Java Institute for Advanced Technology

UNIT NAME: SOFTWARE ENGINEERING II

UNIT ID: HF2W 04

ASSIGNMENT ID: HF2W 04/AS/02

NAME: M.R.P.N.THARUKSHA RAJAPAKSHA

STUDENT ID: 2019/2020/CO/SE/I2/029

SCN NO: 207977608

NIC: 200019401866

BRANCH: JAVA INSTITUTE, COLOMBO

1. Why do you think quality standards should use when testing software?

Testing is essential as defects/bugs are recently discovered during shipment to the client who guarantees the quality of the program. It makes the program more robust and easy to use. The fully tried program ensures solid and high-performance software functionality.
Mind-boggling reports of program failures. The potential damage caused by software frustration can be seen in astonishing reports around the world. The basic testing method is to identify and evaluate whether the designed application meets the business requirements specifications. In addition to the specific software testing lifecycle (STLC) program enhancement life cycle, it can also be a non-stop handle. At each stage, it verifies functionality and approves the implementation of applications as needed.
Basically, the importance of software testing can be traced back to the user's reaction. It ensures the quality of the product and serves the customers as well. Moreover, it ensures better trade optimization (less support), unwavering quality, and reach out to popular clients. Significantly, however, the recurring efforts to invest in designing a flawless program are far-reaching and far-reaching. The importance of program testing is clear in the final quality assurance report. By successfully passing the extensive test levels, the final items can exceed the expected results. At each level, testers cannot identify faults but cannot anticipate such entanglements in the future. Other than that, each botch investigation leads to the birth of a comprehensive adaptation of the software.

2. Explain levels in CMM (Capability Maturity Model).

1. Initial (Level 1)
   The software process is characterized as inconsistent and sometimes confusing. Defined processes and standard practices in a crisis are abandoned. The success of the organization largely depends on the effort, skill, and heroism of the individual. Heroes eventually take their knowledge or lessons learned with them and move on to other organizations.

2. Repeatable/Managed (Level 2)
   The software development organization has basic and consistent project management processes to monitor this level of cost, schedule, and performance. The process is active to repeat the previous successes of projects with similar applications. Program management is a key feature of a second-tier organization.

3. Defined (Level 3)
   The software process in both management and engineering activities is documented, standardized, and integrated into a standard software program for the entire organization, using an approved, customized version of the organization's standard software process for the development, testing, and maintenance of all projects throughout the organization. Application.

4. Quantitatively Managed (Level 4)
   Management can effectively control the software development effort by using accurate measurements. At this level, the organization has set a quantitative quality target for both software processing and software maintenance. At this mature level, the activity of processes is controlled using statistical and other quantitative techniques and can be quantitatively predicted.

5. Optimizing (Level 5)
   The main feature of this level is the focus on continuous improvement of process performance through enhancements and innovative technological improvements. At this level, process changes are intended to improve process performance and at the same time maintain the statistical probability of achieving established quantitative process-improvement objectives.

3. Name 2 standards affected to the software testing and explain them.

1. ISO/IEC 9126
   This standard deals with the aspects to determine the quality of a software application such as quality model, external metrics, internal metrics, and quality in use metrics. This standard presents some set of quality attributes for any software such as functionality, reliability, usability, efficiency, maintainability, and portability. The above-mentioned quality attributes are further divided into sub-factors, which you can study when you study the standard in detail.

2. ISO/IEC 25000:2005
   ISO/IEC 25000:2005 is commonly known as the standard that provides the guidelines for Software Quality Requirements and Evaluation (SQuaRE). This standard helps in organizing and enhancing the process related to software quality requirements and their evaluations. In reality, ISO-25000 replaces the two old ISO standards, i.e. ISO-9126 and ISO-14598. SQuaRE is divided into sub-parts such as ISO 2500n − Quality Management Division, ISO 2501n − Quality Model Division, ISO 2502n − Quality Measurement Division, ISO 2503n − Quality Requirements Division, and ISO 2504n − Quality Evaluation Division. The main contents of SQuaRE are terms and definitions, reference models, general guide, individual division guides, and standards related to Requirement Engineering (i.e. specification, planning, measurement, and evaluation process).

4. Explain differences between validation and verification with examples.

Validation	**Verification**
Includes testing and validating the actual product.	Includes checking documents, designs, codes, and programs.
Dynamic testing.	Static testing.
Includes the execution of the code.	Does not include the execution of the code.
Checks whether the software meets the requirements and expectations of a customer or not.	Checks whether the software conforms to specifications or not.
Methods used are Black Box Testing, White Box Testing, and non-functional testing.	Methods used are reviews, walkthroughs, inspections, and desk-checking.
Can only find the bugs that could not be found by the verification process.	Can find the bugs in the early stage of development.

The goal is an actual product.	The goal is application and software architecture and specification.
Comes after verification.	Comes before validation.
Validation is executed on software code with the help of the testing team.	The quality assurance team does verification.
Consists of the execution of the program and are performed by the computer.	Consists of checking documents/files and is performed by humans.

Examples for validation and verification:

E.G. 1:-

A clickable Submit button

- Validation is done after the code is ready. Due to the validation test, the development team will make the submit button clickable.
- Verification will check the design document and correct spelling errors. Otherwise, the development team will create a button.

E.G. 2:-

Imagine you go to a restaurant/dinner party and order blueberry pancakes. When the waiter brings out your order, how do you tell if the food that came out is according to your order?

First, we look at it and observe these things such as Does the food look like pancakes in general?, Can you see the blueberries?, and Do they smell right?

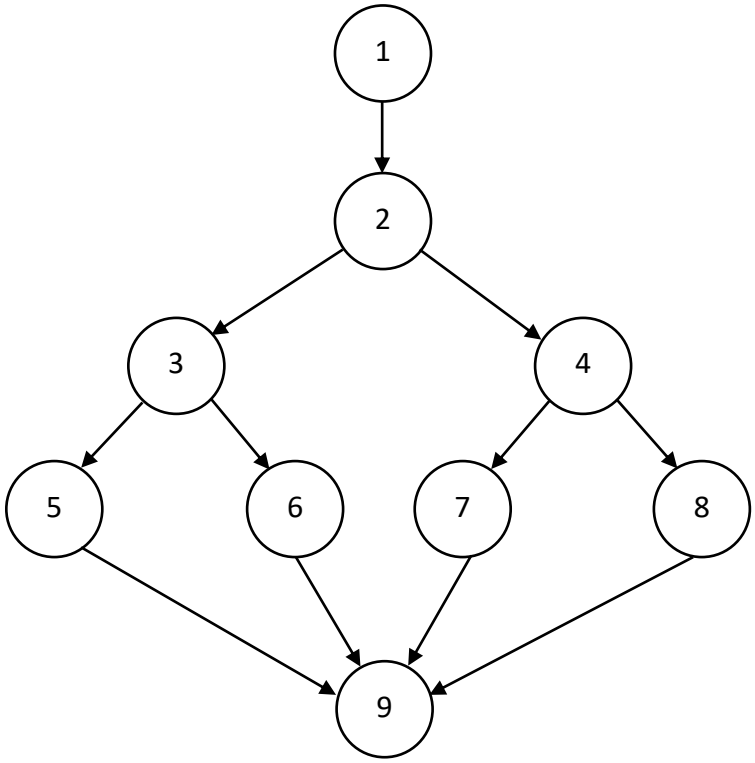
On the other hand, when you have to be absolutely sure that the food is as you expected, you will have to eat it.

- Validation is when you eat a product to see if it is right.
- Verification is when you have not yet eaten but check a few points by reviewing the subjects.

5. Draw the flow graph and find out the cyclomatic complexity of the following.

A. Printing the large number of three numbers.

1. Read a, b, c
2. If (a > b)
3. If (a > c)
4. Else if (b > c)
5. m = a
6. Else m = c
7. m = b
8. Else m = c2
9. Print m

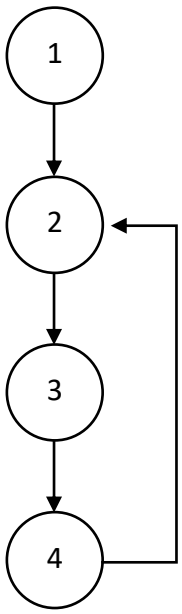


e = 11, n = 9, p = 1

Complexity = e – n + 2p
= 11 – 9 + 2*1
= 4 //

B. Printing numbers from 10 to 100. (20 marks)

1. m = 10
2. While m <= 100
3. Print m
4. m = m + 1



Complexity = e – n + 2p
= 4 – 4 + 2
= 2 //

6. What are the difference between white box and blackbox testing?

White Box Testing	Black Box Testing
A way of testing the software in which the tester has knowledge about the internal structure or the code or the program of the software.	A way of software testing in which the internal structure or the program or the code is hidden and nothing is known about it.
Mostly done by software developers.	Mostly done by software testers.
Require the knowledge of the implementation.	No need the knowledge of the implementation.
A structural test of the software	A functional test of the software.
Mandatory to have knowledge of programming.	No knowledge of programming is required.
The inner or the internal software testing.	Can be referred to as outer or external software testing.
The logic testing of the software.	The behavior testing of the software.
Started after detail design document.	Can be initiated on the basis of requirement specifications document.
Most time-consuming.	Least time-consuming.
Generally applicable to the lower levels of software testing.	Applicable to the higher levels of testing of software.
Suitable for algorithm testing.	Not suitable or preferred for algorithm testing.
Data domains along with inner or internal boundaries can be better tested.	Can be done by trial and error ways and methods.
Also called clear box testing.	Also called closed testing.