Question #1: Match the items on the left column to the related items on the right one: (12 Marks):

No		Letter	
1	**Configuration management**	**A**	The extent to which a software performs its intended functions without failure
2	**Codeline**	**B**	The software should conform to its specification
3	**Validation**	**C**	Any type of measurement which relates to a software system, process or related documentation
4	**Fog index**	**D**	A set of versions of a software components and other configuration items on which that component depends
5	**Change management**	**E**	A collection of component versions that make up a system
6	**Test case**	**F**	The policies, processes and tools for managing changing software systems
7	**Software metric**	**G**	The extent to which a software tolerates the unexpected problems
8	**Reliability**	**H**	The process that is intended to ensure that system evolution is a managed process and that priority is given to the most urgent and cost-effective changes
9	**Verification**	**I**	The software should do what the user really requires.
10	**System building**	**J**	a measure of the average length of words and sentences in documents
11	**Robustness**	**K**	A program with input and expected output used to test the real program
12	**Baseline**	**L**	The process of creating a complete, executable system by compiling and linking the system components, external libraries, configuration files, etc

No	Letter
1	**F**
2	**D**
3	**I**
4	**J**
5	**H**
6	**K**
7	**C**
8	**A**
9	**B**
10	**L**
11	**G**
12	**E**

Question #2 (12 Marks): Multiple Choice Questions:

1. Which of the following need to be assessed during unit testing?
   a. algorithmic performance
   b. error handling
   c. execution paths
   d. both b and c
2. Regression testing should be a normal part of integration testing because as a new module is added to the system new
   a. control logic is invoked
   b. data flow paths are established
   c. drivers require testing
   d. **both a and b**
3. The testing technique that requires devising test cases to exercise the internal logic of a software module is called
   a. behavioral testing
   b. black-box testing
   c. grey-box testing
   d. **white-box testing**
4. The cyclomatic complexity metric provides the designer with information regarding the number of
   a. cycles in the program
   b. errors in the program
   c. **independent logic paths in the program**
   d. statements in the program
5. Black-box testing attempts to find errors in which of the following categories
   a. incorrect or missing functions
   b. interface errors
   c. performance errors
   d. **All of the above**
6. Which of the following is not one of the dimensions of quality used to assess a WebApp?
   a. Content
   b. **Maintainability**
   c. Navigability
   d. Usability

Question #3 (6 Marks): True/False questions:

1. In software quality assurance work there is no difference between software verification and software validation (**False**)

2. When testing object-oriented software it is important to test each class operation separately as part of the unit testing process (**False**)
3. Debugging is not testing, but always occurs as a consequence of testing (**True**)
4. With thorough testing it is possible to remove all defects from a program prior to delivery to the customer (**False**)
5. Graph-based testing methods can only be used for object-oriented systems (**False**)
6. Boundary value analysis can only be used to do white-box testing (**False**)

Question #4 (20 Marks): Answer the following questions:

A. Why are software changes inevitable? What is the difference between software evolution and software servicing?

Software change is inevitable because:

- New requirements emerge when the software is used;
- The business environment changes;
- Errors must be repaired
- New computers and equipment is added to the system;
- The performance or reliability of the system may have to be improved.

Evolution

The stage in a software system's life cycle where it is in operational use and is evolving as new requirements are proposed and implemented in the system

Servicing

At this stage, the software remains useful but the only changes made are those required to keep it operational i.e. bug fixes and changes to reflect changes in the software's environment. No new functionality is added

B. Explain the Benefits of test-driven development

Code coverage

Every code segment that you write has at least one associated test so all code written has at least one test.

Regression testing

A regression test suite is developed incrementally as a program is developed.

Simplified debugging

When a test fails, it should be obvious where the problem lies. The newly written code needs to be checked and modified.

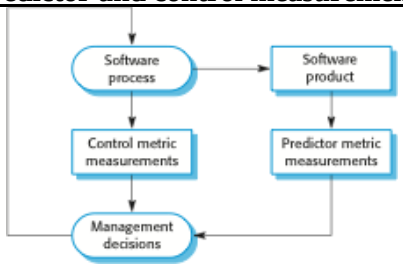
System documentation

The tests themselves are a form of documentation that describe what the code should be doing.

C. What is software measurements? Explain what is meant by Predictor and Control measurements? Explain Product Metrics

Software measurement is concerned with deriving a numeric value for an attribute of a software product or process

Predictor and control measurements

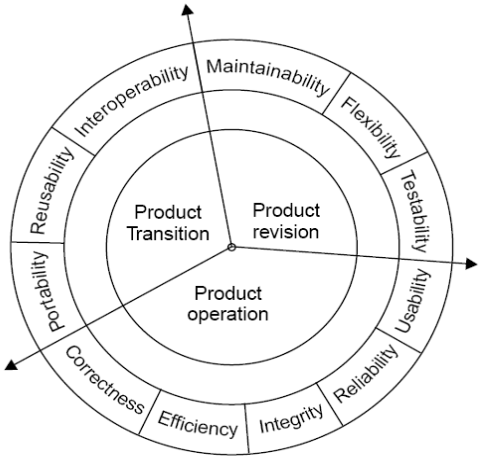


Product metrics

- A quality metric should be a predictor of product quality.
- Classes of product metric
  - ❖ Dynamic metrics which are collected by measurements made of a program in execution;
  - ❖ Static metrics which are collected by measurements made of the system representations;
  - ❖ Dynamic metrics help assess efficiency and reliability
  - ❖ Static metrics help assess complexity, understandability and maintainability.

D. Explain McCall Software Quality Model

McCall Software Quality Model



Product Operation

Factors which are related to the operation of a product are combined. The factors are:

Correctness – Efficiency – Integrity – Reliability – Usability.

These five factors are related to operational performance, convenience, ease of usage and its correctness. These factors play a very significant role in building customer's satisfaction.

Product Revision

The factors which are required for testing and maintenance are combined and are given below

Maintainability – Flexibility - Testability

These factors pertain to the testing & maintainability of software. They give us idea about ease of maintenance, flexibility and testing effort. Hence, they are combined under the umbrella of product revision.

Product Transition

We may have to transfer a product from one platform to another platform or from one technology to another technology. The factors related to such a transfer are combined and given below:

Portability – Reusability - Interoperability

E. What are the software configuration management activities

CM Activities

Change management: Keeping track of requests for changes, working out the costs and impact of changes, and deciding the changes

Version management: Keeping track of the multiple versions of system components

System building: The process of assembling program components, data and libraries, then compiling these to create an executable

Release management: Preparing software for external release and keeping track of the system versions that have been released

Question #4 (20 Marks): The following program is written in C++ programming language. Review it and answer the below-mentioned questions:

```
1: using namespace std;
2: #include <string>
3: #include <iostream>
4: #include <cmath>
5: #include <iomanip>
6: int main()
7: {
8:       string shape;
9:       double height;
10:      double width;
11:      double length;
12:      double radius;
13:      const double PI = 3.1416;
14:      cout << "Enter the shape type: (rectangle, circle, cylinder) ";
15:      cin >> shape;
16:      cout << endl;
17:      if (shape == "rectangle")
18:            {
19:             cout << "Enter the length of the rectangle: ";
20:             cin >> length;
21:             cout << endl;
22:             cout << "Enter the width of the rectangle: ";
23:             cin >> width;
24:            cout << endl;
25: cout << fixed << showpoint << setprecision(2);
26: cout << "Area of the rectangle = "
27:             << length * width << endl;
28:             cout << "Perimeter of the rectangle = "
29:             << 2 * (length + width) << endl;
30:             }
31:             else if (shape == "circle")
32:             {
33:             cout << "Enter the radius of the circle: ";
34:             cin >> radius;
35:             cout << endl;
36:             cout << "Area of the circle = "
37:             << PI * pow(radius, 2.0) << endl;
38:             cout << "Circumference of the circle: "
39:             << 2 * PI * pow(radius, 2.0) << endl;
40:             }
41:              else if (shape == "cylinder")
42:             {
43:             cout << "Enter the height of the cylinder: ";
44:             cin >> height;
45:             cout << endl;
46:             cout << "Enter the radius of the base of the cylinder: ";
47:             cin >> radius;
48:             cout << endl;
49:             cout << "Surface area of the cylinder: "
50:             << 2 * radius * + 2 * PI * pow(radius, 2.0) << endl;
51:             cout << "Volume of the cylinder = "
52:             << PI * pow(radius, 2.0)* height << endl;
53:                                       }
54:               else
55:               cout << "The program does not handle " << shape << endl;
56: return 0;
57: }
```

a. Specify the objective, input and output of the program
b. Draw the flowchart for that program
c. Design the test cases for the program using the robustness testing methodology (suggest input and output domain values and any other requirements)
d. Draw the program flow graph and DD path graph
e. Draw the mapping table for the DD path graph and specify the independent paths in the program
f. Calculate the program cyclomatic complexity using the different known methods

---

Question #1: Match the items on the left column to the related items on the right one: (12 Marks):

No		Letter			
1	**Configuration management**	**A**	The extent to which a software performs its intended functions without failure	1	**F**
2	**Codeline**	**B**	The software should conform to its specification	2	**D**
3	**Validation**	**C**	Any type of measurement which relates to a software system, process or related documentation	3	**I**
4	**Fog index**	**D**	A set of versions of a software components and other configuration items on which that component depends	4	**J**
5	**Change management**	**E**	A collection of component versions that make up a system	5	**H**
6	**Test case**	**F**	The policies, processes and tools for managing changing software systems	6	**K**
7	**Software metric**	**G**	The extent to which a software tolerates the unexpected problems	7	**C**
8	**Reliability**	**H**	The process that is intended to ensure that system evolution is a managed process and that priority is given to the most urgent and cost-effective changes	8	**A**
9	**Verification**	**I**	The software should do what the user really requires.	9	**B**
10	**System building**	**J**	a measure of the average length of words and sentences in documents	10	**L**
11	**Robustness**	**K**	A program with input and expected output used to test the real program	11	**G**
12	**Baseline**	**L**	The process of creating a complete, executable system by compiling and linking the system components, external libraries, configuration files, etc	12	**E**

Question #2 (10 Marks): Multiple Choice Questions:

7. The testing technique that requires devising test cases to exercise the internal logic of a software module is called
   e. behavioral testing
   f. black-box testing
   g. grey-box testing
   h. **white-box testing**
8. The cyclomatic complexity metric provides the designer with information regarding the number of
   e. cycles in the program
   f. errors in the program
   g. **independent logic paths in the program**
   h. statements in the program
9. Which of the following need to be assessed during unit testing?
   e. algorithmic performance
   f. error handling
   g. execution paths
   h. both b and c
10. Black-box testing attempts to find errors in which of the following categories
    e. incorrect or missing functions
    f. interface errors
    g. performance errors
    h. **All of the above**
11. Testing OO class operations is made more difficult by
    a. encapsulation
    b. inheritance
    c. polymorphism
    d. both b and c

Question #3 (10 Marks): True/False questions:

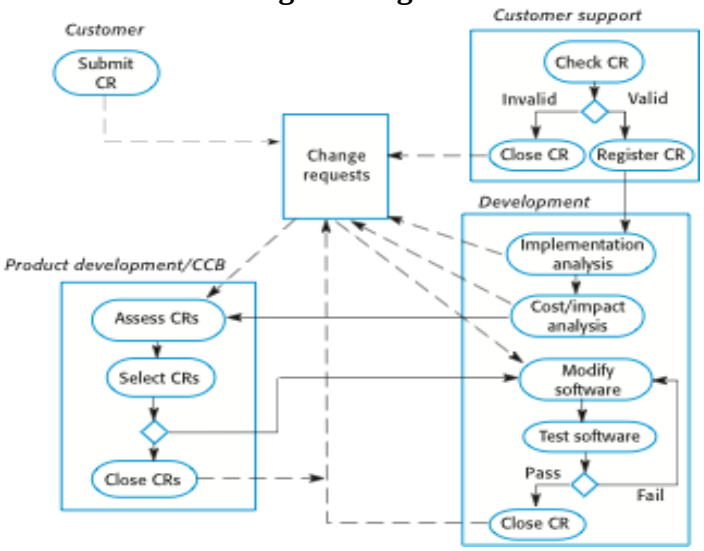
7. When testing object-oriented software it is important to test each class operation separately as part of the unit testing process (**False**)
8. Black-box testing technique requires devising test cases to exercise the internal logic of a software module **(F)**
9. In software quality assurance work there is no difference between software verification and software validation **(F)**
10. Software validation is achieved through a series of tests performed by the user once the software is deployed in his or her work environment **(F)**
11. Debugging is not testing, but always occurs as a consequence of testing **(T)**

12. With thorough testing it is possible to remove all defects from a program prior to delivery to the customer **(F)**
13. Graph-based testing methods can only be used for object-oriented systems **(F)**
14. Boundary value analysis can only be used to do white-box testing **(F)**
15. Validation of object-oriented software focuses on user visible actions and outputs from the system **(T)**
16. Since WebnApps evolve continuously, the testing process is an on-going activity, conducted by the Web support staff using regression tests **(T)**

Question #4 (18 Marks): Answer the following questions:

A. Explain the software change management process

The Change Management Process



B. What are the components of software release

Release Components

As well as the executable code of the system, a release may also include:
Configuration files: defining how the release should be configured for particular installations;
Data files, such as files of error messages that are needed for successful system operation;
An installation program that is used to help install the system on target hardware;
Electronic and paper documentation describing the system;
Packaging and associated publicity that have been designed for that release.

C. Why are software changes inevitable? What is the difference between software evolution and software servicing?

Software changes are inevitable because:

- New requirements emerge when the software is used;
- The business environment changes;
- Errors must be repaired;
- New computers and equipment is added to the system;
- The performance or reliability of the system may have to be improved.

Software Evolution:

The stage in a software system's life cycle where it is in operational use and is evolving as new requirements are proposed and implemented in the system.

Software Servicing

At this stage, the software remains useful but the only changes made are those required to keep it operational i.e. bug fixes and changes to reflect changes in the software's environment. No new functionality is added.

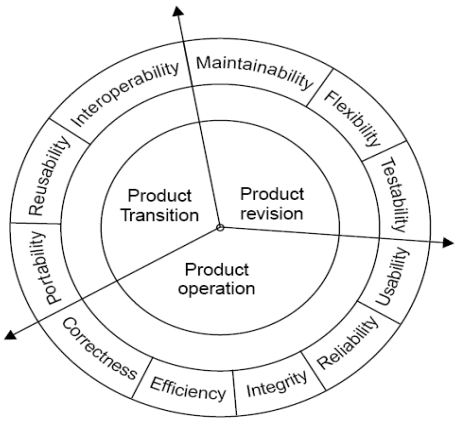
D. Explain the Benefits of test-driven development

- **Code coverage**: Every code segment that you write has at least one associated test so all code written has at least one test.

- **Regression testing:** A regression test suite is developed incrementally as a program is developed.
- **Simplified debugging:** When a test fails, it should be obvious where the problem lies. The newly written code needs to be checked and modified.
- **System documentation:** The tests themselves are a form of documentation that describe what the code should be doing.

E. Explain McCall Software Quality Model

McCall Software Quality Model



Product Operation

Factors which are related to the operation of a product are combined. The factors are:

Correctness – Efficiency – Integrity – Reliability - Usability

These five factors are related to operational performance, convenience, ease of usage and its correctness. These factors play a very significant role in building customer's satisfaction.

Product Revision

The factors which are required for testing and maintenance are combined and are given below

Maintainability – Flexibility - Testability

These factors pertain to the testing & maintainability of software. They give us idea about ease of maintenance, flexibility and testing effort. Hence, they are combined under the umbrella of product revision.

Product Transition

We may have to transfer a product from one platform to another platform or from one technology to another technology. The factors related to such a transfer are combined and given below:

Portability – Reusability - Interoperability

F. What are the software configuration management activities

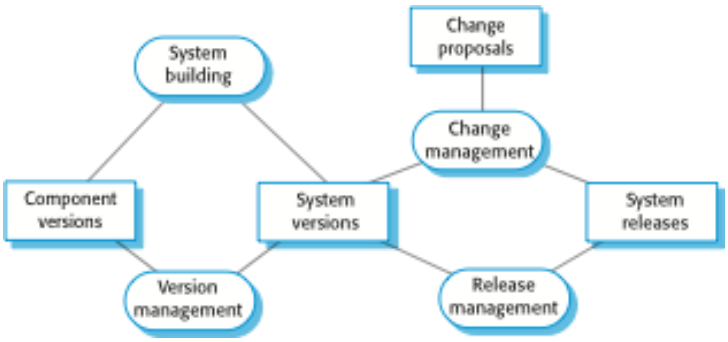
CM Activities

Change management: Keeping track of requests for changes, working out the costs and impact of changes, and deciding the changes

Version management: Keeping track of the multiple versions of system components

System building: The process of assembling program components, data and libraries, then compiling these to create an executable

Release management: Preparing software for external release and keeping track of the system versions that have been released



Question #4: Solve the following problem (20 Marks):

Draw the flowchart and write the C++ program in order to find the Roots of the quadratic equation: $\mathbf{ax^2 + bx + c = 0;}$ the program input is a triple of positive integers (**a, b, c**) and values may be from interval [**0,100**]. The program output may have one of the following [**Not a quadratic equation**; **Real roots**; **Imaginary roots**; **Equal roots**]

- Design the ROBUST test cases for the above mentioned program?
- Draw the flow graph, the DD path graph and find the Cyclomatic Complexity (by the three known methods) for the above mentioned program?

Note:

Roots are real if (b2-4ac) > 0, Roots are imaginary if (b2-4ac) < 0, Roots are equal if (b2-4ac) = 0.
Equation is not quadratic if a = 0

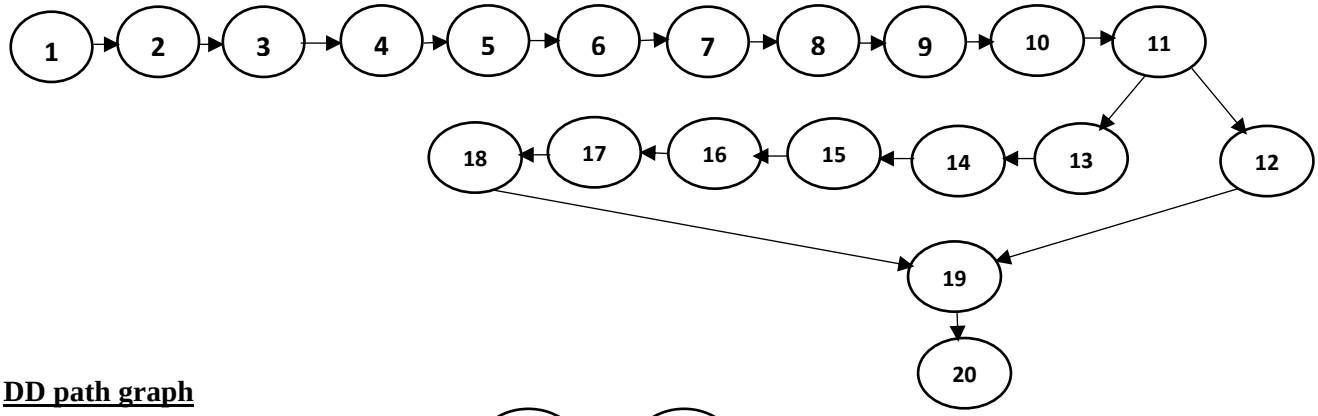
Solution

Quadratic equation will be of type: ax2+bx+c=0
Roots are real if (b2-4ac) > 0 - Roots are imaginary if (b2-4ac) < 0 - Roots are equal if (b2-4ac) = 0
Equation is not quadratic if a = 0

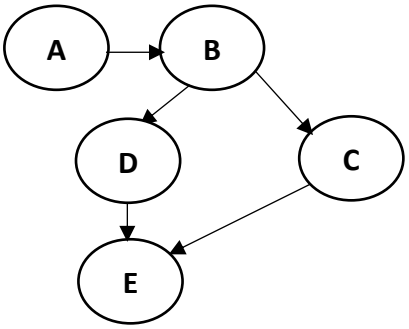
C++ Program

```
#include <iostream>
#include <cmath>
using namespace std;
int a, b, c;
double x1, x2, root;
int main() {
cin >> a;
cin >> b;
cin >> c;
root=(b*b)-(4*a*c);
        if (root<0)
        cout<<"imaginary roots";
        else
        {
        x1=-b+sqrt(root);
        x2=-b-sqrt(root);
        cout << x1 << '\n';
        cout << x2 << '\n';}
return 0;
}
```

Flow graph



DD path graph



A=1-10
B=11
C=12
D=13-18
E=19-20

Number of nodes (n) = 5 - Number of edges (e) = 5

(i) V(G) = e – n + 2P = 5 – 5 + 2 = 2

(ii) V(G) = π + 1 = 1 + 1 = 2

(iii) V(G) = Number of regions = 2

Hence, Cyclomatic Complexity is **2**, meaning thereby **TWO** independent paths in the DD path graph

The ROBUST test cases are:

Test case	*a*	*b*	*c*	*Expected Output*
1	-1	50	50	Invalid input`
2	0	50	50	Not quadratic equation
3	1	50	50	Real roots
4	50	50	50	Imaginary roots
5	99	50	50	Imaginary roots
6	100	50	50	Imaginary roots
7	101	50	50	Invalid input
8	50	-1	50	Invalid input
9	50	0	50	Imaginary roots
10	50	1	50	Imaginary roots
11	50	99	50	Imaginary roots
12	50	100	50	Equal roots
13	50	101	50	Invalid input
14	50	50	-1	Invalid input
15	50	50	0	Real roots
16	50	50	1	Real roots
17	50	50	99	Imaginary roots
18	50	50	100	Imaginary roots
19	50	50	101	Invalid input

Question #1: Match the items on the left column to the related items on the right one: (18 Marks):

No		Letter	
1	**Configuration management**	**A**	The extent to which a software performs its intended functions without failure
2	**Codeline**	**B**	The software should conform to its specification
3	**Validation**	**C**	Any type of measurement which relates to a software system, process or related documentation
4	**Fog index**	**D**	A set of versions of a software components and other configuration items on which that component depends
5	**Change management**	**E**	A collection of component versions that make up a system
6	**Test case**	**F**	The policies, processes and tools for managing changing software systems
7	**Software metric**	**G**	The extent to which a software tolerates the unexpected problems
8	**Reliability**	**H**	The process that is intended to ensure that system evolution is a managed process and that priority is given to the most urgent and cost-effective changes
9	**Verification**	**I**	The software should do what the user really requires.
10	**System building**	**J**	a measure of the average length of words and sentences in documents
11	**Robustness**	**K**	A program with input and expected output used to test the real program
12	**Baseline**	**L**	The process of creating a complete, executable system by compiling and linking the system components, external libraries, configuration files, etc

No	Letter
1	F
2	D
3	I
4	J
5	H
6	K
7	C
8	A
9	B
10	L
11	G
12	E

Question #2: Mark "True" or "False" (10 Marks):

12. Black-box testing technique requires devising test cases to exercise the internal logic of a software module **(F)**
13. In software quality assurance work there is no difference between software verification and software validation **(F)**
14. Software validation is achieved through a series of tests performed by the user once the software is deployed in his or her work environment **(F)**
15. Debugging is not testing, but always occurs as a consequence of testing **(T)**
16. Boundary value analysis can only be used to do white-box testing **(F)**

Question #3: Answer the following questions (24 Marks):

F. Explain software quality reviews and its application on Agile Methods?

- Software Quality Reviews involve a group of people who carefully examine part or all of a software system and its associated documentation.
- Code, designs, specifications, test plans, standards, etc. can all be reviewed.
- Software or documents may be 'signed off' at a review which signifies that progress to the next development stage has been approved by management

The software review process

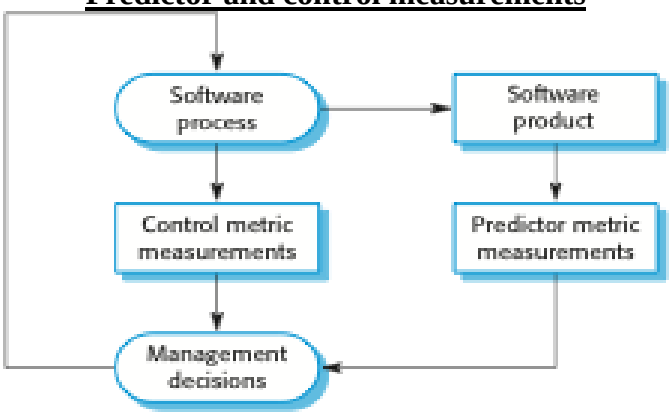
Reviews and agile methods

- The review process in agile software development is usually informal.
  - ❖ In Scrum, for example, there is a review meeting after each iteration of the software has been completed (a sprint review), where quality issues and problems may be discussed.
- In extreme programming, pair programming ensures that code is constantly being examined and reviewed by another team member.
- XP relies on individuals taking the initiative to improve and refactor code. Agile approaches are not usually standards-driven, so issues of standards compliance are not usually considered.

G. What is software measurements? Explain what is meant by Predictor and Control measurements? Explain Product Metrics?

Software measurement is concerned with deriving a numeric value for an attribute of a software product or process

Predictor and control measurements

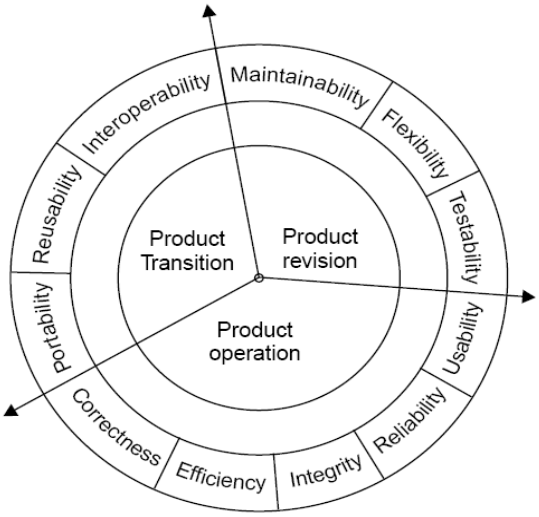


Product metrics

- A quality metric should be a predictor of product quality.
- Classes of product metric
  - ❖ Dynamic metrics which are collected by measurements made of a program in execution;
  - ❖ Static metrics which are collected by measurements made of the system representations;
  - ❖ Dynamic metrics help assess efficiency and reliability
  - ❖ Static metrics help assess complexity, understandability and maintainability.

H. Explain McCall Software Quality Model?

McCall Software Quality Model



Product Operation

Factors which are related to the operation of a product are combined. The factors are:

Correctness – Efficiency – Integrity – Reliability - Usability

These five factors are related to operational performance, convenience, ease of usage and its correctness. These factors play a very significant role in building customer's satisfaction.

Product Revision

The factors which are required for testing and maintenance are combined and are given below

Maintainability – Flexibility - Testability

These factors pertain to the testing & maintainability of software. They give us idea about ease of maintenance, flexibility and testing effort. Hence, they are combined under the umbrella of product revision.

Product Transition

We may have to transfer a product from one platform to another platform or from one technology to another technology. The factors related to such a transfer are combined and given below:

Portability – Reusability - Interoperability

I. What are the software configuration management activities?

CM Activities

Change management: Keeping track of requests for changes, working out the costs and impact of changes, and deciding the changes

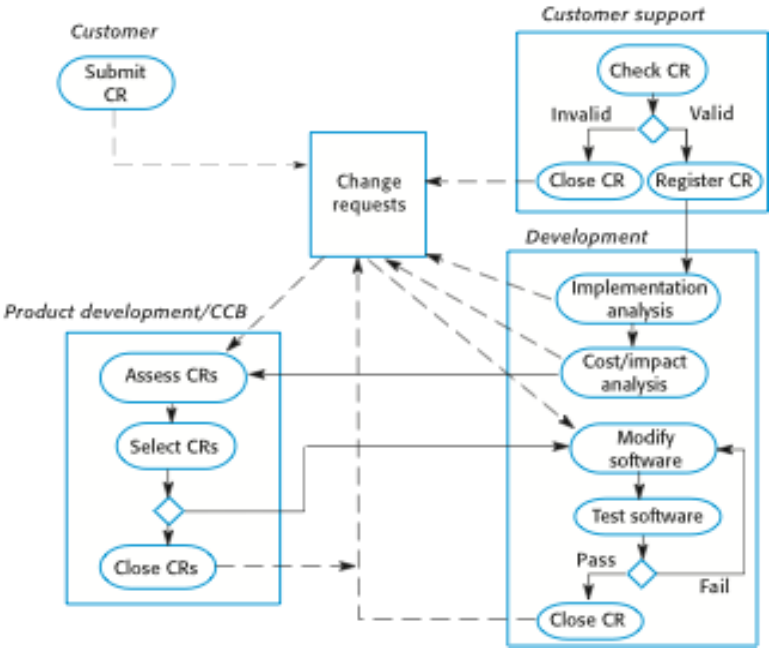
Version management: Keeping track of the multiple versions of system components

System building: The process of assembling program components, data and libraries, then compiling these to create an executable

Release management: Preparing software for external release and keeping track of the system versions that have been released

J. Explain the software change management process?

The Change Management Process



K. What are the components of software release?

Release Components

As well as the executable code of the system, a release may also include:
Configuration files: defining how the release should be configured for particular installations;
Data files, such as files of error messages that are needed for successful system operation;
An installation program that is used to help install the system on target hardware;
Electronic and paper documentation describing the system;
Packaging and associated publicity that have been designed for that release.

Question #4: Solve the following problem (18 Marks):

Draw the flowchart and write the C++ program in order to find the Roots of the quadratic equation: $\mathbf{ax^2 + bx + c = 0;}$ the program input is a triple of positive integers (**a, b, c**) and values may be from interval [**0,100**]. The program output may have one of the following [**Not a quadratic equation**; **Real roots**; **Imaginary roots**; **Equal roots**]

Design the ROBUST test cases for the above mentioned program?

Draw the flow graph, the DD path graph and find the Cyclomatic Complexity (by the three known methods) for the above mentioned program?

Note:

Roots are real if (b2-4ac) > 0, Roots are imaginary if (b2-4ac) < 0, Roots are equal if (b2-4ac) = 0. Equation is not quadratic if a = 0

Solution

Quadratic equation will be of type: ax2+bx+c=0
Roots are real if (b2-4ac) > 0 - Roots are imaginary if (b2-4ac) < 0 - Roots are equal if (b2-4ac) = 0
Equation is not quadratic if a = 0
The ROBUST test cases are:

Test case	a	b	c	Expected Output
1	-1	50	50	Invalid input`
2	0	50	50	Not quadratic equation
3	1	50	50	Real roots
4	50	50	50	Imaginary roots
5	99	50	50	Imaginary roots
6	100	50	50	Imaginary roots
7	101	50	50	Invalid input
8	50	-1	50	Invalid input
9	50	0	50	Imaginary roots
10	50	1	50	Imaginary roots
11	50	99	50	Imaginary roots
12	50	100	50	Equal roots
13	50	101	50	Invalid input
14	50	50	-1	Invalid input
15	50	50	0	Real roots
16	50	50	1	Real roots
17	50	50	99	Imaginary roots
18	50	50	100	Imaginary roots
19	50	50	101	Invalid input

Number of nodes (n) = 19 - Number of edges (e) = 24

(i) V(G) = e – n + 2P = 24 – 19 + 2 = 7

(ii) V(G) = π + 1 = 6 + 1 = 7

(iii) V(G) = Number of regions = 7

Hence, Cyclomatic Complexity is 7, meaning thereby seven independent paths in the DD path graph