Software Reliability

(Lecture 13)

Dr. R. Mall

Organization of this Lecture:

- Introduction.
- Reliability metrics
- Reliability growth modelling
- Statistical testing
- Summary

Introduction

- Reliability of a software product:
  - a concern for most users especially industry users.
  - An important attribute determining the quality of the product.
- Users not only want highly reliable products:
  - want quantitative estimation of reliability before making buying decision.

Introduction

- Accurate measurement of software reliability:
  - a very difficult problem
  - Several factors contribute to making measurement of software reliability difficult.

Major Problems in Reliability Measurements

- Errors do not cause failures at the same frequency and severity.
  - measuring latent errors alone not enough
- The failure rate is observer-dependent

Software Reliability: 2 Alternate Definitions

- Informally denotes a product’s trustworthiness or dependability.
- Probability of the product working “correctly” over a given period of time.

Software Reliability

- Intuitively:
  - a software product having a large number of defects is unreliable.
- It is also clear:
  - reliability of a system improves if the number of defects is reduced.

Difficulties in Software Reliability Measurement (1)

- No simple relationship between:
  - observed system reliability
  - and the number of latent software defects.
- Removing errors from parts of software which are rarely used:
  - makes little difference to the perceived reliability.

The 90-10 Rule

- Experiments from analysis of behavior of a large number of programs:
  - 90% of the total execution time is spent in executing only 10% of the instructions in the program.
- The most used 10% instructions:
  - called the core of the program.

Effect of 90-10 Rule on Software Reliability

- Least used 90% statements:
  - called non-core are executed only during 10% of the total execution time.
- It may not be very surprising then:
  - removing 60% defects from least used parts would lead to only about 3% improvement to product reliability.

Difficulty in Software Reliability Measurement

- Reliability improvements from correction of a single error:
  - depends on whether the error belongs to the core or the non-core part of the program.

Difficulty in Software Reliability Measurement (2)

- The perceived reliability depends to a large extent upon:
  - how the product is used,
  - In technical terms on its operation profile.

Effect of Operational Profile on Software Reliability Measurement

- If we select input data:
  - only "correctly" implemented functions are executed,
    - none of the errors will be exposed
    - perceived reliability of the product will be high.

Effect of Operational Profile on Software Reliability Measurement

- On the other hand, if we select the input data:
  - such that only functions containing errors are invoked,
  - perceived reliability of the system will be low.

Software Reliability

- Different users use a software product in different ways.
  - defects which show up for one user,
    - may not show up for another.
- Reliability of a software product:
  - clearly observer-dependent
  - cannot be determined absolutely.

Difficulty in Software Reliability Measurement (3)

- Software reliability keeps changing through out the life of the product
  - Each time an error is detected and corrected

Hardware vs. Software Reliability

- Hardware failures:
  - inherently different from software failures.
- Most hardware failures are due to component wear and tear:
  - some component no longer functions as specified.

Hardware vs. Software Reliability

- A logic gate can be stuck at 1 or 0,
  - or a resistor might short circuit.
- To fix hardware faults:
  - replace or repair the failed part.

Hardware vs. Software Reliability

- Software faults are latent:
  - system will continue to fail:
    - unless changes are made to the software design and code.

Hardware vs. Software Reliability

- Because of the difference in effect of faults:
  - Though many metrics are appropriate for hardware reliability measurements
    - Are not good software reliability metrics

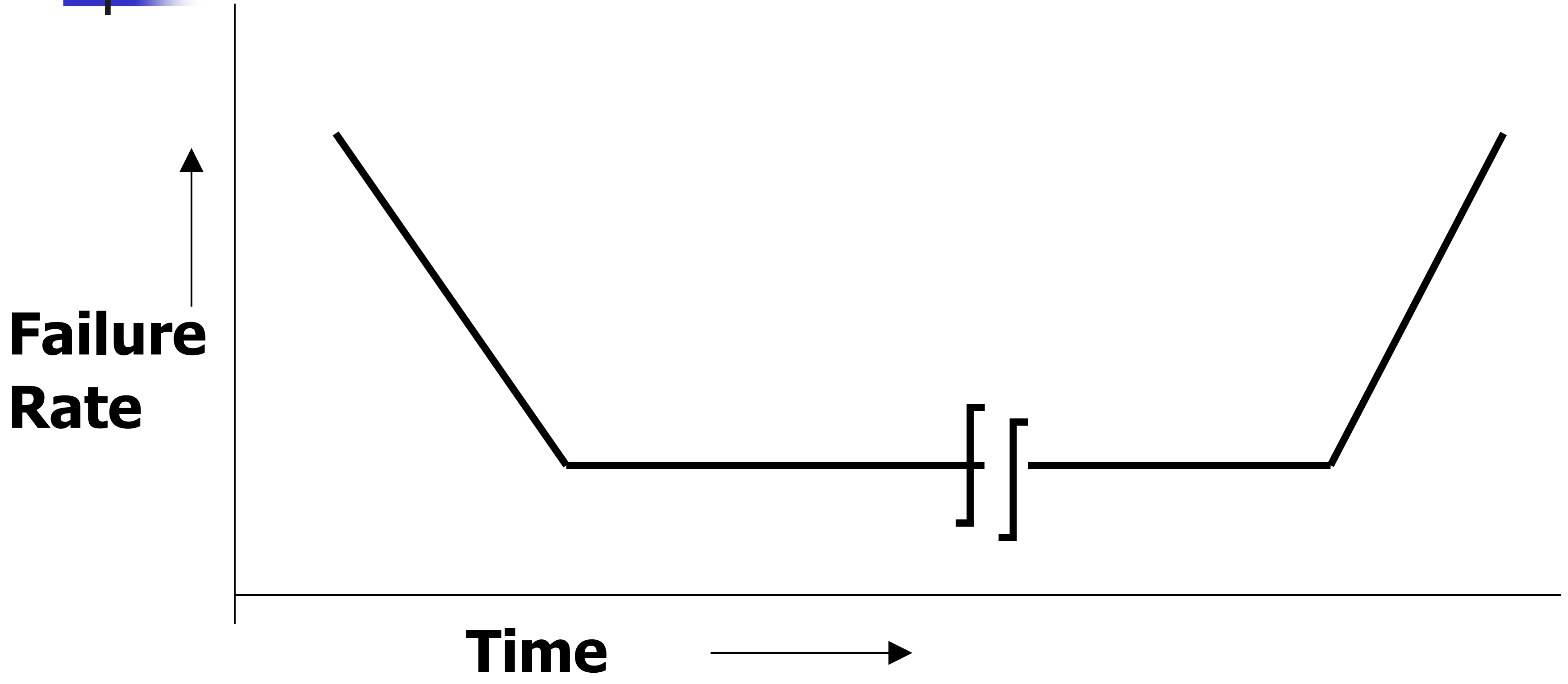
Hardware vs. Software Reliability

- When a hardware is repaired:
  - its reliability is maintained
- When software is repaired:
  - its reliability may increase or decrease.

Hardware vs. Software Reliability

- Goal of hardware reliability study :
  - stability (i.e. interfailure times remains constant)
- Goal of software reliability study
  - reliability growth (i.e. interfailure times increases)

Digression: The Bath Tub Curve



Reliability Metrics

- Different categories of software products have different reliability requirements:
  - level of reliability required for a software product should be specified in the SRS document.

Reliability Metrics

- A good reliability measure should be observer-independent,
  - so that different people can agree on the reliability.

Rate of occurrence of failure (ROCOF):

- ROCOF measures:
  - frequency of occurrence failures.
  - observe the behavior of a software product in operation:
    - over a specified time interval
    - calculate the total number of failures during the interval.

Mean Time To Failure (MTTF)

- Average time between two successive failures:
  - observed over a large number of failures.

Mean Time To Failure (MTTF)

- MTTF is not as appropriate for software as for hardware:
  - Hardware fails due to a component's wear and tear
    - thus indicates how frequently the component fails
  - When a software error is detected and repaired:
    - the same error never appears.

Mean Time To Failure (MTTF)

- We can record failure data for n failures:
  - let these be t1, t2, …, tn
  - calculate (ti+1-ti)
  - the average value is MTTF (ti+1-ti)/(n-1)

Mean Time to Repair (MTTR)

- Once failure occurs:
  - additional time is lost to fix faults
- MTTR:
  - measures average time it takes to fix faults.

Mean Time Between Failures (MTBF)

- We can combine MTTF and MTTR:
  - to get an availability metric:
  - MTBF=MTTF+MTTR
- MTBF of 100 hours would indicae
  - Once a failure occurs, the next failure is expected after 100 hours of clock time (not running time).

Probability of Failure on Demand (POFOD)

- Unlike other metrics
  - This metric does not explicitly involve time.
- Measures the likelihood of the system failing:
  - when a service request is made.
  - POFOD of 0.001 means:
    - 1 out of 1000 service requests may result in a failure.

Availability

- Measures how likely the system shall be available for use over a period of time:
  - considers the number of failures occurring during a time interval,
  - also takes into account the repair time (down time) of a system.

Availability

- This metric is important for systems like:
  - telecommunication systems,
  - operating systems, etc. which are supposed to be never down
  - where repair and restart time are significant and loss of service during that time is important.

Reliability metrics

- All reliability metrics we discussed:
  - centered around the probability of system failures:
  - take no account of the consequences of failures.
    - severity of failures may be very different.

Reliability metrics

- Failures which are transient and whose consequences are not serious:
  - of little practical importance in the use of a software product.
  - such failures can at best be minor irritants.

Failure Classes

- More severe types of failures:
  - may render the system totally unusable.
- To accurately estimate reliability of a software product:
  - it is necessary to classify different types of failures.

Failure Classes

- Transient:
  - Transient failures occur only for certain inputs.
- Permanent:
  - Permanent failures occur for all input values.
- Recoverable:
  - When recoverable failures occur:
    - the system recovers with or without operator intervention.

Failure Classes

- Unrecoverable:
  - the system may have to be restarted.
- Cosmetic:
  - These failures just cause minor irritations,
    - do not lead to incorrect results.
  - An example of a cosmetic failure:
    - mouse button has to be clicked twice instead of once to invoke a GUI function.

Reliability Growth Modelling

- A reliability growth model:
  - a model of how software reliability grows
    - 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.
  - i.e. how long to test the system?

Reliability Growth Modelling

- There are two main types of uncertainty:
  - in modelling reliability growth which render any reliability measurement inaccurate:
- Type 1 uncertainty:
  - our lack of knowledge about how the system will be used, i.e.
    - its operational profile

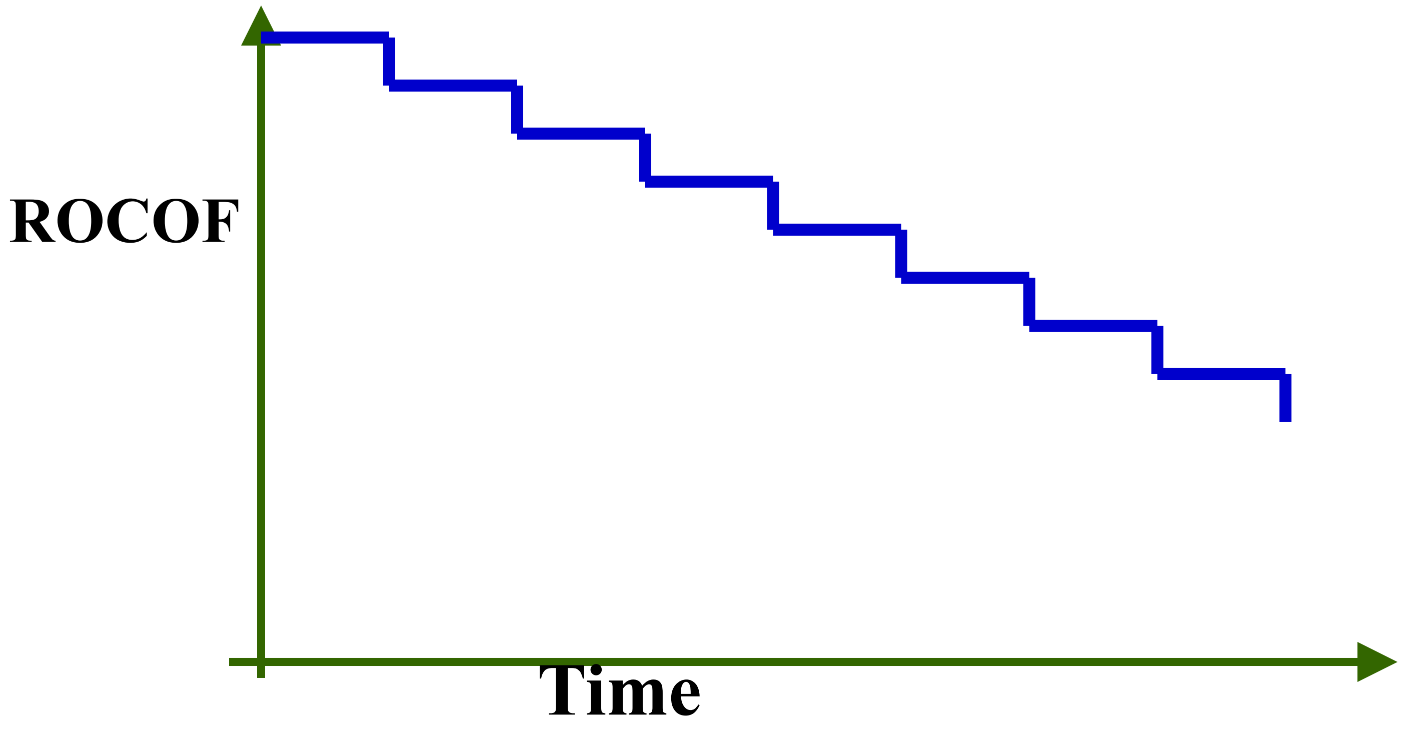
Reliability Growth Modelling

- Type 2 uncertainty:
  - reflects our lack of knowledge about the effect of fault removal.
  - When we fix a fault
    - we are not sure if the corrections are complete and successful and no other faults are introduced
  - Even if the faults are fixed properly
    - we do not know how much will be the improvement to interfailure time.

Step Function Model

- The simplest reliability growth model:
  - a step function model
- The basic assumption:
  - reliability increases by a constant amount each time an error is detected and repaired.

Step Function Model



Step Function Model

- Assumes:
  - all errors contribute equally to reliability growth
  - highly unrealistic:
    - we already know that different errors contribute differently to reliability growth.

Jelinski and Moranda Model

- Realizes each time an error is repaired:
  - reliability does not increase by a constant amount.
- Reliability improvement due to fixing of an error:
  - assumed to be proportional to the number of errors present in the system at that time.

Jelinski and Moranda Model

- Realistic for many applications,
  - still suffers from several shortcomings.
  - Most probable failures (failure types which occur frequently):
    - discovered early during the testing process.

Jelinski and Moranda Model

- Repairing faults discovered early:
  - contribute maximum to the reliability growth.
- Rate of reliability growth should be large initially:
  - slow down later on,
  - contrary to assumption of the model

Littlewood and Verall's Model

- Allows for negative reliability growth:
  - when software repair introduces further errors.
  - Models the fact that as errors are repaired:
    - average improvement in reliability per repair decreases.

Littlewood and Verall's Model

- Treats a corrected bug's contribution to reliability improvement:
  - an independent random variable having Gamma distribution.
- Removes bugs with large contributions to reliability:
  - earlier than bugs with smaller contribution
  - represents diminishing return as test continues.

Reliability growth models

- There are more complex reliability growth models,
  - more accurate approximations to the reliability growth.
  - these models are out of scope of our discussion.

Applicability of Reliability Growth Models

- There is no universally applicable reliability growth model.
- Reliability growth is not independent of application.

Applicability of Reliability Growth Models

- Fit observed data to several growth models.
  - Take the one that best fits the data.

Statistical Testing

- A testing process:
  - the objective is to determine reliability rather than discover errors.
  - uses data different from defect testing.

Statistical Testing

- Different users have different operational profile:
  - i.e. they use the system in different ways
  - formally, operational profile:
    - probability distribution of input

Operational profile: Example

- An expert user might give advanced commands:
  - use command language interface, compose commands
- A novice user might issue simple commands:
  - using iconic or menu-based interface.

How to define operational profile?

- Divide the input data into a number of input classes:
  - e.g. create, edit, print, file operations, etc.
- Assign a probability value to each input class:
  - a probability for an input value from that class to be selected.

Steps involved in Statistical testing (Step-I)

- Determine the operational profile of the software:
  - This can be determined by analyzing the usage pattern.

Step 2 in Statistical testing

- Manually select or automatically generate a set of test data:
  - corresponding to the operational profile.

Step 3 in Statistical testing

- Apply test cases to the program:
  - record execution time between each failure
  - it may not be appropriate to use raw execution time

Step 4 in Statistical testing

- After a statistically significant number of failures have been observed:
  - reliability can be computed.

Statistical Testing

- Relies on using large test data set.
- Assumes that only a small percentage of test inputs:
  - likely to cause system failure.

Statistical Testing

- It is straight forward to generate tests corresponding to the most common inputs:
  - but a statistically significant percentage of unlikely inputs should also be included.
- Creating these may be difficult:
  - especially if test generators are used.

Advantages of Statistical Testing

- Concentrate on testing parts of the system most likely to be used:
  - results in a system that the users find more reliable (than actually it is!).

Advantages of Statistical Testing

- Reliability predictions based on test results:
  - gives an accurate estimation of reliability (as perceived by the average user) compared to other types of measurements.

Disadvantages of Statistical Testing

- It is not easy to do statistical testing properly:
  - there is no simple or repeatable way to accurately define operational profiles.
- Statistical uncertainty.

Summary

- Reliability of a software product:
  - essentially denotes its trustworthiness or dependability.
  - probability of the product working “correctly” over a given period of time.

Summary

- Operational profile of a software
  - reflects how it will be used in practice.
  - Consists of specification of:
    - classes of inputs
    - probability of their occurrence.

Summary

- Statistical testing:
  - uses large data set selected based on operational profile.
  - Provides more realistic reliability figures.