

Capabilities Overview

April 2007

*reproduces research and ideas from reliability engineering articles, papers and books.

Three-part presentation to cover aspects of Infonetica QA & Testing practice:

- ◆ **Testing for Reliability [this presentation]**
- ◆ **Test Automation**
- ◆ **Security & Vulnerability**

What is Software Reliability Engineering?

- ◆ **Relationship with Infonetica QMS**

Engagement Opportunities with you

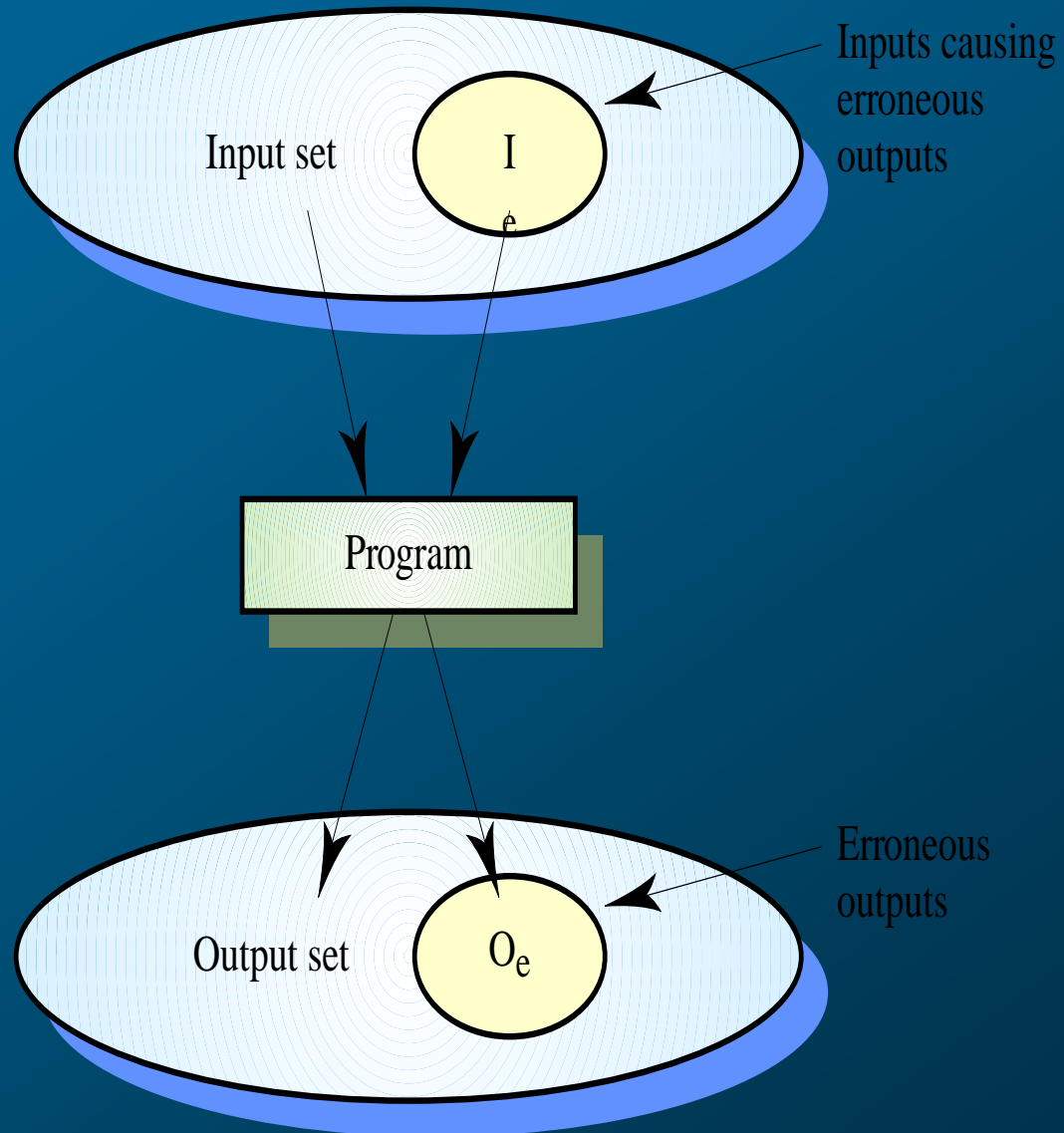
The probability that a system or a capability of a system will continue to function without failure for a specified period in a specified environment.

The period may be specified in natural or time units.

R (for 1h mission time)	Failure intensity
0.386	1 failure/h
0.9	105 failures/1000h
0.959	1 failure/day
0.99	1 failure/100 h
0.994	1 failure/week
0.9986	1 failure/month
0.999	1 failure/1000 h
0.99989	1 failure/year

It helps to involve customers in defining requirements regarding failure rates

CMM levels 4 and 5 (and 3, indirectly), recommend reliability measurement.



The reliability of software-based product depends on how the computer and other external elements use it.

Reliability is improved when software faults which occur in the most frequently used parts of the software are removed

- ◆ **Removing x% of software faults will not necessarily lead to an x% reliability improvement. In a study, removing 60% of software defects actually led to a 3% reliability improvement**
- ◆ **Removing faults with serious consequences is the most important objective**

The set of best practices that empower testers and developers to

- ◆ **Ensure product reliability meets users needs**
- ◆ **Speed the product to market faster**
- ◆ **Reduce product cost**
- ◆ **Improve customer satisfaction (fewer angry users)**
- ◆ **Increase their productivity**

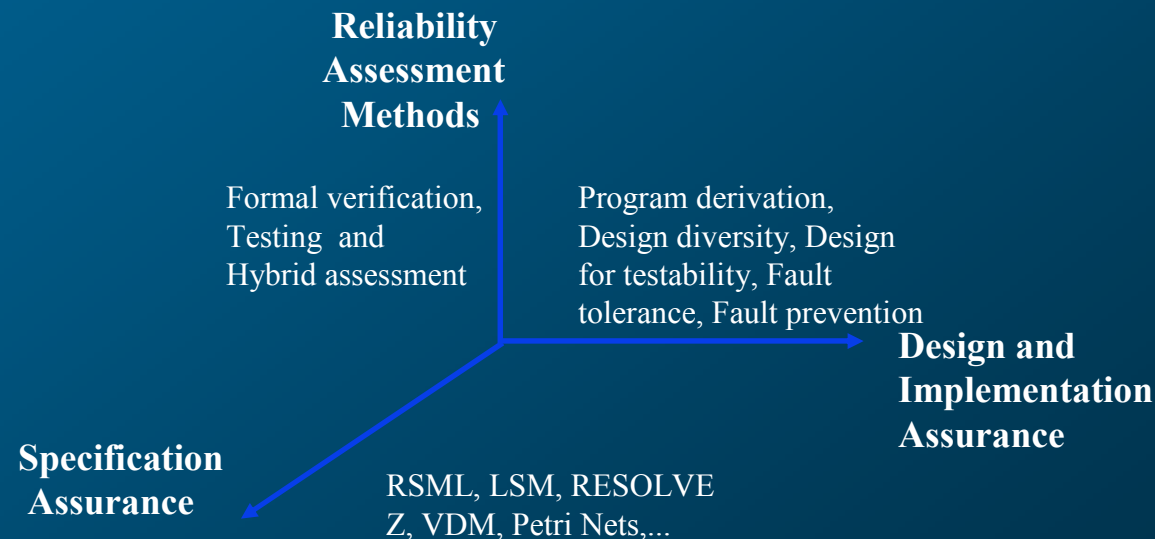
Applicable to all software based systems

Two fundamental ideas

- ◆ **Focus resources on the most used/critical functions**
- ◆ **Make testing realistically represent field conditions**

Software faults introduced in all phases of the life-cycle: specification, design, implementation, testing, maintenance.

- ◆ **Reliable operation requires assurance in all the phases of the life-cycle**



Widely used and accepted, especially by the large corporations and ISVs

Increase in project cost: less than 1%

Predominant SRE workflow:

Define Necessary
Reliability

Develop Operational Profiles (Customer, User,
Mode, Functions, Operation profile itself)

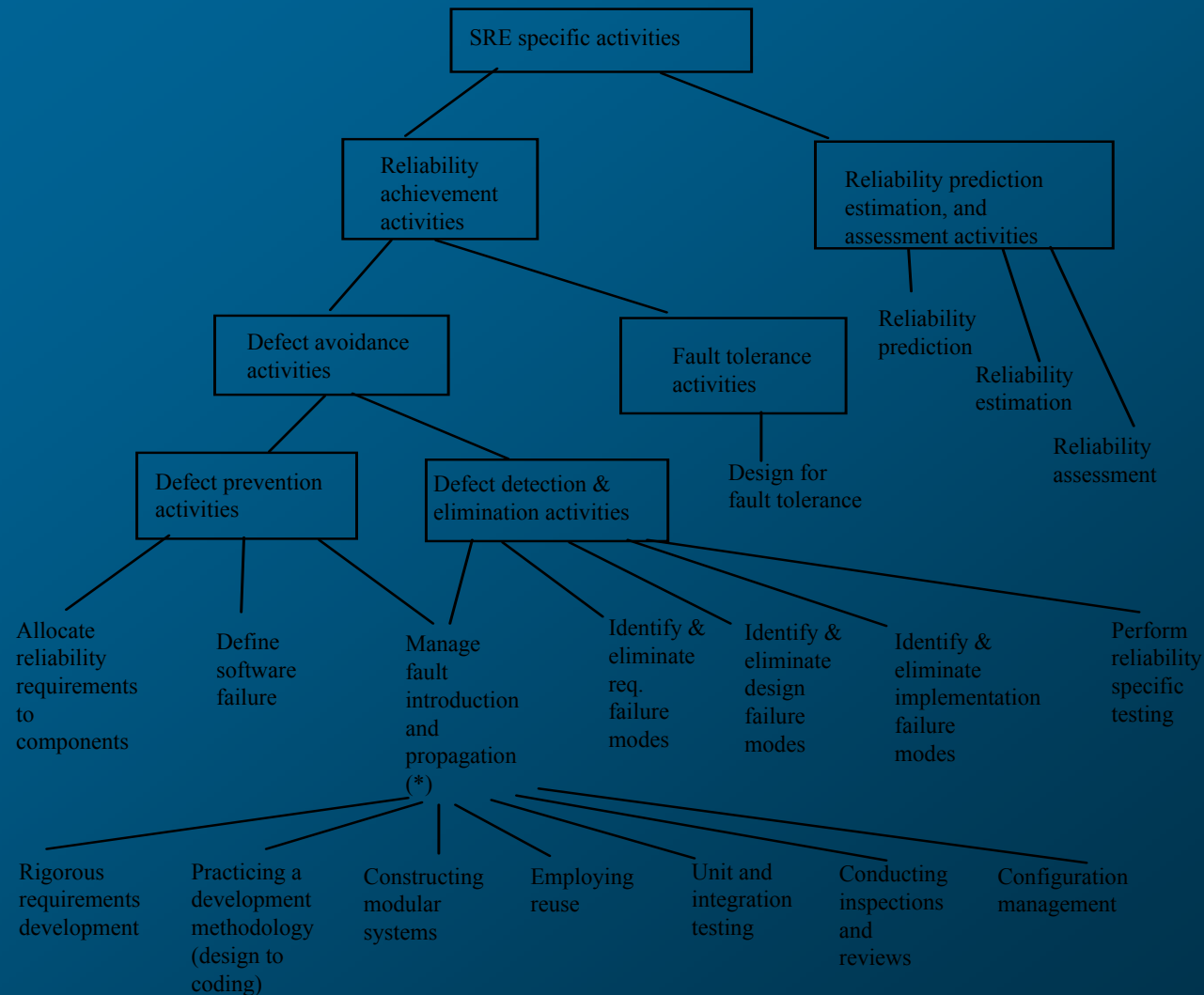
Prepare for Test

Execute Tests & Apply Failure Data
to Guide Decisions

Requirements and
architecture

Design and
Implementation

Test & Validation



(*) not SRE specific

Reliability based approach for new development

Testing for reliability

- ◆ **Statistical testing**
- ◆ **Reliability growth tests (find and remove faults)**
 - ◆ *Feature* (minimize impact of the environment),
 - ◆ *load* (maximize environmental impacts),
 - ◆ *regression* tests (following a major change)
- ◆ **Certification tests**
 - ◆ no debugging, accept or reject software under test

Reliability Assessments

Reliability Growth Modeling

Reliability Predictions

Reliability Estimations

Reliability Optimization testing using OOAD

Develop a complete object model, including all use cases

Identify key operational variables

Establish operational relationships with each use case

Develop an operational profile for all use cases

Estimate the frequencies of use cases, then operations.

Estimate testing productivity

Evaluate system test effort budget

Evaluate coverage, add more tests, if necessary. Conduct testing according to operational profile until acceptable low failure rate is observed using reliability techniques

Use Case	Actor	Scenario
Cash Withdrawal	Bank Customer	Wrong pin, request \$75
	Bank Customer	PIN OK, deposit \$300, request \$50
	Crook	Stolen card, Valid PIN
ATM Restocking	Operator & Guard	ATM opened, Cash dispenser empty, \$150000 added
	Operator & Guard	ATM Opened, cash dispenser full

Operational Variables				Expected Result	
Card PIN	Entered PIN	Customer Bank reply	Customer Acct. Status	Message Displayed	Card Action
Invalid	-	-	-	Insert ATM Card	Eject
Valid	Matches Card PIN	OK	Closed	Account Closed	Eject
Valid	Matches	OK	Open	Enter Amount	Keep
Valid	Matches	No reply	-	Try Later	Eject
Valid	Doesn't Match	-	-	Reenter PIN	Keep
Revoked	-	Bank Replies	-	Card Revoked	Retain
Revoked	-	No reply	-	Card Invalid	Eject

Assumptions

- ◆ It takes 1 hour to design and run one test
- ◆ 5 percent of tests reveal faults
- ◆ It takes 4 hours to correct faults
- ◆ Test budget = 1000 hours

Number of tests to be performed:

$$T + (0.05 * 4T) = 1000$$

$$T = 833 \text{ (total number of tests)}$$

Use Case	Occurrence Probability	Number of Tests
Cash Withdrawal	0.53	441
Checking Deposit	0.15	125
Savings Deposit	0.14	117
Funds Transfer	0.08	67
Balance Inquiry	0.06	50
Restock	0.02	17
Collect Deposits	0.02	16
Total	1.00	833

Testing software for reliability rather than fault detection

Test data selection should follow the predicted usage profile for the software

Measuring the number of errors allows the reliability of the software to be predicted

An acceptable level of reliability should be specified and the software tested and amended until that level of reliability is reached

Determine operational profile of the software

Generate a set of test data corresponding to this profile

Apply tests, measuring amount of execution time between each failure

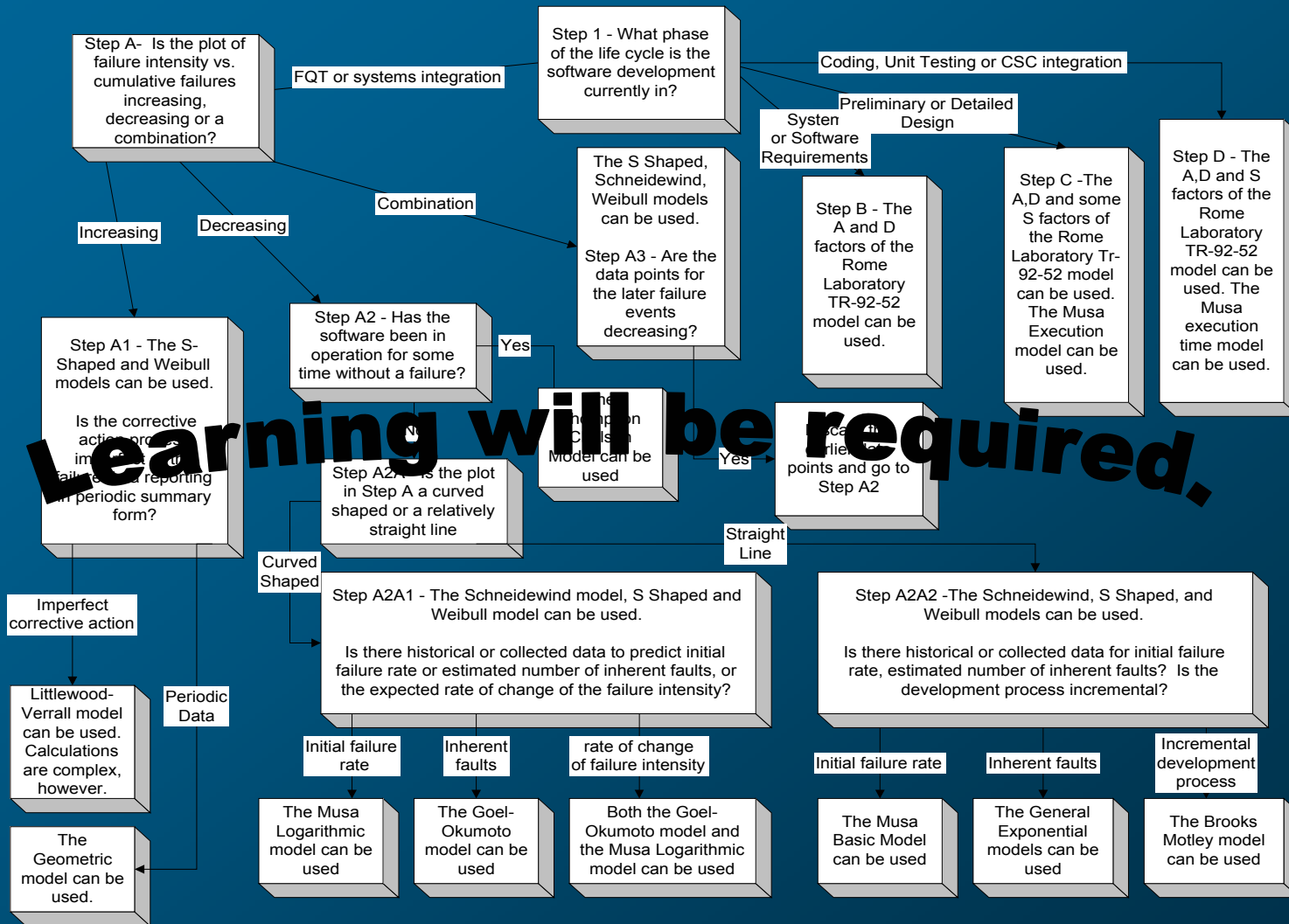
After a statistically valid number of tests have been executed, reliability can be measured

Growth model is a mathematical model of the system reliability change as it is tested and faults are removed

Used as a means of reliability prediction by extrapolating from current data

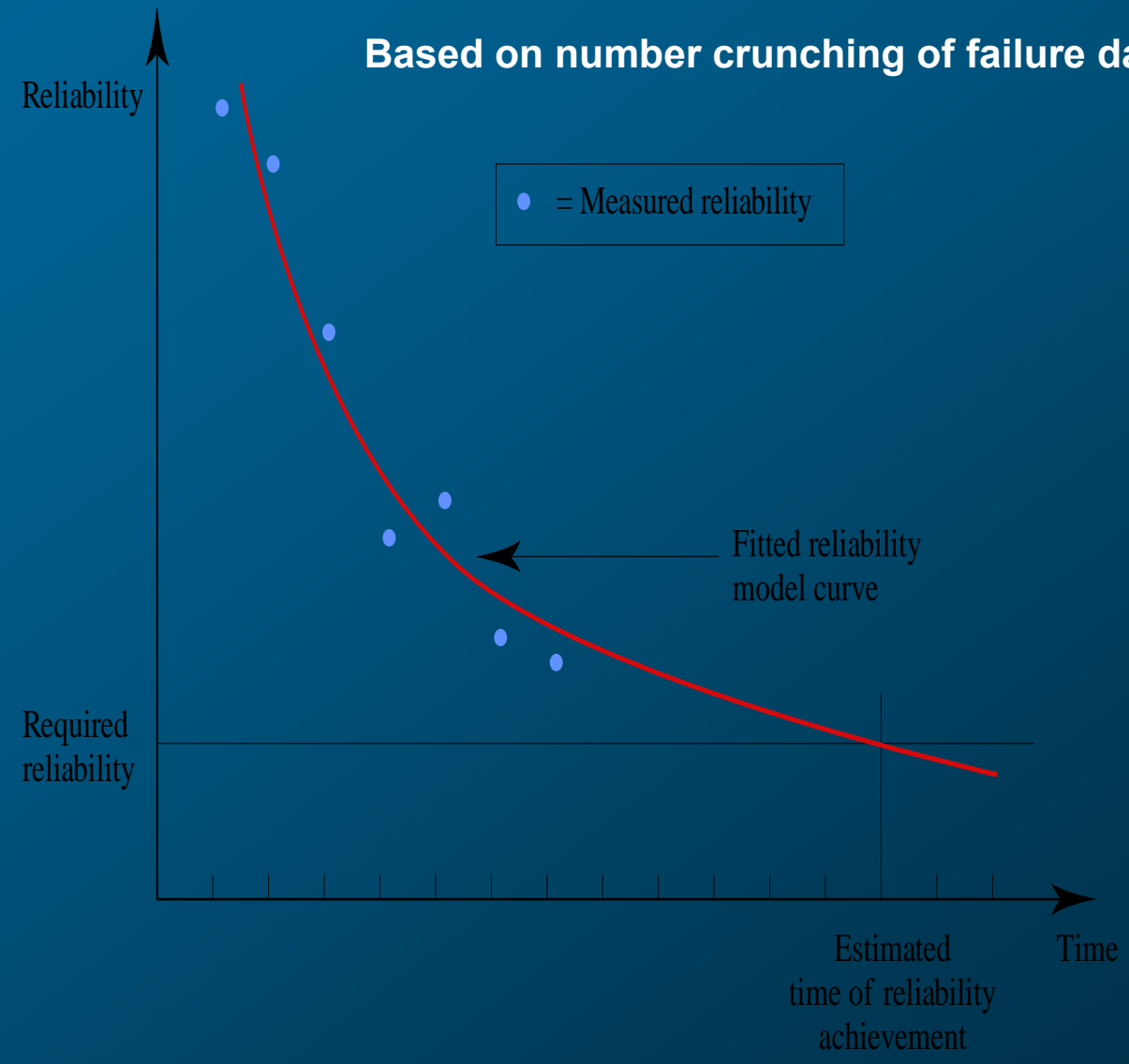
Depends on the use of statistical testing to measure the reliability of a system version

Many different reliability growth models have been proposed. Reliability should be measured and observed data should be fitted to several models. Best-fit model should be used for reliability prediction



Learning will be required.

Based on number crunching of failure data and rates.



Reliability Models

Example: Assume that a program will experience 100 failures in infinite time.
 The initial failure intensity was 10 failures/CPU-hr, the present failure intensity is 3.68 failures/CPU-hour and our objective intensity is 0.000454 failure/CPU-hr.
 Predict the additional testing time to achieve the stated objective.

Ans.: We know that $\lambda(\tau) = \lambda_0 \exp(-\lambda_0 \tau / v_0)$

At time τ_1 , $\lambda(\tau_1) = \lambda_0 \exp(-\lambda_0 \tau_1 / v_0) = \lambda_p$

At time τ_2 , $\lambda(\tau_2) = \lambda_0 \exp(-\lambda_0 \tau_2 / v_0) = \lambda_f$

$\tau_2 - \tau_1 = (v_0 / \lambda_0) \cdot \ln(\lambda_p / \lambda_f)$

$v_0 = 100$ faults, $\lambda_0 = 10$ failures/CPU-hr

$\lambda_p = 3.68$ failures/CPU-hr, $\lambda_f = 0.000454$ failure/CPU-hr

Testing time = $(\tau_2 - \tau_1) = 90$ CPU-hr

Reliability is usually the most important dynamic software characteristic

Reliability directly affects cost, price, schedule and other important decisions. Infonetica Professionals should aim to produce reliable software

Reliability depends on the pattern of usage of the software. Faulty software can be reliable.

Reliability requirements should be defined quantitatively whenever possible

There are many different reliability metrics. The metric chosen should reflect the type of system and the application domain

Statistical testing is used for reliability assessment. Depends on using a test data set which reflects the use of the software

Reliability growth models may be used to predict when a required level of reliability will be achieved