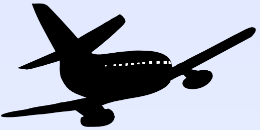
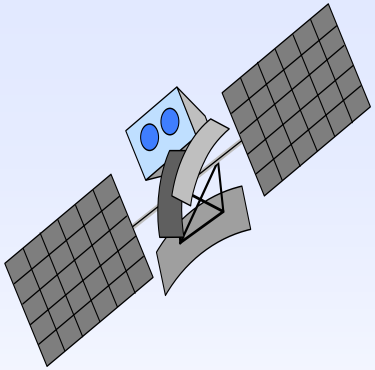


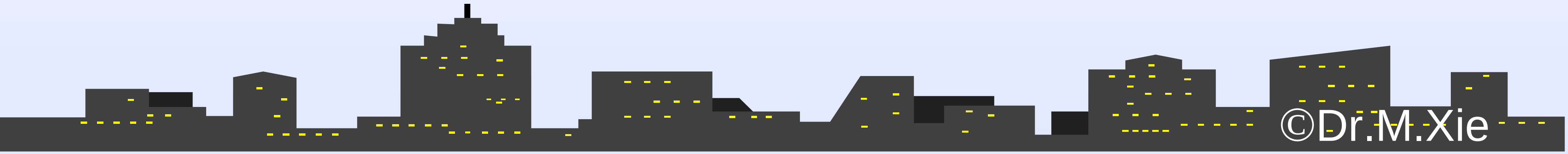


SOFTWARE SYSTEM SAFETY AND RELIABILITY

M. Xie, PhD (Quality), Fellow of IEEE

Professor, Dept of Industrial & Systems Engineering

National University of Singapore



Quality, Reliability, Safety

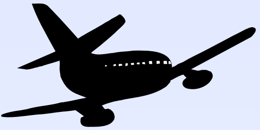
- Quality: multi-dimensional measurement
  - Plenty of data
- Reliability: most important attribute of product quality, study of failures, their causes and consequences
  - Some data
- Safety: dealing with most critical failures
  - Lack of data/information

My work/experience in Reliability

- Nuclear power plant monitoring system
- Telecommunication system
- Traffic control system
- Automobile
- Aerospace
- Mostly concerns software, complex, and safety-critical system

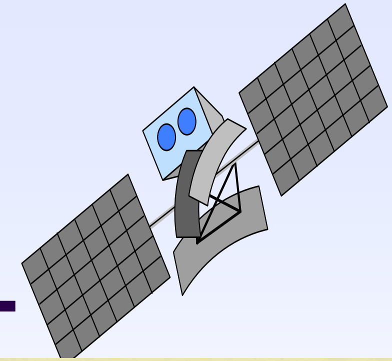
Reliability of Software System

- Complex systems contain both software and hardware
- Software is different from hardware in many aspects
- Hardware failures are easier to deal with
- Software problems are usually solved only by the developer
- For software system
  - Failure cause is identified after a failure
  - Action is taken to remove the cause
  - Same type of failure will not occur
  - Time to next failure is likely to be longer



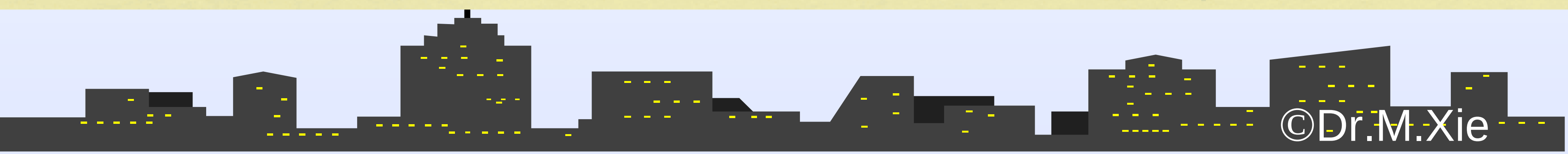
" Software Hall of Shame"

(from IEEE Spectrum, Sept 05 issue)



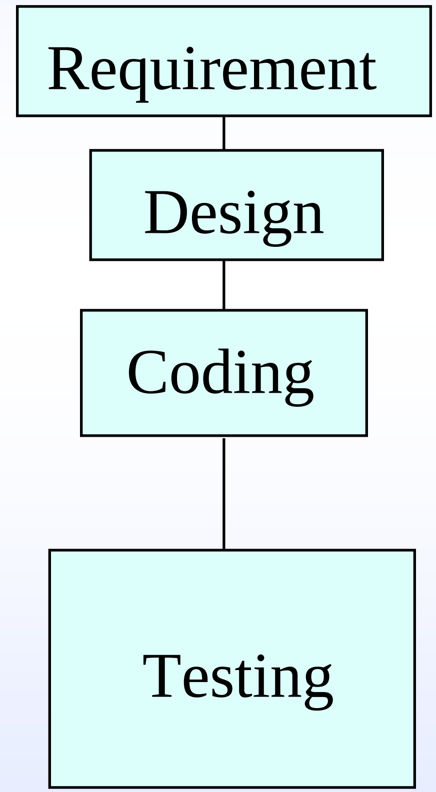
YEAR	COMPANY	OUTCOME (COSTS IN US $)
2005	Hudson Bay Co. [Canada]	Problems with inventory system contribute to $33.3 million* loss.
2004–05	UK Inland Revenue	Software errors contribute to $3.45 billion* tax-credit overpayment.
2004	Avis Europe PLC [UK]	Enterprise resource planning (ERP) system canceled after $54.5 million[†] is spent.
2004	Ford Motor Co.	Purchasing system abandoned after deployment costing approximately $400 million.
2004	J Sainsbury PLC [UK]	Supply-chain management system abandoned after deployment costing $527 million.[†]
2004	Hewlett-Packard Co.	Problems with ERP system contribute to $160 million loss.
2003–04	AT&T Wireless	Customer relations management (CRM) upgrade problems lead to revenue loss of $100 million.
2002	McDonald's Corp.	The Innovate information-purchasing system canceled after $170 million is spent.
2002	Sydney Water Corp. [Australia]	Billing system canceled after $33.2 million[†] is spent.
2002	CIGNA Corp.	Problems with CRM system contribute to $445 million loss.
2001	Nike Inc.	Problems with supply-chain management system contribute to $100 million loss.
2001	Kmart Corp.	Supply-chain management system canceled after $130 million is spent.
2000	Washington, D.C.	City payroll system abandoned after deployment costing $25 million.
1999	United Way	Administrative processing system canceled after $12 million is spent.
1999	State of Mississippi	Tax system canceled after $11.2 million is spent; state receives $185 million damages.
1999	Hershey Foods Corp.	Problems with ERP system contribute to $151 million loss.

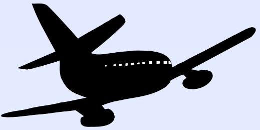
1999	United Way	Administrative processing system canceled after $12 million is spent.
1999	State of Mississippi	Tax system canceled after $11.2 million is spent; state receives $185 million damages.
1999	Hershey Foods Corp.	Problems with ERP system contribute to $151 million loss.
1998	Snap-on Inc.	Problems with order-entry system contribute to revenue loss of $50 million.
1997	U.S. Internal Revenue Service	Tax modernization effort canceled after $4 billion is spent.
1997	State of Washington	Department of Motor Vehicle (DMV) system canceled after $40 million is spent.
1997	Oxford Health Plans Inc.	Billing and claims system problems contribute to quarterly loss; stock plummets, leading to $3.4 billion loss in corporate value.
1996	Arianespace [France]	Software specification and design errors cause $350 million Ariane 5 rocket to explode.
1996	FoxMeyer Drug Co.	$40 million ERP system abandoned after deployment, forcing company into bankruptcy.
1995	Toronto Stock Exchange [Canada]	Electronic trading system canceled after $25.5 million** is spent.
1994	U.S. Federal Aviation Administration	Advanced Automation System canceled after $2.6 billion is spent.
1994	State of California	DMV system canceled after $44 million is spent.
1994	Chemical Bank	Software error causes a total of $15 million to be deducted from 100 000 customer accounts.
1993	London Stock Exchange [UK]	Taurus stock settlement system canceled after $600 million** is spent.
1993	Allstate Insurance Co.	Office automation system abandoned after deployment, costing $130 million.
1993	London Ambulance Service [UK]	Dispatch system canceled in 1990 at $11.25 million**; second attempt abandoned after deployment, costing $15 million.**
1993	Greyhound Lines Inc.	Bus reservation system crashes repeatedly upon introduction, contributing to revenue loss of $61 million.
1992	Budget Rent-A-Car, Hilton Hotels, Marriott International, and AMR [American Airlines]	Travel reservation system canceled after $165 million is spent.



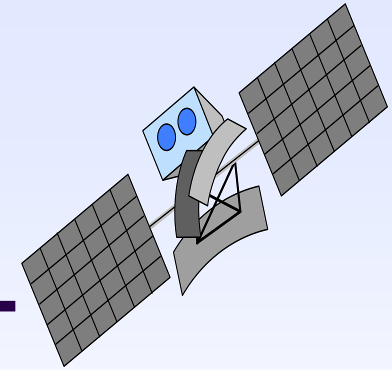
Difficulties in SR Analysis

- Software failures can be tracked to individual mistake
- Although in theory we can make it correct, in reality it is impossible
- Testing is costly
- Testing cannot prove the correctness
- There are many testing techniques with varying degree of efficiency
- Difficult to improve reliability





Software Reliability compared to hardware



- The process is essentially a design process
- Mainly human errors involved in creating the software
- No physical aging of the software
- Traditional redundancy is not useful
- Problems can be removed permanently
- Theoretically it can be made perfect
- Testing takes up to 50% of development resource
- …

SOFTWARE RELIABILITY MODELS

Past, Present, and Future

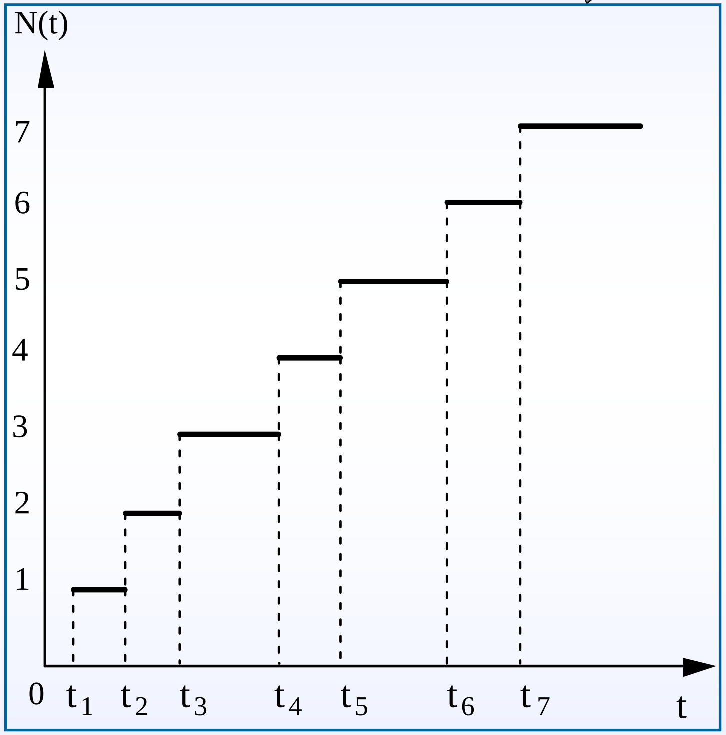
M. Xie

Dept of Industrial & Systems Engineering

National University of Singapore

Markov Process Models

- Jelinski-Moranda
- Earliest model
- Equal contribution of all faults
- Finite number of possible failures
- Debugging assumed to be perfect



The Jelinski-Moranda Model

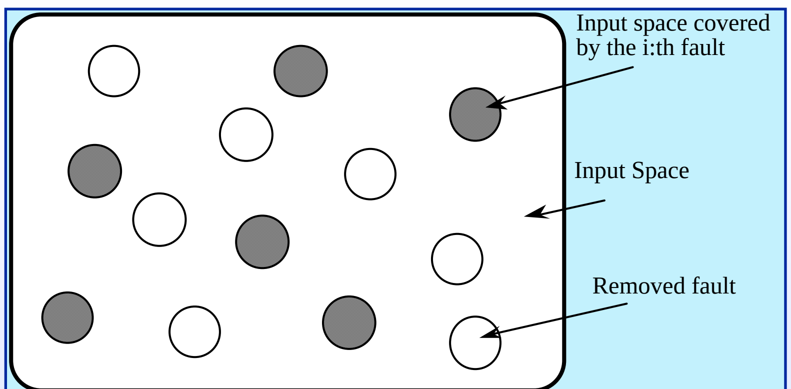
- the number of initial faults is an unknown but fixed constant;
- a detected fault is removed immediately and no new fault is introduced;
- times between failures are independent, exponentially distributed random quantities
- all remaining software faults contribute the same amount to the software failure intensity
- The time between the (i-1):st and the i:th failures is exponentially distributed with

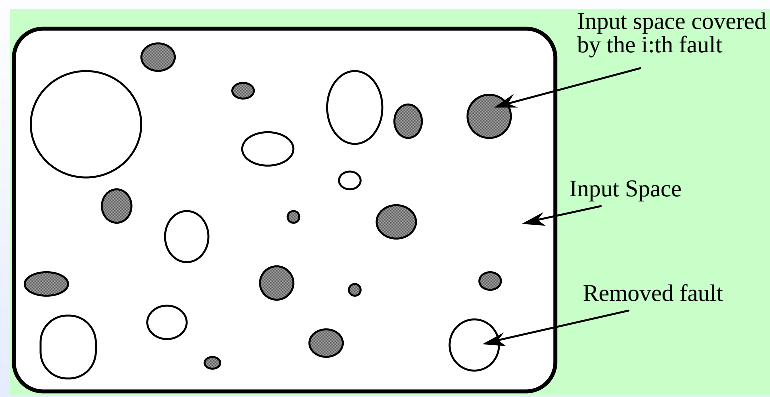
$$\lambda_i = \phi[N-(i-1)],\ i = 1,2,\ldots,N_0.$$

“Equal Size” Assumption

- Many models assumes that all faults contribute the same to the total failure probability
- This is equivalent to that all faults are of the same “size”

- Faults are not of equal size
- “Large” faults are likely to be detected at the beginning
- “Small” faults are difficult to detect





Input-Domain Based Models

- Started with the concept of correctness
- Select test cases and show the percentage of those that leads to a failure

- Closely related to operational profile
- Can be modified incorporating probability of input-domain data

NHPP Models

- An important class of SRGMs that has been widely studied by researchers and used by practitioners.

- The testing process is assumed to follow an NHPP whose mean value function is $m(t)$.

- The instantaneous failure intensity at time t can be calculated by $\lambda(t) = dm(t)/dt$

The Goel-Okumoto Model

- Probably the most well-known SRM
- Many similar models
- Derived assuming the same detection rate of remaining faults
- Simple model for finite number of faults

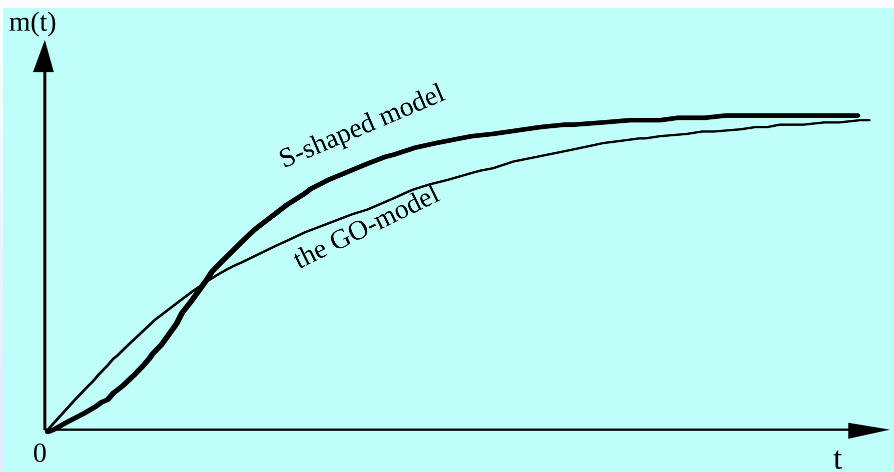
$$m(t) = a\left(1 - e^{-bt}\right), \quad a > 0, \; b > 0;$$

$$\lambda(t) = \frac{dm(t)}{dt} = abe^{-bt}.$$

S-shaped NHPP Model

- Failure intensity increases at the beginning
- Suitable for the modeling of a learning process
- Has shown to be good for a number of data sets

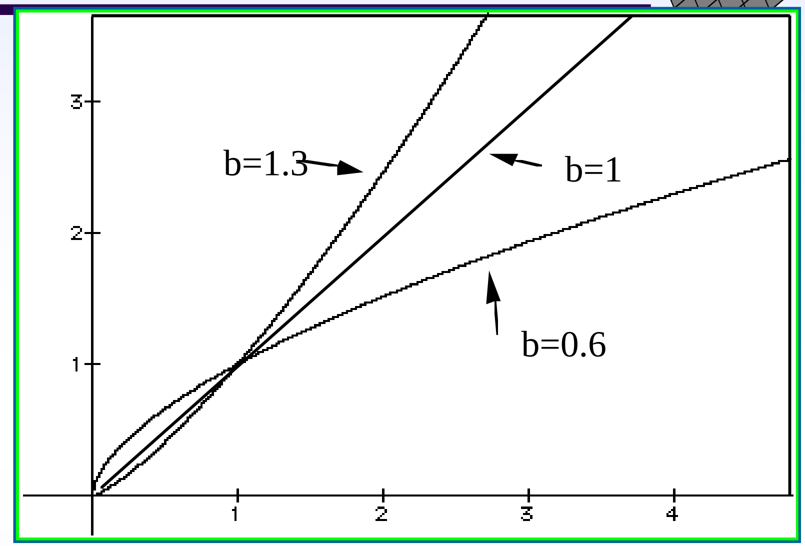
$$m(t) = a[1 - (1 + bt)e^{-bt}]; \quad b > 0.$$



The Duane Model

- Mean value function

$$m(t) = at^b$$

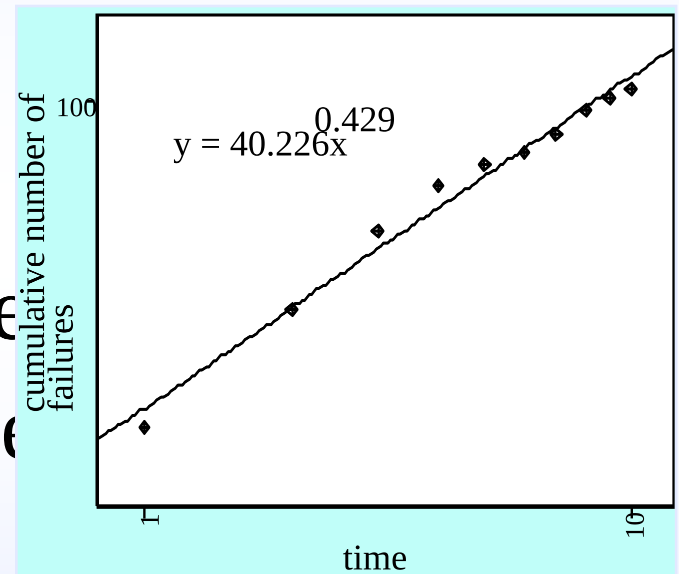


- Very flexible model
  - $b<1$ improving
  - $b<1$ deteriorating
- Duane plot and graphical interpretation available
- Simple and reasonably accurate
- Widely used for repairable systems

$$\lambda(t) = \frac{dm(t)}{dt} = abt^{b-1}$$

The Duane Plot

- A useful relationship:
- $\ln m(t) = \ln a + b \ln t$
- Plot cumulative number of failures vs *t* on a log-log scale
- Fit the plot with a straight line
- slope=*b* and intercept=ln*a*
- **The validity of the model can be checked BEFORE its use**

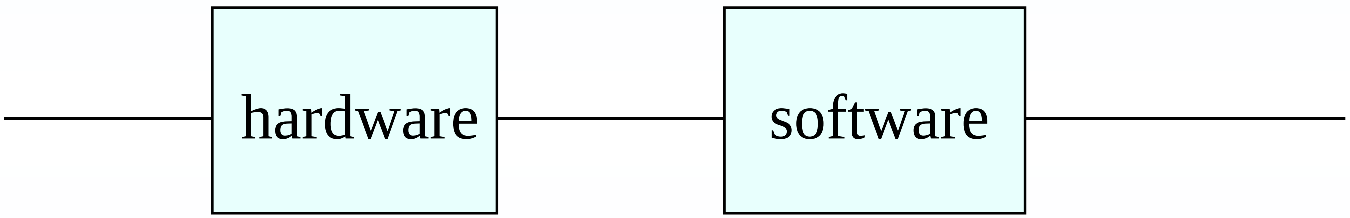


Advantages of Graphical Approach

- (a) Model verification is very simple
- (b) Parameter estimation can be carried out easily
- (c) Model can be validated BEFORE parameter estimation
- (d) Plotting can be done using simple spreadsheet software

Reliability of Combined System

$$Reliability_{system} = R_{hardware} \cdot R_{software}$$



- Assuming both are needed for the system to work
- Failure of one should not affect the other
- The failure causes should be able to be isolated
- Software may not be more reliable than hardware
- Important to consider serious failures

Definition of software reliability

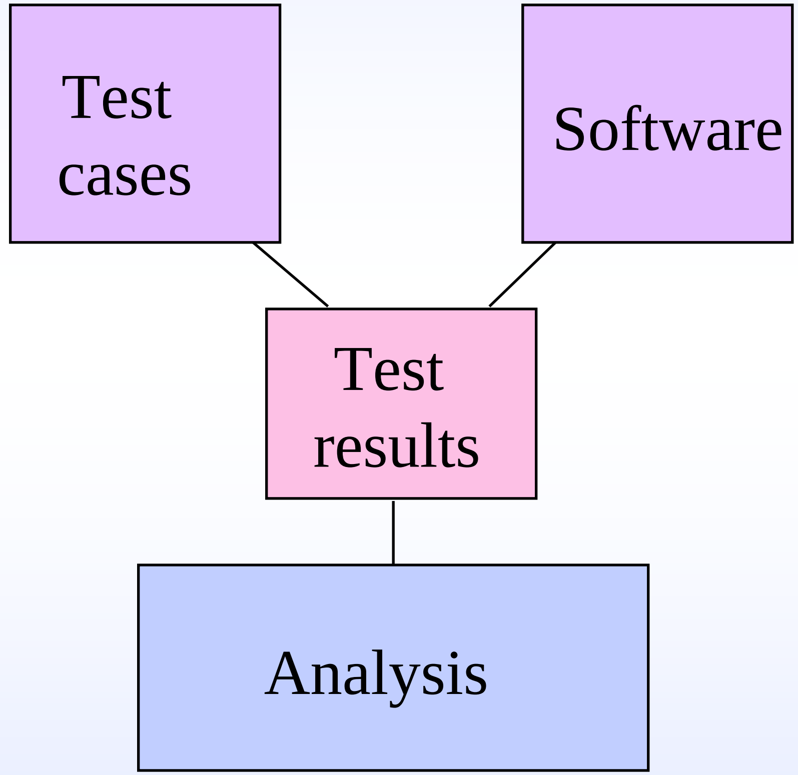
- Many different measures used (not appropriate)
- the number of faults
- defect density
- defect per module
- defect per KLOC
- defect per FP

Reliability vs # Faults

- The number of faults is not a good reliability measure
- Testing should focus on reliability improvement rather than removing more faults
- Reliability depends on the number of faults
- Software metrics can be used to estimate the number of faults
- Estimates of the number of faults are not accurate

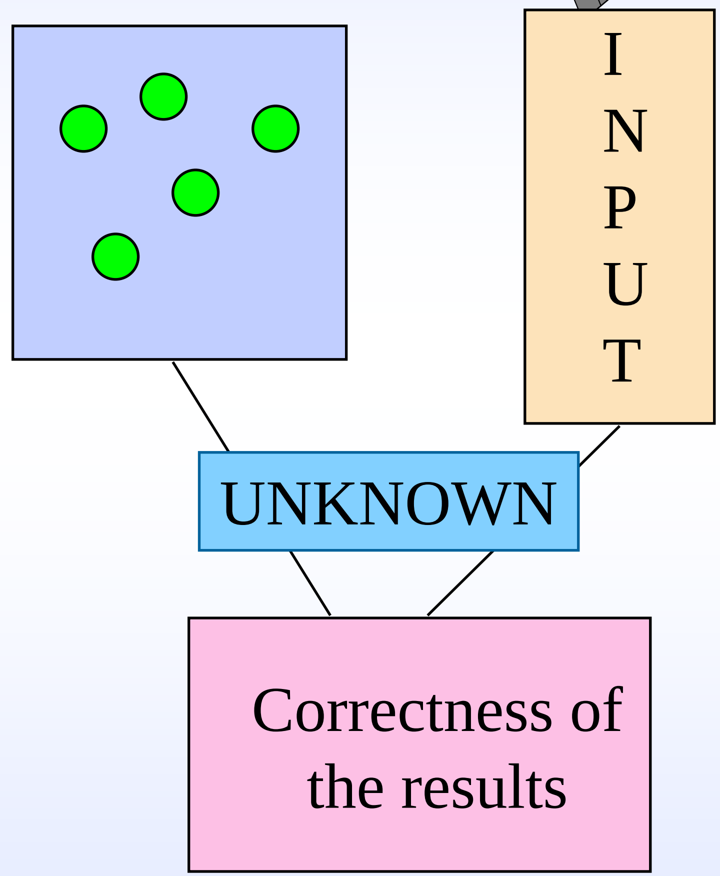
Random Testing

- Test cases are selected randomly
- Test cases should follow the operational profile - input states are selected in accordance of the probabilities of occurrence when used
- This will minimize the probability of failure experienced by the customers



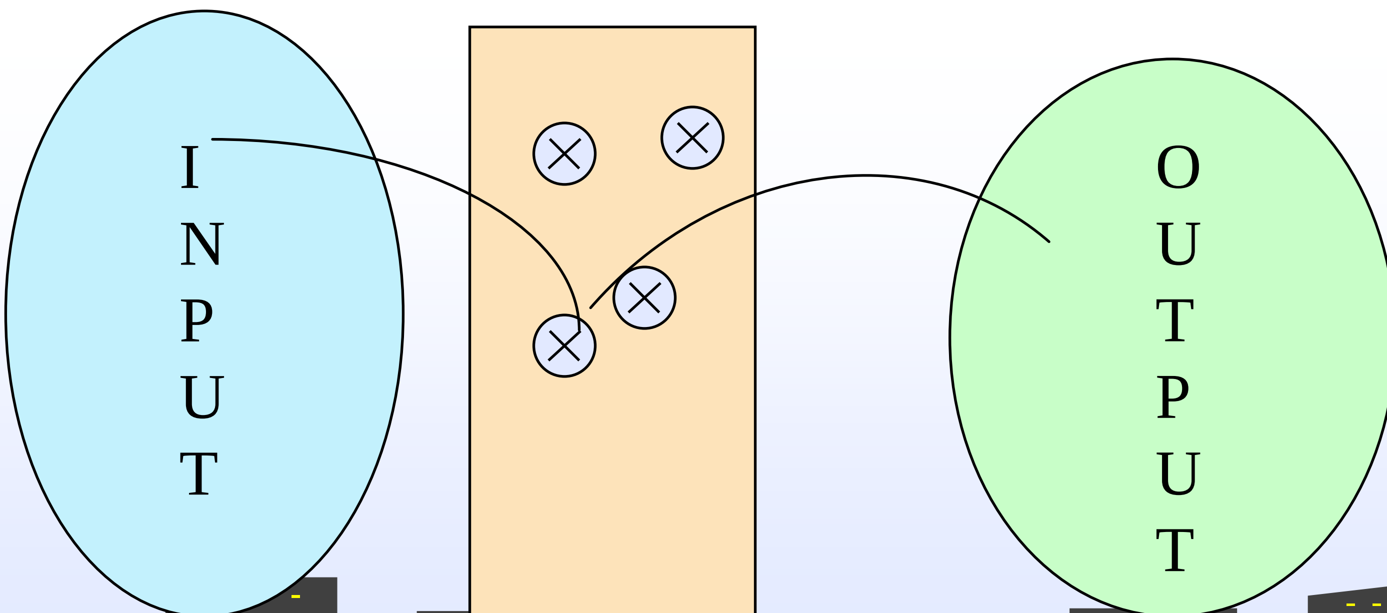
Randomness of Failures

- Number of failures per unit time is random
- Time to next failure is random
- This is because
  - the location of faults in the programme is unknown
  - the usage of programme is not predictable



“Theory” of Testing

- Input space, software, output space
- Some inputs lead to a failure because of a fault
- The fault can be identified and removed



Effect of Imperfect Debugging

- Most of the software testing processes belong to the imperfect debugging ones.
- The development of the software is extremely time-consuming and costly.
- It is important to know the effect of imperfect debugging on software cost.

Models using Software Metrics

- Relate the number of faults to various software metrics and a relationship can be derived using earlier projects
- Existing studies focus on the number of faults
- Useful for the planning
- Require information from earlier and similar projects

Need for and Availability of Data

- Data (collection) can be used
  - to help with quantitative analysis
  - to study the current system/project
  - to help identify weak spots in the process and system
  - to be used as a record
- Data are and should be available

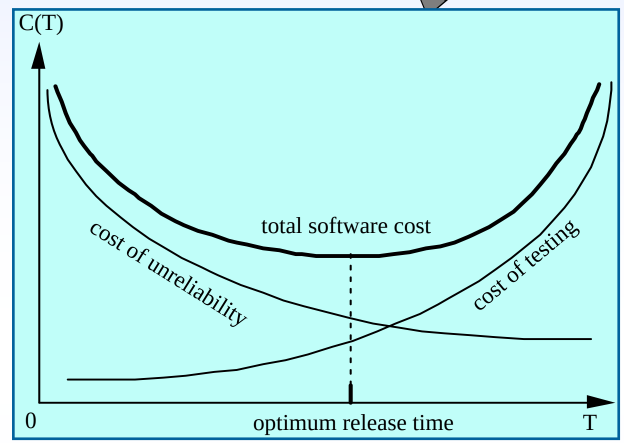
Uses of SR Models

- To assess the reliability of software
- To predict future failure behavior
- To study the effective testing technique
- To help allocating resources
- To provide information how to improve the process and product

Release Time Determination
- cost minimization

- Time to minimize total cost
  - – need a cost model

$$c(T) = c_1 m(T) + c_2\left[m(\infty) - m(T)\right] + c_3 T.$$



- c_1 = expected cost of removing a fault in testing
- c_2 = expected cost of removing a fault in field
- c_3 = expected cost per unit time of software testing including the cost of testing, the cost due to a delay in releasing the software, etc.

Summary on use of software reliability models

- Need to incorporate software metrics
- Need to consider testing strategies
- Reliability as an aspect of quality
- Understanding of randomness and statistical errors a necessity
- Suitable model selection approaches should be developed
- Models should be used in decision-making