

ISO 9001:2008 Certified Institute

**JAVA INSTITUTE FOR ADVANCED TECHNOLOGY**

Department of Examinations



|  |  |
| --- | --- |
| COURSE(S) – (LEADING TO) | PROFESSIONAL HIGHER DIPLOMA IN SOFTWARE ENGINEERING |
| ASSIGNMENT STARTING DATE | 05th May 2020 |
| ASSIGNMENT CLOSING DATE | 12th May 2020 |
| UNIT NAME | SOFTWARE ENGINEERING II (SOFTWARE TESTING, QUALITY ASSURANCE AND MAINTENANCE) |
| UNIT ID | HF2W 04 |
| ASSIGNMENT ID | HF2W 04/AS/02 |
| DESCRIPTION | Understanding the term software test process’ (Individual Assignment) |
| DURATION | 1 Day |

**GUIDE LINES FOR CANDIDATES**

Students should describe step by step testing process which has specific steps to be executed in a definite sequence to ensure that the quality goals have been met. And student will understand each activity is carried out in a planned and systematic way.

|  |  |
| --- | --- |
| NAME : | KANDAGE DON ISHAN VIHANGA VIMUKTHI  .................................................................................................................................... (BLOCK CAPITALS) |
| NIC : | 960263812v  .................................................................................................................................... |
| SCN NO : | 177646148  .................................................................................................................................... |

Java Institute for Advanced Technology Sri Lanka

1. Why do you think quality standards should use when testing software? (10 marks)

ISO/IEC 29119-1: Deals with concepts and definitions of software (published: Sept ’13).

ISO/IEC 29119-2: Deals with test processes in a product (published: Sept ’13).

The ISO/IEC 29119-3: Deals with test documentation of the product (published: Sept ’13).

ISO/IEC 29119-4: Deals with testing techniques and strategies (published: 2014).

ISO/IEC 29119-5: Deals with keyword-based software testing (published: 2015).

1. Explain levels in CMM (Capability Maturity Model). (10Marks)

Capability Maturity Model is a bench-mark for measuring the maturity of an organization’s software process. It is a methodology used to develop and refine an organization’s software development process. CMM can be used to assess an organization against a scale of five process maturity levels based on certain Key Process Areas (KPA). It describes the maturity of the company based upon the project the company is dealing with and the clients. Each level ranks the organization according to its standardization of processes in the subject area being assessed.

A maturity model provides:

A place to start

The benefit of a community’s prior experiences

A common language and a shared vision

A framework for prioritizing actions

A way to define what improvement means for your organization

In CMMI models with a staged representation, there are five maturity levels designated by the numbers 1 through 5 as shown below:

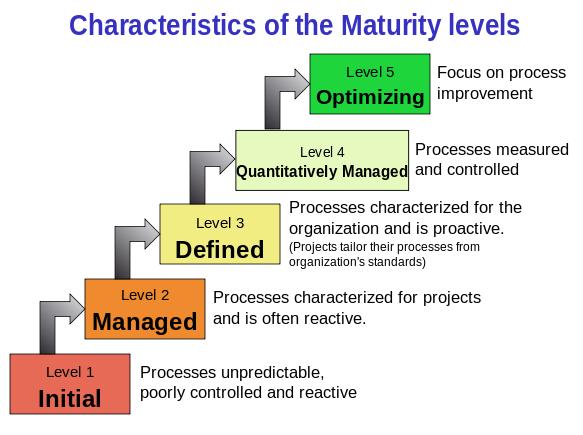
Initial

Managed

Defined

Quantitatively Managed

Optimizing



Maturity levels consist of a predefined set of process areas. The maturity levels are measured by the achievement of the specific and generic goals that apply to each predefined set of process areas. The following sections describe the characteristics of each maturity level in detail.

Maturity Level 1 – Initial: Company has no standard process for software development. Nor does it have a project-tracking system that enables developers to predict costs or finish dates with any accuracy.

In detail we can describe it as given below:

At maturity level 1, processes are usually ad hoc and chaotic.

The organization usually does not provide a stable environment. Success in these organizations depends on the competence and heroics of the people in the organization and not on the use of proven processes.

Maturity level 1 organizations often produce products and services that work but company has no standard process for software development. Nor does it have a project-tracking system that enables developers to predict costs or finish dates with any accuracy.

Maturity level 1 organizations are characterized by a tendency to over commit, abandon processes in the time of crisis, and not be able to repeat their past successes.

Maturity Level 2 – Managed: Company has installed basic software management processes and controls. But there is no consistency or coordination among different groups.

In detail we can describe it as given below:

At maturity level 2, an organization has achieved all the specific and generic goals of the maturity level 2 process areas. In other words, the projects of the organization have ensured that requirements are managed and that processes are planned, performed, measured, and controlled.

The process discipline reflected by maturity level 2 helps to ensure that existing practices are retained during times of stress. When these practices are in place, projects are performed and managed according to their documented plans.

At maturity level 2, requirements, processes, work products, and services are managed. The status of the work products and the delivery of services are visible to management at defined points.

Commitments are established among relevant stakeholders and are revised as needed. Work products are reviewed with stakeholders and are controlled.

The work products and services satisfy their specified requirements, standards, and objectives.

Maturity Level 3 – Defined: Company has pulled together a standard set of processes and controls for the entire organization so that developers can move between projects more easily and customers can begin to get consistency from different groups.

In detail we can describe it as given below:

At maturity level 3, an organization has achieved all the specific and generic goals.

At maturity level 3, processes are well characterized and understood, and are described in standards, procedures, tools, and methods.

A critical distinction between maturity level 2 and maturity level 3 is the scope of standards, process descriptions, and procedures. At maturity level 2, the standards, process descriptions, and procedures may be quite different in each specific instance of the process (for example, on a particular project). At maturity level 3, the standards, process descriptions, and procedures for a project are tailored from the organization’s set of standard processes to suit a particular project or organizational unit.

The organization’s set of standard processes includes the processes addressed at maturity level 2 and maturity level 3. As a result, the processes that are performed across the organization are consistent except for the differences allowed by the tailoring guidelines.

Another critical distinction is that at maturity level 3, processes are typically described in more detail and more rigorously than at maturity level 2.

At maturity level 3, processes are managed more proactively using an understanding of the interrelationships of the process activities and detailed measures of the process, its work products, and its services.

Maturity Level 4 – Quantitatively Managed: In addition to implementing standard processes, company has installed systems to measure the quality of those processes across all projects.

In detail we can describe it as given below:

At maturity level 4, an organization has achieved all the specific goals of the process areas assigned to maturity levels 2, 3, and 4 and the generic goals assigned to maturity levels 2 and 3.

At maturity level 4 Sub-processes are selected that significantly contribute to overall process performance. These selected sub-processes are controlled using statistical and other quantitative techniques.

Quantitative objectives for quality and process performance are established and used as criteria in managing processes. Quantitative objectives are based on the needs of the customer, end users, organization, and process implementers. Quality and process performance are understood in statistical terms and are managed throughout the life of the processes.

For these processes, detailed measures of process performance are collected and statistically analyzed. Special causes of process variation are identified and, where appropriate, the sources of special causes are corrected to prevent future occurrences.

Quality and process performance measures are incorporated into the organizations measurement repository to support fact-based decision making in the future.

A critical distinction between maturity level 3 and maturity level 4 is the predictability of process performance. At maturity level 4, the performance of processes is controlled using statistical and other quantitative techniques, and is quantitatively predictable. At maturity level 3, processes are only qualitatively predictable.

Maturity Level 5 – Optimizing: Company has accomplished all of the above and can now begin to see patterns in performance over time, so it can tweak its processes in order to improve productivity and reduce defects in software development across the entire organization.

In detail we can describe it as given below:

At maturity level 5, an organization has achieved all the specific goals of the process areas assigned to maturity levels 2, 3, 4, and 5 and the generic goals assigned to maturity levels 2 and 3.

Processes are continually improved based on a quantitative understanding of the common causes of variation inherent in processes.

Maturity level 5 focuses on continually improving process performance through both incremental and innovative technological improvements.

Quantitative process-improvement objectives for the organization are established, continually revised to reflect changing business objectives, and used as criteria in managing process improvement.

The effects of deployed process improvements are measured and evaluated against the quantitative process-improvement objectives. Both the defined processes and the organization’s set of standard processes are targets of measurable improvement activities.

Optimizing processes that are agile and innovative depends on the participation of an empowered workforce aligned with the business values and objectives of the organization.

The organization’s ability to rapidly respond to changes and opportunities is enhanced by finding ways to accelerate and share learning. Improvement of the processes is inherently part of everybody’s role, resulting in a cycle of continual improvement.

A critical distinction between maturity level 4 and maturity level 5 is the type of process variation addressed. At maturity level 4, processes are concerned with addressing special causes of process variation and providing statistical predictability of the results. Though processes may produce predictable results, the results may be insufficient to achieve the established objectives. At maturity level 5, processes are concerned with addressing common causes of process variation and changing the process (that is, shifting the mean of the process performance) to improve process performance (while maintaining statistical predictability) to achieve the established quantitative process-improvement objectives.

1. Name 2 standards affected to the software testing and explain them. (20 marks)

1. ISO/IEC/IEEE 29119-1

This software testing standard focuses on definitions and concepts of all other standards in the 29119 series of quality standards. It helps user understand the vocabulary on which the other standards in the series are built as well as provides relevant examples to show the way in which each concept works in practice.

A complete knowledge bank, the 29119-1 can also be considered to be the foundation of IEEE software testing standards. Some of the topics that are included in this series are introduction to software testing, testing processes in SDLC, risk-based testing, common test practices, defect management, etc.

2. ISO/IEC/IEEE 29119-2

This standard has been designed with an aim to develop a generic process model that can be used for conducting testing in any SDLC.

As per this standard, the testing process will work on a three-layer process that would include organizational test specifications such as organizational test policy and test strategy, test management and dynamic testing.

Laying a special emphasis on alleviation of risks, this standard allows the process of testing to focus on product’s key features and attributes under test.

1. Explain differences between validation and verification with examples. (10 marks)

|  |  |
| --- | --- |
| **Verification** | **Validation** |
| 1. Verification is a static practice of verifying documents, design, code and program. | 1. Validation is a dynamic mechanism of validating and testing the actual product. |
| 2. It does not involve executing the code. | 2. It always involves executing the code. |
| 3. It is human based checking of documents and files. | 3. It is computer based execution of program. |
| 4. Verification uses methods like inspections, reviews, walkthroughs, and Desk-checking etc. | 4. Validation uses methods like black box (functional) testing, gray box testing, and white box (structural) testing etc. |
| 5. Verification is to check whether the software conforms to specifications. | 5. Validation is to check whether software meets the customer expectations and requirements. |
| 6. It can catch errors that validation cannot catch. It is low level exercise. | 6. It can catch errors that verification cannot catch. It is High Level Exercise. |
| 7. Target is requirements specification, application and software architecture, high level, complete design, and database design etc. | 7. Target is actual product-a unit, a module, a bent of integrated modules, and effective final product. |
| 8. Verification is done by QA team to ensure that the software is as per the specifications in the SRS document. | 8. Validation is carried out with the involvement of testing team. |
| 9. It generally comes first-done before validation. | 9. It generally follows after verification. |

1. Draw the flow graph and find out the cyclomatic complexity of the following.
   1. Printing the large number of three numbers. (20 marks)

**cyclomatic complexity:**

cyclomatic complexity can be designed through the given formula

v(6) = e - n + 2p

where

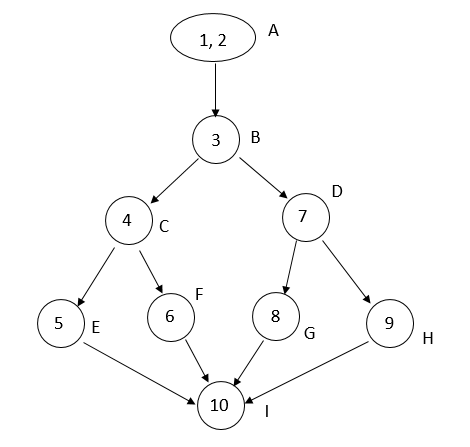
**e** is the no. of edges

**n** is the no. of nodes in the graph

**p** is the no. of components in the whole graph.

void main c { int a, b, c, max,

1. print f ('enter 3 integers"),
2. scan f (" %d %d %d", & a, & , & c),b
3. if (a >) b
4. if ( a > c)
5. Max = a
6. else max = c,
7. else if (b > c)
8. Max = b
9. else max = c
10. print and ( "a max = % d", max), 3



No. of independent paths

v (a) = e - n + 2

= 11 - 9 + 2

= 4

No. of procedures = 1

∴∴ cyclomatic complexity

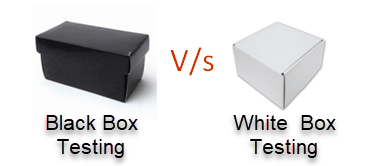
= e - n + 2 P

= 11 - 9 + 2

= 4

* 1. Printing numbers from 10 to 100. (20 marks)

1. What are the difference between white box and black box testing? (10 marks)



| **Parameter** | **Black Box testing** | **White Box testing** |
| --- | --- | --- |
| **Definition** | It is a testing approach which is used to test the software without the knowledge of the internal structure of program or application. | It is a testing approach in which internal structure is known to the tester. |
| **Alias** | It also knowns as data-driven, box testing, data-, and functional testing. | It is also called structural testing, clear box testing, code-based testing, or glass box testing. |
| **Base of Testing** | Testing is based on external expectations; internal behavior of the application is unknown. | Internal working is known, and the tester can test accordingly. |
| **Usage** | This type of testing is ideal for higher levels of testing like System Testing, Acceptance testing. | Testing is best suited for a lower level of testing like Unit Testing, Integration testing. |
| **Programming knowledge** | Programming knowledge is not needed to perform Black Box testing. | Programming knowledge is required to perform White Box testing. |
| **Implementation knowledge** | Implementation knowledge is not requiring doing Black Box testing. | Complete understanding needs to implement WhiteBox testing. |
| **Automation** | Test and programmer are dependent on each other, so it is tough to automate. | White Box testing is easy to automate. |
| **Objective** | The main objective of this testing is to check what functionality of the system under test. | The main objective of White Box testing is done to check the quality of the code. |
| **Basis for test cases** | Testing can start after preparing requirement specification document. | Testing can start after preparing for Detail design document. |
| **Tested by** | Performed by the end user, developer, and tester. | Usually done by tester and developers. |
| **Granularity** | Granularity is low. | Granularity is high. |
| **Testing method** | It is based on trial and error method. | Data domain and internal boundaries can be tested. |
| **Time** | It is less exhaustive and time-consuming. | Exhaustive and time-consuming method. |
| **Algorithm test** | Not the best method for algorithm testing. | Best suited for algorithm testing. |
| **Code Access** | Code access is not required for Black Box Testing. | White box testing requires code access. Thereby, the code could be stolen if testing is outsourced. |
| **Benefit** | Well suited and efficient for large code segments. | It allows removing the extra lines of code, which can bring in hidden defects. |
| **Skill level** | Low skilled testers can test the application with no knowledge of the implementation of programming language or operating system. | Need an expert tester with vast experience to perform white box testing. |
| **Techniques** | Equivalence partitioning is Black box testing technique is used for Black box testing.  Equivalence partitioning divides input values into valid and invalid partitions and selecting corresponding values from each partition of the test data.  Boundary value analysis  checks boundaries for input values. | Statement Coverage, Branch coverage, and Path coverage are White Box testing technique.  Statement Coverage validates whether every line of the code is executed at least once.  Branch coverage validates whether each branch is executed at least once  Path coverage method tests all the paths of the program. |
| **Drawbacks** | Update to automation test script is essential if you to modify application frequently. | Automated test cases can become useless if the code base is rapidly changing. |