Unit. 10 Software Reliability Engineering

1. **Introduction**
2. **System Reliability**
3. **Concurrent Systems**
4. **Time and Platforms Conversions**
5. **Software Reliability Engineering Process**
6. **Reliability Acceptance Testing**

Reading: TB-Chapter 15 (15.1-15.7)

1. Introduction

-Software Reliability is defined as the ***probability of a software system to perform its specified functions correctly over a certain period of time*** in a specified execution environment.

-Software reliability models characterize the failure arrival process as a stochastic process, which can be analyzed and used for various purposes, e.g., reliability assessment and prediction.

-Software reliability engineering (SRE) is the branch of software engineering that studies the ***issues related to the measurement, modeling, and improvement of software reliability***.

1. Introduction (ctd.)

Applications of Software Reliability Engineering

- Comparison of software engineering technologies
  - What is the cost of adopting a technology?
  - What is the return from the technology -- in terms of cost and quality?
- Measuring the progress of system testing
  - Key question: How of testing has been done?
  - The failure intensity measure tells us about the present quality of the system: high intensity means more tests are to be performed.
- Controlling the system in operation
  - The amount of change to a software for maintenance affects its reliability. Thus the amount of change to be effected in one go is determined by how much reliability we are ready to potentially lose.
- Better insight into software development processes
  - Quantification of quality gives us a better insight into the development processes.

2. System Reliability

-System reliability encompasses three aspects: ***hardware reliability***, ***software reliability***, and ***operator reliability***.

- Hardware errors can generate inputs outside the expected range for software, causing unpredictable outcome
- Software errors are often transient rather than permanent
- Operator errors may result from lacks in users understanding of the system, or confusing output creating stressful situations.

-The study of system reliability requires determining the system boundary, its components and configurations.

- Reliabilities for the system components are first determined and then combined based on system configurations yielding overall system reliability measures.

-Decomposition of the system into a specific set of components is often guided by the following considerations:

- Physical or architectural structure of the system
- Existing components with known reliabilities
- Special requirements set for specific components
- Effort required for data collection.

-We also need to determine the combinatorial relationships existing between the system components.

-Two kinds of relationships are commonly considered: concurrent and sequential.

- ***Concurrent systems***: collection of components that function at the same time
- ***Sequential systems***: only one component functions at a time.

-The most common combinatoric rules underlying concurrent systems are AND and OR combinations, also called series and parallel.

- AND-OR configurations with independent failures are the most common and simplest.

Reliability Block Diagram

-Reliability Block Diagram (RBD) is a graphical representation of how the components of a system are connected from reliability standpoint.

-The most common configurations of a RBD are the ***series*** and ***parallel*** configurations.

- In a serial configuration, all the elements must work for the system to work: the system fails if one of the components fails.
- In parallel configuration, the components are considered to be redundant and the system will still cease to work if all the parallel components fail.

-A system is usually composed of combinations of serial and parallel configurations.

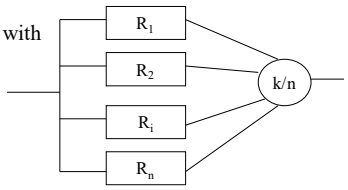
-RBD analysis is used in practice to ***determine reliability, availability and down time of the system***.

3. Concurrent Systems

K-out-of-n Structure

-The most general form of concurrent systems is the so-called *k-out-of-n* structure (k≤n): at least *k* components must be working for the system to work

•Example: airplane with 4 engines can fly with only 2 engines.



•Note that:
- k=n: corresponds to a serial system
- k=1: corresponds to a parallel system

-In case where all the components have the same reliability R_0, the reliability of the system is given by:

$$R = \sum_{j=k}^{n} \binom{n}{j} R_0^{\,j} (1 - R_0)^{n-j}$$

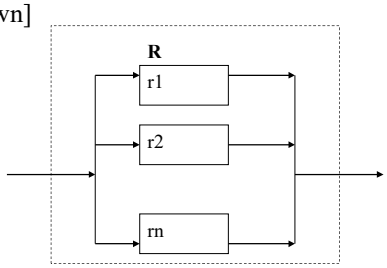
Example 11.1: Calculate the reliability of the airplane engines assuming R_0=0.99 for each engine.

Parallel Structure

-The reliability of a system of *n* components connected in parallel, of respective reliability r_i, and that fail independently is obtained by:

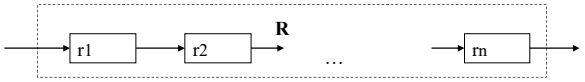
R = 1 –P[all components are down]

$$R = 1 - \prod_{i=1}^{n} (1 - r_i)$$



•***Remark***: the probability of a system of *n* identical parallel components (i.e., $R=1-(1-r)^n$) increases with the number of components.

Serial Structure



-The reliability of *n* components connected in series, of respective reliability *ri* and that fail independently, can be computed by

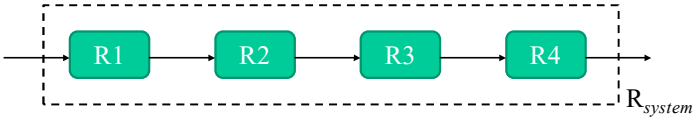
$$R = r_1 \times \ldots \times r_n = \prod_{i=1}^{n} r_i$$

•***Remark***:
-Components reliabilities r_i must be expressed with respect to a common interval.
-The reliability of a system of similar components connected in series (i.e., $R=r^n$) decreases very fast as the number of components increases.

-Assuming that each of the *n* components has a constant failure intensity λ_i , then the system failure intensity can be computed by

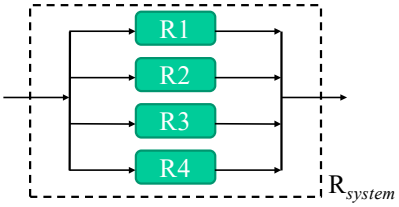
$$\lambda = \sum_{i=1}^{n} \lambda_i$$

Example 11.2: A system is composed of 4 independent serially connected components *R1 = 0.95 R2 = 0.87 R3 = 0.82 R4 = 0.73*



$R_{system} = 0.95 \times 0.87 \times 0.82 \times 0.73 = 0.4947$

Example 11.3: A system is composed of 4 independent components connected in parallel *R1 = 0.95 R2 = 0.87 R3 = 0.82 R4 = 0.73*



$R_{system} = 1 - ((1 - 0.95) \times (1 - 0.87) \times (1 - 0.82) \times (1 - 0.73)) = 0.9996$

4. Time and Platform Conversions

Time Conversion

-When combining components reliabilities, it is expected that the same calendar period will be used for all the reliabilities.

-In practice software and hardware reliabilities are expressed using different time base:

•Software reliabilities are often expressed in terms of execution time,

•Hardware reliabilities are often expressed with respect to operating or clock time.

-So, before combining with hardware components, it is necessary to ***convert software components reliabilities to a common reference*** period of clock time with hardware components.

•Failure intensity conversion in terms of clock time is as follows: $\lambda_{ti} = \rho_i \lambda_{\tau i}$

Where, for software component *i*:
-**λ_{ti} is the failure intensity with respect to clock time,**
-**$\lambda_{\tau i}$ is the failure intensity with respect to execution time**
-**ρ_{ti} is the average computer utilization**

•Reliability with respect to clock time can be computed for component *i* as:

$$R_i = e^{-\lambda_{ti} t}$$

Platform Adjustments

-When a program is moved from one platform to another, failure intensities and hence reliabilities should be adjusted.

•Suppose a program is moved from computer 1 of average instruction execution rate r_1 to computer 2 of average instruction rate r_2 , then:

$$\lambda_2 = \frac{r_2}{r_1} \lambda_1$$

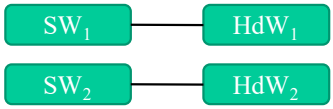
Where:
- λ_2 is the failure intensity on computer 2
- λ_1 is the failure intensity on computer 1

Example 11.4: Consider a system made up of two software components, 1 and 2. Both operate concurrently, and both must operate without failure for the system to operate without failure.

Assume that all hardware components have reliability 1. The software components have reliabilities, respectively, of 0.670 and 0.527 for a period of 8 hr of execution time. Calculate their respective failure intensities.
Calculate the system reliability and failure intensity under each of the following scenarios:

a. Both software components run on the same machine, each taking up half the execution time.

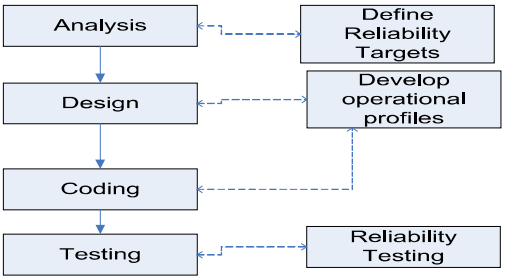
b. The software components run on different machines, and we assume that both machines have the same average instruction execution rate and are fully utilized by these components.



5. Software Reliability Engineering Process

-Typical SRE process involves three main steps:

- *Reliability goals definition*: consists of capturing the customer expectations in terms of reliable product or operations.
- *Operational profiles definition*: consists of identifying and specifying the **probable** patterns of usage of the software.
- *Reliability testing*: the operational profiles are used to design test cases and drive the reliability validation process.



Reliability Targets

-Defining reliability targets involves:

1. Defining consistently the notion of failure for the software product.
2. Choosing common reference measure for all failure intensities (e.g., failures per some natural unit or per hour).
3. Setting system Failure Intensity Objective (FIO) for each associated system (software/hardware)
4. For any software you develop:
   A. Find developed software FIO by subtracting the total of the FIOs of all hardware and acquired software components from the system FIOs.

Example 12.5: Various FIO may be defined for various parts of a system:

-System failure intensity objective λ= 30 failure/1,000,000 transactions
-MTTF for OS is 3,000 hours for 10 million transactions
-MTTF for hardware is 30 hours of operation

One must define a unique scale for all FIOs

Example 11.6: System FIO = 100 failure/1,000,000 transactions.
Failure intensity for hardware = 0.1 failure/hour
OS failure for a load of 100,000 transactions = 0.4 failure/hour
Calculate developed software FIO.

Example 11.7: Database system running on Win 2K
System failure intensity objective = 30 failure/1,000,000 transactions
MTTF for Win 2K is around 3,000 hours for 10 million transactions
Average hardware failure is 1 per 30 hours
Failure rate for other systems is 9 for one million transactions
What is FIO for the developed software?

Operational Profiles

- Developed at AT&T Bell Labs.
- An OP describes how actual users operate a system.
  - An OP is a quantitative characterization of how a system will be used.
- Two ways to represent operational profiles
  - Tabular
  - Graphical

Operation	Operations per hour	Probability
Book checked out	450	0.45
Book returned in time	324	0.324
Book renewed	81	0.081
Book returned late	36	0.036
Book reported lost	9	0.009
...	...	...
Total	1000	1.0

Table: An example of operational profile of a library information system.

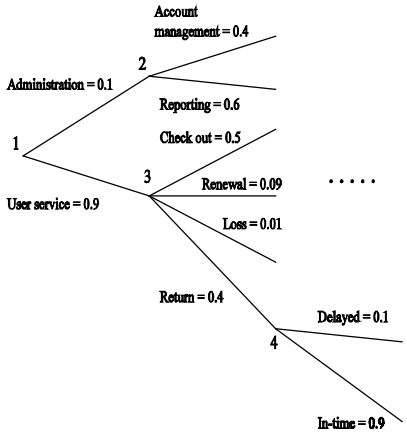


Figure: Graphical representation of operational profile of a library information system.

Operational Profiles (ctd.)

- Use of operational profiles
  - For accurate estimation of the reliability of a system, test the system in the same way it will be actually used in the field.
- Other uses of operational profiles
  - Use an OP as a guiding document in designing user interfaces.
    - The more frequently used operations should be easy to use.
  - Use an OP to design an early version of a software for release.
    - This contains the more frequently used operations.
  - Use an OP to determine where to put more resources.

6. Reliability Acceptance Testing

-Reliability acceptance or demonstration testing occurs in situations where it is desirable or necessary to prove that a given level of reliability has been achieved.

-The purpose of the acceptance testing is to determine ***if the failure intensity objective (FIO) is met with high confidence or not***.

-In conducting the acceptance testing, the following assumptions are made:

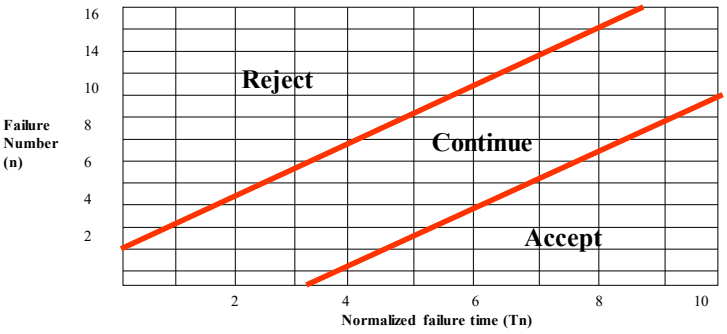
- Proper operational profile has been executed; it must represent actual operation.
- The software is being tested as it has been or would be delivered; no repair will take place.

-The reliability acceptance or demonstration testing approach is based on sequential sampling theory.

- Sequential sampling theory provides an efficient approach to testing a hypothesis by taking just enough data and no more.
- The (failure) data points are added one by one, and data collection stops as soon as a decision can be reached.
- The collected data points are used to build a ***reliability demonstration chart***, which serves for decision making.

-A ***reliability demonstration chart*** consists of a graph in which:

- *Vertical axis:* failure number (***n***)
- *Horizontal axis:* normalized failure data (***Tn***), i.e., failure time × λ_F



-The steps for conducting a reliability demonstration include:

1. Establish the FIO being tested.
2. Run the test and record the time of the next failure that is experienced.
2. Normalize the failure time by multiplying it by the FIO.
3. Locate the normalized failure time on the reliability demonstration chart.
   - If it falls in the "accept" region, then the FIO is considered with high confidence.
   -If it falls in the "reject" region, there is little possibility of meeting the objective; it is not worth testing further.
   -If it falls in the "continue" region, the result is uncertain; testing should continue.

-The boundaries between the different regions of the chart (i.e., reject-continue, continue-accept) can be determined by specifying the following parameters:

- ***Discrimination ratio (γ):*** Acceptable error in estimating failure intensity.
- ***Customer risk (β) :*** Probability that the developer is willing to accept of falsely saying the failure intensity objective is met (i.e., ***acceptance***) when it is not.
- ***Developer risk (α) :*** Probability that the developer is willing to accept of falsely saying the failure intensity objective is **not** met (i.e., ***rejection***) when it is.

-Example: For **α** =10% and **β**= 10% and **γ** =2

- There is 10% risk (**β**) of wrongly accepting the software when its failure intensity objective is actually equal or greater than twice (**γ** =2) the failure intensity objective.
- There is 10% risk (**α**) of wrongly rejecting the software when its failure intensity objective is actually equal or less than half (**γ** =2) the failure intensity objective.

-Based on α, β, and γ, the following parameters are computed and used to determine the different regions of the chart:

$$A = \ln\frac{\beta}{1-\alpha}$$

$$B = \ln\frac{1-\beta}{\alpha}$$

- ***A*** changes rapidly with *customer risk* but very slightly with *developer risk* and it determines the intercept of ***accept*** boundary with the line *n=0*
- ***B*** changes rapidly with *developer risk* but very slightly with *customer risk* and it determines the intercept of ***reject*** boundary with the line *Tn=0*

-Boundary between ***accept*** and ***continue*** regions:

$$T_n = \frac{A}{1-\gamma} - \frac{\ln\gamma}{1-\gamma}n$$

-Boundary between ***reject*** and ***continue*** regions:

$$T_n = \frac{B}{1-\gamma} - \frac{\ln\gamma}{1-\gamma}n$$

- When risk levels (α and β) decrease, the system will require more test before reaching the accept or reject regions, *i.e.*, the continue region becomes wider.
- When discrimination ratio (γ) decreases, the system will require more test before reaching the accept or reject regions, *i.e.*, the continue region becomes wider.

Example 11.8: A personal computer manufacturer wishes to know if the software company delivering the operating system for a new machine has met the failure intensity objective of 0.1 failure/CPU hr. A series of tests are applied and the following failure data at different CPU hr are recorded (see table).

Decide the outcome of testing based on the following parameters:
α= 10%, β=10%, γ=2

Failure Number	Failure Time (CPU hr)
1	8
2	19
3	60

Example 11.9: We have developed a program for a Web server with the failure intensity of 1 failure/100,000 transactions. 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 11.10: Suppose that a new component is added to the program serially to make a new package. The failure intensity of the new component is 0.5 failures/100,000 transactions. The new package fails after 10 hrs. Can we release the new package now?

Example 11.11: 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.

a) What shall be the failure intensity objective for the system?

b) 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. Using the reliability demonstration chart, what can we conclude about this printer?