Towards More Reliable Software: Introduction

Duke University

ECE 590, ECE 590K, Spring 2024

Instructor: Prof. Kishor S. Trivedi
Department of Electrical and Computer Engineering
Duke University, Durham, NC 27708-0291
E-mail: ktrivedi@duke.edu URL: www.ee.duke.edu/~ktrivedi
Office: 401 Wilkinson

DUKE

Goals of the Course

- To discuss techniques and tools that can help make software more reliable
- To discuss techniques and tools that can help quantify the potential reliability/availability improvement
- The course was taught first time in Spring 2020 and then in Spring 2021, Spring 2022 & Spring 2023
- This course was also taught to practicing engineers from Hill Air Force Base and Northrup Grumman during July-August 2021

ECE 590 Spring 2024

- https://canvas.duke.edu/courses/22499
- Lectures are in hybrid mode
  - January 11, 16, 18, 23, 25, 30
  - February 1, 6, 8, 13, 15, 20, 22, 27, 29
  - March 5, 7, 19,21, 23, 26, 28
  - April 2, 4, 9, 11, 16
- Lecture times: Tue & Thu 8.30-9.45 am,

ECE 590 Spring 2024

- Four TAs:
- TA: Fangyun Qin (fyqin@cnu.edu.cn)
  - office hours:
- TA: Kun Qiu (qiukun@hfut.edu.cn)
  - office hours:
- TA: Xinyi Li (xinyi.li@duke.edu)
  - office hours
- TA: Fan Yang (fan.yang@duke.edu)
  - office hours:

Outline of Lectures in ECE 590

- Introduction: Motivations and Basic Definitions – Jan. 11, 16
- Fault Avoidance (or Fault Prevention) – Jan 18, 23
- Fault Removal: Software Testing and Software Debugging – Jan. 25
- Fault Removal: Software Reliability Growth Models – Jan. 30, Feb. 1
- Architecture-based Software Reliability – Feb. 6
- Traditional Software Fault Tolerance – Feb. 8, 13
- Rethinking Software Fault Tolerance – Feb. 15
- Software Bug Classification – Feb. 20, 22
- Software Aging and Rejuvenation – Feb. 27, 29, Mar. 5, 7, 19, 21, 26
- Patterns of Software Fault Tolerance – Mar. 28
- Software Fault Tolerance via Environmental Diversity – Apr 2, Apr. 4
- Software Security – Apr. 9, 11
- Model Checking – Apr. 16

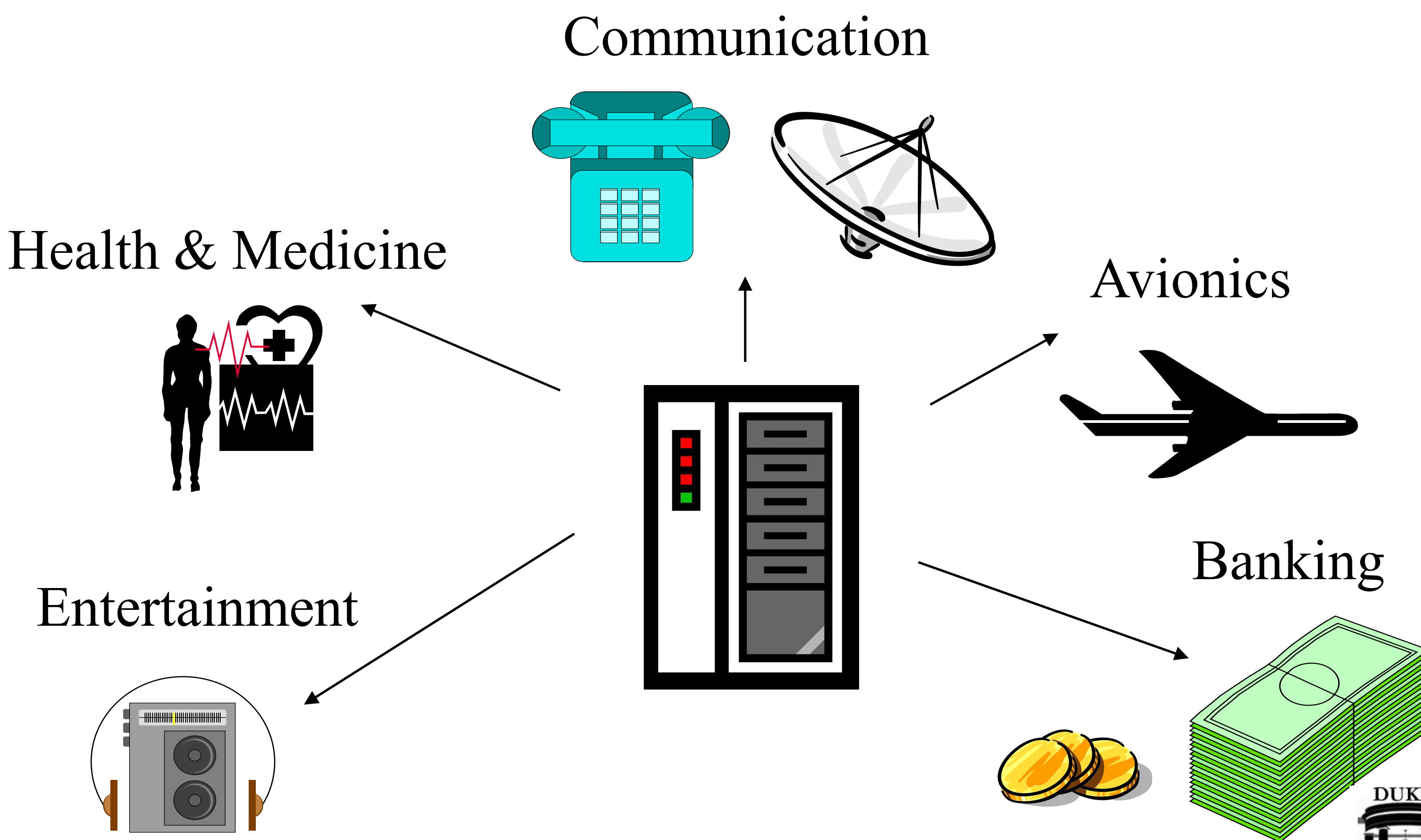
Guest Lecturers

- Dr. Ivan Mura: Software Architecture – Jan 18
- Dr. Veena Mendiratta: Microservice Architecture – Jan. 23
- Dr. Ivan Mura: Software Testing – Jan 25
- Prof. Hiroyuki Okamura: Software Reliability Growth Models – Feb. 1
- Prof. Rivalino Matias, Jr.: Quality of Experience and Software Reliability – Feb. 7
- Prof. Swapna Gokhale: Arch.-based Software Reliability – Feb. 6
- Prof. Roberto Pietrantuono: Software Bug classification – Feb. 20
- Prof. Fangyun Qin: Automated Bug Classification -- Feb. 22
- Prof. Roberto Natella: software aging Monitoring & data analysis – Feb. 29
- TAs: ARB Removal – Mar. 5
- Dr. Michael Grottke: Fundamentals of Software Aging – Mar. 7
- Dr. Rivalino Matias: Experimental methods in SAR – Mar.19
- Dr. Kalyan Vaidyanathan: Implementing Software Rejuvenation – Mar. 21
- Dr. Robert Hanmer: Patterns of Software Fault Tolerance – Mar. 28
- Dr. Kun Qiu: Concurrency Bugs – Apr. 4
- Prof. Neil Gong: Software Security – Apr. 9
- Prof. Deong Song Kim: Security vulnerabilities, mitigation and models – Apr. 11
- Prof. Gianfranco Ciardo: Model Checking – Apr. 16

ECE 590: General Introduction

- Motivation
- Definitions
- Methods of Achieving High Reliability/Availability
- Quantitative Assessment Methods
  - Measurements
  - Modeling: Simulative, Analytic, Hybrid solutions
  - Measurements + Modeling

Pervasive Dependence on Computer Systems → Need for High Reliability/Availability



Telecommunications system outages

- Externally caused events
  - Hinsdale, Illinois central office switch fire, May 1988
  - San Francisco Bay Area earthquake, October 1989
  - Oakland fire storm, October 1991
  - Judge Thomas senate vote, October 1991
  - Events of September 11, 2001
  - North America power outage, August 14, 2003
- Internally caused events
  - Signaling System 7 (SS7) outage, January 1990
  - Newark fiber cut, January 1991
  - New York power outage, September 1991

Many other important outages

- Feb. 2010 Bank of America
- Sep. 2010 JP Morgan Chase Bank
- Oct. 2010 Facebook
- Jun. 2011 “Go Daddy”
- Aug. 2011 EMIS systems outage

More Outages

- Black Sept. 2011, In the same week!!!!:
  - Microsoft Cloud service outage (2.5 hours)
  - Google Docs service outage (1 hour)
    - A memory leak due to a software update
- Sept. 2012 GoDaddy (4 hours)
  - 5 millions of websites affected
- Oct. 2012 Amazon
  - 10/15/2012 Web services – 6 hours (Memory leak)
  - 10/27/2012 EC2 – 2 hours

Real failure examples from High Tech companies

Jan. 2014 , Gmail was down for 25 – 50 min.

Oct. 2013, services like post photos and "likes" unavailable

Feb. 2013, Windows Azure down for 12 hours

Jan. 2013, AWS down for an hour approx.

Sept. 2012 - GoDaddy (4 hours and 5 millions of websites affected)

Real failure examples from High Tech companies

Mar. 2015 , Gmail was down for 4 hours and 40 min.

Mar. 2015, Down for 3 hours affecting Europe and US

Dec. 2015, Microsoft Office 365 and Azure down for 2 hours

Sept. 2015, AWS DynamoDB down for 4 hours impacting among others Netflix, AirBnB, Tinder

Mar. 2015, Apple ITunes, App Stores long outage: 12 hours

More recent examples of real failures

amazon.com — Feb. 2017 - Amazon S3 service outage (almost 6 hours)

Google — Jul. 2017 - Google Cloud Storage service outage (3 hours and 14 min.) - API low-level software defect

Microsoft — Jul. 2017 - Microsoft Azure service outage (4 hours) – Load Balancer Software bug

Google — Dec. 2020- Google released a statement confirming the authentication system outage for approximately 45 minutes due to an internal storage quota issue.

facebook — Oct. 2021, Facebook became globally unavailable for a period of six to seven hours.

- These examples indicate that even the most advanced tech companies are offering not that satisfying reliability/availability

DUKE

Jan 11, 2023

- FAA grounds all domestic flights due to safety alert system computer outage
- Senate Commerce Committee chair Maria Cantwell, a Democrat, said “We will be looking into what caused this outage and how redundancy plays a role in preventing future outages"
- Republican Senator Ted Cruz called the failure "completely unacceptable" and said the issue should lead to reforms

Failures & Downtime Lead to

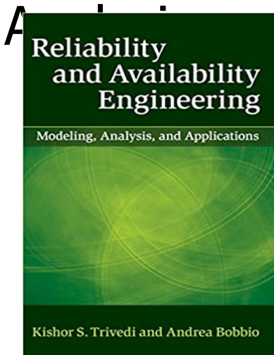
- Loss of Reputation
- Loss of Revenue
- Possible Loss of Mission
- Possible Loss of Life

Need Methods

- That reduce system failures and reduce downtime due to these failures (contributed by hardware, software and humans)

- System Reliability/Availability assessment and bottleneck detection to help decide the most cost-effective path to improvement of reliability/availability
  - During Design phase
  - During Testing/Debugging phase
  - During Operational phase

Ref: Trivedi & Bobbio, Reliability and Availability: Modeling, A
Applications, Cambridge University Press, 2017



1. Introduction

Basic Definitions

Fault, Error, Failure

- *Failure* occurs when the delivered service no longer complies with the desired service as in specifications.
- *Error* is an intermediate system state which is liable (may or may not) lead to subsequent failure.
- *Fault* is adjudged or hypothesized cause of an error.

Faults/Bugs may cause errors may lead to failures

.... Fault → Error → Failure ....

Fault, Error, Failure

cause of failure

An erroneous internal state that may lead to failure

service departs from specification

fault → error → failure

(1) Design/manu/integration
(2) physical

May raise

Errors can be detected and/or corrected

exception

Fault, Error, Failure

$$
\begin{array}{ccccc}
\text{fault} & \longrightarrow & \text{error} & \longrightarrow & \text{failure} \\
\downarrow & & \downarrow & & \downarrow \\
\text{Detected and corrected} & & \text{Detected/corrected} & & \text{Detected}
\end{array}
$$

Fault Classification

- Node vs. Link
- Physical vs. Design/Development vs. Manufacturing vs. Interaction
- Hardware vs. Software vs. Human
- Hardware:
  - Permanent, Intermittent, Transient
- Software
  - Bohrbugs, Mandelbugs, Heisenbugs, Aging-related bugs

Failure Modes

- Omission failures
  - Crash failures
  - Infinite loop
- Response or Value failures
- Timing failures
  - Late (performance or dynamic failures)
  - Early
- Safe vs. Unsafe failure
- Security failures: Breach of confidentiality vs. breach of integrity vs. loss of use

Failure Severity

Critical: A failure which affects critical functionality or critical data

Major (High): A failure which affects major functionality or major data

Minor (Medium): The failures that affect minor requirements or non-critical data

Trivial (Low): This failure does not affect functionality or data. It is just only an inconvenience.

Need for a new term

- *Reliability* is used in a generic sense as an umbrella term.
- *Reliability* is also used as a precisely defined mathematical function.
- To remove confusion, IFIP WG 10.4 has proposed *Dependability* as an umbrella term and *Reliability* is to be used as a well-defined mathematical function.

DUKE

Dependability– An umbrella term

- Trustworthiness of a computer system such that reliance can justifiably be placed on the service it delivers

- Dependability
  - Attributes
    - Availability
    - Reliability
    - Safety
    - Maintainability
  - Means
    - Fault Avoidance/Prevention
    - Fault Removal
    - Fault Tolerance
    - Fault Forecasting
  - Threats
    - Faults
    - Errors
    - Failures

System Attributes of Concern

- **Reliability**
  - Continuity of service; that is, how long does system work w/o system failure
  - "*The ability of a system to perform a required function under given conditions for a given time interval*." No recovery is assumed once ***system*** fails (there can be recovery after a component/subsystem failure)
- **System Mean time to failure (MTTF)** – *derived from Reliability*:
  - The expected time that a system will operate before the first system failure occurs.
- **Availability**
  - Readiness of service; that is, how frequently system fails and how quickly can it be repaired
  - "*The ability of a system to be in a state to perform a required function at a given instant of time.*" Recovery after ***system*** failure allowed

DUKE

Basic Definitions

- One shot Reliability *R:*

  When is this applicable?

- (time-dependent) Reliability $R(t)$:

  X : Time to failure of a system (TTF), or lifetime random variable

  $F(t)$: cumulative distribution function of system lifetime

$$F(t) = P(X \leq t)$$
$$R(t) = P(X > t) = 1 - F(t)$$

Reliability: complementary distribution function of TTF

DUKE

Basic Definitions

- Mean Time To system Failure:

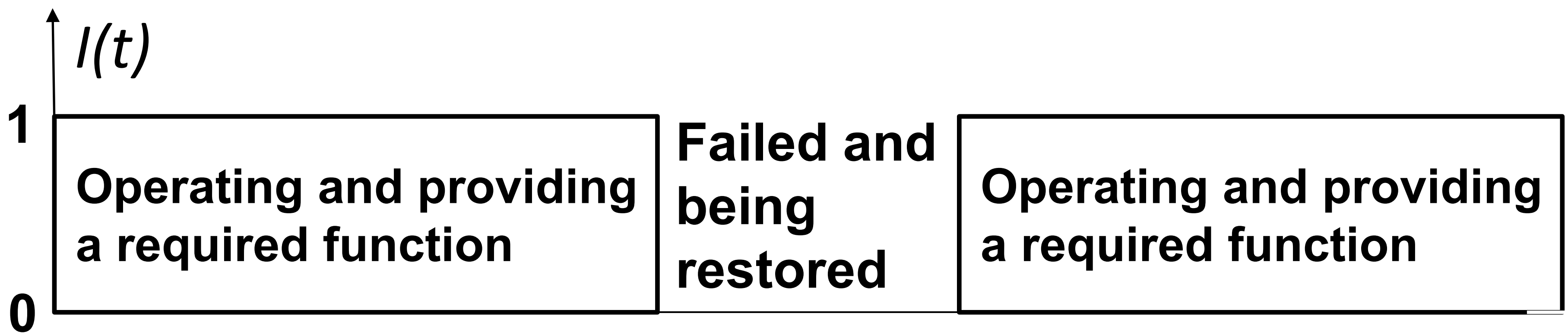
Let $f(t)$: prob. density function of system TTF

$$MTTF = E[X] = \int_0^\infty t f(t)\,dt = \int_0^\infty R(t)\,dt$$

Make a clear distinction between *TTF, R(t)* and *MTTF*

Basic Definitions

- Availability



System Failure and Restoration Process

I(t) is the indicator function

Basic Definitions

- Instantaneous Availability $A(t)$:

$$A(t) = P(\text{system working at } t)$$

  - Using the example in the figure, the availability at time t becomes:

$$A(t)=P(I(t)=1)$$

  - This is sometimes called point-wise availability, instantaneous availability, or transient availability. $A(t)$ can be asked for at any point t in time.

Basic Definitions

- Limiting or Steady-state availability or just availability
  - Long-term probability that the system is available when requested (limit of $A(t)$ as $t \rightarrow \infty$):

$$A_{ss} = \frac{MTTF}{MTTF + MTTR}$$

  - MTTF is the system mean time to failure, a complex combination of component MTTFs
  - MTTR is the system mean time to recovery

    - TTR may consist of many phases

For a non-fault-tolerant system, the formula holds without any distributional assumptions

DUKE

Basic Definitions

- Downtime in minutes per year
  - In industry, (un)availability is usually presented in terms of annual downtime.
  - Downtime = $8760 \times 60 \times (1 - A_{ss})$ minutes.
  - In Industry it is common to define the availability in terms of number of nines
    - 5 NINES ($A_{ss} = 0.99999$) → 5.26 minutes annual downtime
    - 4 NINES ($A_{ss} = 0.9999$) → 52.56 minutes annual downtime

Number of Nines– Reality Check

- 49% of Fortune 500 companies experience at least 1.6 hours of downtime per week
  - Approx. 80 hours/year=4800 minutes/year
  - A_{ss}=(8760-80)/8760=0.9908
  - **That is, between 2 NINES and 3 NINES!**

Failures & Downtime Lead to

- A Loss of Reputation
- A Loss of Revenue
- Possible Loss of Mission
- Possible Loss of Life

DUKE

Downtown Costs per Hour

Brokerage operations	$6,450,000
Credit card authorization	$2,600,000
eBay (1 outage 22 hours)	$225,000
Amazon.com	$180,000
Package shipping services	$150,000
Home shopping channel	$113,000
Catalog sales center	$90,000
Airline reservation center	$89,000
Cellular service activation	$41,000
On-line network fees	$25,000
ATM service fees	$14,000

Source: InternetWeek 4/3/2000; *Fibre Channel: A Comprehensive Introduction*, R. Kembel 2000, p.8. "...based on a survey done by Contingency Planning Research."

Loss of Life

- In Commercial aircrafts (Boeing 737 Max software problem)
  - Ethiopian Airlines Flight, March 2019, 149 people died
  - Lion Air Flight crash, Oct. 2018, 189 people died

Need Methods

- That reduce system failures and reduce downtime due to these failures (contributed by hardware, software and humans)

- For System Reliability/Availability assessment and bottleneck detection to help decide the most cost-effective path to improvement of reliability/availability – ECE 555 Deals with this topic

Means to Improve Dependability

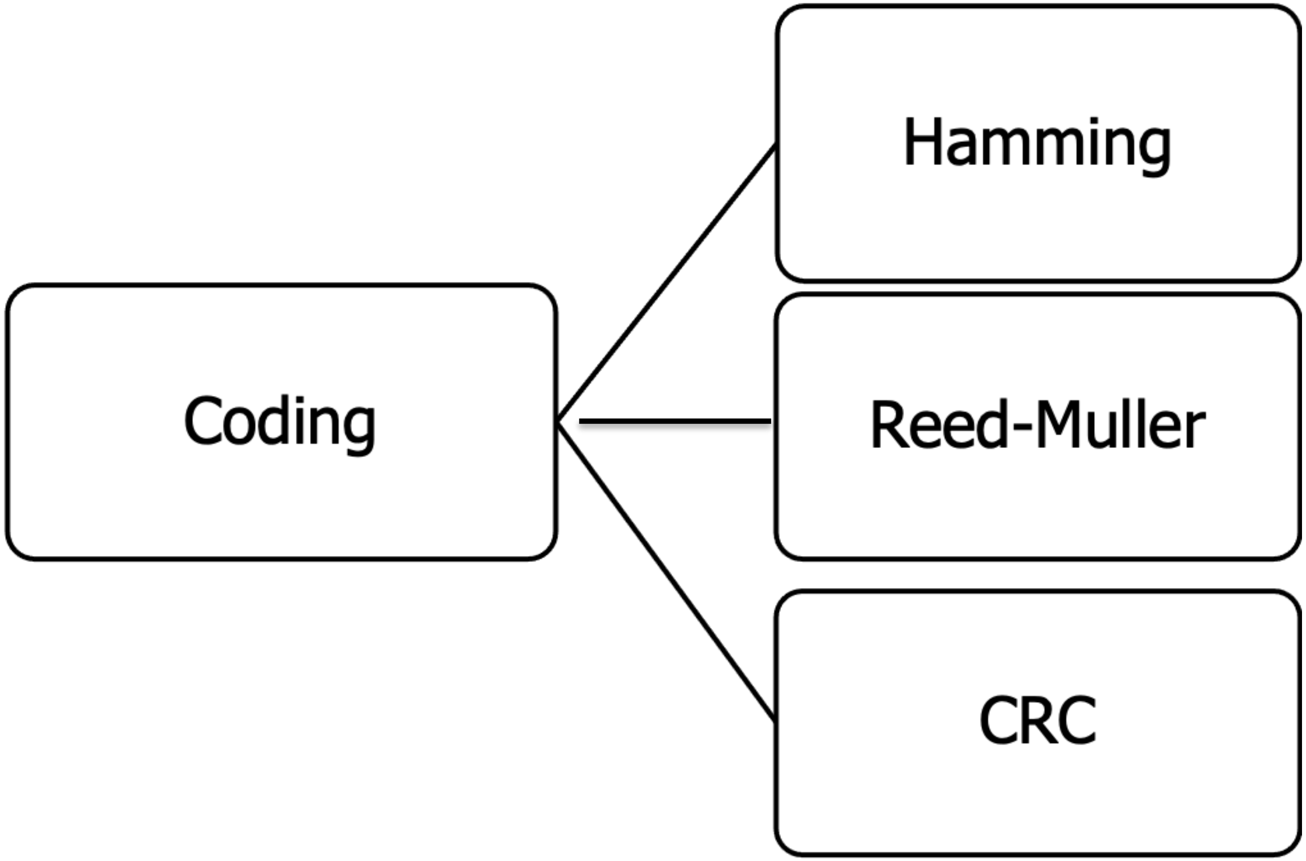
- Fault Avoidance (or Fault Prevention)
  - Employ highly reliable components
  - Employ good software engineering practices
- Fault Removal
  - Careful Testing to remove faults
- Fault Tolerance
  - Utilize Redundancy
- Fault Forecasting
  - Identify bottlenecks/ fault-prone modules (at design time)
  - Predict when failures may occur and thence carry out preventive maintenance (at operational time)

DUKE

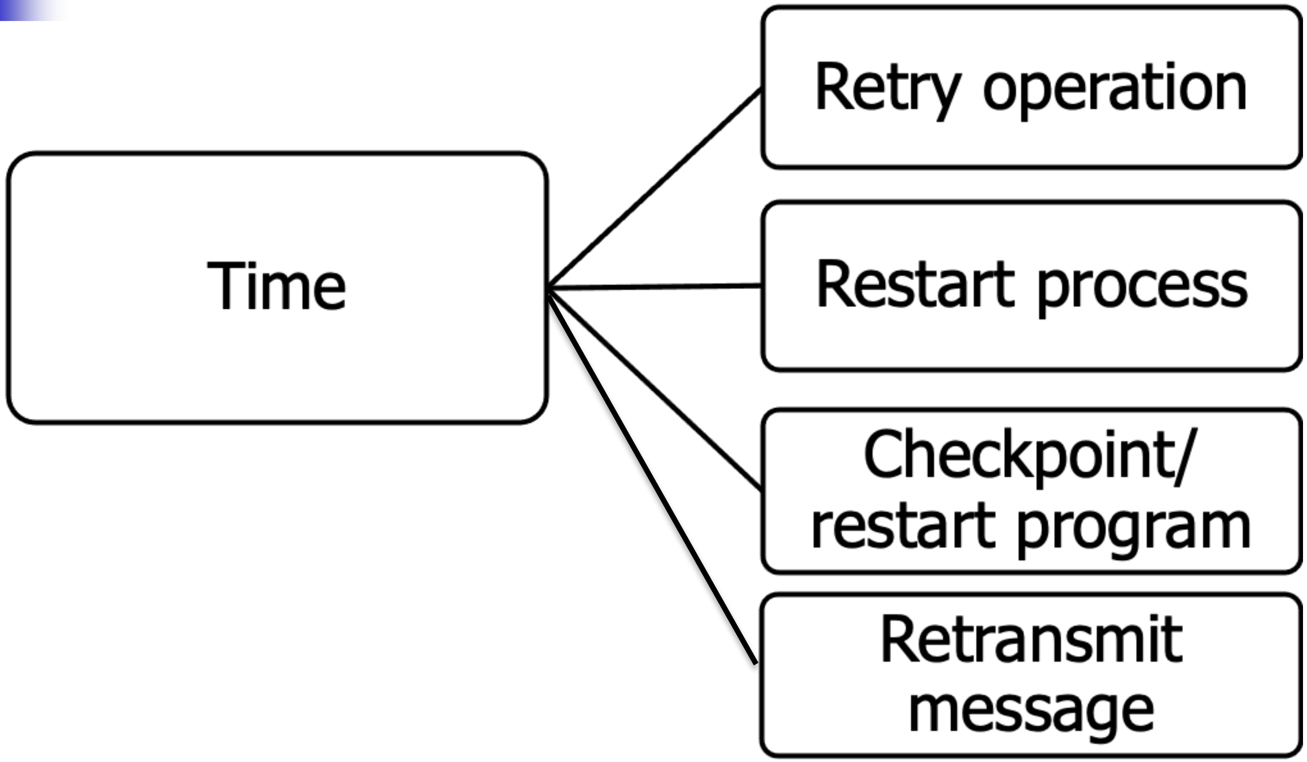
Means Overview (Redundancy)

- Redundancy
  - Coding
  - Time
  - Use of Multiple Redundant Components, i.e., more components than required for the performance needs

Coding Redundancy



Time Redundancy

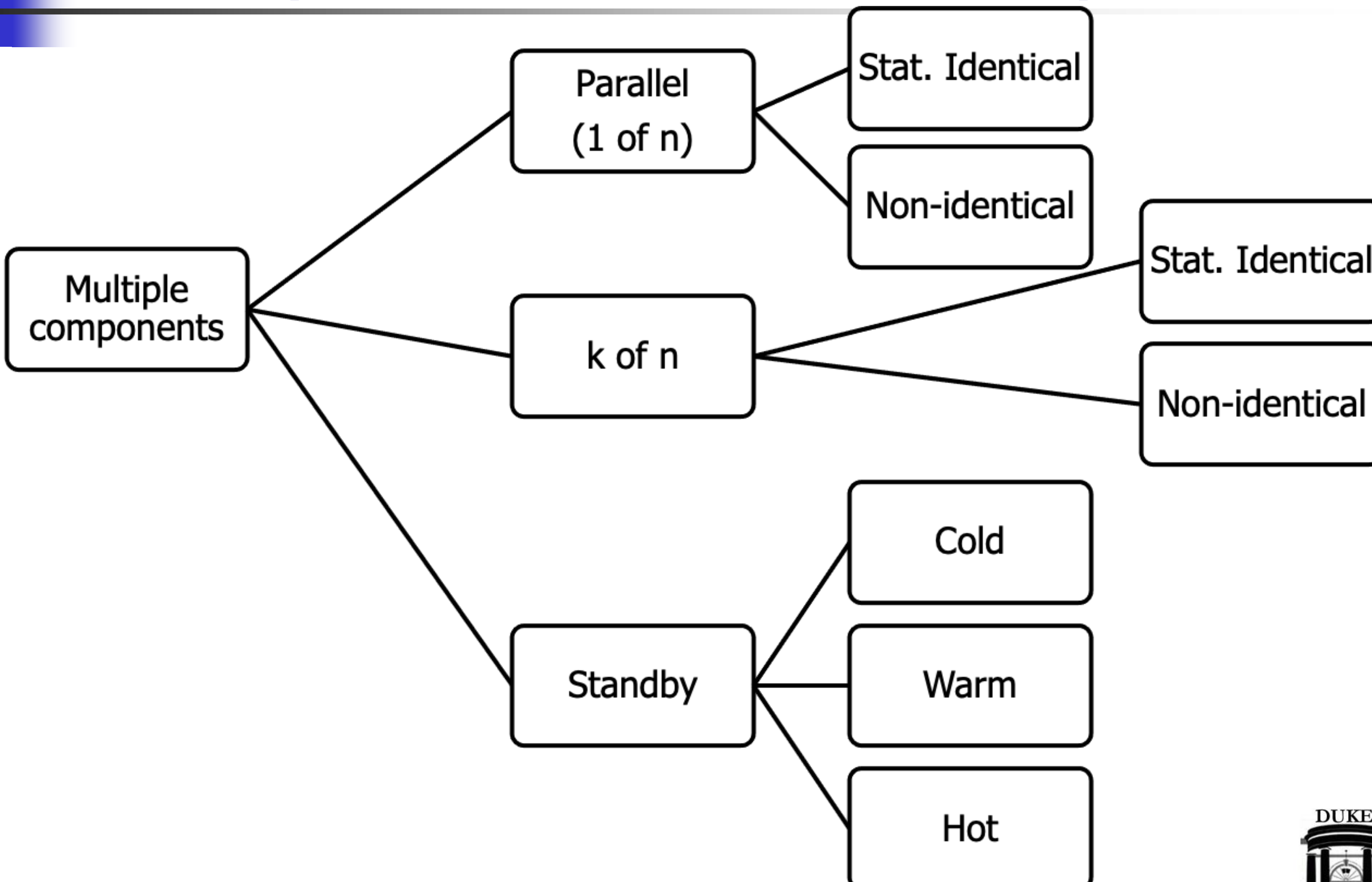


If at first you don't succeed, try and try again

Some Notes

- Time redundancy is time-honored method to tolerate hardware transient faults
- It is now recognized that time redundancy (retry, restart, reboot) can also be used to recover from software failures – more on this later

Multiple Redundant Components

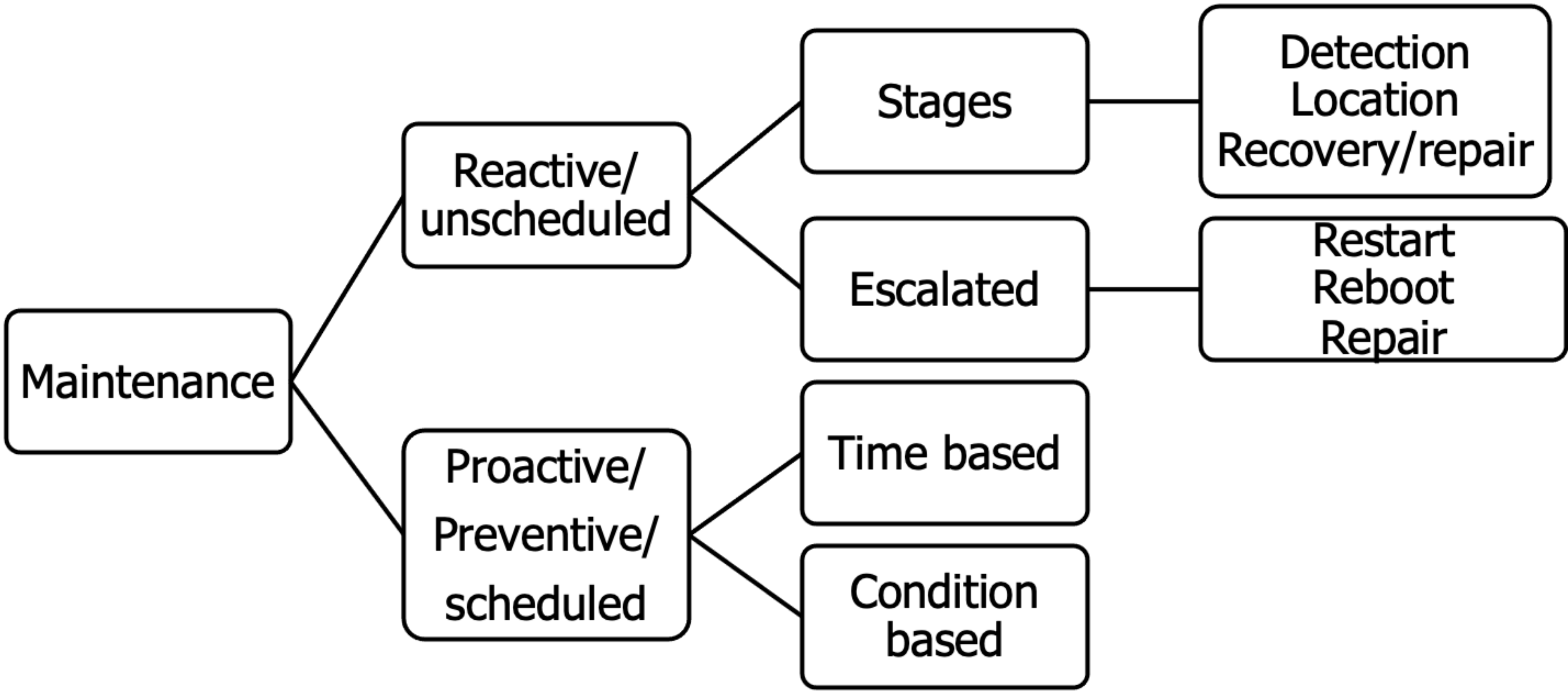


Some More Notes

- In hardware redundancy, statistically identical components are commonly (though not always) used
- In software, it has been recognized since 1970's that identical redundant copies of software will not be useful for fault tolerance
- So, classical techniques for Software Fault Tolerance are based on the idea of **design diversity**
  - Recovery block
  - N-version programming
- It is now recognized that failover to identical software copy does help in recovering after software failures – more on this later

DUKE

Maintenance

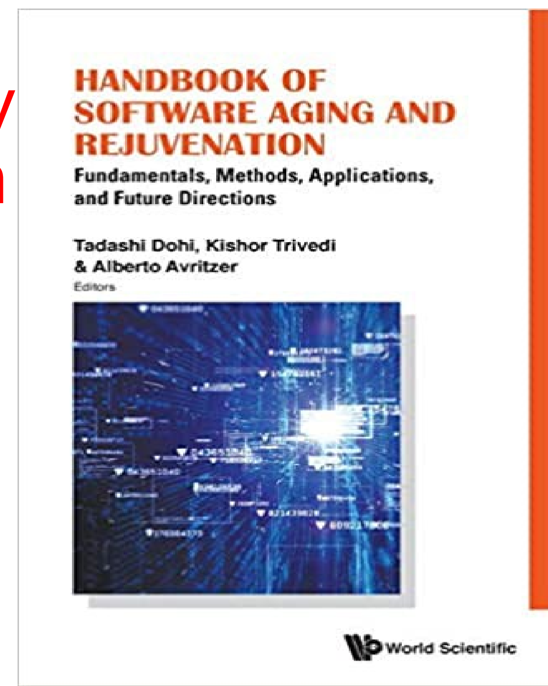


DUKE

Software Aging

- Conventional wisdom is that unlike hardware, software does not age, so proactive recovery will not help
- However, since 1995 it has been recognized that software does age and software rejuvenation (proactive recovery) does help improving software reliability/availability
- We (my group at Duke) helped study & implement software rejuvenation in IBM X-series Director:

Proactive Management of Software Aging, Castelli, Harper, Heidelberger, Hunter, Trivedi, Vaidyanathan, Zeggert, IBM JRD, 2001



Need Methods

- That reduce system failures and reduce downtime due to these failures (contributed by hardware, software and humans)
  - Fault-Tolerant Computing
  - Dependable Computing
  - Resilient Computing
  - Trustworthy Computing

- System Reliability/Availability assessment and bottleneck detection methods can be used:
  - To compare alternative designs/architectures
  - Find bottlenecks, answer what if questions, design optimization and conduct trade-off studies
  - At certification time
  - At design verification/testing time
  - Configuration selection phase
  - Operational phase for system tuning/on-line control

DUKE

Quantitative Assessment methods for system reliability and availability

- Black-box or Data-driven (measurement data + statistical inference):
  - The system is treated as a monolithic whole, without explicitly taking its internal structure into account
  - Very expensive especially for ultra-reliable systems
    - ALT can help reduce the cost
  - Generally applicable to small systems that are not very highly reliable
  - Not feasible for system under design/development

DUKE

Quantitative Assessment approaches

- White-box (or Model-driven):
  - When no data is available for the system as a whole
  - Stochastic Model (e.g., RBD, Ftree, Markov chain) constructed based on the known internal structure of system – its components, their characteristics and interactions between components
  - Derive the behavior of ensembles (combinations of components to form a system) from first principles of probability theory
  - Used to analyze a system with many interacting and interdependent components

DUKE

Quantitative Assessment approaches

- White-box (or Model-driven):
  - Probability Model (e.g., RBD, Ftree, Relgraph, Markov chain, SMP, hierarchical…) constructed based on the known internal structure of system – its components, their characteristics and interactions between components
  - Need input parameters for components and subsystems

DUKE

Quantitative Assessment approaches

- Combined approach
  - Use black-box approach at subsystem/component level
  - Use white-box approach at the system level
  - Thus, a combined Data + Model driven approach

DUKE

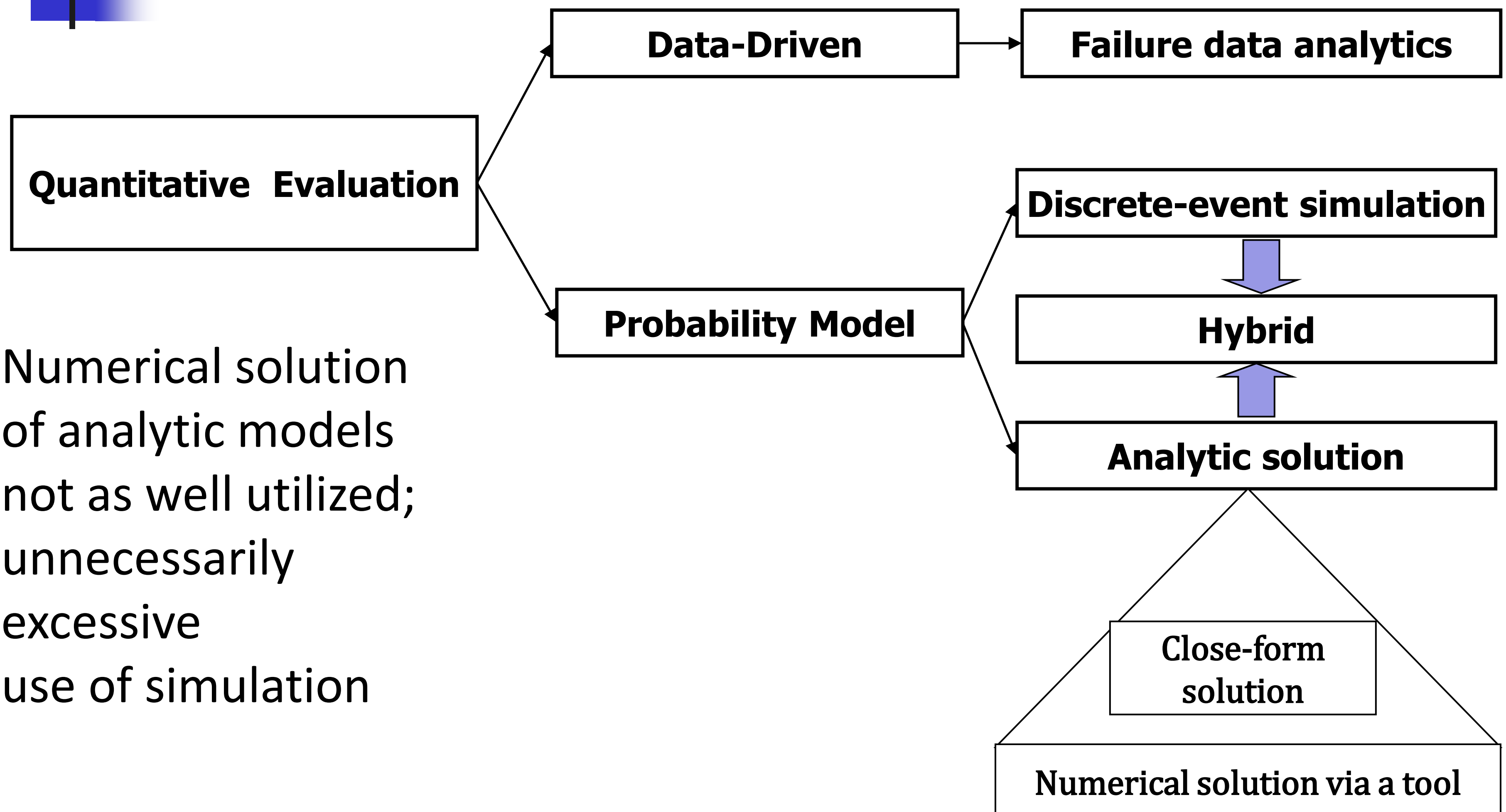
Software Reliability Assessment

- Black-box or Data-driven (measurements + statistical inference):
  - System is treated as a monolithic whole, considering its input, output and transfer characteristics without explicitly taking into account its internal structure – SRGM
- White-box (or grey box) or Model-driven:
  - Internal structure of system explicitly considered using a Probability Model (e.g., CTMC,SMP,MRGP,PFQN,FTREE)
  - Used to analyze a system with many interacting and interdependent components – Architecture-based soft. Rel.
- Combined approach
  - Use black-box approach at subsystem/component level
  - Use white-box approach at the system level

DUKE

Overview of Assessment Methods

ECE 555



Numerical solution
of analytic models
not as well utilized;
unnecessarily
excessive
use of simulation

Data-Driven Assessment

- Measurement-Based (black-box) or Data-driven
  - More Accurate, more expensive
  - Not possible in system design phase but during debugging phase
  - Not cost effective when many system configurations or parameter settings are to be compared/evaluated
  - More difficult for overall system reliability, availability, performability then for pure performance assessment
- Statistical techniques are very important here
  - Inference (point estimate, confidence intervals)
  - Hypothesis testing
  - Regression and Analysis of Variance
  - Design of experiments (DoE)
  - Accelerated life testing (ALT)

Model-based Evaluation

- White Box or Probability Model-Based

Less Accurate, Less expensive, possible at design time

  - Discrete-Event Simulation vs. Analytic solution
  - Hybrid: Simulation + Analytic **(SPNP)**

Black+Whitebox

- Data + Models: Measurements at subsystem level and probability model at the system level
  - Mei-Chen Hsueh, Ravi Iyer, Kishor Trivedi: Performability Modeling Based on Real Data: A Case Study. IEEE Trans. Computers 37(4): 478-484 (1988)
  - Kalyan Vaidyanathan, Kishor Trivedi: A Comprehensive Model for Software Rejuvenation. IEEE Trans. Dependable Secur. Comput. 2(2): 124-137 (2005)
  - Swapna Gokhale, W. Eric Wong, Robert Horgan, Kishor Trivedi: An analytical approach to architecture-based software performance and reliability prediction. Perform. Evaluation 58(4): 391-412(2004)
  - Kishor Trivedi, Dazhi Wang, Jason Hunt, Andrew Rindos, Earl Smith, Vashaw: Availability Modeling of SIP Protocol on IBM WebSphere. PRDC 2008: 323-330

High Reliability/Availability

- Hardware fault tolerance, fault management, reliability/availability/performance assessment methods relatively well developed

- System outages more due to software faults

Key Challenge:

Software reliability is one of the weakest links in system reliability/availability

DUKE

Software Reliability: Means

- **Fault prevention or Fault avoidance**
- **Fault Removal**
- **Fault Tolerance**
- **Fault Forecasting**

Key References

- **Probability and Statistics with Reliability, Queuing, and Computer Science Applications**, Trivedi, second edition, paperback, John Wiley, 2016
- **Reliability and Availability Engineering**, Trivedi & Bobbio, Cambridge Univ. Press, 2017
- **Why do computers stop and what can be done about it?** Gray, SRDS 1986
- **A census of tandem system availability between 1985 and 1990,** Gray, IEEE-TR, 1990
- **Basic concepts and taxonomy of dependable and secure computing**, Avizienis, Laprie, Randell, Landwehr, IEEE TDSC,2004.
- **Lessons Learned From the Analysis of System Failures at Petascale: The Case of Blue Waters**, Martino, Baccanico, Fullop, Kramer, Kalbaczyk, and Iyer, DSN 2014