Unit III: Software Reliability

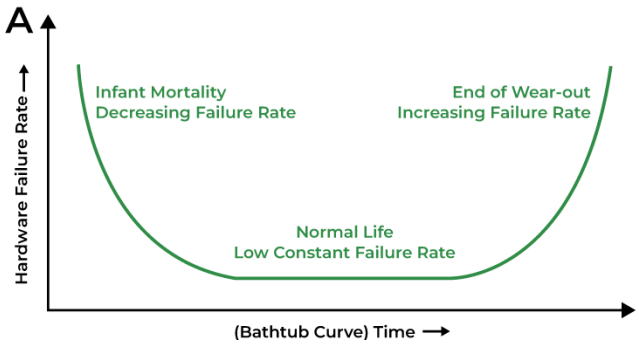
Dr. Sagar Deep Deb

In software engineering reliability is defined as the probability of failure free operation in a specified environment over a defined period of time.

Hardware Reliability

Hardware reliability is the probability that a physical component or system will perform its intended function without failure for a specific period in a defined environment. It is often measured by Mean Time Between Failures (MTBF) and follows a "bathtub curve" representing early, useful, and wear-out life phases.

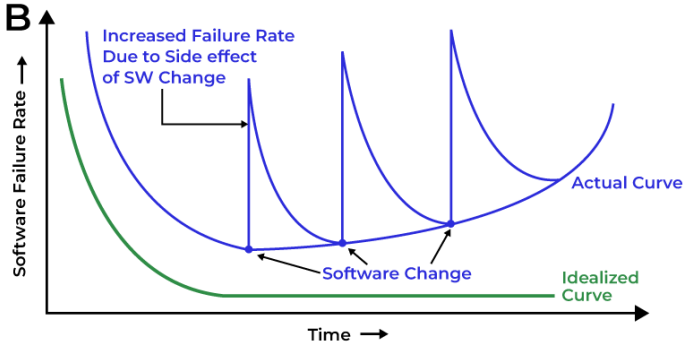
In Software Engineering, the concept of a software failure curve is part of the hardware bathtub curve. According to the diagram, the failure rate is a function of time for Hardware. This curve shows that hardware exhibits a relatively high failure rate early in its life (these failures are often design or manufacturing defects); defects are corrected, and the failure rate drops to a steady-state level for some time.



After some time according to time passes, the failure rate rises again as hardware components suffer from multiple effects and many other environmental maladies. Stated simply, the hardware begins to wear out.

Software Reliability

Initially, the failure rate is high due to defects and we correct these defects then the failure rate is reduced and fixed, but this is an idealized curve and it never happens in real life scenario.



The actual curve is drawn above the idealized curve in the actual software curve initially the failure rate is high because of undetected defects and when we correct these defects then steady

leave is started as we know changes are very common in the software when we added these changes this failure rate will increase due to update in the code and addition of new defects when this failure is increased we corrected all the defects and failure rate comes down this process continuously goes on and no wear out situation occur in the graph.

Hardware Reliability vs Software Reliability

Features	Hardware Reliability	Software Reliability
Source of Failure	Failures are caused due to defects in design, production, and maintenance.	Failures are caused due to defects in design.
Wear and Tear	Failure occurs due to physical deterioration in wear and tear.	In software reliability, there is no wear and tear.
Deterioration Warning	In this prior deterioration warning about failure.	In this no prior deterioration warning about failure.
Failure Curve	The bathtub curve is used for failure rates apply.	There is no Bathtub curve for failure rates.
Is Failure Time-dependent?	In this failures are time-dependent.	In this failures are not time-dependent.
Reliability Prediction	In this reliability can be predicted from design.	In this reliability can not be predicted from design.
Reliability Complexity	The complexity of hardware reliability is very high.	The complexity of software reliability is low.
External Environment Impact	Hardware reliability is related to environmental conditions.	External environment conditions do not affect software reliability.

Features	Hardware Reliability	Software Reliability
Reliability Improvement	Reliability can't be improved through redundant of hardware.	Reliability can be improved through redundancy of software.
Maintenance	Repairs can be made that make hardware more reliable through maintenance.	No equivalent preventive maintenance for software.

Mean Time Between Failures MTBF

Mean Time Between Failures (MTBF) is a reliability metric that represents the average time between two consecutive failures of a system during operation. It is widely used to measure the reliability of repairable systems, especially in software systems, servers, and hardware. MTBP = Total Operating Time/Number of failures.

On average, how long the system works before it fails again.

Higher MTBF ⇒ Higher reliability

Lower MTBF ⇒ Lower reliability

Q1. Suppose software runs for 1000 hours and fails 5 times.

$$MTBF = \frac{1000}{5} = 200 \text{ hours}$$

Meaning: On average, failure occurs every 200 hours.

Q2. Failure occurs at:

10 hr, 30 hr, 70 hr, 100 hr

Intervals between failures:

- 30 − 10 = 20
- 70 − 30 = 40
- 100 − 70 = 30

$$MTBF = \frac{20 + 40 + 30}{3}$$
$$MTBF = 30 \text{ hours}$$

$$\boxed{MTBF = MTTF + MTTR}$$

where:

- MTTF = Mean Time To Failure (average working time before failure)
- MTTR = Mean Time To Repair (average repair time)

Q3. System works for 90 hours before failure.
Repair takes 10 hours.

$MTBF = 90 + 10$; $MTBF = 100$ hours

Relationship with Failure Rate (λ)

Failure rate:

$$\lambda = \frac{1}{MTBF}$$

Reliability function:

$R(t) = e^{-\lambda t}$ or $R(t) = e^{-t/MTBF}$

Q4. If MTBF = 100 hours
Find reliability for 50 hours.

$$R(t) = e^{-t/MTBF}$$
$$R(50) = e^{-50/100}$$
$$R(50) = e^{-0.5}$$
$$R(50) = 0.6065$$

Reliability = 60.65%

Faults and Failures

Fault: A fault is a defect or bug in software caused by human error in design, coding, or specification. Example: Wrong formula used in attendance percentage calculation.
Failure: A failure occurs when the software does not perform its intended function. Example: Attendance system shows incorrect attendance percentage.

1. Fault Prevention
   Definition Fault prevention refers to techniques used to avoid introducing faults into software.
   Methods
   - Proper system design
   - Coding standards
   - Developer training
   - Use of formal methods
   - Modular programming: Modular programming is a software design technique that breaks large, complex programs into smaller, independent, and interchangeable units called modules.

   Example Designing proper validation for attendance input prevents faults.
2. Fault Removal
   Definition: Fault removal refers to detecting and fixing faults after they are introduced.
   Methods
   - Unit testing
   - Integration testing
   - System testing
   - Debugging Code reviews

   Example Fixing incorrect attendance calculation logic.
3. Fault Tolerance
   Definition: Fault tolerance is the ability of software to continue functioning even when faults exist.
   Methods
   Exception handling
   Backup systems
   Redundancy: Redundancy means having extra or backup components in a system so that if one component fails, another can take over and the system continues to function.
   Checkpointing: Checkpointing is the process of saving the current state of the system periodically so that the system can restart from that point in case of failure.
   Example If main server fails, backup server continues attendance recording.
4. Fault Forecasting
   Definition Fault forecasting refers to predicting future faults and failures using statistical models.
   Methods Failure rate calculation
   MTBF calculation
   Reliability models
   Failure data analysis