Chap 4. Software Reliability

4.1 Reliability Concepts and Metrics

1. Introduction
2. Software Reliability Concepts
3. Reliability Model
4. Availability

1. Introduction

Motivating Examples

Example 1: A company is planning to purchase several new color laser printers. Before finalizing the purchase, they acquire a similar printer for the test run and conduct certification test on it.

Vendor's data shows that the toner should be changed every 10,000 pages. The goal of the company is to have the system running without any failure between the two consecutive toner changes and in the worst case having only one failure during the same period.

During the test run, it is observed that failures occur at 4,000 pages, 6,000 pages, 10,000 pages, 11,000 pages, 12,000 pages and 15,000 pages of output. Should the company purchase this kind of printer?

Example 2: We have developed a program for a Web server with the failure intensity objective of 1failure/100,000 transactions. During testing, the program runs for 50 hours, handling 10,000 transactions per hour on average with no failures occurring. How confident are we that the program has met its objective? Can we release the software now?

Example 3: Unit manufacturing cost of a software product is $50. The company decides to offer one year free update to its customers. Suppose that failure intensity of the product at the release time is $\lambda = 0.01$ failures/month. What should be the unit cost of the product including warranty services?

Overview

-Critical systems such as spacecraft, aircraft, nuclear power plant and pacemakers require a high level of *dependability* in their operation.

-Two different categories of techniques are used in the design and implementation of dependable software systems: *fault avoidance* and *fault tolerance* techniques.

- ***Fault avoidance***: primary goal of any sound software engineering process.
- ***Fault tolerance***: address the shortcoming of fault avoidance by mitigating the risk that there are some potential or hidden faults remaining in the software.

-Reliability is a popular aspect of software dependability, which relies, in particular, on *fault forecasting* and *fault removal*.

- ***Fault forecasting*** consists of estimating the presence of faults and the occurrence and consequences of failures.
- ***Fault removal*** uses techniques such as testing or inspection to track and remove faults in software.

2. Software Reliability Concepts

-Software reliability measures can be used to improve software engineering processes, by:

- supporting quantitative evaluation of software technologies, tracking development status, conducting upgrades and maintenance activities.

-Software reliability is the probability that the software system will function properly without **failure** over a certain time period.

- Reliability is one of the most important software quality attributes.
- It is an external quality attribute, which relates internally to the notion of program faults or defects.

-A ***failure*** corresponds to an occurrence where the operational behavior of a program deviates from the requirements.

- A failure is triggered only in operation and as such it is dynamic in nature.
- A failure is different from a fault, although both notions are related.

-A ***fault*** or a "bug" is a program defect, which triggers a failure when such program is executed under specific operational conditions.

- According to the execution conditions different failures may be triggered.
- So a fault may lead to several failures.

Hardware vs. Software Reliability

-Many of the concepts and models used in software reliability are derived from hardware reliability, which is an established field.

-There are, however, some fundamental differences between both fields. For instance:

- While hardware reliability tends to be stable or constant over time, software reliability has tendency to change during test periods; this phenomenon is referred to as ***reliability growth***.
- The sources of failures are different. Hardware faults arise mostly from wear and physical deterioration, while software faults arise mostly from design issues.

-The source of software faults include:

- Incorrect requirements, even though the implementation may match them.
- Implementation (software design and coding) deviating from (correct) requirements.
- Uncontrolled or unexpected changes in operational usage or incorrect modifications.

Execution Variables

-Reliability models express the probability of failure over a certain *execution exposure* variable or metric for the system.

- Examples of exposure metrics include time (of execution), number of executed test cases or runs, number of transactions etc.

-Time is the most common exposure metric used. Three kinds of times variables are commonly used: *calendar time*, *clock time*, and *(CPU) execution time*.

- ***Execution time*** corresponds to the effective time used by the processor to execute the program instructions.
- ***Calendar time*** is the regular time we use in our daily business; as such it is important for users and managers.
- ***Clock time*** corresponds to the elapsed time between the start and end of program execution. Clock time and execution time are equivalent when computer utilization is constant.

Reliability quantities are often computed using execution time, and later converted into calendar time for managerial purposes.

Failure Behavior

-Failure behavior directly depends on the environment and the number of faults present in the program during execution.

-Failure occurrences are expressed as ***random variables***, because of the unpredictable nature of fault commission by programmers, and the unpredictability of the conditions under which programs are executed.

-Using time variable, failure occurrences may be expressed in four different ways:

1. Time of failure
2. Time interval between failures
3. Cumulative failures occurred up to a specified time
4. Failures occurred in a specified time interval.

-Two important functions are derived from the random process associated with failure occurrence:

- ***Mean value function*** expresses the average cumulative failures at each point in time
- ***Failure intensity function*** is the number of failures per time unit; it is computed as the derivative of the mean value function with respect to time.

Reliability Metrics

-Reliability metrics are derived from failure occurrence expressions and data.

-Common reliability metrics include

- ***Probability Of Failure On Demand (POFOD)*** is likelihood that a transaction request will fail.
- ***Rate Of Occurrence Of Failures*** (***ROCOF)*** corresponds to the failure intensity.
- ***Mean Time To Failure*** (***MTTF)*** is the average time between consecutive system failures.
- ***Availability*** is the likelihood that the system will be working at a given time.

Metrics	**Reliability Specification**
POFOD	For systems where (critical) services requests happen in an unpredictable way, or when there is a long time interval between consecutive requests.
ROCOF	For systems where (critical) services are demanded in more regular way.
MTTF	For systems involving long transactions, during which a guarantee of service continuity and delivery should be expected.
AVAIL	For systems where continuous service delivery is a major concern.

3. Reliability Model

-Software reliability relies on three basic models:

1. ***Usage model***: describes how the software is used
2. ***Trend model***: describes how reliability evolve over time as certain bugs are fixed or new bugs are introduced.
3. ***Probabilistic failure model***: captures the fact that failures may happen randomly.

-Reliability models characterize the occurrence of software failures as a stochastic process.

- Software failures are characterized by studying failure occurrence time or number of failures occurring at specific time.
- Software reliability models assume that failures are independent of each other.

-Let us denote by **M(t)** the random process representing failure occurrence at time **t** :

- The expected number of failures at time t is computed by the mean value function:

$$m(t) = E[M(t)]$$

- The failure intensity function of M(t) quantifies the rate of change of the expected number of faults at time t; it is computed as follows:

$$l(t) = \frac{d\,m(t)}{d\,t}$$

-Let **T** denote a random variable representing the system failure time.

- Failure density **f(t)** corresponds to the probability distribution function of **T**.

- Failure probability **F(t)** is the probability that the failure time is less or equal to time **t**:

$$F(t) = \mathrm{Prob}(T \leq t) = \int_0^t f(u).du$$

- Reliability **R(t)** is the probability that the system will be working in the time interval:

$$R(t) = 1 - F(t) = \mathrm{Prob}(T \geq t) = \int_t^{+\infty} f(u).du$$

Deferred Repair

-Standard reliability modeling assumes that faults are not repaired immediately (i.e., **deferred repair**);

- This is usually the case in **operation**, where repair is deferred until the next release. In this case the failure intensity remains constant.
- The situation with repair, which usually occurs in **production**, is referred to as **reliability growth**, because the failure intensity decreases as failures are removed.

-Under deferred repair assumption:

- Let divide the observation time interval [0,t] into *i* equal segments, each of length *t/i*, such that the system fails at the end of each segment with probability $\lambda t/i = -u$.
- Reliability R(t) is the *probability of no failure over time interval [0,t]*. So:

$$R(t) = \left(1 - \frac{\lambda t}{i}\right)^{i} = (1 + u)^{\frac{-\lambda t}{u}}$$

- For greater values of *i*, we get:

$$R(t) \approx \lim_{i \to +\infty} (1 + u)^{\frac{-\lambda t}{u}} = e^{-\lambda t}$$

- Hence, the reliability model is:

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

Example: for λ=0.001 or 1 failure for 1000 hours, reliability (*R*) is around 0.992 for 8 hours of operation.

4. Availability

-*Reliability* is the probability that the system is working properly over a fixed period of time.

-*Availability* is the probability that the system is operational at any point in time:

$$Availability = \frac{uptime}{uptime + downtime}$$

Classes of Systems According to Availability

Availability class	***Availability (%)***	***Unavailability (min/year)***	***System type***
1	90.0	52560	Unmanaged
2	99.0	5256	Managed
3	99.9	526	Well-managed
4	99.99	52.6	Fault-tolerant
5	99.999	5.3	Highly available
6	99.9999	0.53	Very highly available
7	99.99999	0.0053	Ultra available

Metrics

-Availability is characterized by defining some basic concepts that describe quantitatively the operational state of the system. These include:

***MTTF (Mean Time To Failure*):** average time it takes for a system to fail.

MTTR (Mean Time To Recover): average time for the system to recover; correspond to the average time to repair the system.

MTBF (Mean Time Between Failure): average time between consecutive system failures.

-The *MTBF* is equal to the sum of the *MTTF* and the *MTTR*:

MTBF = MTTF + MTTR

- Let's denote by ***A*** and ***U*** respectively the probability that the system is up (e.g. ***available***), and that the system is down (e.g. ***unavailable***). We can write that A + U = 1
- Let's denote by l and m respectively the failure rate of the system (e.g., the system going from up to down) and the repair rate of the system (e.g., the system going from down to up). We can write that $\lambda = 1/MTTF$ and $\mu = 1/MTTR$, and that $A \times \lambda = U \times \mu$

By combining the above equations, we get that:

A = m/(l + m) = MTTF/(MTTF +MTTR) = MTTF/MTBF
U = l/(l + m) = MTTR/(MTTR + MTTF) = MTTR/MTBF

In general MTTF >> MTTR, hence U can be approximated as:

$$U \cong MTTR/MTTF$$

Example: If a product must be available 99% of time and downtime is 6 min, then λ is about 0.1 failure per hour (1 failure per 10 hours) and MTTF=594 min.