UNIT 5

SOFTWARE QUALITY MODELS

1. McCall Model

McCall’s model was developed by the Rome air development center (RADC), the US air-force electronic system decision (ESD), general electric, in order to improve the quality of software products at **software development companies**. The model was developed to assess the relationships between external factors and product quality criteria. The quality characteristics were classified in three major types, eleven such factors which describe the external view of the software (user view), 23 quality criteria which describe the internal view of the software (developer view), and the metrics which define and are used to provide a scale and method for measurement. The total number of factors was reduced to eleven in order to simplify it. Those factors are Correctness, Integrity, Reliability, Efficiency, Usability, Flexibility, Maintainability, Reusability, Testability, Portability, and Interoperability. The major contribution of this model is the relationship between the quality characteristics and metrics. But, this model does not consider directly on the functionality of software products.

2. Boehm Model

Boehm added new factors to McCall’s model with emphasis on the maintainability of software product at **software development companies**. The main aim of this model is to address the contemporary shortcomings of models that automatically and quantitatively evaluate the quality of software. Thus, Boehm model represents the characteristics of the software product hierarchically in order to get contribute in the total quality. Also, the software product evaluation considered with respect to the utility of the program. But, this model contains only a diagram without any suggestion about measuring the quality characteristics.

3. FURPS Model

FURPS model was proposed by and Hewlett-Packard Co and Robert Grady. The attributes were classified into two main categories according to the user’s requirements, the functional and non-functional requirements. Functional requirements (F): Defined by input and expected output. Non-functional requirements (URPS):

Usability, reliability, performance, supportability. Also, this model was extended by IBM Rational Software – into FURPS+. Thus, this model considered only the user’s requirements and disregards the developer consideration. But, this model fails to take into account the software some of the product characteristics, like maintainability and portability.

4. Dromey Model

Dromey (1995) states that the evaluation is different for each product, thus a dynamic idea for process modeling is required. Thus, the main idea of the proposed model was to obtain a model broad enough to work for different systems. Also the model seeks to increase understanding of the relationship between the attributes (characteristics) and the sub-attributes (sub-characteristics) of quality. Also this model defined two layers, the high-level attributes and subordinate attributes. Therefore, this model suffers from lack of criteria for measurement of software quality.

5. ISO IEC 9126 Model

As, many software quality models were proposed, the confusion happened and new standard model was required. Thus, ISO/IEC JTC1 began to develop the required consensus and encourage standardization world-wide. The ISO 9126 is part of the ISO 9000 standard, and it is the most important standard for quality assurance. The first considerations originated in 1978, and in the year 1985 the development of ISO/IEC 9126 was started.

In this model, for **software development companies,** the totality of software product quality attributes was classified in a hierarchical tree structure of characteristics and sub characteristics. And the highest level of this structure consists of the quality characteristics and the lowest level consists of the software quality criteria. This model specified six characteristics including Functionality, the Reliability, Usability, Efficiency, Maintainability and the Portability; all of which are further divided into 21 sub characteristics. All these sub characteristics are manifested externally when the software is used as part of a computer system, and thus are the result of internal software attributes. All the defined characteristics are applicable to every kind of software, including computer programs and data contained in firmware and provide consistent terminology for software product quality. And they also provide a framework for making trade-offs between software product capabilities.