UG END SEM EXAM

Semester - Vth

Date : 22nd December, 2021

Name : Subhojit Ghimire
Sch. Id : 1912160
Branch : CSE - B
Subject : Software Engineering
Subject Code : CS304

Q.1.

(a).
Ans → Software quality is defined as a field of study and practice that describes the desirable attributes of software products. There are two main approaches to software quality : defect management and quality attributes.

The McCall quality model (or McCall's Triangle of Quality) is organised around three types of quality characteristics:

1. Product Revision: Maintainability, Flexibility, Testability
2. Product Operations: Correctness, Reliability, Efficiency, Integrity, Usability.
3. Product Transition: Portability, Reusability, Interoperability.

(b)
Ans → Software quality management activities are generally spilt into three core components:

1. Quality Assurance
2. Quality Planning
3. Quality Control

The approaches for the same are:
(i) Startup, (ii) Initiation, (iii) Controlling a Stage, (iv) Stage Boundry and, (v) Close.

Q.2.

(a)

Ans → The unit is the smallest piece of code that can be logically isolated in a system. Unit testing is a type of software testing where individual units or components of a software are tested.

Considerations of Unit Testing:

i) To isolate a section of code.

ii) To verify the correctness of code.

iii) To test every function and procedure.

iv) To fix bug early in development cycle and to save costs.

v) To help for code reuse.

Procedures of Unit Testing:

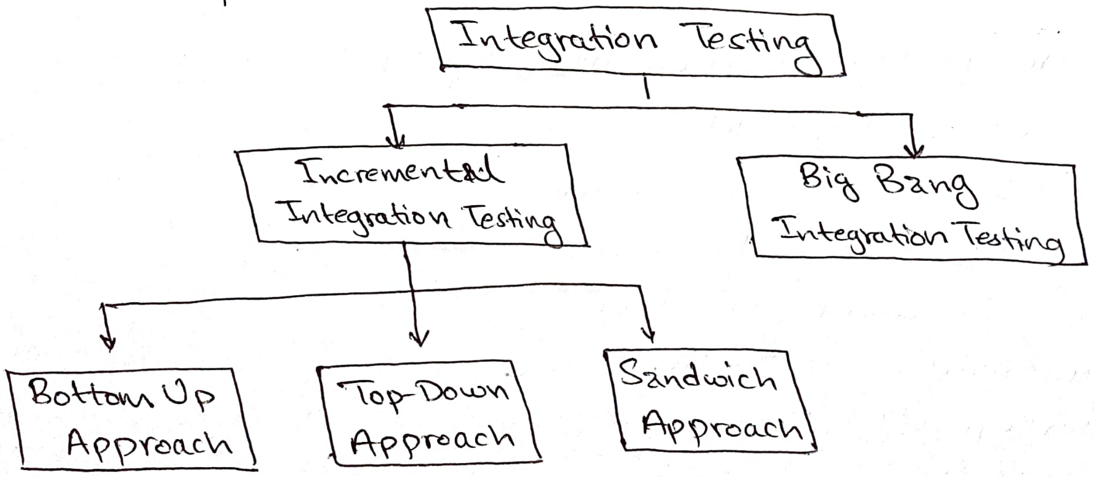
i) White-Box Testing.

ii) Black-Box Testing.

iii) Gray-Box Testing.

(b)

Ans → Integration Testing is a division of software testing that tests interfaces between different software components:



Q. 3.

(a).

Ans → Given, μ_1 = Number of unique operators

μ_2 = Number of unique operands

N_1 = Total Occurrences of operators.

N_2 = Number of unique operators.

Now,

Halstead Program length, $N = N_1 + N_2$

Halstead Vocabulary, $\mu = \mu_1 + \mu_2$

Program Volume, $V = N * \log_2(\mu)$

Program Difficulty, $D = \left(\frac{\mu_1}{\mu_2}\right) * \left(\frac{N_2}{2}\right)$

Therefore, Programming effort, $E = D * V$

$$= \left(\frac{\mu_1}{\mu_2}\right) * \left(\frac{N_2}{2}\right) * N * \log_2(\mu)$$

$$\therefore E = \frac{\mu_1 N_2 (N_1 + N_2)}{2\mu_2} * \log_2(\mu_1 + \mu_2)$$

(b)

Ans → The steps to calculate final function count in Albrecht's function Point Method are as follows:

(i) Compute value adjustment factor (VAF) based on 14 general system characteristics (GSC)

(ii) Weigh eache GSC on a scale of 0 to 5 based on whether it has no influence to strong influence.

(iii) Compute the FPC as follows:

$FPC = UFC * (0.65 + (sum(GSC) * 0.01))$

Q. 4.

(a)

Ans → The ~~components~~ principles for designing Class-based Components are :

(i) Open Closed Principle (OCP) : Any module in OCP should be available for extension and modification.

(ii) Liskov Substitution Principle (LSP) : The subclass must be substitutable for their base class.

(iii) Dependency Inversion Principle (DIP) : It depends on the abstraction and not on concretion. Abstraction is the place where the design is extended without difficulty.

(iv) Interface Segregation Principle (ISP) : Many client specific interface is better than the general purpose interface.

(v) Release Reuse Equivalency Principle (REP) : A fragment of reuse is the fragment of release.

(vi) Common Closure Principle (CCP) : The classes are packed as part of design which should have the same address and functional area.

(vii) Common Reuse Principle (CRP) : The classes that are not reused together should not be grouped together.

(b)

Ans → The steps for Component Level Design are :

(i) Identify Design Classes in Program Domain.
(ii) Identify Infrastructure Design Classes.
(iii) Elaborate Design Classes.
(iv) Describe Persistent Data Sources.
(v) Elaborate Behavioural Representations.
(vi) Elaborate Deployment Diagrams.
(vii) Refactor Design and Consider Alternatives.

Q. 6.

A (a)

Ans → The different classes of software product metrics are:

(i) Dynamic Metrics: It helps in accessing the efficiency and reliability of a program. Dynamic metrics are the class of software metrics that capture the dynamic behaviour of the software system and are usually obtained from the execution traces of the code or from the executable models.

(ii) Static Metrics: It helps in understanding and maintaining the complexity of a software system. A large number of these matrices have been proposed to try to derive and validate the relationship between the complexity, understandability and maintainability.

(b)
Ans) → Effort estimate in COCOMO model is calculated as follows:

(i) Basic COCOMO:

$$\text{EFFORT} = a_1 * (\text{KLOC})^{a_2} \quad \text{PM}$$

(ii) Intermediate COCOMO:

$$\text{EFFORT} = a_1 * (\text{KLOC})^{b_1} * \text{EAF}$$

where,

a_1, a_2, b_1 are constants for each category of software projects.

KLOC is estimated thousands of source lines of code

PM is unit for Person Months.

EAF is effort adjustment factor.

for example,

Let us suppose a project was estimated to be 400 KLOC. Using Basic COCOMO, calculate the effort and development time for each of the three models, i.e, organic, semi-detached and embedded.

Here,

for Basic COCOMO,

$$EFFORT = a_1 * (KLOC)^{a_2} \; PM$$

In organic mode,

$$EFFORT = 2.4 * (400)^{1.05} = 1295.31 \; PM$$

In semi-detached mode,

$$EFFORT = 3.0 * (400)^{1.12} = 2462.79 \; PM$$

In Embedded mode,

$$EFFORT = 3.6 * (400)^{1.20} = 4772.81 \; PM.$$