
N
L
O