Software Reliability

by
Manik Chand Patnaik

Topics

- Characteristics of Software
- Reliability
- Designing a Software with Reliability in view
- Estimating Reliability
- Error Classification
- Statistical Testing

Important Design objectives

The two operational design objectives continually sought by developer are systems reliable & maintainable systems

- A system is said to be reliable if it doesn't produce dangerous or costly failures when used in a reasonable manner
- Maintainability refers to the ability to keep the system up-to-date. This is possible through proper design practices

Software Vs. Hardware Characteristics

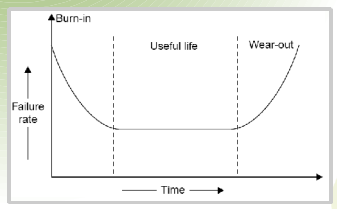
There are three phases in the life of any hardware component i.e., burn-in, useful life & wear-out.

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.

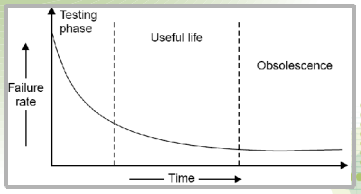
Failure rate increase in **wear-out phase** due to wearing out/aging of components. The best period is useful life period. The shape of this curve is like a "bath tub" and that is why it is known as bath tub curve. The "bath tub curve" is given in Figure >

Bath Tub Curve of Hardware



Curve of Software

We do not have wear out phase in software. The expected curve for software is given in figure



Why Software becomes Obsolete?

- 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

Hardware Vs Software Reliability

- When a hardware is repaired:
  - its reliability is maintained
- When software is repaired:
  - its reliability may increase or decrease.
- Goal of hardware reliability:
  - Stability (i.e. interfailure times remains constant)
- Goal of software reliability:
  - Growth of reliability (i.e. inter-failure times increases)

What is Software Reliability? >

- "Software reliability means operational reliability". Who cares how many bugs are in the program?
- As per IEEE standard: "Software reliability is defined as the ability of a system or component to perform its required functions under stated conditions for a specified period of time".

What is Software Reliability? >

- **Probability that a software system fulfills its assigned task in a given environment for a predefined number of input cases**, assuming that the hardware and the inputs are free of error.
- "It is the probability of a failure free operation of a program for a specified time in a specified environment".

Extra points on Reliability

- Informally: denotes a product's trustworthiness or dependability.
- Also: Probability of the product working "correctly" over a given period of time.
- 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.

Approaches to reliability

- Error Avoidance
  - Prevention of errors from occurring in a system
- Error Detection & Correction
  - Recognizing errors when they occur and correcting them so that the system does not fail or the effect of the error is minimized
- Error Tolerance
  - Recognizing errors when they occur, but enabling the system to keep running, possibly with degraded performance.

Designing Maintainable systems

The following precautions can reduce the need for maintenance.

- Accurately define user's requirements
- Use effective methods for designing processing logic and communicate it to project team members
- Make use of existing tools & techniques

Types of system maintenance

Category	Activity	Relative frequency
Corrective	Routine debugging	20%
Adaptive	Accommodating changes to data, files, h/w, s/w	20%
Perfective	Enhancements, re-coding for computational efficiency	60%

Modular design of S/W

- Modularity & partitioning
  - Hierarchical arrangement of modules
  - Lower level modules are smaller in scope & size
- Coupling (modules should have little dependence on other modules in a system)
- Cohesion (modules should carry out a single processing function)
- Span of control (modules should interact/manage limited number of lower-level modules)
- Size (the no. of instructions within a module should be kept low)
- Shared use (modules should be reused as far as possible)

Reliability Assessment

- 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.

Is it Easy?

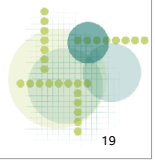
- Accurate measurement of software reliability:
  - a **very difficult** problem
  - Several factors contribute to making measurement of software reliability difficult.
- Errors do not cause failures at the same frequency and severity.
  - measuring latent-errors alone not enough
  - The failure rate is **observer-dependent**

Difficulties in Reliability Estimation

- 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.

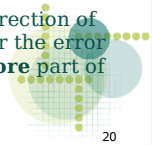
90-10 Rule

- Experiments from analysis of behavior of a large number of programs shows:
  - 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.



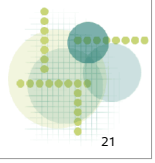
Effect of 90-10 Rule on Reliability

- So, **90%** statements: are executed only **10%** of the 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.
- 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.



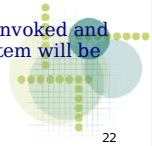
Operational Profile

- The perceived reliability depends to a large extent upon:
  - how the product is used,
- In technical terms it is **operational profile of the software**.



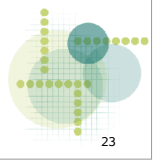
Effect of Operational Profile on Reliability Estimation

- If your input results in the invocation of only "correctly" implemented functions then :-
  - none of the errors will be exposed
  - perceived reliability of the product will be **high.**
- Otherwise :-
  - functions **containing errors** are invoked and the perceived reliability of the system will be **low.**



Perception of 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.



Reliability Metrics

- **Different categories** of software products have **different reliability requirements**: e.g. Video player Vs. Banking Application
- level of reliability required for a software product **should be specified in the SRS** document.
- 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.
- **To calculate Average only program run time is taken.**
  - **Startup, Poweroff etc eliminated.**

Mean Time To Failure (MTTF)

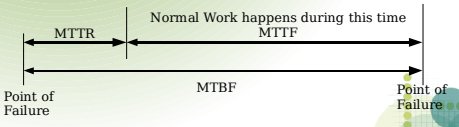
- Average time between two successive failures: observed over a large number of failures.
- **MTTF is appropriate for hardware.**
- Hardware fails due to a component's wear and tear thus indicates how frequently the component fails but
- When a software error is detected and repaired and the same error never appears again.

Mean Time to Repair (MTTR)

- It is the average of **"Time Required to Isolate the error and Fix it"** after the failure.

Mean Time Between Failures (MTBF)

- It combines both MTTF and MTTR metrics.
- It is the next expected failure time after one failure.



MTBF of 100 hours would indicate, Once a failure occurs, the next failure is expected after 100 hours of clock time (not running time).

Probability of Failure on Demand (POFOD)

- Instead of time, this metric takes number of requests into account.
- 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

- This measures in percentage, the time normally the system is expected to be Up and running.
- Total Time – {(Startup/)+ Routine work + MTTR / Shutdown Maintenance}
- This metric is important for systems like :-
- **telecommunication systems, operating systems, web services,** etc. which are supposed to be **never down** and where repair and restart time are significant and loss of service during that time is important.

About: 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.
- Failures which are transient and are not serious:such failures can at best be minor irritants.

Failure Classes

Failures are classified to the following types:-

- Transient
- Permanent
- Recoverable
- Unrecoverable
- Cosmetic

Transient Failure >

- Failure occurs only for some input values.
- Otherwise the system is normally functioning.
- It is a temporary error.
- It does not cause the system to crash.
- Small bits of data may be lost.
- Generally system has some kind of feedback which shows information about the error.

Transient Contd.

- No serious damage to user session and data. The user is not logged out / session is not lost.
- There is instant recovery from the error.
- e.g.:- In a networked system, if a single user gets the error then a small amount of data pertaining to that single user is lost. The user's login remains valid and other user's data also does not get damaged. Mostly this error would result due to improper data input.

Permanent Failure>

- When a function results in failure for all kinds of input values.
- The inputs can be anything like: clicking any button, entering any value like username & password, for any kind of operation error occurs in software. The software does not seem to accept any input.

Permanent Contd.

- e.g. This may happen due to
  - loss of network,
  - filled up HDD,
  - database error,
  - Deletion of system files.
  - Corruption of configuration files. (A configuration file is a file which contains startup/configuration parameters. It should not contain bugs or errors.)

Permanent Contd.

- Recovery from a permanent error:-
  - May be as simple as a reboot
  - Recovery routines may help
  - If network problem is the cause, a proper network setup can restore it.
- However it is known that permanent errors may cause some data loss, may be little / more / none depending on the situation.

Recoverable >

- When after a failure the system can be successfully recovered back, with/without the operator's intervention, it is called recoverable error.
- Presence of MTTR metrics indicates the recoverable nature of the system.
- Recovery is of two types
  - Manual and
  - Automatic

Recoverable: Manual Recovery

- Start the recovery routine software manually.
- E.g. Repair option of installer.
- E.g. Recovery routine in database. In database crash, from the debris, data can be salvaged.
- E.g. Window System restore.
- E.g. Norton Go Back. If we install Norton Go Back , we have periodic snapshots and if error is there in a software, it will restore it using a recent snapshot.

Recoverable: Automatic Recovery

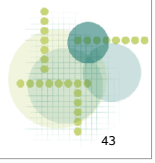
- When recovery feature is in-built in the application, automatic recovery is possible.
- E.g. Autosave of word processors.
- E.g. Automatic Checkdisk after unclean shutdown
- E.g. Automatic recovery in databases.
- Preventing errors from occurring in large systems is almost impossible. So recovery is required.

Recoverable: role of journal >

- A set of operations to be performed is stored as a list journal. A journal can prevent data loss.
- There are two kinds of operations done on the Journal.
- 1) Writing of the journal:-Appending the “list of actions to be done” as entries in journal.
- 2) Playing of a journal:-Doing the action and after that individual entries are removed to indicate the completion of task.

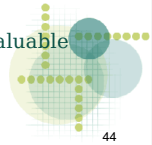
Recoverable: role of journal

- Restoration with Journal:-
  - If an error occurs, the entry remains pending in the journal. During recovery the entry will be re-done.



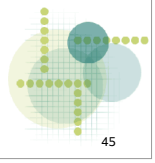
Unrecoverable Failure

- Failures which need to be attended by a trained operator from the vendor for restoring of the system which may result in data-loss are known as unrecoverable errors.
- There is also the likelihood that the system will be back in operation after a plain restart of the system.
- These are the error where some valuable data are lost.



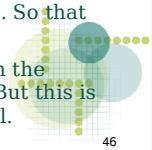
Unrecoverable Contd.

- Unrecoverable errors often requires system restart which is not good. So we should try to build some features which can keep the system running at-least with degraded performance.
- This requires us to use the concept of graceful degradation.



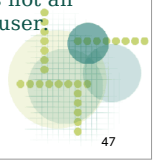
Unrecoverable: solution Contd..

- **Graceful Degradation** "Recognizes error when they occur, but enables the system to keep running through degraded performance or by applying rules that instruct the system how to continue processing."
- e.g.:-Shut down part of the system. So that system is not down.
- The user receives less service than the system was designed to provide. But this is better than having no service at all.



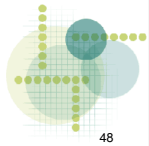
Cosmetic Errors

- These are only minor irritations.
- They don't tend to wrong results
- e.g.:-In desktop, if background color is black & text color is black, the user does not see anything. That means this does not cause damage to the data or text. So it is not an error. It is just an irritation to the user.



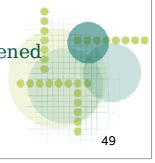
Statistical Testing

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



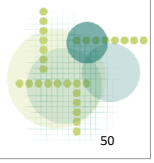
Step-1 Operation Profile

- Different users have different operational profile: i.e. using the system in differently.
- Formal def:probability distribution of input.
- E.g. (Each operation is a class)
- Certain options regularly used
- Certain I/O regularly performed
  - Certain Input dialogs regularly opened
  - Certain Reports normally created



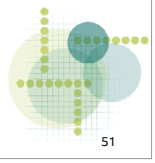
Step-2 Designing the Tests

- Test cases are created to test each class of operations.
- Number of such cases must be statistically significant.
- Cases for failure condition also must be there.



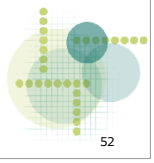
Step-3 Running the Tests

- Tests are run and failure times are noted.



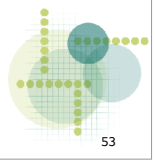
Step-4 Computation

- The statistical data about failures is taken and calculation on reliabilities are made.



Merits

- It focuses to test the most likely used parts of the system.
- Reliability estimates are more accurate.



Limitations

- The process is quite involved and difficult.
- There is no straight forward way to find operation profiles.
- Number of Test Cases must be statistically significant (A Large Number of)

Finally,

There

Is

no

End

The show will go on and . . .