7.1 Basic Concepts

Software Reliability means Operational reliability. It is described as the ability of a system or component to perform its required functions under static conditions for a specific period.

Software reliability is also defined as the probability that a software system fulfills its assigned task in a given environment for a predefined number of input cases, assuming that the hardware and the input are free of error.

Software Reliability is an essential connect of software quality, composed with functionality, usability, performance, serviceability, capability, installability, maintainability, and documentation. Software Reliability is hard to achieve because the complexity of software turn to be high. While any system with a high degree of complexity, containing software, will be hard to reach a certain level of reliability, system developers tend to push complexity into the software layer, with the speedy growth of system size and ease of doing so by upgrading the software.

For example, large next-generation aircraft will have over 1 million source lines of software on-board; next-generation air traffic control systems will contain between one and two million lines; the upcoming International Space Station will have over two million lines on-board and over 10 million lines of ground support software; several significant life-critical defense systems will have over 5 million source lines of software. While the complexity of software is inversely associated with software reliability, it is directly related to other vital factors in software quality, especially functionality, capability, etc.

7.2 Software quality

Software quality product is defined in term of its fitness of purpose. That is, a quality product does precisely what the users want it to do. For software products, the fitness of use is generally explained in terms of satisfaction of the requirements laid down in the SRS document.

Example: Consider a functionally correct software product. That is, it performs all tasks as specified in the SRS document. But, has an almost unusable user interface. Even though it may be functionally right, we cannot consider it to be a quality product.

The modern view of a quality associated with a software product several quality methods such as the following:

Portability: A software device is said to be portable, if it can be freely made to work in various operating system environments, in multiple machines, with other software products, etc.

Usability: A software product has better usability if various categories of users can easily invoke the functions of the product.

Reusability: A software product has excellent reusability if different modules of the product can quickly be reused to develop new products.

Correctness: A software product is correct if various requirements as specified in the SRS document have been correctly implemented.

Maintainability: A software product is maintainable if bugs can be easily corrected as and when they show up, new tasks can be easily added to the product, and the functionalities of the product can be easily modified, etc.

Software Quality Management System

A quality management system is the principal methods used by organizations to provide that the products they develop have the desired quality.

A quality system subsists of the following:

Managerial Structure and Individual Responsibilities: A quality system is the responsibility of the organization as a whole. However, every organization has a several quality department to perform various quality system activities. The quality system of an arrangement should have the support of the top management.

Quality System Activities: The quality system activities encompass the following:

- Auditing of projects
- Review of the quality system
- Development of standards, methods, and guidelines, etc.
- Production of documents for the top management summarizing the effectiveness of the quality system in the organization.

7.3 Software reliability model

A software reliability model indicates the form of a random process that defines the behaviour of software failures to time.

Software reliability models have appeared as people try to understand the features of how and why software fails, and attempt to quantify software reliability.

Over 200 models have been established since the early 1970s, but how to quantify software reliability remains mostly unsolved.

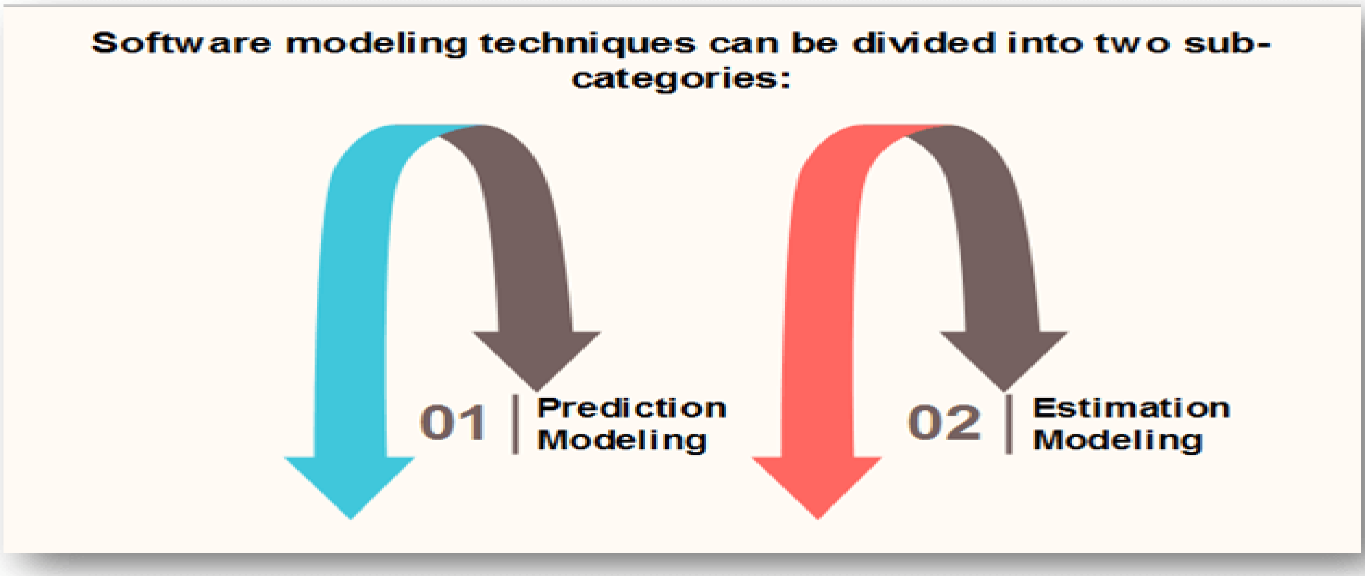
There is no individual model that can be used in all situations. No model is complete or even representative.

Most software models contain the following parts:

- Assumptions
- Factors

A mathematical function that includes the reliability with the elements. The mathematical function is generally higher-order exponential or logarithmic.

Software Reliability Modelling Techniques


Software modeling techniques can be divided into two sub-categories:
01 | Prediction Modeling
02 | Estimation Modeling

Both kinds of modeling methods are based on observing and accumulating failure data and analyzing with statistical inference.

Differentiate between software reliability prediction models and software reliability estimation models

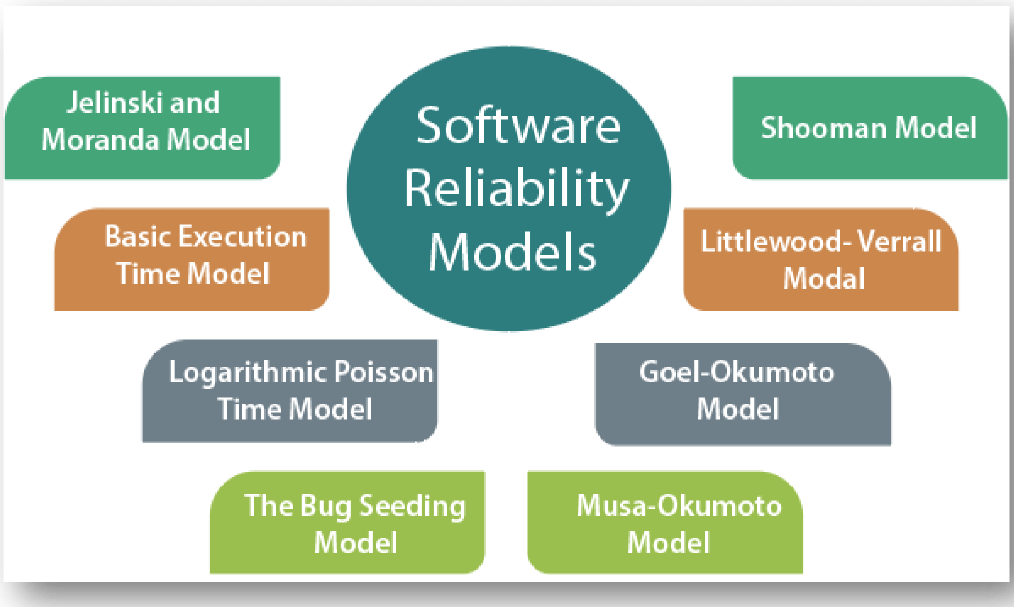
Basics	Prediction Models	Estimation Models
Data Reference	Uses historical information	Uses data from the current software development effort.
When used in development cycle	Usually made before development or test phases; can be used as early as concept phase.	Usually made later in the life cycle (after some data have been collected); not typically used in

		concept or development phases.
Time Frame	Predict reliability at some future time.	Estimate reliability at either present or some next time.

Reliability Models

A reliability growth model is a numerical model of software reliability, which predicts how software reliability should improve over time as errors are discovered and repaired. These models help the manager in deciding how much efforts should be devoted to testing. The objective of the project manager is to test and debug the system until the required level of reliability is reached.

Following are the Software Reliability Models are:


Software Reliability Models
Jelinski and Moranda Model
Shooman Model
Basic Execution Time Model
Littlewood- Verrall Modal
Logarithmic Poisson Time Model
Goel-Okumoto Model
The Bug Seeding Model
Musa-Okumoto Model

7.4 Capability maturity model(CMM)

CMM was developed by the Software Engineering Institute (SEI) at Carnegie Mellon University in 1987.

- It is not a software process model. It is a framework that is used to analyze the approach and techniques followed by any organization to develop software products.
- It also provides guidelines to further enhance the maturity of the process used to develop those software products.

- This model describes a strategy for software process improvement that should be followed by moving through 5 different levels.
- Each level of maturity shows a process capability level. All the levels except level-1 are further described by Key Process Areas (KPA's).

Key Process Areas (KPA's):

Each of these KPA's defines the basic requirements that should be met by a software process in order to satisfy the KPA and achieve that level of maturity.

Conceptually, key process areas form the basis for management control of the software project and establish a context in which technical methods are applied, work products like models, documents, data, reports, etc. are produced, milestones are established, quality is ensured and change is properly managed.

The 5 levels of CMM are as follows:

Level-1: Initial –

- No KPA's defined.
- Processes followed are Adhoc and immature and are not well defined.
- Unstable environment for software development.
- No basis for predicting product quality, time for completion, etc.

Level-2: Repeatable –

- Focuses on establishing basic project management policies.
- Experience with earlier projects is used for managing new similar natured projects.
- Project Planning- It includes defining resources required, goals, constraints, etc. for the project. It presents a detailed plan to be followed systematically for the successful completion of good quality software.
- Configuration Management- The focus is on maintaining the performance of the software product, including all its components, for the entire lifecycle.
- Requirements Management- It includes the management of customer reviews and feedback which result in some changes in the requirement set. It also consists of accommodation of those modified requirements.
- Subcontract Management- It focuses on the effective management of qualified software contractors i.e. it manages the parts of the software which are developed by third parties.

- Software Quality Assurance- It guarantees a good quality software product by following certain rules and quality standard guidelines while developing.

Level-3: Defined –

- At this level, documentation of the standard guidelines and procedures takes place.
- It is a well-defined integrated set of project-specific software engineering and management processes.
- Peer Reviews- In this method, defects are removed by using a number of review methods like walkthroughs, inspections, buddy checks, etc.
- Intergroup Coordination- It consists of planned interactions between different development teams to ensure efficient and proper fulfillment of customer needs.
- Organization Process Definition- Its key focus is on the development and maintenance of the standard development processes.
- Organization Process Focus- It includes activities and practices that should be followed to improve the process capabilities of an organization.
- Training Programs- It focuses on the enhancement of knowledge and skills of the team members including the developers and ensuring an increase in work efficiency.

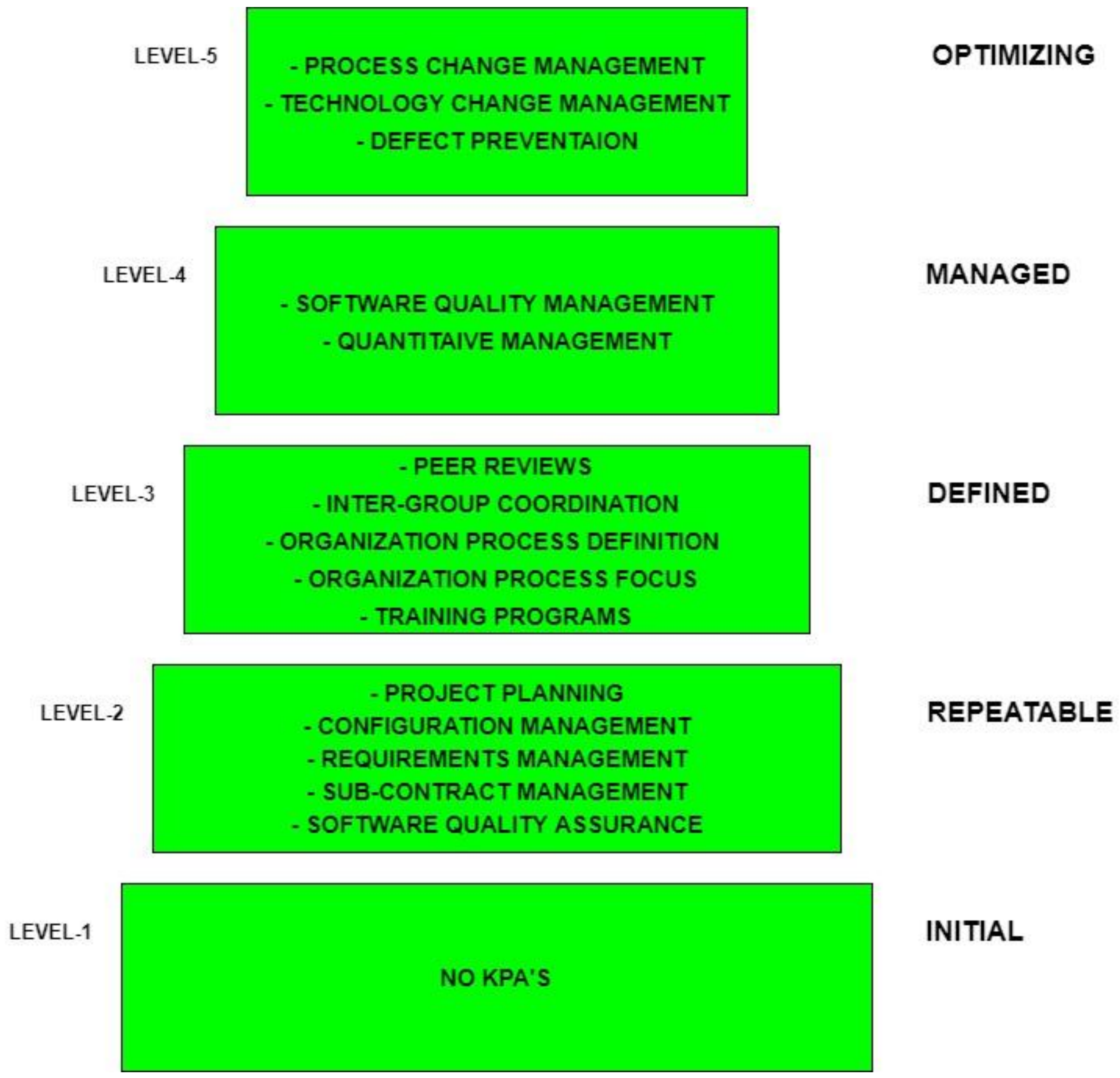
Level-4: Managed –

- At this stage, quantitative quality goals are set for the organization for software products as well as software processes.
- The measurements made help the organization to predict the product and process quality within some limits defined quantitatively.
- Software Quality Management- It includes the establishment of plans and strategies to develop quantitative analysis and understanding of the product's quality.
- Quantitative Management- It focuses on controlling the project performance in a quantitative manner.

Level-5: Optimizing –

- This is the highest level of process maturity in CMM and focuses on continuous process improvement in the organization using quantitative feedback.
- Use of new tools, techniques, and evaluation of software processes is done to prevent recurrence of known defects.
- Process Change Management- Its focus is on the continuous improvement of the organization's software processes to improve productivity, quality, and cycle time for the software product.

- Technology Change Management- It consists of the identification and use of new technologies to improve product quality and decrease product development time.
- Defect Prevention- It focuses on the identification of causes of defects and prevents them from recurring in future projects by improving project-defined processes.


LEVEL-5
- PROCESS CHANGE MANAGEMENT
- TECHNOLOGY CHANGE MANAGEMENT
- DEFECT PREVENTAION
OPTIMIZING
LEVEL-4
- SOFTWARE QUALITY MANAGEMENT
- QUANTITAIVE MANAGEMENT
MANAGED
LEVEL-3
- PEER REVIEWS
- INTER-GROUP COORDINATION
- ORGANIZATION PROCESS DEFINITION
- ORGANIZATION PROCESS FOCUS
- TRAINING PROGRAMS
DEFINED
LEVEL-2
- PROJECT PLANNING
- CONFIGURATION MANAGEMENT
- REQUIREMENTS MANAGEMENT
- SUB-CONTRACT MANAGEMENT
- SOFTWARE QUALITY ASSURANCE
REPEATABLE
LEVEL-1
NO KPA'S
INITIAL