9

Software Relibility and Quality Assurance

Software reliability refers to the dependability of the software product and is one of the important non-functional characteristics of software. Software reliability is the probability of failure-free operation of a computer program in a specified operating environment for a specified period of time. Software product having a large number of defects is unreliable and the reliability improves if the numbers of defects are reduced. According to the IEEE standard, software reliability is defined as the ability of a system or a component to perform its required functions under stated conditions for a specified period of time. Software reliability means operational reliability; it reflects the product's ability to operate failure free in the environment for which it was designed.

9.1 VERIFICATION AND VALIDATION

Software Verification and Validation (V&V) is the process of checking and analyzing software to ensure that it conforms to its specifications. V&V activities should be carried out at each stage of the software development process to ensure that the results of each stage are as specified. According to the IEEE standard, "Software V&V is a disciplined approach to assess software products throughout the product life-cycle. A V&V effort strives to ensure that quality is built into the software and that the software satisfies user requirements". V&V activities directly address the quality of software product, and use testing techniques to locate and fix the defects. The V&V process determines whether or not products of a given development or maintenance activity conform to the requirement of that activity, and whether or not the final software product fulfills its intended purpose and meets user requirements.

Verification is an attempt to ensure that the product is built right, that is, the products of an activity meet the requirements / specifications imposed at the beginning of the activity. *Validation* is an attempt to

ensure that the right product is built, that is, the product fulfills its specific intended purpose.

The V&V process begin early in the development or maintenance phase. The purpose of planning V&V activities is to ensure that each resource, role, and responsibility is clearly assigned. The resulting V&V plan documents and describes the various activities, resources and their roles, as well as the techniques and tools to be used. A proper understanding of each V&V activity will help in carefully planning the resources and techniques needed to fulfill the different purposes of the activity.

9.2 SOFTWARE RELIABILITY ISSUES

Reliability is a complex concept that should always be considered at the system level rather than at the level of an individual component. Failure is non-conformance to system requirements and is a function of inputs to the system. As the components in a system are interdependent, a failure in one component can be propagated through the system and affect the operation of other components. In a computer-based system, both hardware reliability and software reliability must be considered when specifying the overall system reliability:

- ***Hardware Reliability:*** Most hardware failures are caused due to wear and tear of its components. A failed component is either repaired or replaced to fix the faults. What is the probability of failure of a hardware component and how long does it take to repair that component?

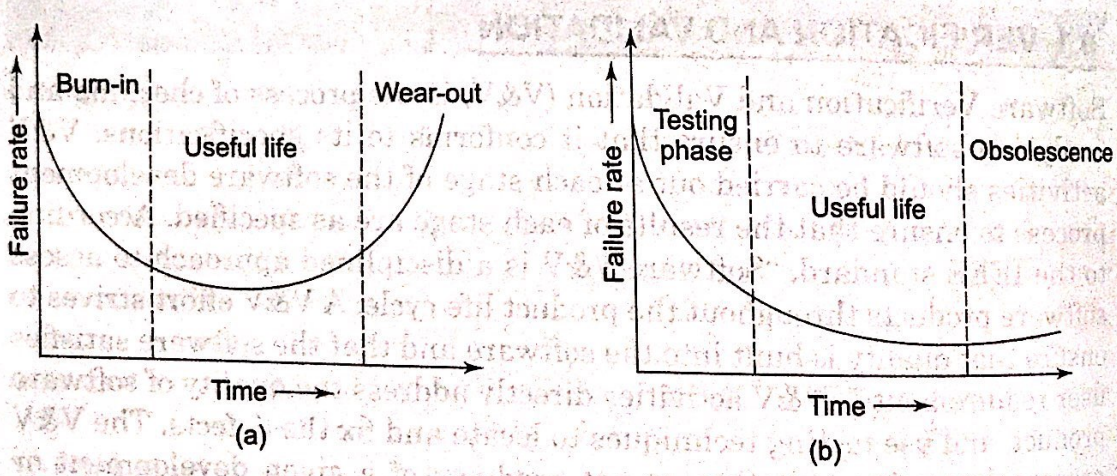

Fig. 9.1: Reliability Curve (Failure Rate vs. Time)
(a) Hardware product (b) Software product

- ***Software Reliability:*** Software failures are different from hardware failures in that software does not wear out. It can continue to operate correctly even with errors that are yet to be uncovered. However, errors should be tracked down to increase

software reliability, and either the design or the code is changed to fix the errors. The aim is to increase the time between failures. How likely is it that a software component will produce an incorrect output?

The plot of hardware reliability and software reliability over time is given in Fig. 9.1. For hardware products, it can be seen that the failure rate is high initially but decreases as the faulty components are first identified and then either repaired or removed. The system then enters its useful life. After some time (called product life time) the components wear out, and the failure rate increases. On the other hand, for software, the failure rate is highest during system integration and testing phase. As the system is tested, more and more errors are identified and removed resulting in reduced failure rate during the useful life of the product. As the software becomes obsolete due to changing requirements or the operating environment, no error corrections occurs and the failure rate remains unchanged.

Failure behavior for software is affected by the number of defects in the software being executed and the operating environment. There are four general ways of characterizing failure occurrences in time:

1. time of failure,
2. time interval between failures,
3. cumulative failure experienced up to a given time, and
4. failures experienced in a time interval.

9.3 SOFTWARE RELIABILITY METRICS

Reliability metrics are used to quantitatively express the reliability of a software product that may be useful due to a number of reasons as stated below:

- To evaluate software engineering technology quantitatively.
- To evaluate development status of a project.
- To monitor the operational performance of software.
- To control enhancement of features and changes to the design of the software.
- A quantitative understanding of software quality and the various factors influencing it and affected by it enriches the software development process and hence the software product.

Reliability metrics used to quantify the reliability of a software product are as follows:

1. **MTTF (Mean Time to Failure)**: MTTF is the mean time for which a component is expected to be operational. It is the most widely

used hardware reliability metric. The MTTF is the average time between observed system failures. The units used to express time are totally dependent on the system and it can even be specified by the number of transactions, as in the case of database query systems.

2. **MTTR (Mean Time to Repair):** MTTR measures the average time it takes to track the errors causing the failure and then to fix them. MTTR can also be defined as the average time required to replace a defective component in case of hardware components.
3. **MTBF (Mean Time Between Failures)**

   MTBF metric is derived by combining the MTTF and MTTR metrics, that is,

   $$MTBF = MTTF + MTTR$$

   Thus, a MTBF of 300 hours indicates that once a failure occurs, the next failure is expected to occur only after 300 hours. In this case, the time measurements are in real time and not given by the execution time as in MTTF.
4. **POFOD (Probability of Failure on Demand)**

   POFOD is the likelihood that the system will fail when a service request is made. A POFOD of 0.001 means that one out of a thousand service requests may result in failure. POFOD is an important measure for safety critical systems and should be kept as low as possible.
5. **ROCOF (Rate of Occurrences of Failure)**

   ROCOF or the failure intensity is the frequency of occurrence with which unexpected behavior is likely to occur. It is relevant for operating systems and transaction-processing systems (e.g. credit-card processing systems) where the system has to process a large number of similar requests that are relatively frequent.
6. **Availability**

   Availability is the probability that the system is available for use at a given time.

9.3.1 Measurement of Reliability

Most hardware-related reliability models are predicted for failure due to wear rather than failure due to design defects. In case of hardware, failures due to physical wear (e.g., effect of temperature corrosion) are more likely than a design-related failure. For a computer-based system, a simple measure of reliability is mean-time-between-failure (MTBF). MTBF is considered as a more useful measure than defects/KLOC or defects/FP by the researchers because an end-user is concerned with failures, not with

the total error count. MTBF can be expressed as:

$$MTBF = MTTF + MTTR \quad \text{where,}$$

MTTF = Mean Time to Failure

MTTR = Mean Time to Repair.

9.3.2 Measurement of Availability

Software availability is the probability that a program is operating according to requirements at a given point in time and is defined as:

$$\text{Availability} = \frac{MTTF}{MTTF + MTTR} \times 100\%$$

9.4 SOFTWARE RELIABILITY MODELS

A reliability growth model mathematically models improvement in software reliability as errors are detected and repaired. A reliability growth model can be used to predict when (or if at all) a particular level of reliability is likely to be attained. Among the several different reliability growth models, two very simple reliability growth models are discussed below.

9.4.1 Jelinski and Moranda Model

Jelinski and Moranda model is the simplest reliability growth model. It is a step function model (Fig. 9.2) where it is assumed that the reliability increases by a constant increment each time an error is detected and repaired, that is, all errors contribute equally to reliability growth. However, the assumption is highly unrealistic as it is already known that correction of different types of errors contributes differently to reliability growth.

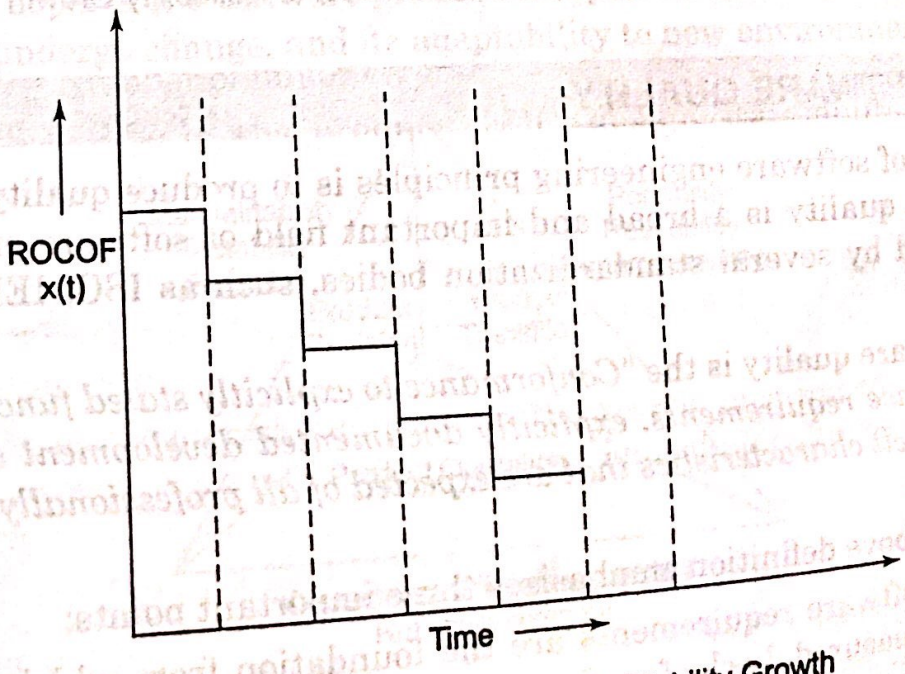

Fig. 9.2: Step Function Model of Reliability Growth

The model proposes a failure intensity function in the form of:

$$\lambda(t) = \phi(N - i + 1)$$

where,

ϕ = Constant of proportionality

N = Total number of errors present

I = number of errors found by time interval t

9.4.2 Littlewood and Verall's 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 as errors are repaired, the average improvement in reliability per repair decreases (Fig. 9.3). An error's contribution to reliability improvement is treated as an independent random variable having Gamma distribution. Gamma distribution models the fact that those errors are corrected first which when corrected have large contributions to reliability growth.

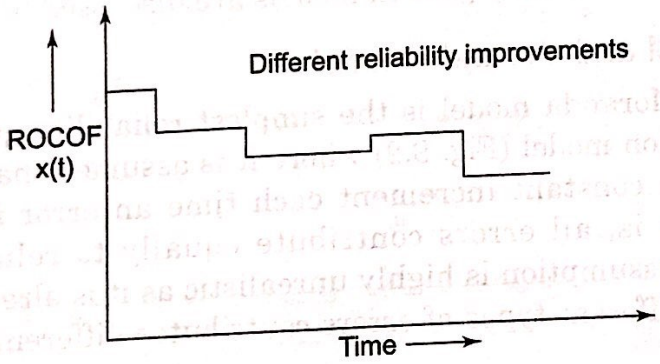

Fig. 9.3: Random-Step Function Model of Reliability Growth

9.5 SOFTWARE QUALITY

The aim of software engineering principles is to produce quality software. Software quality is a broad and important field of software engineering addressed by several standardization bodies, such as ISO, IEEE, ANSI, etc.

Software quality is the *"Conformance to explicitly stated functional and performance requirements, explicitly documented development standards, and implicit characteristics that are expected of all professionally developed software"*.

The above definition emphasizes three important points:

1. Software requirements are the foundation from which quality is measured. Lack of conformance to requirements is lack of quality.