Software Reliability

18-849b Dependable Embedded Systems

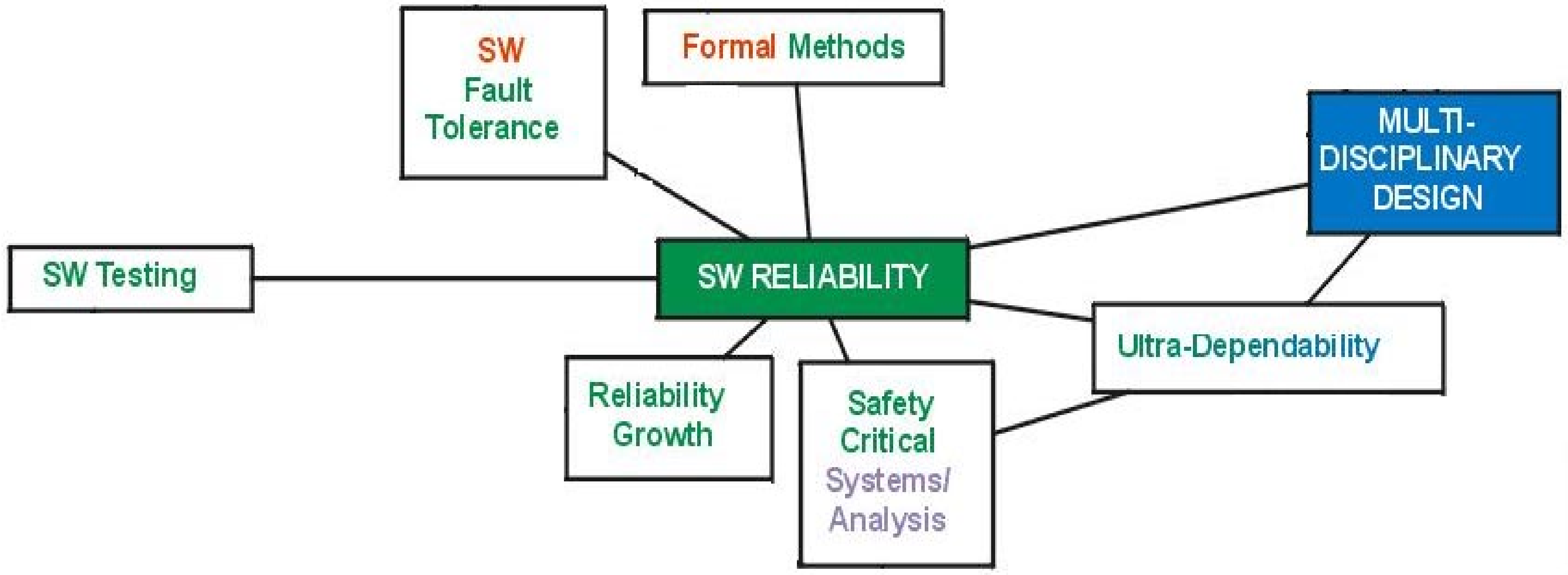
Jiantao Pan

Feb 2, 1999

Required Reading:	Handbook of Software Reliability Engineering, Chapter 1
Best Tutorial:	Handbook of Software Reliability Engineering, Michael R. Lyu
Authoritative Books:	Handbook of Software Reliability Engineering, Michael R. Lyu
	Introduction to Software Reliability: A state of the Art Review

Carnegie Mellon

You Are Here


SW Fault Tolerance
Formal Methods
MULTI-DISCIPLINARY DESIGN
SW Testing
SW RELIABILITY
Ultra-Dependability
Reliability Growth
Safety Critical Systems/ Analysis

Issues

- **More and more computers, and more …**
  - Increased control by software
    - Everyday life
    - Critical applications
- **Can we trust software?**
  - Software never breaks!?
    - Therac 25
    - Ariane 5
    - NASA Voyager Uranus encounter jeopardy
    - Telephone network outages

Software & Hardware Differences

- **Major differences for software:**
  - *Failure cause*: Software defects are mainly design defects
  - *Wearout*: Software does not rust
  - *Repairable system concept*: Periodic restarts can help fix problems
  - *Time dependency and life cycle*: SR not related to operational time
  - *Environmental factors*: External environment does not affect SR
  - *Reliability prediction*: SR human factors, not physical factors
  - *Redundancy*: Can not improve SR using identical components
  - *Interfaces*: Purely conceptual; not visual
  - *Failure rate motivators*: Usually not predictable
  - *Standard components*: Usually no standard parts. Reuse limited
- **Additional differences:**
  - SW Cannot be touched
  - SW has no size, material, etc
  - No weight/energy($E=mc^2$)

Key Concepts

- **Software Reliability (SR)**
  - the probability of failure-free software operation for a specified period of time in a specified environment. [ANSI]
  - It is not a function of operational time!
- **SR is an attribute of software quality**
  - Together with: functionality, usability, performance, serviceability, capability, installability, maintainability, and documentation.
  - Robustness is an aspect of SR
- **Why SR is so hard to achieve:**
  - Complexity
    - Software is not intrinsically buggy than hardware, but people tend to push complexity into software

SR: Bathtub Curves

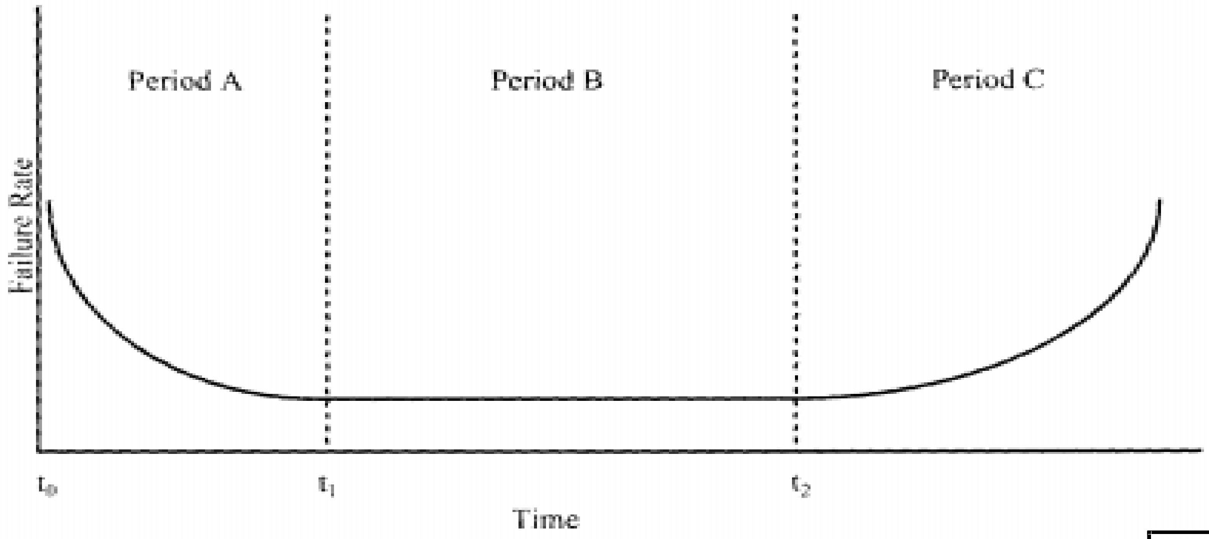


Figure 1-2. Bathtub Curve for Hardware Reliability

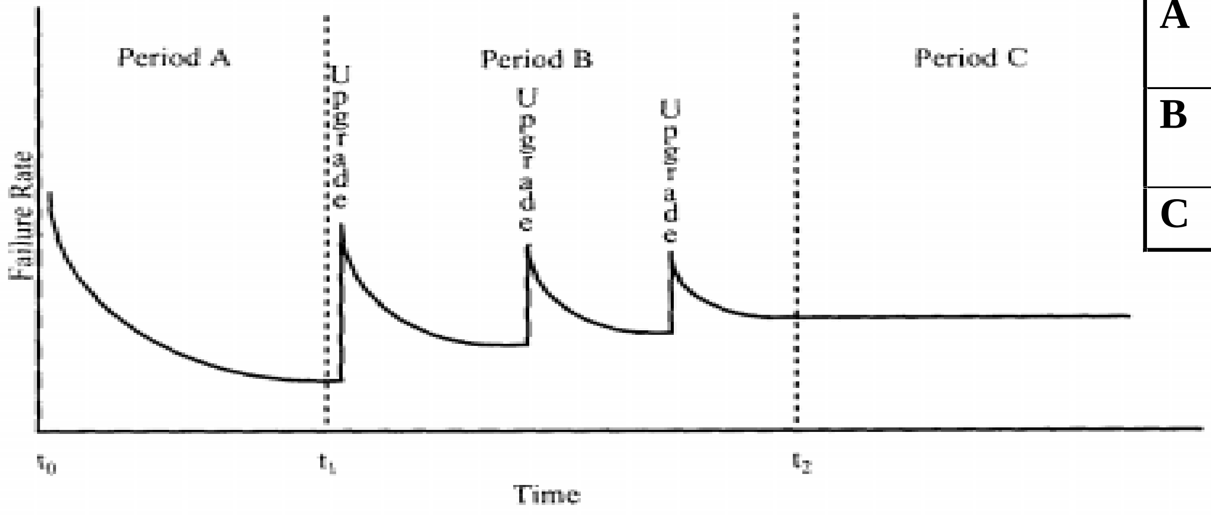
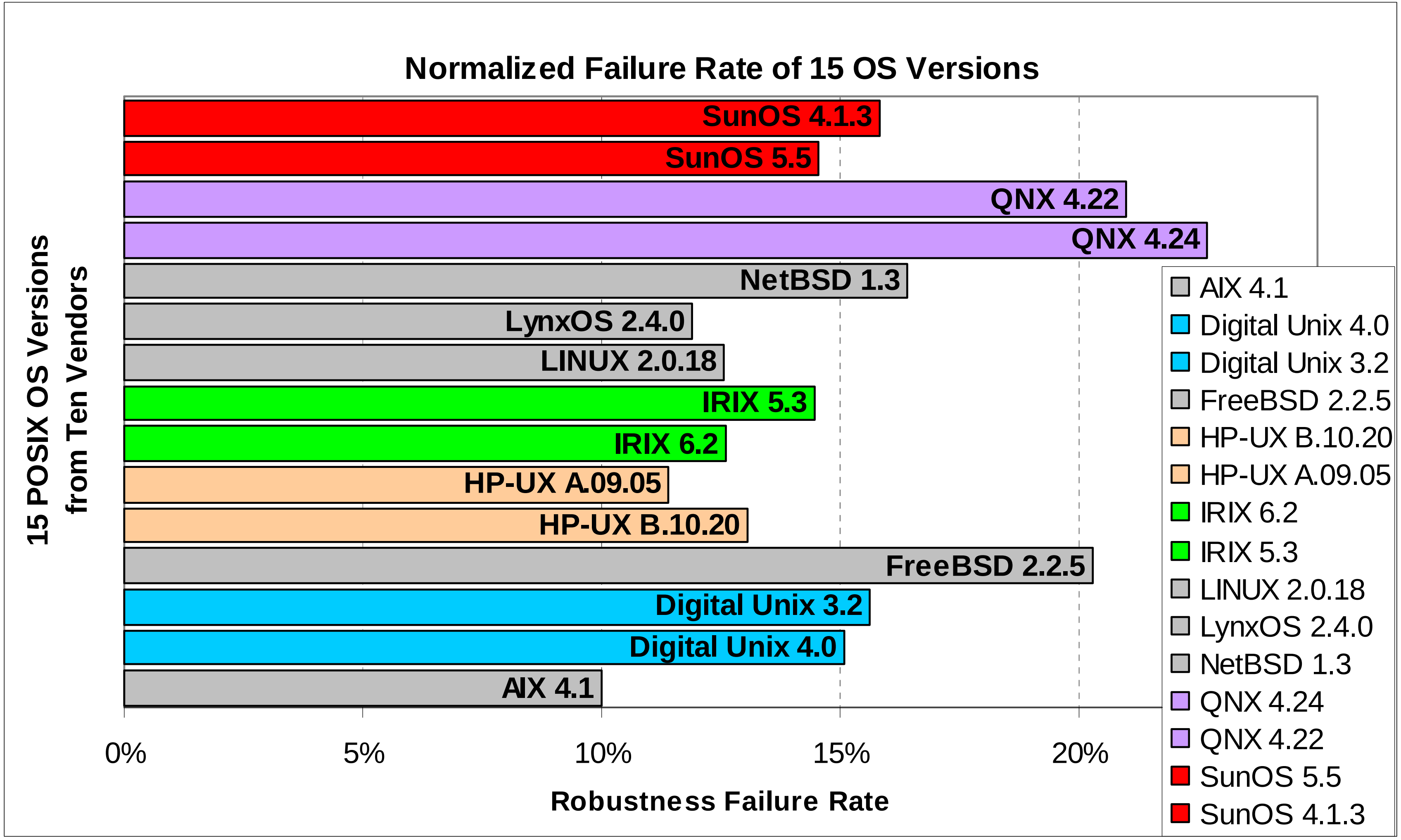


Figure 1-3. Revised Bathtub Curve for Software Reliability

Period	SW	HW
A	Test/Debug	Infant mortality
B	Useful life	Useful life
C	Obsolescence	Wearout

Legend

True?


Normalized Failure Rate of 15 OS Versions
SunOS 4.1.3
SunOS 5.5
QNX 4.22
QNX 4.24
NetBSD 1.3
LynxOS 2.4.0
LINUX 2.0.18
IRIX 5.3
IRIX 6.2
HP-UX A.09.05
HP-UX B.10.20
FreeBSD 2.2.5
Digital Unix 3.2
Digital Unix 4.0
AIX 4.1
15 POSIX OS Versions from Ten Vendors
0%
5%
10%
15%
20%
Robustness Failure Rate
AIX 4.1
Digital Unix 4.0
Digital Unix 3.2
FreeBSD 2.2.5
HP-UX B.10.20
HP-UX A.09.05
IRIX 6.2
IRIX 5.3
LINUX 2.0.18
LynxOS 2.4.0
NetBSD 1.3
QNX 4.24
QNX 4.22
SunOS 5.5
SunOS 4.1.3

Software Reliability: Pieces of the Puzzle

- **SR: Models**
  - Prediction
  - Estimation
- **SR: Measurement**
  - Metrics
- **SR: Improvement**
  - Time
  - Budget
- **Other techniques (and many more emerging)**
  - Software Reliability Simulation
    - Trace-driven, self-driven
    - Observing the result
    - Sensitivity analysis
  - The Operational Profile

SR: Models

- **Observed failure data + statistical inference**
- **Prediction Models**
  - In-House Historical Data Collection Model
  - Musa’s Execution Time Model
  - Putnam’s Model
  - Rome Laboratory prediction Model: RL-TR-92-15
  - Rome Laboratory prediction Model: RL-TR-92-52
- **Estimation Models**
  - Classical Fault Count/Fault Rate Estimation Models
    - Exponential Distribution Models
    - Weibull Distribution Model
  - Bayesian Fault Rate Estimation Models
    - Thompson and Chelson’s Model
- **Neural Networks for SRE** *New!*

SR: Models Summary

- **There are so many models**
  - You can probably find the model that can *produce* the result you want!
- **Matured to the degree that**
  - can be applied in practical situations
  - give meaningful results
- **There is no one model that is best in all situations**
  - Select the model that is most appropriate for he data set and the environment in which the data were collected
- **Results can not be blindly applied**

SR: Measurement

- **“Measurement is far from commonplace in the software engineering world ... ”**
- **SR itself is hard to measure, so we measure other aspects**
  - Product metrics
    - Lines Of Code(LOC, KLOG, SLOC, KSLOC) with relation to defects
    - Function Point Metric
    - Complexity-Oriented Metrics
    - Test Coverage Metrics
  - Project Management Metrics
  - Process metrics
  - Fault and Failure Metrics

SR: Improvement

- **Before deployment**
  - Software testing
  - Verification, validation
  - Software system analysis tools
    - Fault Tree, ODC, Formal methods, etc
    - Trend analysis
- **After deployment**
  - Field data analysis
  - Dealing with faults:
    - Fault prevention
    - Fault removal
    - Fault tolerance
    - Fault/failure forecasting

Relationship To Other Topic Areas

- **It relates to any area that uses software …**
- **Traditional/Hardware Reliability**
  - SR is an analogy of Hardware Reliability(HR)
    - SR focuses on design perfection
    - HR focuses manufacturing perfection
- **Software Fault Tolerance**
  - Achieve high reliability using software methods
- **Software Testing**
  - Can be used to improve, measure software reliability
- **Social & Legal Concerns**
  - Bugs will always exist; I am not liable.
  - It is a specification problem.
  - No known bugs!

Conclusions & Future Work

- **Conclusions**
  - Models are affluent
    - Too many models (but which one suits your case?)
  - Measurement is naïve
    - "Just how good is the software, quantitatively?"
  - Improvement is hard
    - Need to balance time and cost issues.
- **Future work:**
  - Metrics?
    - Study common failure modes
    - Find better quantitative metrics to represent software reliability and quality
  - Complexity?
    - Find better engineering method to manage and conquer software complexity
  - Standardization?
    - Standard software components as building blocks
  - Recreate a new area called "Software Quality Assurance"

Software Reliability Questions with answers on topics such as reliability measures and models, model types, applications and usages etc.Â Explain software reliability with respect to reliability measures and models. What is software reliability? It is the probability of failure-free software operation for a specified period of time in a specified environment. It is also an important factor that affects system reliability. Software reliability testing is a field of software testing that relates to testing a software's ability to function, given environmental conditions, for a particular amount of time. Software reliability testing helps discover many problems in the software design and functionality. Software reliability is the probability that software will work properly in a specified environment and for a given amount of time. Using the following formula, the probability of failure is calculated by testing a sample 1 Software Reliability Basic Concepts 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.