Software Metrics Exam

Student Name: Islam Elshakhs

Student Number: 2016095

1. Design phase:

During the Design Phase, the system is designed to satisfy the requirements identified in the

previous phases. The requirements identified in the Requirements Analysis Phase are

transformed into a System Design Document that accurately describes the design of the system

and that can be used as an input to system development in the next phase.

OBJECTIVE/GOALS - DELIVERABLES AND APPROVALS .

2. Determine the list of quality requirements:

quality requirement refers to a condition or a capability that must be present in a requirement. They represent that which is needed to validate the successful completion of a project deliverable and here is a list of their most important requirements

-Functional suitability. -Reliability. -Operability. -Performance efficiency. -Security.

-Compatibility. -Maintainability. -Transferability.

3. Drawback of code coverage measurement:

Drawback of code coverage measurement is that it measures coverage of what has been written, i.e. the code itself; it cannot say anything about the software that has not been written.

If a specified function has not been implemented or a function was omitted from the specification, then structure-based techniques cannot say anything about them it only looks at a structure which is already there.

4.Requirements traceability:

Requirements traceability refers to the capability of a requirements management process or tool which enables the process participant or tool user to follow the life of a requirement both forwards and backwards. It also refers the the ability to link requirements (via specific relationships) to other constructs or artifacts of the product developmetnt lifecycle. Here are some examples of common requirements traceability relationships.

business goal -> detailed requirement

detailed requirement -> test script

detailed requirement -> system component

Requirements traceability often takes the physical form of a requirements traceability matrix (RTM), which is a manual spreadsheet or table that demonstrates the interconnections between requirements and business needs, other requirements, and/or deliverables. (The columns in a table or spreadsheet, for example, might list primary requirements, while the rows might list requirements that are somehow tied to them—thus creating a visual juxtaposition of related requirements.) A traceability matrix is the most common way to demonstrate requirements traceability. An example of a requirements traceability matrix that links requirements to test cases is viewable, the matrix method is commonly considered to be appropriate only for smaller projects. According to BABOK, “It is typically used when there are relatively few requirements or when tracing is limited to high-level requirements (e.g. features or models).” 8 Author Karl Weigers agrees, noting that it’s “impossible to perform requirements tracing manually for any but very small applications. You can use a spreadsheet to maintain traceability data for up to a couple hundred requirements, but larger systems demand a more robust solution.

5. Software Design:

Software design is a mechanism to transform user requirements into some suitable form, which helps the programmer in software coding and implementation. It deals with representing the client's requirement, as described in SRS (Software Requirement Specification) document, into a form, i.e., easily implementable using programming language.

The software design phase is the first step in SDLC (Software Design Life Cycle), which moves the concentration from the problem domain to the solution domain. In software design, we consider the system to be a set of components or modules with clearly defined behaviors & boundaries.

6. Goal/question/metric (GQM) paradigm:

The goal-question-metric (GQM) paradigm has been proposed to support the definition of quantifiable goals and the interpretation of collected measurement data. It is a goal-oriented approach to derive metrics from measurement goals to ensure that collected data is usable and serves a purpose.

In GQM method, measurement is goal-oriented. firstly, the goals need to be described clearly so that it can be measured during the software development.

In this method goals are defined which transforms into questions and metrics. Then the questions are answered and it is checked whether these answers satisfy goals or not. Hence, this method follows a top-down approach through division of goals and then mapping of goals into questions and then these questions are transformed into metrics, and method also follows bottom-up approach by analyzing measurement and checking whether goals are satisfied or not.

Steps in GQM Method

-Goals are defined and described clearly.

-Conversion of goals into appropriate questions.

-Questions are transformed into metrics.

7. Organization measures:

An object for grouping together several measure areas that contain changes to force elements, positions, organizational structures, and object assignments.

8. Cohesion:

Cohesion is a measure of the degree to which the elements of the module are functionally related. It is the degree to which all elements directed towards performing a single task are contained in the component. Basically, cohesion is the internal glue that keeps the module together. A good software design will have high cohesion.

Some Types of Cohesion:

Functional Cohesion: Every essential element for a single computation is contained in the component. A functional cohesion performs the task and functions. It is an ideal situation.

Sequential Cohesion: An element outputs some data that becomes the input for other element, i.e., data flow between the parts. It occurs naturally in functional programming languages.

Communicational Cohesion: Two elements operate on the same input data or contribute towards the same output data. Example - update record in the database and send it to the printer.

Temporal Cohesion: The elements are related by their timing involved. A module connected with temporal cohesion, all the tasks must be executed in the same time span. This cohesion contains the code for initializing all the parts of the system. Lots of different activities occur, all at unit time.

Logical Cohesion: The elements are logically related and not functionally. Ex- A component reads inputs from tape, disk, and network. All the code for these functions is in the same component. Operations are related, but the functions are significantly different.

9. Downtime:

Downtime is a computer industry term for the time during which a computer or IT system is unavailable, offline or not operational. Downtime has many causes, including shutdowns for maintenance (known as scheduled downtime), human errors, software or hardware malfunctions, and environmental disasters such as power outages, fires, flooding or major temperature changes. In industrial environments, downtime may refer to failures in production equipment. This type of downtime is often measured as downtime per work shift or downtime per a 12- or 24-hour period. Downtime duration is the period of time when a system fails to perform its primary function. Communications failures, for example, may cause network downtime , In IT environments, downtime can be one of the metrics used for system availability. Availability is often measured against a 100% operational or never-fails standard. A common standard of availability is 99.999%, known as “five 9s” availability. Two 9s would be a system that guarantees 99% availability in a one-year period, allowing up to 1% downtime, or 3.65 days of unavailability. Service level agreements (SLAs) often use monthly downtime or availability percentages for billing calculation. Scheduled downtime for system updates and routine maintenance is usually not included in the availability percentages for SLA contracts. For provisioning, service level agreements may use uptime and downtime percentages to describe the dependability of the various services available to clients. Such percentages also help determine the value of each service, as most clients desire continuous real-time availability (zero downtime).

10.Depth in Tree (DIT):

DIT definition: Depth of Inheritance Tree (DIT) is the maximum path length from the root to the class under consideration. Especially in the case of multiple inheritance, the maximum path length is of interest, because it builds a complex inheritance hierarchy. The shown class diagram defines the static parts of a system of classes with their associations and methods.

So-called software metrics, which quantify different properties of software products and processes, are used to measure software products. When measuring software, an object-oriented metric takes into account the summary of data structures and the methods applicable to them to form an object, its relationships to other objects and the general structural features of object-oriented programming( OOP).

The metric after Depth OF Inheritance Tree (DIT) is to be classified into the group of the measures, which are based on inheritance hierarchies and defined thereby particularly a measure for the classes standing by inheritance together in relation.