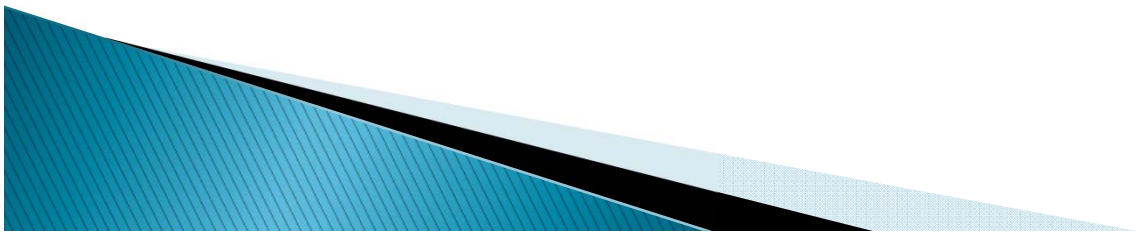
Software Reliability Modeling

Dr. Peter Keiller and Dr. Charles Kim

Presentation Overview

- Definition & History
- Reliability Modeling
- Faults and Failures
- Software Reliability Models
- Failure Data
- Problems
- Conclusion
- Questions

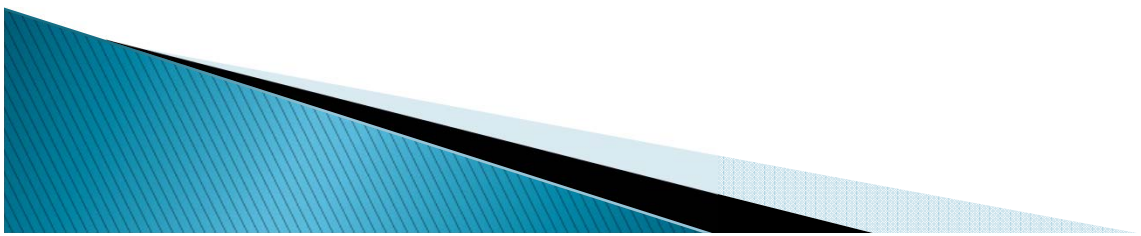


Reliability terminology

Term	Description
System failure	An event that occurs at some point in time when the system does not deliver a service as expected by its users
System error	An erroneous system state that can lead to system behaviour that is unexpected by system users.
System fault	A characteristic of a software system that can lead to a system error. For example, failure to initialise a variable could lead to that variable having the wrong value when it is used.
Human error or mistake	Human behaviour that results in the introduction of faults into a system.

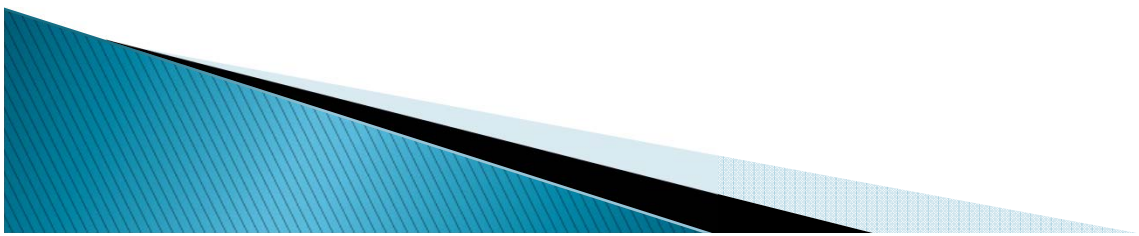
Faults and failures

- Failures are a usually a result of system errors that are derived from faults in the system
- However, faults do not necessarily result in system errors
  - The faulty system state may be transient and 'corrected' before an error arises
- Errors do not necessarily lead to system failures
  - The error can be corrected by built-in error detection and recovery
  - The failure can be protected against by built-in protection facilities. These may, for example, protect system resources from system errors



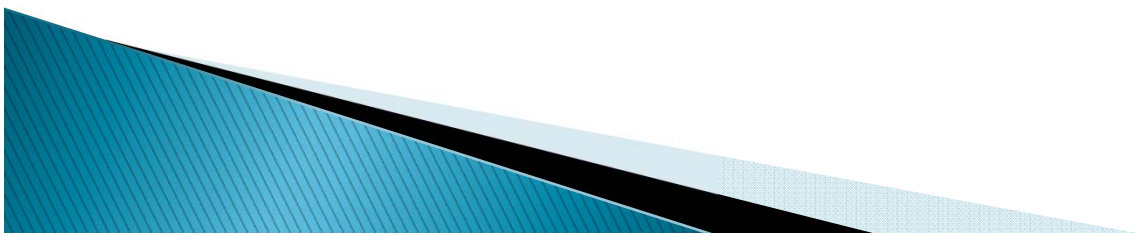
Reliability achievement

- Fault avoidance
  - Development technique are used that either minimise the possibility of mistakes or trap mistakes before they result in the introduction of system faults
- Fault detection and removal
  - Verification and validation techniques that increase the probability of detecting and correcting errors before the system goes into service are used
- Fault tolerance
  - Run-time techniques are used to ensure that system faults do not result in system errors and/or that system errors do not lead to system failures

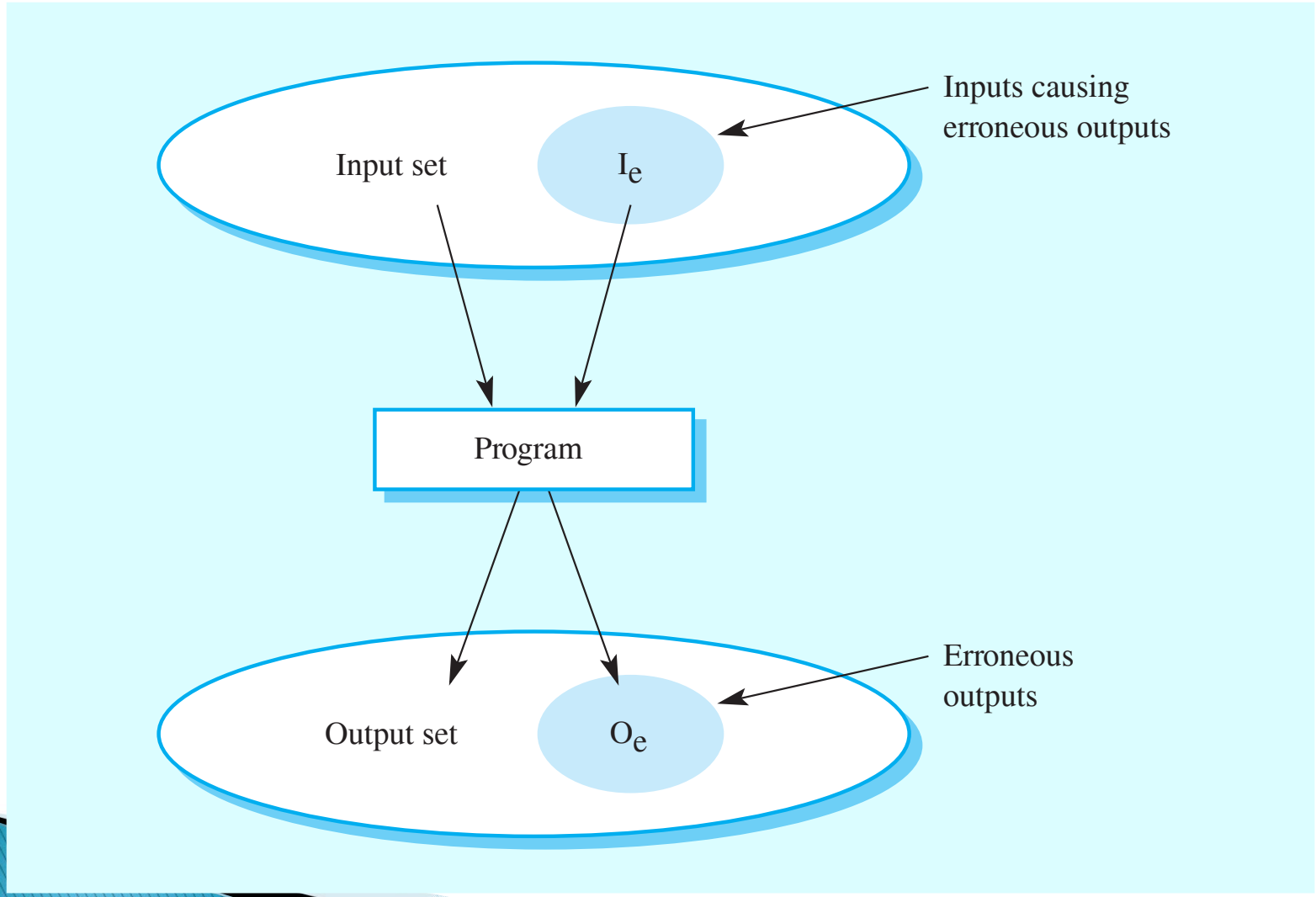


Reliability modeling

- You can model a system as an input-output mapping where some inputs will result in erroneous outputs
- The reliability of the system is the probability that a particular input will lie in the set of inputs that cause erroneous outputs
- Different people will use the system in different ways so this probability is not a static system attribute but depends on the system's environment

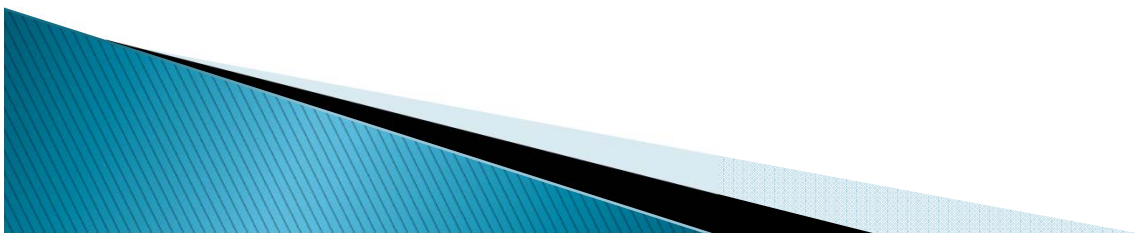


Input/output mapping



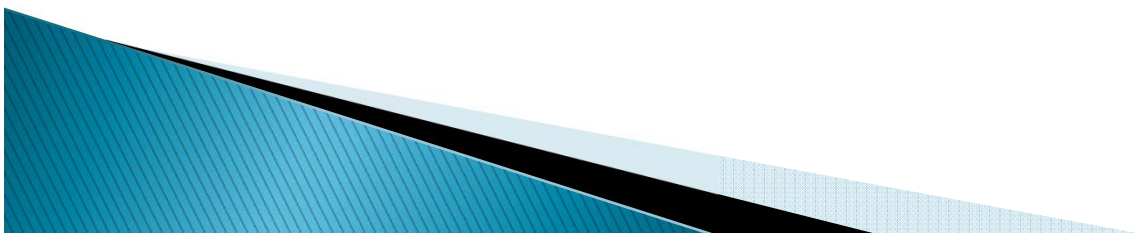
Reliability improvement

- Removing X% of the faults in a system will not necessarily improve the reliability by X%. A study at IBM showed that removing 60% of product defects resulted in a 3% improvement in reliability
- Program defects may be in rarely executed sections of the code so may never be encountered by users. Removing these does not affect the perceived reliability
- A program with known faults may therefore still be seen as reliable by its users



Software Reliability

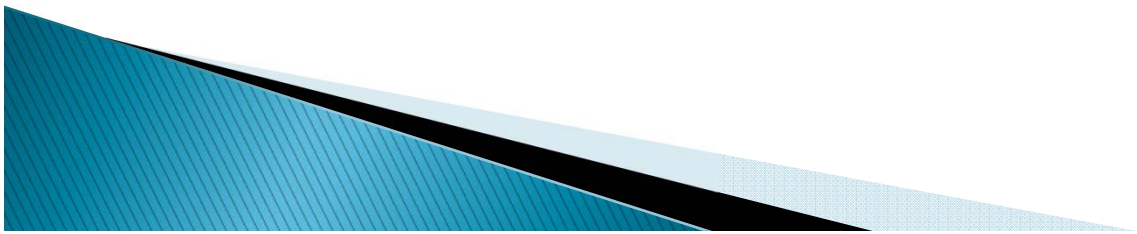
- Software reliability (SR):
  - "The probability that software will not cause the failure of a system for a specified time under specified conditions."



Software Failures 2010

http://www.newtechblog.com/internet/news/ten-enormous-software-failures-for-2010/

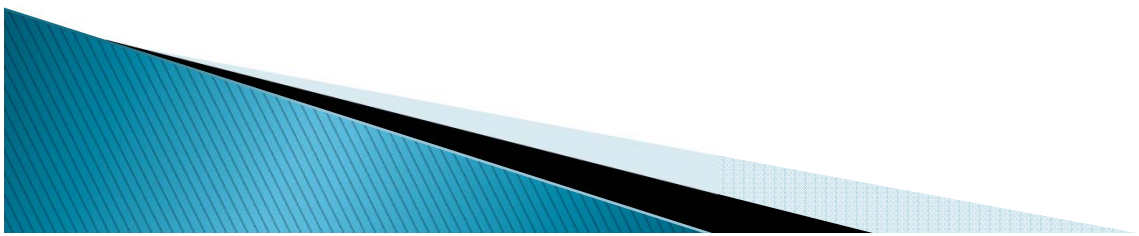
- Toys R Us Double Charges Black Friday Shoppers
- New Hampshire Man Charged 23 Quadrillion Dollars for a
- pack of smokes
- McAfee Software Glitch Proves Costly
- Apple to issue software fix for lingering iPad Wi-Fi problems
- Weapon Software Glitch Hits Close to Home
- Toyota Slapped With $32.4 Million Fine for Mishandling Recalls
- Verizon Wireless to refund $50 million over software glitch
- Faulty software costs NY $114M
- Chase Web crash locks out 16.5 million online customers
- Airborne Laser Test Failure Blamed on Software Error



Software Failure

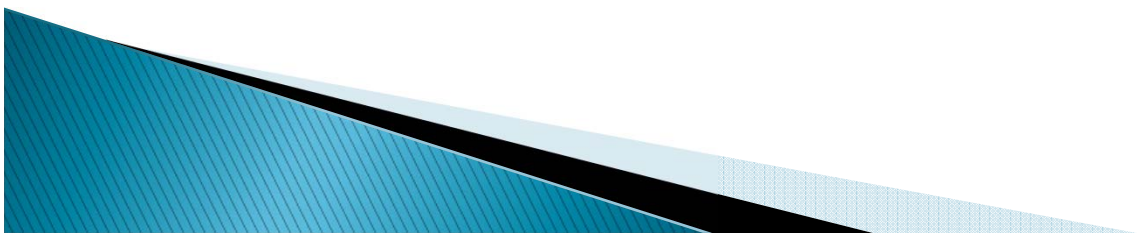
Why Does Software Fail?

- Wrong requirement: not what the customer wants
- Missing requirement
- Requirement impossible to implement
- Faulty design
- Faulty code
- Improperly implemented design



Software Failure (cont.)

- In the early 90's hardware was dominant driver of system failure
- By 1997software became the dominant driver of system's reliability (Everett, Keene and Nikora 1998)
- Software driven outages were exceeding hardware driven outages by a factor of ten (Keene 1997)



Software Reliability Modeling

- Early approaches based on modifications of models derived in Hardware reliability field.

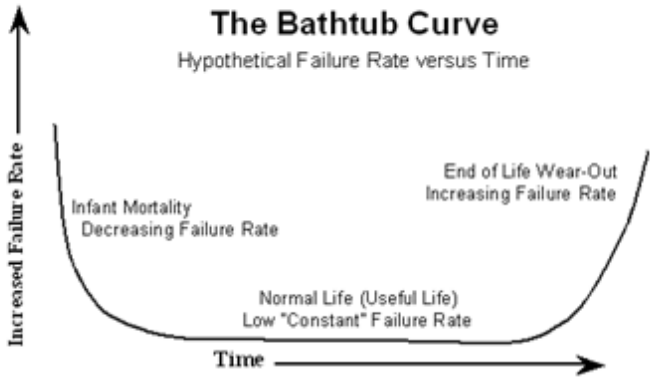
Software Failure	Hardware Failure
Failure does not occur if the software is not used.	Material deterioration or wear can cause failure even when the system is not in use.
Failures are inherently determined during the earlier phase of the development and can be caused by incorrect logic, incorrect statements, or incorrect input data.	Failures are caused by material deterioration, random failures, design errors, misuse, and environmental factors
Software failures are rarely preceded by warnings.	Hardware failures are usually preceded by warnings.
Software cannot usually be tested exhaustively.	Hardware can usually be tested exhaustively.
Reliability is mostly affected by users' operational profiles.	Hardware reliability is mostly affected by physical environmental factors.
Once a software fault is removed it will never cause the same failure again.	Hardware can fail from the same fault but a MTBF is usually known and corrective maintenance can be performed
Copies of software programs are identical and therefore cannot improve reliability through redundancy.	No two pieces of hardware are exactly the same and hardware reliability can be improved with redundancy of identical units.

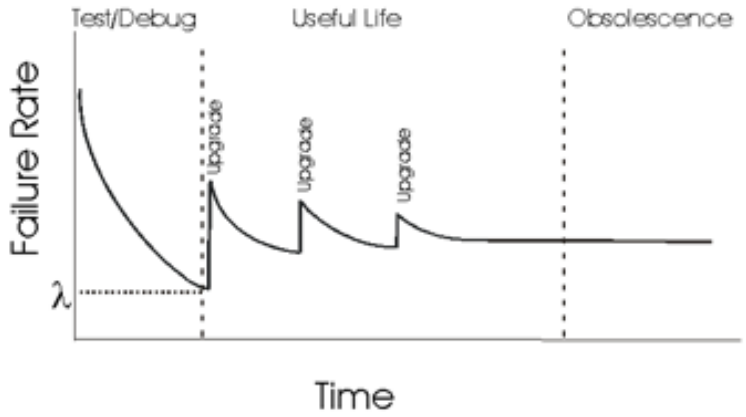
Table comparing the differences between hardware and software failures (Yamada, Ohba and Osaki 1983) (Xie 1991).

Hazard Curves

Comparison of the hazard curves for hardware and software respectively

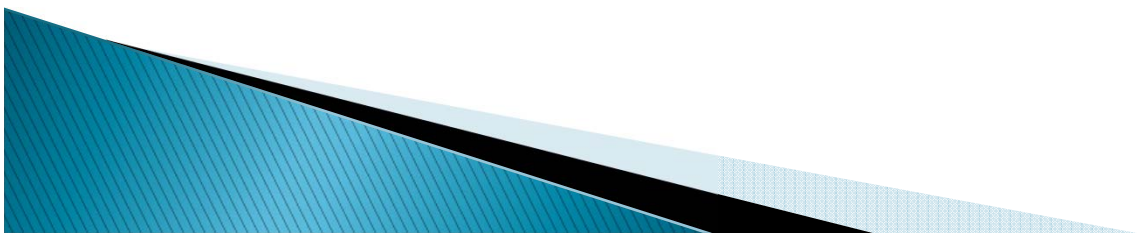
http://www.ece.cmu.edu/~koopman/des_s99/sw_reliability/





History

- The first published approach dedicated to SRM was by G. R. Hudson (Hudson 1967)
- Most authors credit (Jelinski and Moranda 1972) as the major step.
- Jelinski and Moranda suggested stochastic process.
- Since the early 1970's over 80+ models have been proposed in literature



Types of Software Reliability Models

- **Times Between Failure Models**

  Times between failures follow a distribution whose parameters depend on the number of faults remaining in the program during the interval.

  (K/L; L/V; J/M; …)

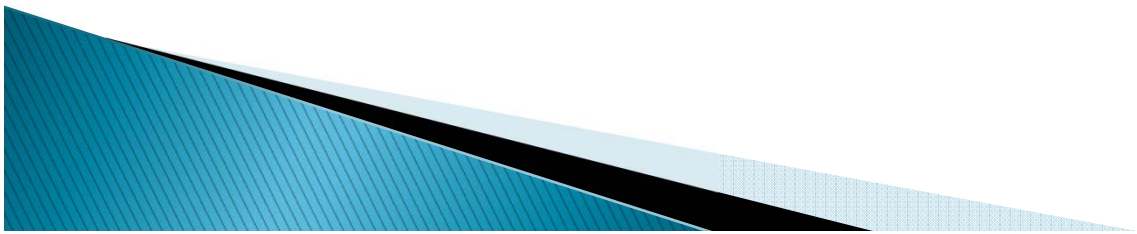
- **Failure Count Models**

  Number of faults detected in a given testing interval follow a Poisson Distribution

  (M/O; Shooman; G/O; …)

Collection of data

- Time of failure
- Time interval between failures
- Cumulative failures experienced
- Failures experienced in a time interval

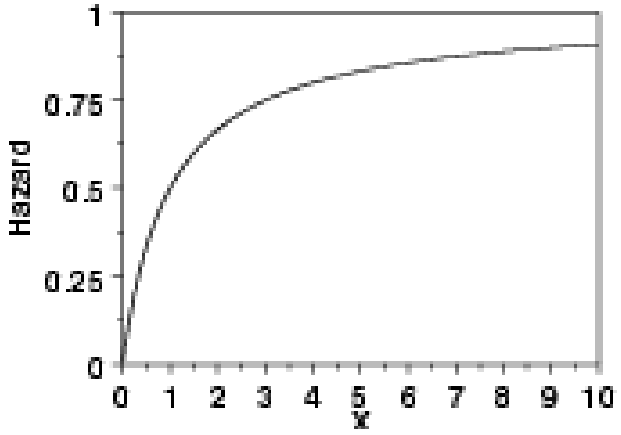


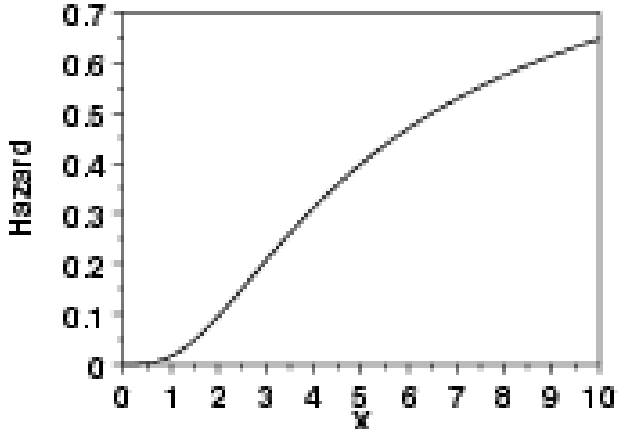
Execution times between successful failures
System SYS40

320.	14390.	9000.	2880.	5700.
21800.	26800.	113540.	112137.	660.
2700.	28793.	2173.	7263.	10865.
4230.	8460.	14805.	11844.	5361.
6553.	6499.	3124.	51323.	17010.
1890.	5400.	62312.	24826.	26335.
363.	13989.	15058.	32377.	41632.
4160.	82040.	13189.	3426.	5833.
640.	640.	2880.	110.	22080.
60654.	52163.	12546.	784.	10193.
7841.	31365.	24313.	298890.	1280.
22099.	19150.	2611.	39170.	55794.
42632.	267600.	87074.	149606.	14400.
34560.	39600.	334395.	296015.	177399.
214622.	156400.	166800.	10800.	267000.
2098833.	614080.	7680.	2629667.	2948700.
187200.	18000.	178200.	487800.	639200.
334560.	1468800.	86720.	199200.	215200.
86400.	88640.	1814400.	4160.	3200.
199200.	356160.	518400.	345600.	31360.
265600.				

Model Shape

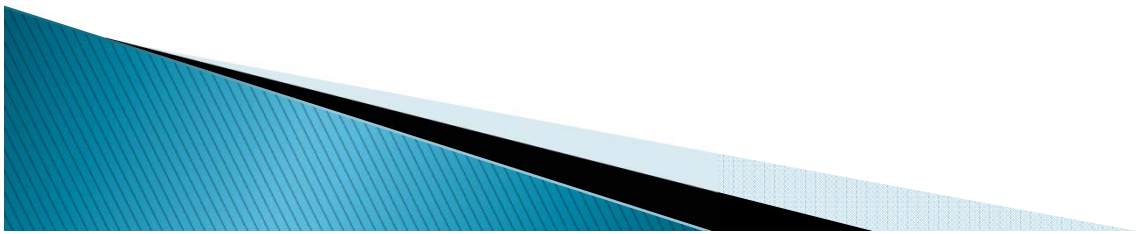
Most models follow the basic Concave or S-shaped curve as seen in Figure below





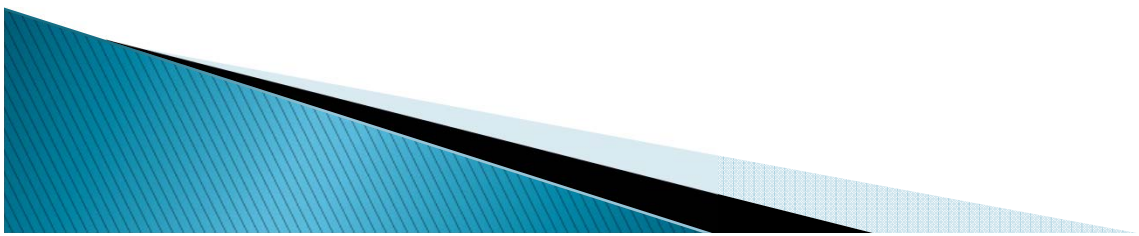
Combined Methods

- Other work in the field have suggested a unification of different NHPP models (Huang, Kuo, et al. 2000) (Gokhale, Philip, et al. 1996) (Pfefferman and Cernuschi-Frias December 2002).
- Super Model use one of multiple models over different phases. (Keiller and Mazzuchi 2005)



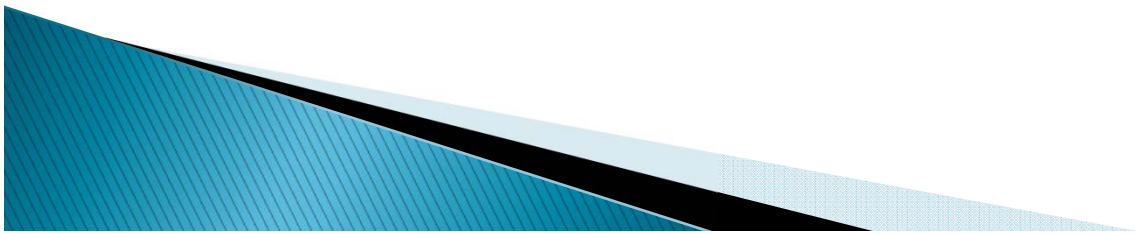
Steps to Software Reliability Modeling

1. Defining and collecting necessary failure metrics.
2. An analysis of failure data to make sure that its reliability is actually growing and that the data meets the assumptions of the models to be assessed.
3. Identification of suitable models and parameterization of the mean value functions.
4. Validating the result of the models using a goodness of fit test.
5. Computation of target metrics.
6. Using computed metrics for tradeoff analysis.



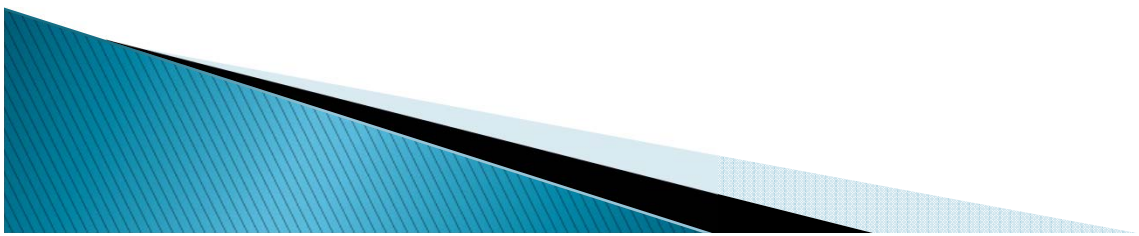
Validation

- No standardized metrics or methods that address how well a model should fit data.
  - Traditional Goodness of Fit methodologies (Kolmogrov Smirnov Test)
  - Sum of Square Errors (SSE) (Zhang 2000)
  - Loglikelihood
  - Akaike's information criterion (AIC) (Akaile 1974)
  - Bayesian information criteria (BIC) (Shibata, Rinsaka and Dohi 2006)



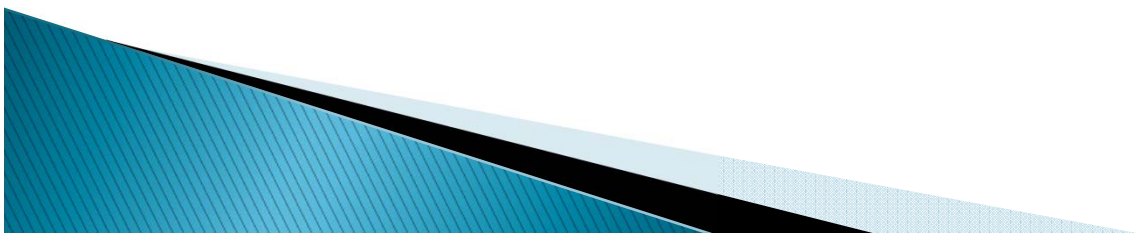
Validation

- Littlewood's metric "U-Plot"
- Prequential Likelihood (Abdalla-Ghally, Chan and Littlewood 1986).
- U-plot and Prequential Likelihood in combination (Brocklehurst and Littlewood 1992)
- Y-plot (Keiller and Miller 1991).



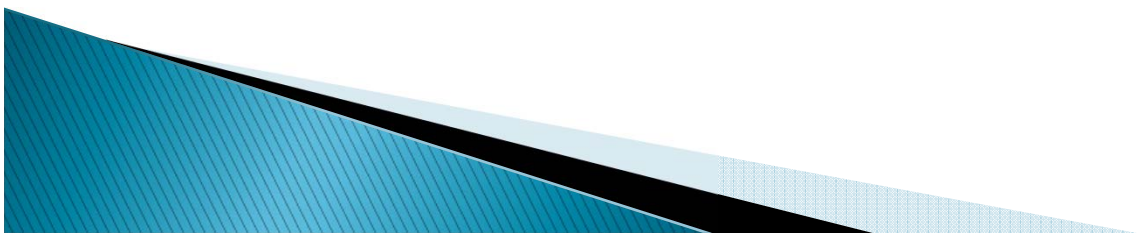
Problems

- "Early software reliability prediction models are often too insubstantial, seldom executable, insufficiently formal to be analyzable and typically not linked to the target system." (M. R. Lyu May 2007).
- Data quality.
- Many prediction models tend to model only part of the underlying problem. (Fenton and Martin 1999)
- The field of SR is still maturing there is no common body of knowledge. (Benlarbi and Stortz 2007).



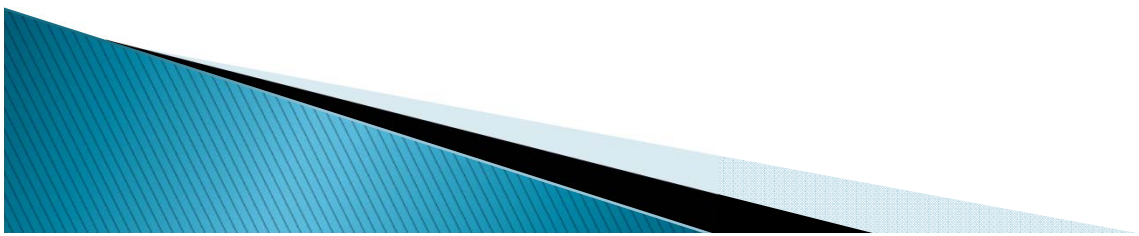
Problems

- No model can truly, fully represent the totality of what is being modeled.
- A lack of acceptance of SRM has hurt the SR modeling community because failure data collection is still seen as an unnecessary expense in some realms.
- Corporate entities that are successfully developing reliable software see their techniques, approaches and data as trade secrets.



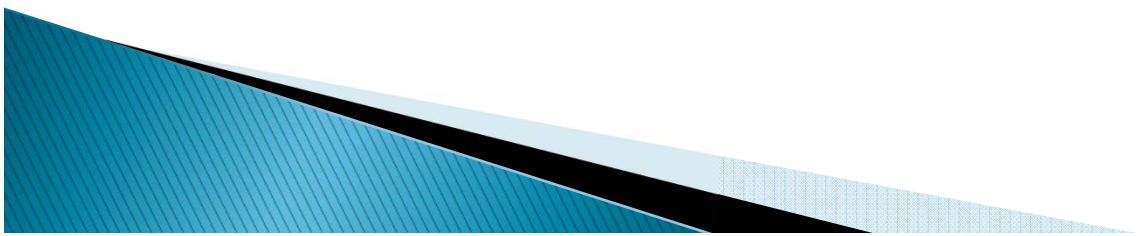
Problems

- Characteristics of the software topology, development process and the software itself do not accompany the data.
- The traditional assumptions of Software Reliability models may not hold true because corrections or changes to some functions may not be possible.



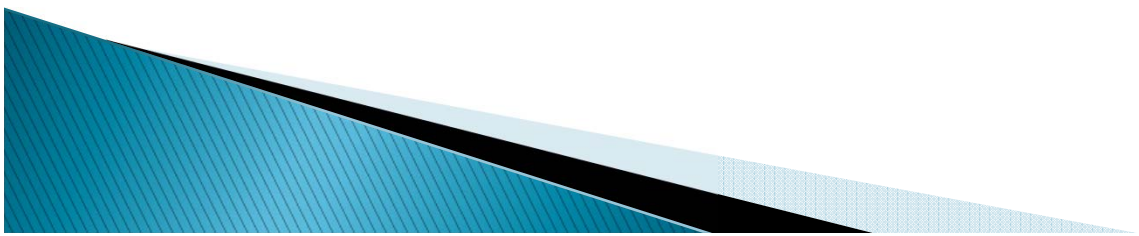
Problems

- It is still difficult to assign rank or measure to the quality of a company's software development process.
- It is still very hard to understand and model the relationships between process and software quality.



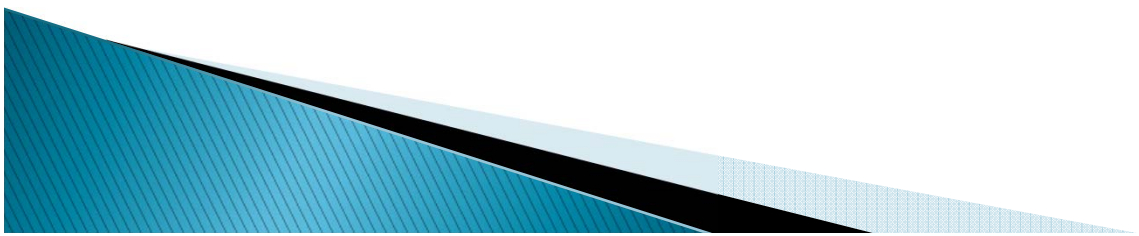
Future classifications

- “Future of software reliability modeling is compound models and tools that can predict and assess the reliability of software from start to finish incorporating the ‘best of breed’ in the predictive as well as assertive models”. (M. R. Lyu May 2007)



Conclusion

- There is no “Silver Bullet” for SRM.
- Many authors have shown that the use of modeling can enhance the decision making process.
- Need for a greater simplification of the modeling process by means of automated tools so that models are more readily available to the layman



Questions