# Software Reliability

**NILESH PANCHOLI**
**B.E. ( Mech.), M.E. (Mech.), Ph. D.**
**Email:** ***nhpancholi@gmail.com***
***www.nileshpancholi.com***

## Contents

- Software reliability concept, Software life cycle, Historical software reliability techniques, Current trends and problems and possible future directions, Software reliability models – Static and dynamic models, Reliability Growth Modeling with Covariates, When to stop testing software, Capabilities and comparisons of commercially available reliability and maintenance software, Software based case study

# Software Reliability

- Basic definitions:
  - *S/W reliability*: probability that the software will not cause a failure for some specified time.
  - *Failure*: divergence in expected external behavior.
  - *Fault*: cause/representation of an error, i.e., a bug
  - *Error*: a programmer mistake (misinterpretation of specifications?)

# Software Reliability

- **Basic question:** How to estimate the growth in software reliability as its errors are being removed?
- **Major issues:**
    - testing - (how much? When to stop!)
    - field use ( # of trained personnel? Support staff?)
- **S/W reliability growth models:** observe past failure history and give an estimate of the future failure behavior; about 40 models have been proposed.

# Reliability and Availability

- A <u>simple measure</u> of reliability can be given as:

  MTBF = MTTF + MTTR , where

  MTBF is mean time between failures

  MTTF is mean time to fail

  MTTR is mean time to repair

## Reliability and Availability

• **Availability** can be defined as the probability that the system is still operating within requirements at a given point in time and can be given as:

$$\text{Availability} = \frac{\text{MTTF}}{(\text{MTTF} + \text{MTTR})} \times 100\%$$

• availability is more sensitive to MTTR which is an indirect measure of the maintainability of software.

## Reliability and Availability

- Since each error in the program does not have the same failure rate,
  - the total error count does not provide a good indication of the reliability of the system.
- MTBF is perhaps more useful (meaningful) than defects/KLOC since the user is concerned with failures not the total error count.

## Historical SoftwareSoftware Reliability Techniques

**Fault lifecycle techniques:**

- 1) Fault prevention: to avoid, by construction, fault occurrences.
- 2) Fault removal: to detect, by verification and validation, the existence of faults and eliminate them.
- 3) Fault tolerance: to provide, by redundancy, service complying with the specification in spite of faults having occurred or occurring.
- 4) Fault/failure forecasting: to estimate, by evaluation, the presence of faults and the occurrences and consequences of failures. This has been the main focus of software reliability modeling.

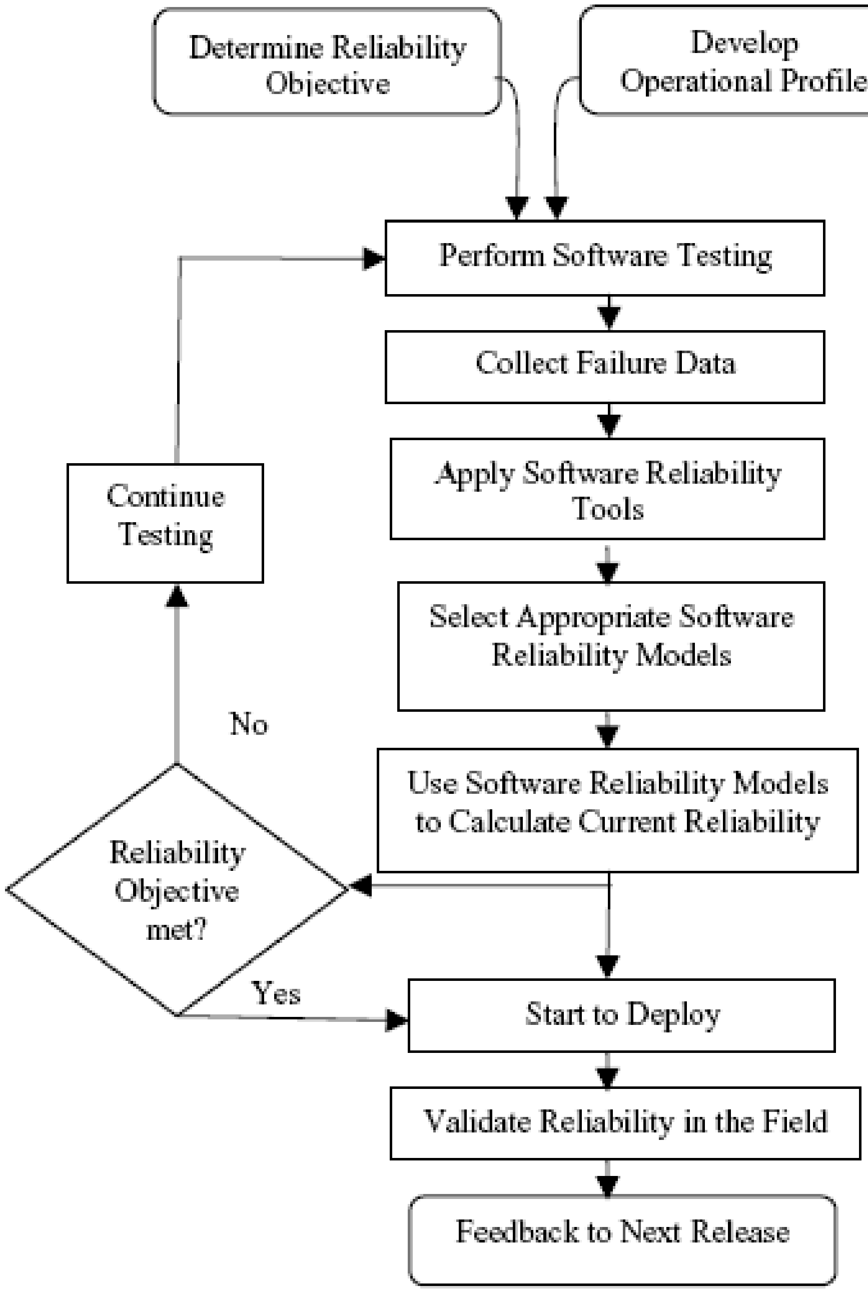
Determine Reliability Objective
Develop Operational Profile
Perform Software Testing
Collect Failure Data
Apply Software Reliability Tools
Select Appropriate Software Reliability Models
Use Software Reliability Models to Calculate Current Reliability
Continue Testing
No
Reliability Objective met?
Yes
Start to Deploy
Validate Reliability in the Field
Feedback to Next Release

**Figure Software Reliability Engineering Process Overview**