

Software Reliability & Quality Assurance

Software Reliability

- It is considered at the system level rather than at the individual component level.
- The components in a system are interdependent ,so failures in one component can be propagated through the system and affect the operation of other components.

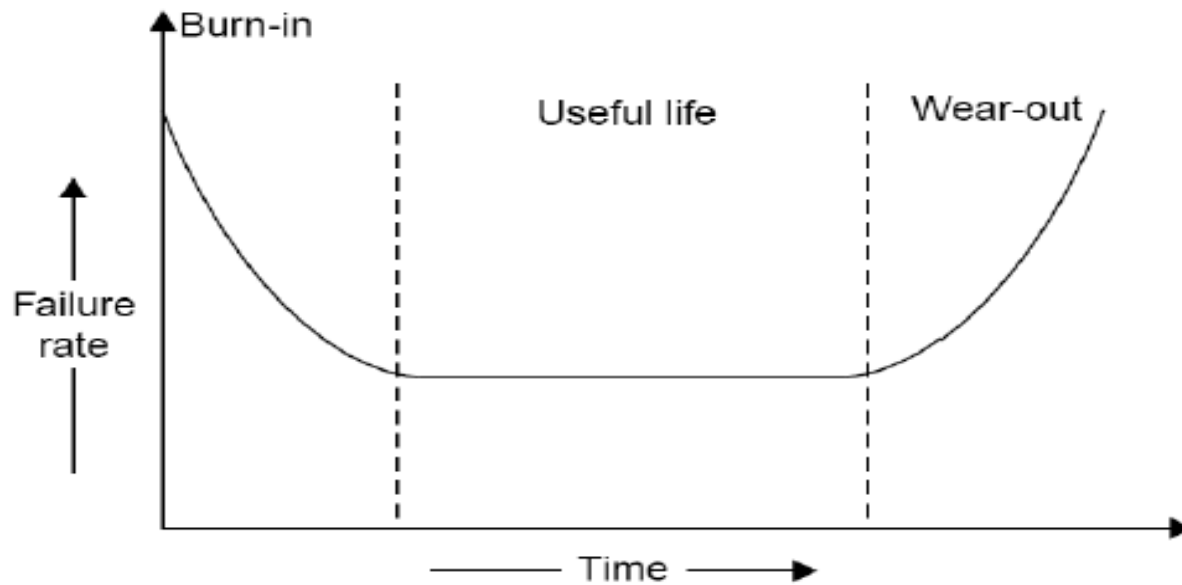
System reliability can be:

1. Hardware Reliability
2. Software Reliability
3. Operator Reliability

Hardware Reliability

- There are three phases in the life of any hardware component i.e. ,burn-in, useful life & wear-out.
- In **burn-in phase**, **failure rate is quite high initially**, **and it starts** decreasing gradually as the time progresses.
- During **useful life period**, **failure rate is approximately constant**.
- Failure rate increase in **wear-out phase due to wearing out/aging of** components. The best period is useful life period.

Hardware Reliability



Bath tub curve of Hardware Reliability

Software Reliability

Software may be retired only if it becomes obsolete.
Some of contributing factors are given below:

- **change in environment**
- **change in infrastructure/technology**
- **major change in requirements**
- **increase in complexity**
- **extremely difficult to maintain**
- **deterioration in structure of the code**
- **slow execution speed**
- **poor graphical user interfaces**

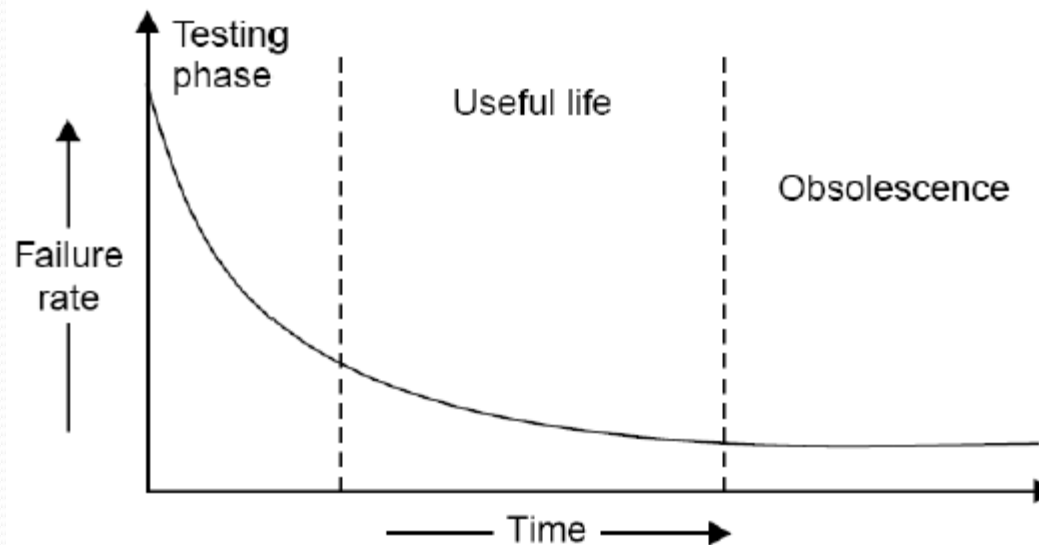
Software Reliability

- Software Reliability means operational reliability.
- As per IEEE standard: “Software reliability is defined as the ability of a system or component to perform its required functions understated conditions for a specified period of time”.

OR

- “It is the probability of a failure free operation of a program for a specified time in a specified environment”.

Software Reliability curve



Software Reliability curve(Failure rate versus time)

Factors Influencing S/W Reliability

1. **Cost** : Reliability is linearly dependent on cost. Higher level of desired reliability requires more testing effort and consequently more cost.
2. **Efficiency** : The degree to which the software makes optimal use of system resources. Efficiency of system is decreased with increase in reliability.
3. **Operational Profile** : Reliability improvement is largely dependent on the availability of operational profile that statistically models the pattern in which the system is more likely to be used in the operating environment. An accurate profile tends to bring out most frequently occurring failures first.

Measurements and metrics

A measurement is an indication of the size, quantity, amount or dimension of a particular attribute of a product or process. For example the *number of errors in a system* is a measurement.

A Metric is a measurement of the degree that any attribute belongs to a system, product or process. For example the *number of errors per person hours* would be a metric.

Thus, software measurement gives rise to software metrics.

Metric Classification

- Software metrics can be divided into two categories; *product metrics* and *process metrics*.
- Product metrics are used to assess the state of the product, tracking risks and discovering potential problem areas or product metrics are measures for the software product. Examples: Size, Reliability, Complexity etc.
- Process metrics quantify the attributes of software development process and environment. Examples: Productivity, Quality, Efficiency etc.

Product reliability Metrics

- **Rate of occurrence of failure (ROCOF).** ROCOF measures the frequency of occurrence of unexpected behavior (i.e. failures). ROCOF measure of a software product can be obtained by observing the behavior of a software product in operation over a specified time interval and then recording the total number of failures occurring during the interval.

- **Mean Time To Failure (MTTF).** MTTF is the average time between two successive failures, observed over a large number of failures. It is important to note that only run time is considered in the time measurements, i.e. the time for which the system is down to fix the error, the boot time, etc are not taken into account in the time measurements and the clock is stopped at these times.
- **Mean Time To Repair (MTTR).** Once 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.

- **Mean Time Between Failure (MTBR).** MTTF and MTTR can be combined to get the MTBR metric: $MTBF = MTTF + MTTR$. Thus, MTBF of 300 hours indicates that once a failure occurs, the next failure is expected after 300 hours. In this case, time measurements are real time and not the execution time as in MTTF.
- **Probability of Failure on Demand (POFOD).** Unlike the other metrics discussed, this metric does not explicitly involve time measurements. POFOD measures the likelihood of the system failing when a service request is made. For example, a POFOD of 0.001 would mean that 1 out of every 1000 service requests would result in a failure

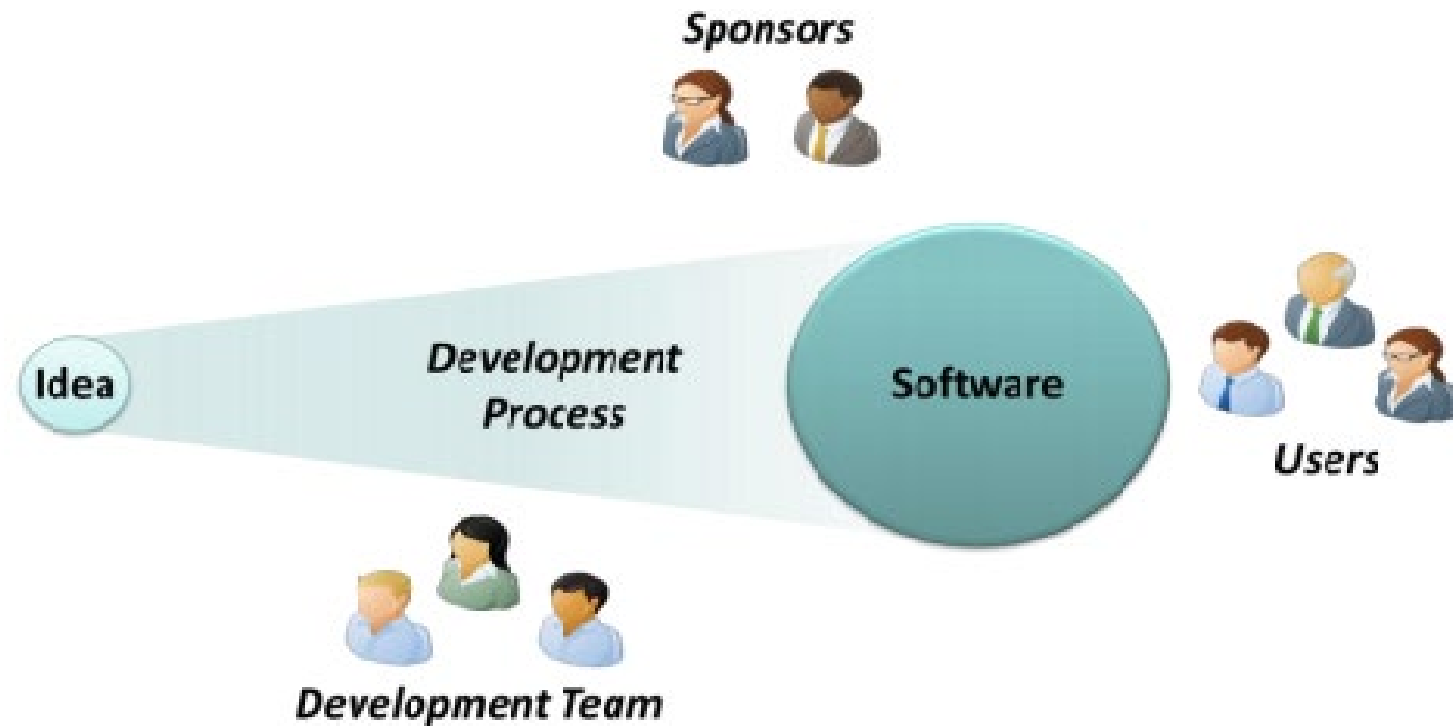
- **Availability.** Availability of a system is a measure of how likely shall the system be available for use over a given period of time. This metric not only considers the number of failures occurring during a time interval, but also takes into account the repair time of a system when a failure occurs. This metric is important for systems such as telecommunication systems, and operating systems, which are supposed to be never down and where repair and restart time are significant and loss of service during that time is important.


Classification of software failures

A possible classification of failures of software products into five different types is as follows:

- **Transient.** Transient failures occur only for certain input values while invoking a function of the system.
- **Permanent.** Permanent failures occur for all input values while invoking a function of the system.
- **Recoverable.** When recoverable failures occur, the system recovers with or without operator intervention.
- **Unrecoverable.** In unrecoverable failures, the system may need to be restarted.
- **Cosmetic.** These classes of failures cause only minor irritations, and do not lead to incorrect results. An example of a cosmetic failure is the case where the mouse button has to be clicked twice instead of once to invoke a given function through the graphical user interface.

Who Cares About Software Quality?

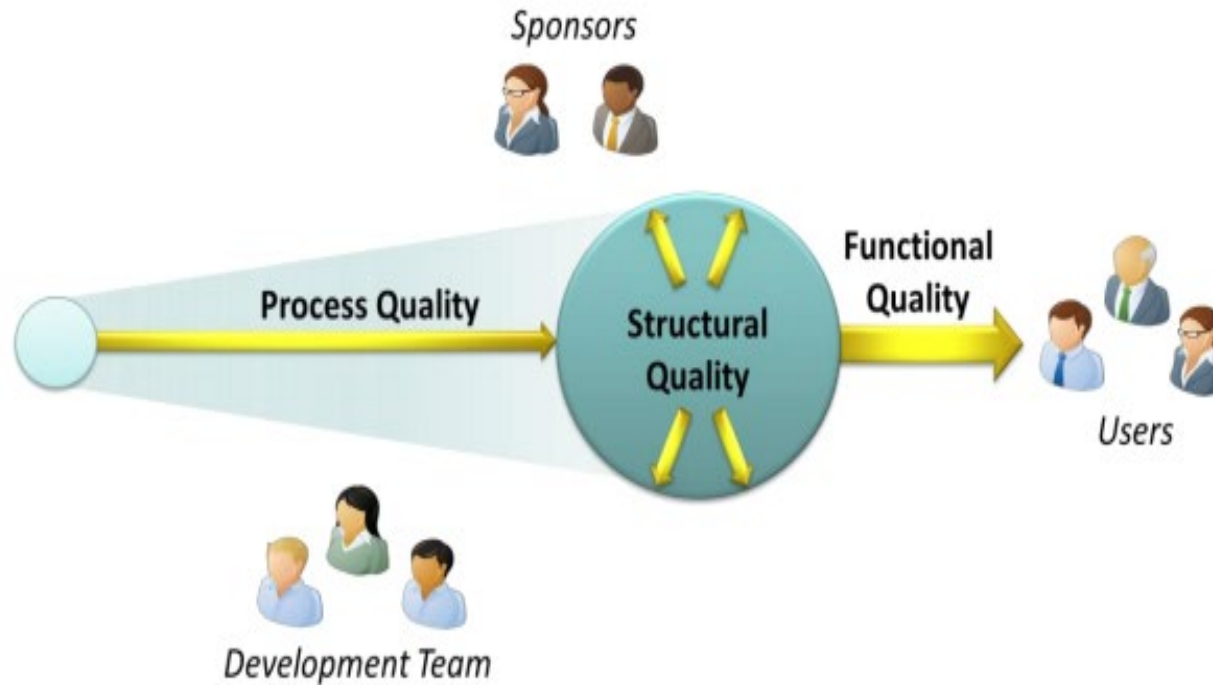




As a development process transforms an idea into working software, three main groups of people care about the software's quality

- The **software's users**, who apply this software to some problem.
- The **development team** that creates the software.
- The **sponsors of the project**, who are the people paying for the software's creation. For software developed by an organization for its own use, for example, these sponsors are commonly business people within that organization.

Defining Software Quality: Three Aspects





The three aspects of software quality are:

- Functional quality
- Structural quality, and
- Process quality

Functional Quality

Functional quality means that the software correctly performs the tasks it's intended to do for its users.

- Meeting the specified requirements.
- Creating software that has few defects.
- Good enough performance.
- Ease of learning and ease of use.

Software testing commonly focuses on functional quality

Structural Quality

The second aspect of software quality, structural quality, means that the code itself is well structured. Unlike functional quality, structural quality is **hard to test** and attributes are:

- **Code testability.** Is the code organized in a way that makes testing easy?
- **Code maintainability.** How easy is it to add new code or change existing code without introducing bugs?
- **Code understandability.** Is the code readable? Is it more complex than it needs to be? These have a large impact on how quickly new developers can begin working with an existing code base.
- **Code efficiency.** Especially in resource-constrained situations, writing efficient code can be critically important.
- **Code security.** Does the software allow common attacks such as buffer overruns and SQL injection?

Process Quality

The quality of the development process significantly affects the value received by users, development teams, and sponsors, and so all three groups have a stake in improving this aspect of software quality.

The most obvious attributes of process quality include these:

- Meeting delivery dates. Was the software delivered on time?
- Meeting budgets. Was the software delivered for the expected amount of money?
- A repeatable development process that reliably delivers quality software.

Who cares about what?

- **Users** care primarily about **functional quality**, since that's what they see. They're also likely to care about some aspects of **process quality**, such as the delivery date of the final software. Users typically don't care at all about **structural quality**, even though its absence might well impact them over the software's lifetime.
- A **development team** certainly does care about structural quality, however, since they're the people who will be affected by the problems caused by low quality here. They also care about functional quality, although perhaps a bit less than users do—cutting features that users want can make life easier for developers. Development teams also care about process quality, in part because it provides many of the metrics by which they're measured.

- 
- The third group, **sponsors**, cares about everything: functional quality, structural quality, and process quality

Software Quality

Software quality is the degree of conformance to explicit or implicit requirements and expectations.

Explanation:

- *Explicit*: clearly defined and documented
- *Implicit*: not clearly defined and documented but indirectly suggested
- *Requirements*: business/product/software requirements
- *Expectations*: mainly end-user expectations

Software Quality(Contd)

Definition by IEEE:

- The degree to which a system, component, or process meets specified requirements.
- The degree to which a system, component, or process meets customer or user needs or expectations.

Software Quality Factors



