**BAHRIA UNIVERSITY, Karachi Campus)**

# Department of Software Engineering

# ASSIGNMENT # 01 – Spring 2024 Description of Software Quality Models CLO 02

|  |  |  |
| --- | --- | --- |
| Course Title: **Software Quality Engineering** |  | Course Code: **SEC-311** |
| Class: **BSE – 6(B)** |  | Shift: **Morning** |
| Course Instructor: **Engr. Misbah Perveen** |  | Date: **12th March 2024** |
| Due Date: **07th March 2024**  Name**: Muhammad Shoaib Akhter Qadri** |  | Max. Marks: **5.0 Marks**  Reg No: **79290** |

**Instructions:**

1. Use only A4 size blank white paper for printing.
2. Use only Times New Roman size 12 font.
3. Do not use color printer to print this title page or assignment.
4. Each heading (underlined, bold and in capital letters) and example must start from a new line.
5. Submit hard copy of your assignment and upload softcopy on LMS as a DOCX file.
6. Do not enclose your assignment in file/folder, staple assignment pages only at the left-top corner.
7. Do not edit (this) assignment file given as a pdf file.
8. Last page of your assignment must contain sources/references (use IEEE referencing style).
9. Assignments will only be accepted in the scheduled class/room.
10. No makeup assignments will be given.
11. Violation of any of the instructions mentioned here will result in marks deduction.

1. Suppose you have to ensure the quality of an online LMS System. You may consider various quality models to address this situation. Keep LMS in mind, describe (separately) each of the Software quality models as given below with respect to the following questions:

[3]

1. ***What*** are they?
2. ***Where*** are they used?
3. ***Why*** are they used?
4. ***How*** are they ensured / implemented?

1. FURPS
2. Dromey
3. ISO 9126
4. Boehm

Follow the format/sample as mentioned below to answer this question:

**Solution:**

**FURPS**

**WHAT?**

FURPS is an acronym representing a model for classifying software quality attributes, both functional and non-functional requirements.

**WHERE?**

FURPS is widely used in the software industry to assess and address quality aspects during software development.

**WHY?**

It helps evaluate critical dimensions of software quality, ensuring that the system meets user needs and performs reliably.

**HOW?**

By considering the following aspects:

**Functionality**: Covers capability, reusability, and security.

**Example**:

Consider an LMS used by a university. Its functionality includes features like course creation, content upload, discussion forums, and grade management.

**Usability**: Focuses on human factors, aesthetics, consistency, and responsiveness.

**Example**:

Imagine an LMS with a user-friendly interface. Learners can easily navigate, find course materials, and submit assignments.

**Reliability**: Addresses availability, predictability, and accuracy.

**Example**:

In an LMS, reliability ensures that students can access course materials consistently without unexpected downtime.

**Performance**: Includes speed, efficiency, and resource consumption.

**Example**:

Suppose an LMS handles thousands of concurrent users during peak times. Its performance metrics (loading speed, resource utilization) must meet expectations.

**Supportability**: Encompasses testability, flexibility, installability, and maintainability.

**Example**:

An LMS should be maintainable. Regular updates, bug fixes, and security patches are essential.

**Dromey:**

**WHAT?**

Dromey’s model emphasizes evaluating software quality by comparing one software product with another.

**Example:**

Suppose we have two competing Learning Management Systems (LMS). We can apply Dromey’s model to compare their quality attributes:

**Correctness**: Verify if both LMSs handle user interactions (e.g., submitting assignments, taking quizzes) correctly.

**Internal**: Assess the code quality, adherence to coding standards, and maintainability.

**Contextual**: Consider factors like the LMS’s intended audience (students, instructors) and the educational context (K-12, higher education).

**Descriptive**: Evaluate documentation, user guides, and support resources.

**WHERE?**

It provides insights into defects and identifies factors contributing to those defects.

**WHY?**

To pinpoint the causes of defects and improve software quality.

**HOW?**

By analyzing relationships between software properties and quality attributes. It considers correctness, internal, contextual, and descriptive factors.

**ISO 9126:**

**WHAT?**

ISO/IEC 9126 is an international standard for software quality. It categorizes quality into six main characteristics.

**WHERE?**

It applies to all software-intensive products, including safety-critical systems.

**WHY?**

To provide a comprehensive specification and evaluation model for software quality.

**HOW?**

By considering the following characteristics:

**Functionality**: Suitability, accuracy, interoperability, and security.

**Example**:

Check if it supports multiple languages, integrates with other systems, and handles various course formats.

**Reliability**: Maturity, fault tolerance, and recoverability.

**Example**:

Ensure that the LMS consistently provides accurate results and doesn’t crash during critical tasks.

**Usability**: Understandability, learnability, operability, and attractiveness.

**Example**:

Evaluate the user interface for clarity, ease of navigation, and accessibility.

**Efficiency**: Time behavior and resource utilization.

**Example**:

Measure loading times, resource utilization, and responsiveness.

**Maintainability**: Effort required for modifications.

**Example**:

Assess how easily the LMS can be updated, debugged, and modified.

**Portability**: Effort required to adapt to new environments.

**Example**:

Consider adaptability to different platforms (web, mobile, desktop).

**Boehm**

**WHAT?**

Introduced by Barry Boehm, this hierarchical quality model defines software quality using predefined attributes and metrics.

**WHERE?**

It serves as a guide for building quality into software products.

**WHY?**

To structure quality characteristics into primary uses, intermediate constructs, and primitive constructs.

**HOW?**

By assessing utility, maintainability, and portability. Quality factors include reliability, efficiency, usability, testability, understandability, and modifiability.

**Example:**

Let’s analyze an LMS using Boehm’s model:

**As Is the Utility**: Evaluate how effectively users can utilize the LMS out of the box.

**Maintainability**: Assess the effort required to fix errors during maintenance (e.g., updating course content, addressing security patches).

**Portability**: Consider how easily the LMS can adapt to different environments (e.g., cloud-based, on-premises).

**Quality Factors:**

**Reliability**: Ensure the LMS performs consistently and doesn’t fail unexpectedly.

**Efficiency**: Measure resource usage (CPU, memory) during peak usage.

**Usability**: Evaluate how intuitive and user-friendly the LMS is.

**Testability**: Assess the ease of verifying correct behavior.

**Understandability**: Consider how well users can grasp the LMS’s concepts.

**Modifiability**: Evaluate the effort required to modify the LMS during maintenance.

1. Summarize the five software quality models (as mentioned in the following table) with respect to the quality factors/attributes available in all the five models (write them in the first column). In the next five columns, mention which quality factor/attribute is available in which model. Sample is given in RED. Complete the following table. [1]

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Quality  Factor/Attribute | McCall | Boehm | FURPS | DROMEY | ISO 9126 |
| Testability | ✓ | ✓ |  |  | ✓ |
| Functionality | ✓ |  | ✓ |  | ✓ |
| Usability | ✓ |  | ✓ |  | ✓ |
| Reliability | ✓ |  | ✓ |  | ✓ |
| Performance | ✓ |  | ✓ |  | ✓ |
| Supportability | ✓ | ✓ | ✓ |  | ✓ |
| Maintainability | ✓ | ✓ | ✓ |  | ✓ |
| Portability | ✓ | ✓ | ✓ |  | ✓ |
| Security |  |  |  |  | ✓ |

1. Briefly explain how one can measure the manufacturer’s view of software quality. [1]

We can measure the manufacturer’s view of software quality in a concise and unique manner:

**Reliability**: Assess the likelihood of software failures or regressions. Measure production incidents, conduct reliability testing, and calculate average failure rates.

**Efficiency**: Evaluate resource usage (e.g., CPU, memory) during peak loads. Efficient software performs well without wasting resources.

**Security**: Gauge the software’s ability to withstand attacks. Consider vulnerability assessments, penetration testing, and adherence to security standards.

**Maintainability**: Measure how easily the software can be updated, debugged, and modified. Low maintenance effort leads to better quality.