5.0 RELIABILITY, QUALITY MANAGEMENT & MAINTENANCE

5.1 Understand the concept of Software Reliability

The reliability of a software product essentially denotes its **trustworthiness** and **dependability**. Alternatively, the reliability of a software product can also be defined as the probability of the product working correctly over a given period of time.

It is obvious that a software product having a large number of defects is unreliable. It is also very reasonable to assume that the reliability of a system improves as the number of defects in it is reduced.

If an error is removed from an instruction that is frequently executed then large improvement of reliability figure. On the other hand, removing errors from parts of the program that are rarely used, may not cause any appreciable change to the reliability of the product.

The main reasons that make software reliability more difficult to measure than Hardware reliability:

- The reliability improvement due to fixing a single bug depends on where the bug is located in the code.
- The perceived reliability of a software product is observer dependent.
- The reliability of a product keep changing as error are detected and fixed.
-

5.1.1 Differentiate Hardware Reliability and Software Reliability;

Hardware components fail due to very different reasons as compared to software components. Hardware components fail mostly due to wear and tear, whereas software components fail due to bugs.

To a fix hardware fault, one has to either replace or repair the failure part. In contrast, a software product would continue to fail until the error is tracked down and either the design or the code is changed to fix the bug. For this reason when a hardware part is repaired its reliability would be maintained at the level that existed before the failure occurred. Whereas when the software failure is repaired, the reliability may either increase or decrease.

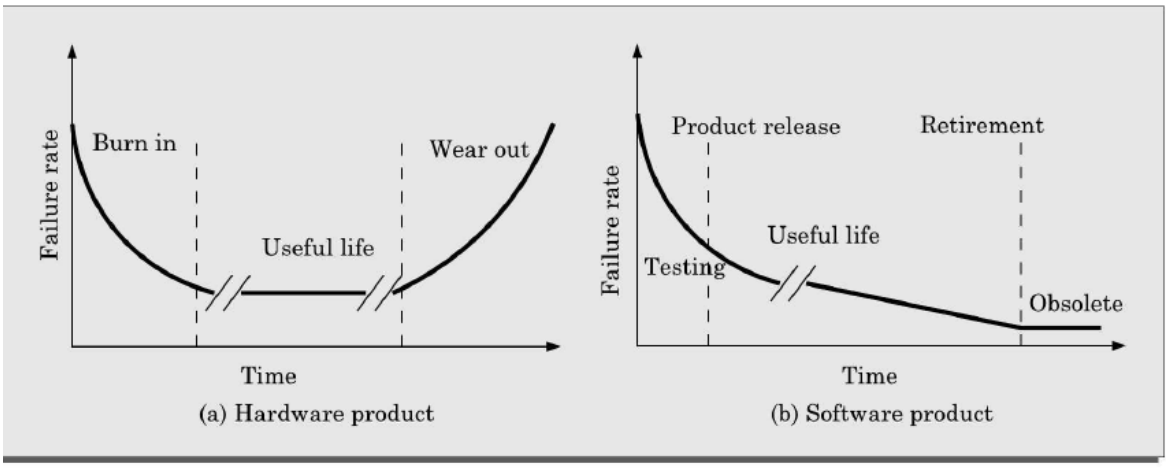


Figure : Change in failure rate of a product.

A comparison of the changes in failure rate over the product lifetime for a typical hardware and software product are shown in figure.

For hardware components the failure rate is initially high, but decreases as the faulty components are identified are either repaired or replaced. The system then enters its useful life, where the rate of failure is almost constant. After some time, the major components wear out and the failure rate increases.

The software product shows the highest failure rate just after purchase and installation. As the system is used, more and more errors are identified and removed resulting in reduced failure rate.

5.1.2 List the different Reliability Metrics

1. Rate of OCurrence Of Failure (ROCOF): ROCOF measures the frequency of occurrence of failures. ROCOF measure of a software product can be obtained by observing the behaviour of a software product in operation over a specified time interval and then calculating the ROCOF value as the ratio of the total number of failures observed and the duration of observation.

2. Mean Time to Failure (MTTF): MTTF is the time between two successive failures, averaged over a large number of failures. To measure MTTF we can record the failure data on *n* failures. Let the failures occur at the time instants *t1,t2,..,tn*. Then MTTF can be calculated as $\sum_{i=1}^{n} \frac{t_{i+1}-t_i}{(n-1)}$.

3. Mean Time To Repair (MTTR): Once the failure occurs, some time is required to fix the error. MTTR measures the average time it takes to track the errors causing the failure and to fix them.

4. Mean Time Between Failure (MTBF): The MTTF and MTTR metrics can be combined to get the MTBF metric: MTBF = MTTF + MTTR. Thus MTBF of 300 hours indicates that once a failure occur, the next failure is expected after 300 hours.

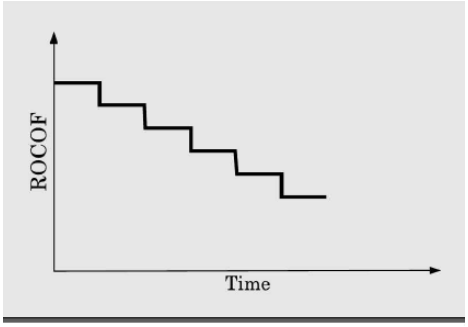
5. Probability Of Failure On Demand (POFOD): POFOD measures the likelihood of the system failing when a service request is made. For example, a POFOD of 0.001 wound mean that 1 out of every 1000 service requests would result in a failure.

6. Availability: Availability of a system is a measure of how likely would the system is available for use over a given period of time. This metric not only considers the number of failures occurring during the time interval, but also takes into account the repair time when a failure occurs.

5.1.3 Understand the Reliability Growth Modelling:

A reliability growth model can be used to predict when (or if at all) a particular level of reliability is likely to be attained. Thus, reliability growth modelling can be used to determine when to stop testing to attain a given reliability level.

Jelinski and Moranda Model: The simplest reliability growth model assumes that reliability increases by a constant increase each time an error is detected and repaired. This model assumes that all errors contribute equally to reliability growth is highly unrealistic.



Step function model of reliability growth.

Littlewood and Veralls Model: This model allows for negative reliability growth to reflect the fact that when a repair is carried out, it may introduce additional errors. It also models the fact that errors are repaired the product reliability per repair decreases.

5.2 Define Statistical Testing

Statistical testing is a testing process whose objective is to determine the reliability of the product rather than discovering errors. To carry out statistical testing we need to first define the operation profile of the product.

Operation Profile: Formally, we can define the operation profile of a software as the probability of a user selecting the different functionalities of the software. If we denote the set of various functionalities offered by the software by {fi}, the operational profile would associate with each function {fi} with the probability with which an average user would select {fi} as his next function to use.

The operational profile of a software product can be determined by observing the usage pattern of the software by number of users

Steps in Statically Testing: The first step is to determine the operation profile of the software. The next step is to generate a set of test data corresponding to the determined operation profile. The third step is to apply the test cases to the software and record the time between each failure. Next the reliability can be computed.

Pros and Cons of Statistical Testing:

Pros: The reliability estimation arrived by using statistical testing is more accurate.

Cons: There is no simple and repeatable way of defining operational profile. The number of test cases should be statistically significant.

5.3 Define Software Quality:

Traditionally the quality of a product is defined in terms of its fitness of purpose. The modern view of a quality associates with a software product several quality factors such as

1. Portability: A software product is said to be portable, if it can be easily make to work in different hardware and OS environments.
2. Usability: If different categories of users can easily invoke the functions of the product.
3. Reusability: If different modules of the product can easily be reused to develop new product.
4. Correctness: If different requirements as specified in the SRS document have been correctly implemented.
5. Maintainability: If errors can be easily corrected, new functions can be easily added to the product.

5.4 Software Quality Management System:

A quality management system (also referred to as quality systems) is the principal methodology used by organisations to ensure that the product they develop has the desired quality.
A quality system is the responsibility of the organisation as a whole. Every organisation has a separate quality department. The quality system of an organisation should have the full support of the top management.
The quality system activities encompass the following

-Auditing of projects
-Review of the quality system
-Development of standards, procedures and guidelines etc.,
-Production of reports for the top management summarizing the effectiveness of the quality system in the organisation.

5.4.1 Understand the Evolution of Quality Systems:

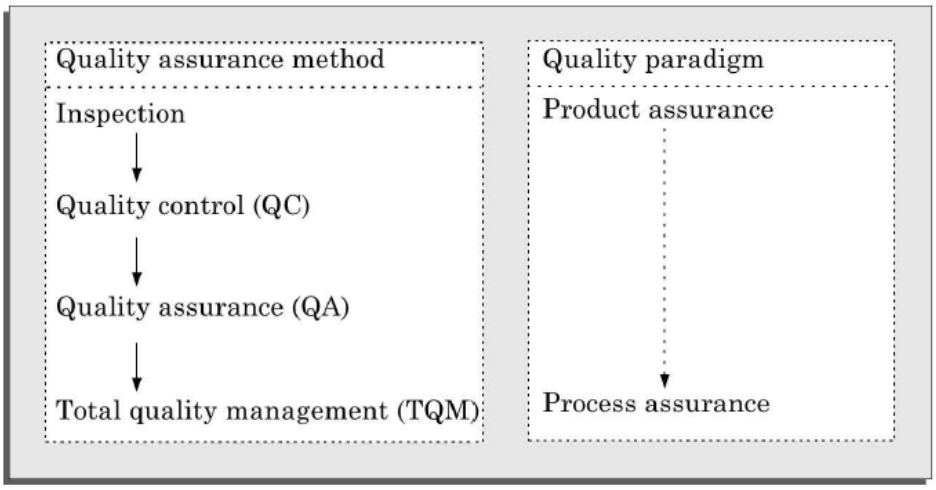
Quality systems have rapidly evolved over the last six decades. Prior to World War II the usual method to produce quality products was to inspect the finished products to eliminate defective products. The initial product inspection method gave way to quality control (QC) principles.

Quality Control (QC) focuses not only on detecting the defective products and eliminating them, but also on determining the causes behind the defects, so that the product rejection rate can be reduced.

The basic principle of modern quality assurance is that if organisations processes are good and are followed rigorously, then the product are bound to be of good quality. The modern Quality assurance model includes guidance for recognizing, defining, analysing, and improving the production process.

Total Quality Management (TQM) advocates that the process followed by an organisation must continuously be improved through process measurements.

Product metrics help measure the characteristics of a product being developed, whereas process metrics help measure how a process is performing.



Evolution of quality system and corresponding shift in the quality paradigm.

5.5 Define SEI Capability Maturity Model:

SEI capability Maturity Model (SEI CMM) was proposed by Software Engineering Institute (SEI) of the Carnegie Mellon University,USA.

CMM is a reference model for appraising the software process maturity into different levels. It must be remembered that SEI CMM can be used two ways capability evaluation and software process assessment. Capability evaluation provides a way to assess the software process capability of an organisation. Software process assessment is used by an organisation with the objective to improve its own process capability.

SEI CMM classifies software development industries into the following five maturity levels.

Level 1: Initial – A software development organisation at this level is characterised by adhoc activities. Very few or no processes are defined and followed.

Level 2: Repeatable – At this level, the basic project management practices such as tracking cost and schedule are established. Configuration management tools are used on items identified for configuration control. Size and Cost estimation techniques such as function point analysis, COCOMO etc., are used.

Level 3: Defined – At this level, the processes for both management and development activities are defined and documented. There is a common organisation wide understanding of activities, roles and responsibilities. The process through defined, the process and product qualities is not measured.

Level 4: Managed – At this level, the focus is on software metrics. Both process and product metrics are collected. Quantitative quality goals are set for the products and at the time of completion of development it checked whether the quantitative quality goals for the product are met.

Level 5: Optimizing – At this stage, process and product metrics are collected. Process and product measurement data are analysed for continuous process improvement. At CMM Level 5 an organisation would identify the best software engineering practices and innovaitions (which may be tools, methods, or processes).

Table Focus areas of CMM levels and their KPAs

CMM Level	*Focus*	*Key Process Areas (KPAs)*
Initial	Competent people	
Repeatable	Project management	Software project planning
		Software configuration management
Defined	Definition of	Process definition
	processes	Training program
		Peer reviews
Managed	Product and	Quantitative process metrics
	process quality	Software quality management
Optimizing	Continuous process	Defect prevention
	improvement	Process change management
		Technology change management