Software Fault Tolerance
via Environmental Diversity

SERE

July 2, 2014

Kishor Trivedi

Dept. of Electrical & Computer Engineering

Duke High Availability Assurance Lab (DHAAL)

Duke University

Durham, NC 27708

ktrivedi@duke.edu

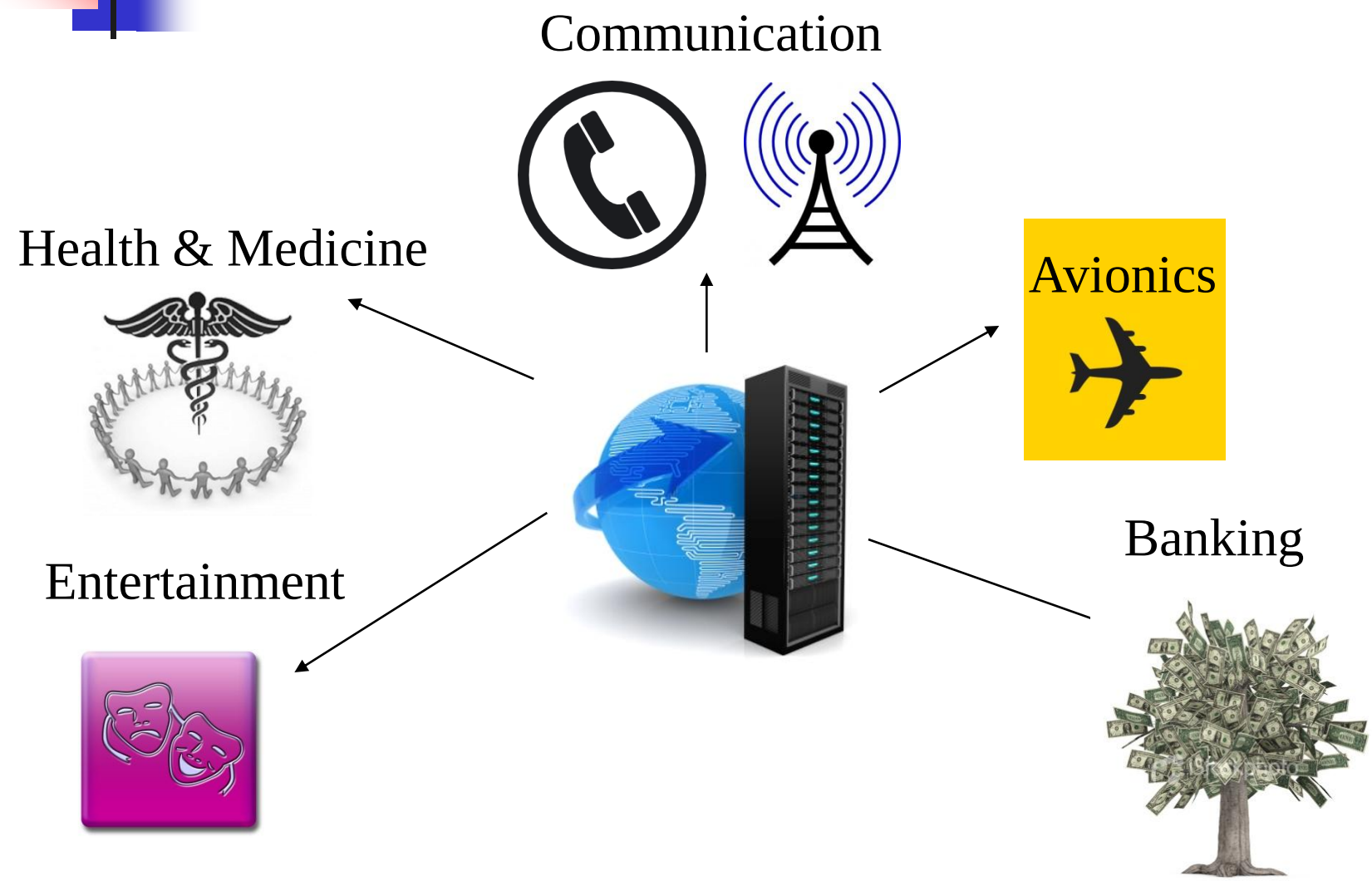
www.ee.duke.edu/~ktrivedi

Outline

- Motivation
- A Real System
- Software Fault Classification
- Environmental Diversity
- Methods of Mitigation
- Software Aging and Rejuvenation
- Conclusions

Duke
UNIVERSITY

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



Basic Definitions

- Steady-state availability (A_{ss}) or just availability
  - Long-term probability that the system is available when requested:

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

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

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.
  - 5 NINES ($A_{ss} = 0.99999$) → 5.26 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!
- This study assumes planned and unplanned downtime, together

Duke UNIVERSITY

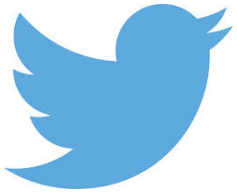
Achieving High Availability is a Challenge

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

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

Feb. 2013, Windows Azure down for 12 hours

Jan. 2013, AWS down for an hour approx.



Duke UNIVERSITY

More Failures

- 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 Webservices – 6 hours (Memory leak)
  - 10/27/2012 EC2 – > 2 hours

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

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

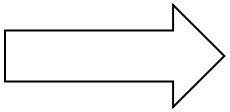
Duke UNIVERSITY

High Reliability/Availability:

Software is the problem

- Hardware fault tolerance, fault management, reliability/availability modeling relatively well developed

- System outages more due to software faults



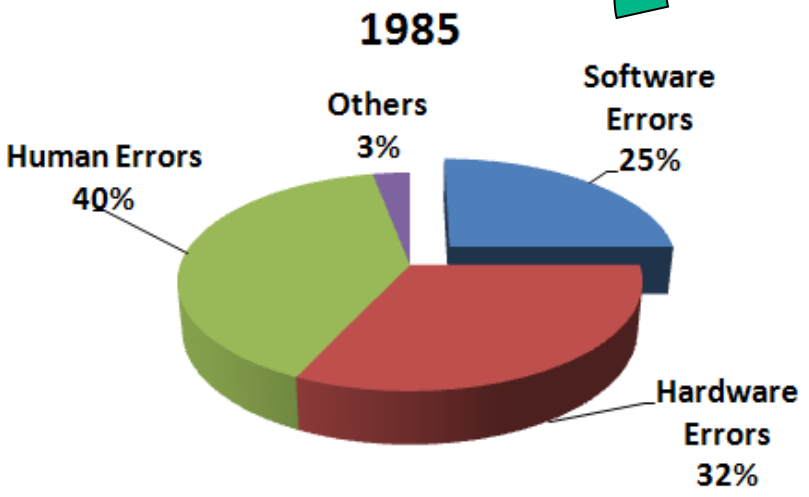
Key Challenge:

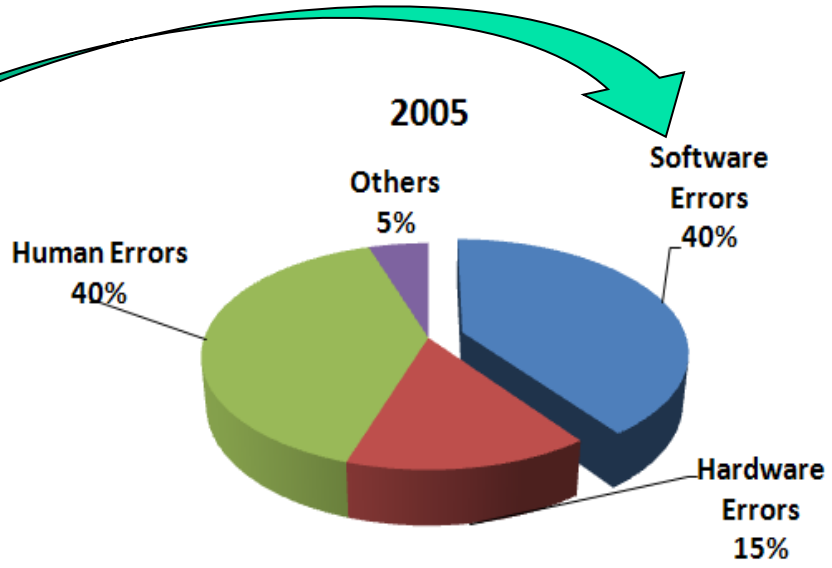
Software reliability is one of the weakest links in system reliability/availability

Duke UNIVERSITY

Software is the problem

Jim Gray's paper titled "Why do computers stop and what can be done about it?" pointed out this trend in 1985, followed by his paper
"A census of tandem system availability between 1985 and 1990"



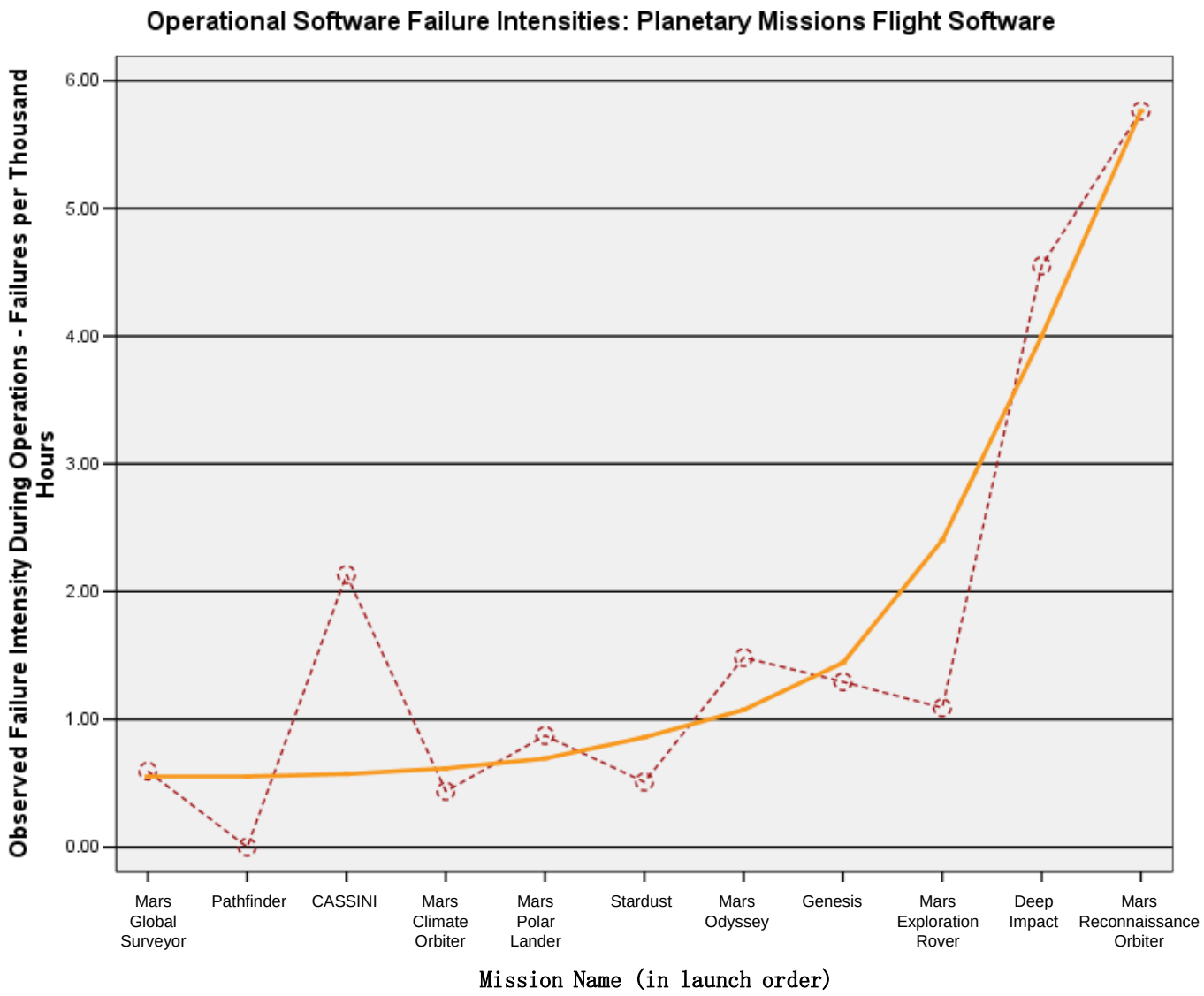


1985 → **2005**

Across different industries…

Increasing SW Failure Rate?

Planetary Missions Flight Software: A. Nikora of JPL



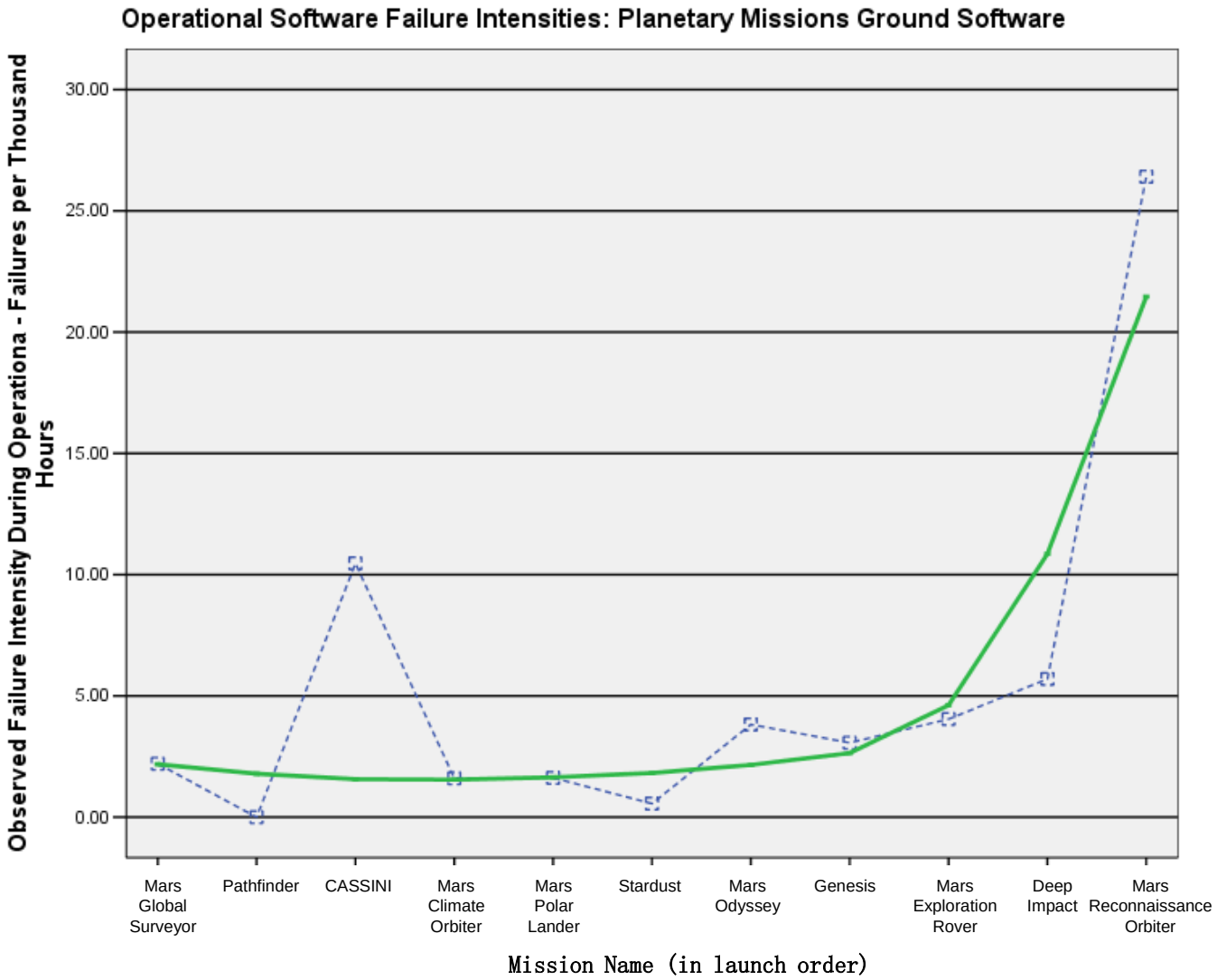
The interval between the first and last launch:
8.76 years.

The interval between successive launches ranges from:
23 to 790 days.

Duke UNIVERSITY

Increasing SW Failure Rate?

Planetary Missions Ground Support Software: A. Nikora of JPL



The interval between the first and last launch: 8.76 years.

The interval between successive launches ranges from:
23 to 790 days.

Duke UNIVERSITY

Software Reliability: Known Means

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

Software Reliability

- **Fault prevention or Fault avoidance**
  - Good software engineering practices
    - Requirement Elicitation (Abuse Case Analysis – TCS SSA)
    - Design Analysis / Review
    - Secure Programming Standard & Review
    - Secure Programming Compilation
    - Software Development lifecycle
    - Automated Code Generation Tools (IDE like Eclipse)
  - Use of formal methods
    - UML, SysML, BPM
    - Proof of correctness
    - Model Checking (SMART, SPIN)
  - Bug free code not yet possible for large scale software systems
- **Yet there is a strong need for failure-free system operation**

Software Reliability

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

Software Reliability

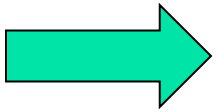
- **Fault removal**
  - Can be carried out during
    - the specification and design phase
    - the development phase
    - the operational phase
  - Failure data may be collected and used to parameterize a software reliability growth model to predict when to stop testing
- Software is still delivered with Many bugs either because of inadequate budget for testing , very difficult to detect/localize/correct bugs or inadequacy of techniques employed/known

Software Reliability

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

Software Reliability

- There are stringent requirements for failure-free operation of software-based systems – next idea



Software fault tolerance is a potential solution to improve software reliability in lieu of virtually impossible fault-free software

Software Fault Tolerance

Classical Techniques

Design diversity

- N-version programming
- Recovery block
- N-self check programming

Expensive (unless based on component-based design)→ not used much in practice!

Yet there are stringent requirements for failure-free operation

Challenge: Affordable Software Fault Tolerance

Duke UNIVERSITY

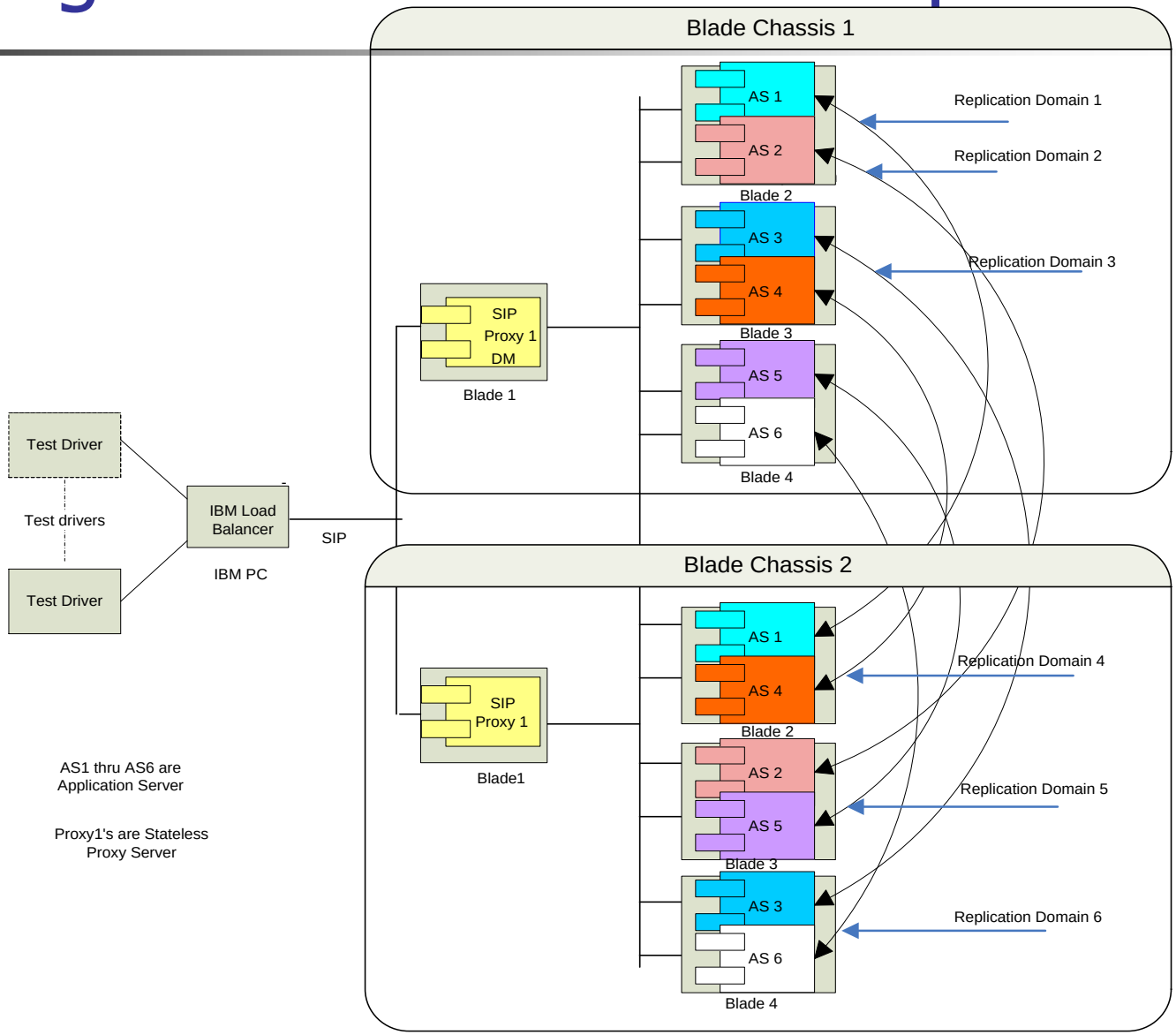
Outline

- Motivation
- A Real System
- Software Fault Classification
- Environmental Diversity
- Methods of Mitigation
- Software Aging and Rejuvenation
- Conclusions

Duke
UNIVERSITY

High availability SIP Application Server Configuration on IBM WebSphere

PRDC 2008 and ISSRE 2010 papers



Duke UNIVERSITY

High availability SIP Application Server configuration on WebSphere

Hardware configuration:

- Two BladeCenter chassis; 4 blades (nodes) on each chassis (**1 chassis sufficient for performance**)

Software configuration:

- 2 copies of SIP/Proxy servers (**1 sufficient for performance**)
- 12 copies of WAS (**6 sufficient for performance**)
- Each WAS instance forms a redundancy pair (**replication domain**) with WAS installed on another node on a different chassis

- The system has hardware redundancy and software redundancy

Duke UNIVERSITY

High availability SIP Application Server configuration on WebSphere

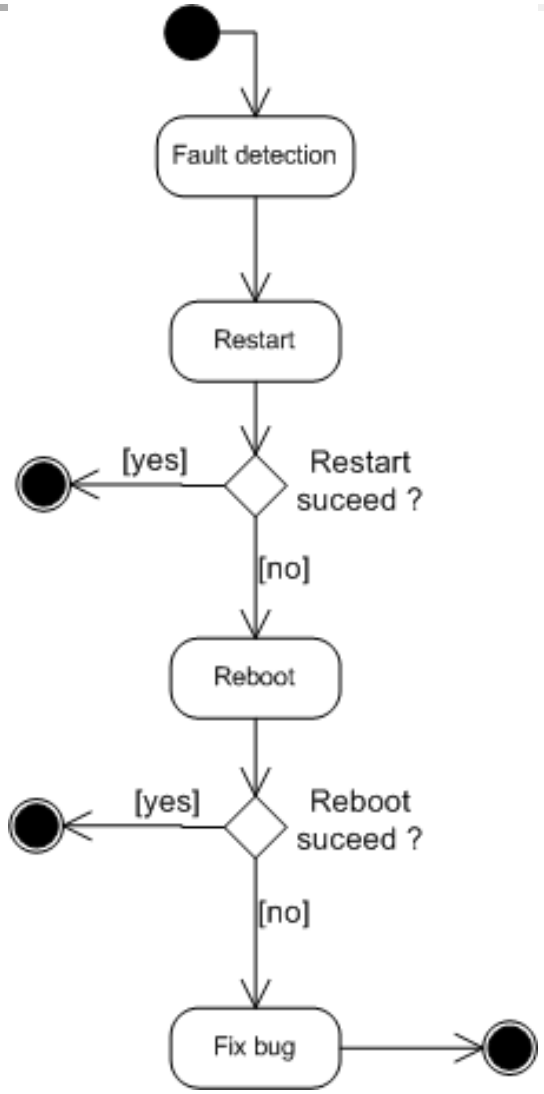
Software Fault Tolerance

- Identical copies of SIP proxy used as backups (**hot spares**)
- Identical copies of WebSphere Applications Server (WAS) used as backups (**hot spares**)
- **Type of software redundancy** – (not design diversity) but replication of identical software copies
- **Normal recovery after a software failure**
  - restart software, reboot node or fail-over to a software replica; only when all else fails, a "software repair" is invoked

Duke UNIVERSITY

Escalated levels of Recovery (Telco)

The flowchart briefly depicts the actions taken for recovery after a failure is detected. Try the simplest recovery method first, then a more complex etc.



Duke
UNIVERSITY

Software Fault Tolerance: New Thinking

Retry, restart, reboot!

- Known to help in dealing with hardware transients
- Do they help in dealing with failures caused by software bugs?
- If yes, why?

Duke UNIVERSITY

Software Fault Tolerance: New Thinking

Failover to an identical software replica (that is not a diverse version)

- Does it help?
- If yes, why?

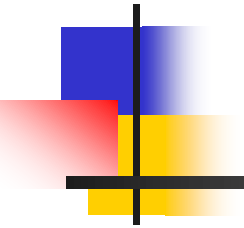
Twenty years ago this would be considered crazy!

Duke
UNIVERSITY

Outline

- Motivation
- A Real System
- **Software Fault Classification**
  - **Fighting Bugs: Remove, Retry, Replicate and Rejuvenate**, M. Grottke and K. Trivedi, *IEEE Computer Magazine,* Feb. 2007
- Environmental Diversity
- Methods of Mitigation
- Software Aging and Rejuvenation
- Conclusions

Duke UNIVERSITY



Software Faults

main threats to high reliability, availability & safety

IFIP Working Group 10.4 (Laprie)

Failure occurs when the delivered service no longer complies with the desired output.

Error is that part of the system state which is liable to lead to subsequent failure.

Fault is adjudged or hypothesized cause of an error.

Faults are the cause of **errors** that may lead to **failures**

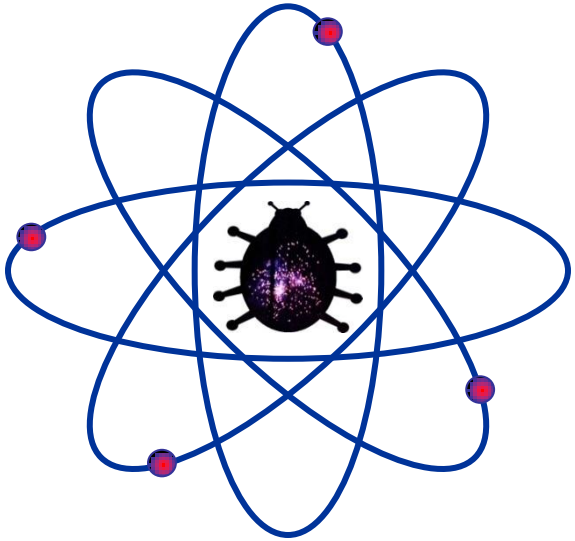
Duke UNIVERSITY

Need to Classify bug types

- We submit that a software fault tolerance approach based on retry, restart, reboot or fail-over to an identical software replica (not a diverse version) works because of a significant number of software failures are caused by **Mandelbugs** as opposed to the traditional software bugs now called **Bohrbugs**

A Classification of Software Faults

- **Bohrbug :=** A fault that is easily isolated and that manifests consistently under a well-defined set of conditions, because its activation and error propagation lack complexity.

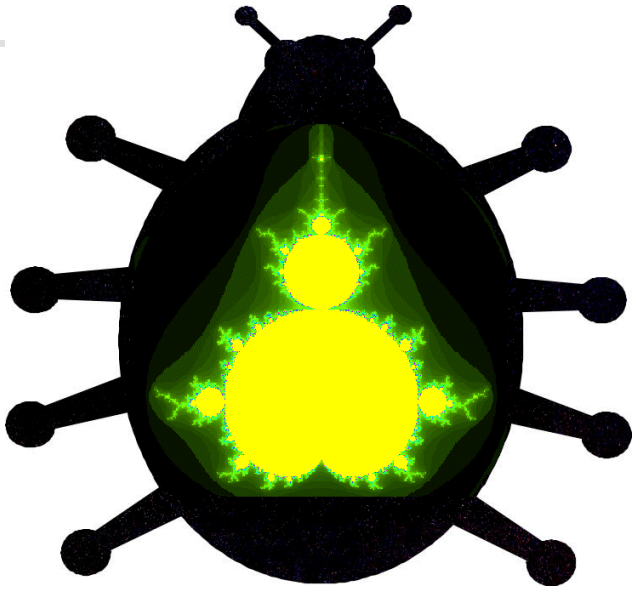


Example: A bug causing a failure whenever the user enters a negative date of birth

- Since they are easily found, Bohrbugs may be detected and fixed during the software testing phase.
- The term alludes to the physicist Niels Bohr and his rather simple atomic model.

Duke UNIVERSITY

Mandelbug – Definition



- **Mandelbug :=** A fault whose activation and/or error propagation are complex. Typically, a Mandelbug is difficult to isolate, and/or the failures caused by a it are not systematically reproducible.
- Example: A bug whose activation is scheduling-dependent
- The residual faults in a thoroughly-tested piece of software are mainly Mandelbugs.
- The term alludes to the mathematician Benoît Mandelbrot and his research in fractal geometry.
- Sometimes called concurrency or non-deterministic bugs

Duke UNIVERSITY

Mandelbugs complexity factors

- A fault is a Mandelbug if its manifestation is subject to the following complexity factors
  - Long time lag between fault activation and failure appearance
  - Operating environment (OS, other applications running concurrently, hardware, network…)
  - Timing among submitted operations
  - Sequencing or Ordering of operations
- A failure due to a Mandelbug may not show up upon the resubmission of a workload if the operating environment has changed enough

Examples of Types of Bugs in IT Systems

- Mandelbugs in IT Systems: Trivedi, Mansharamani, Kim, Grottke, and Nambiar. "*Recovery from failures due to Mandelbugs in IT systems*". PRDC 2011.

- The selected TCS projects ranged across a number of business systems in the banking, financial, government, IT, pharmacy, and telecom sector.

Mandelbug reproducibility

- Mandelbugs are really hard to reproduce
  - Conducted a set of experiments to study the environmental factors (i.e., disk usage, memory occupation and concurrency) that affect the reproducibility of Mandelbugs
    - High usage of environmental factors increases significantly the reproducibility of Mandelbugs
- Submitted to ISSRE 2014

Duke UNIVERSITY

Aging-related Bug – Definition

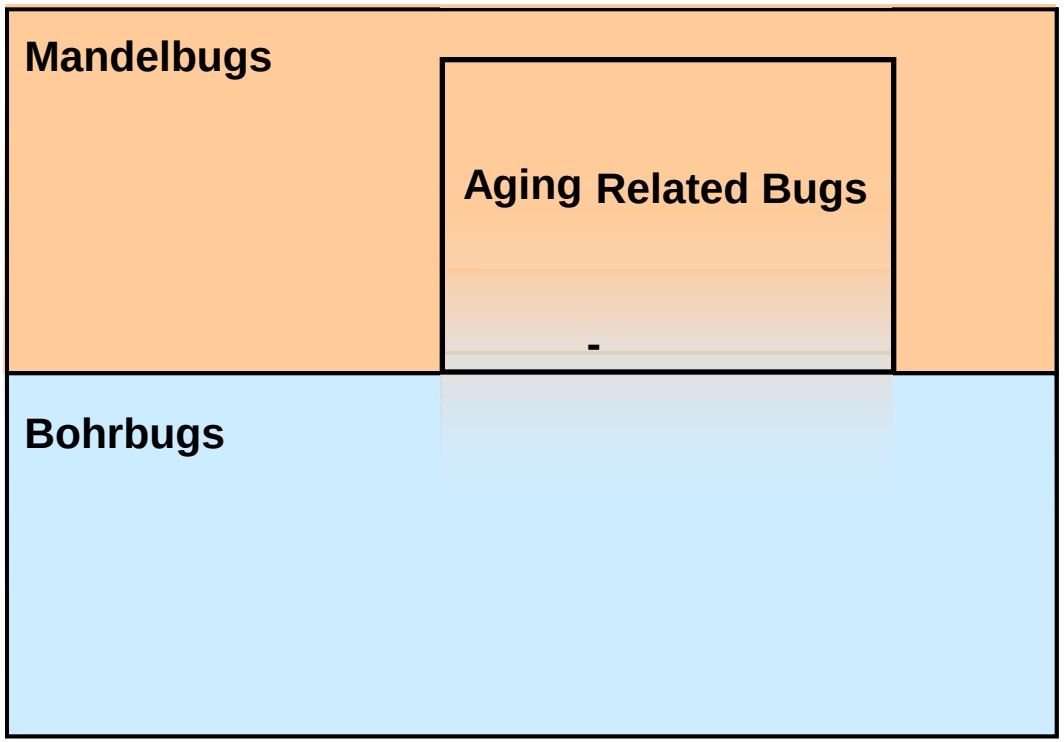
- **Aging-related bug :=** A fault that leads to the accumulation of errors either inside the running application or in its system-context environment, resulting in an increased failure rate and/or degraded performance.

- Example:
  - A bug causing memory leaks in the application
- Note that the aging phenomenon requires a delay between fault activation and failure occurrence.
- Note also that the software *appears to age* due to such a bug; there is no physical deterioration

Duke UNIVERSITY

Relationships

- Bohrbug and Mandelbug are complementary antonyms.
- Aging-related bugs are a subtype of Mandelbugs

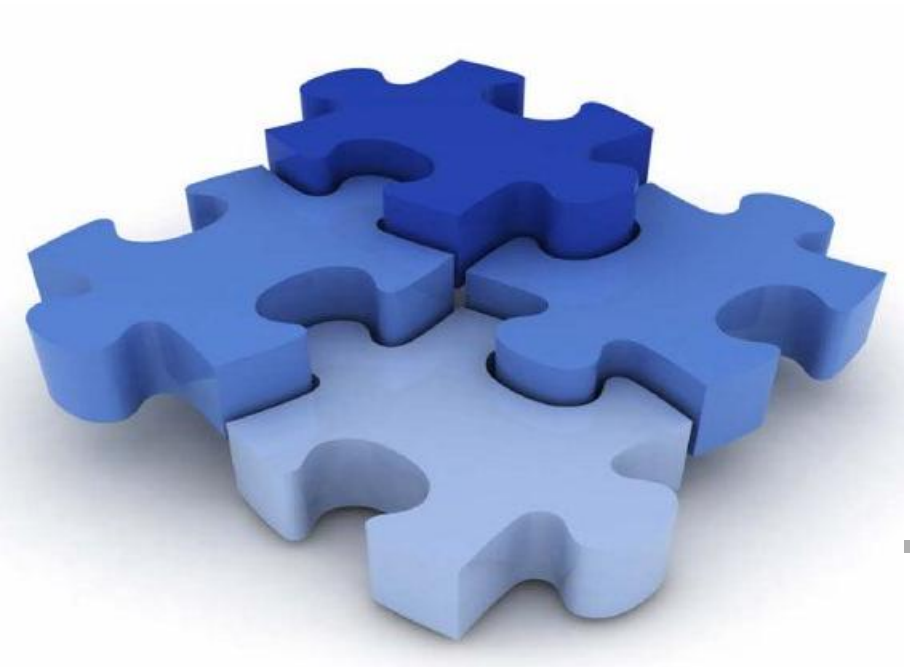


Important Questions about these Bugs

- What fraction of bugs are Bohrbugs, Mandelbugs and aging-related bugs
  - How do these fractions vary
    - over time
    - over projects, languages, application types,…
  - Need Measurements
  - NASA/JPL Project with Allen Nikora & Michael Grottke & Javier Alonso; results:
    - **52% Bohrbugs**
    - **35% Mandelbugs (non-aging-related)**
    - **4% Aging-related bugs**
    - 7% Operator related
    - 2% Unclassified
  - Very similar results for Linux, MySQL, Apache AXIS, httpd

Outline

- Motivation
- A Real System
- Software Fault Classification
- Environmental Diversity
- Methods of Mitigation
- Software Aging and Rejuvenation
- Conclusions



Environmental diversity

A new thinking to deal with software faults and failures

Duke
UNIVERSITY

Software Fault Tolerance: New Thinking

Environmental Diversity as opposed to *Design Diversity*

Our claim is that this (retry, restart, reboot, failover to identical software copy) works since failures due to *Mandelbugs* are not negligible. We thus have an affordable software fault tolerance technique that we call *Environmental Diversity*

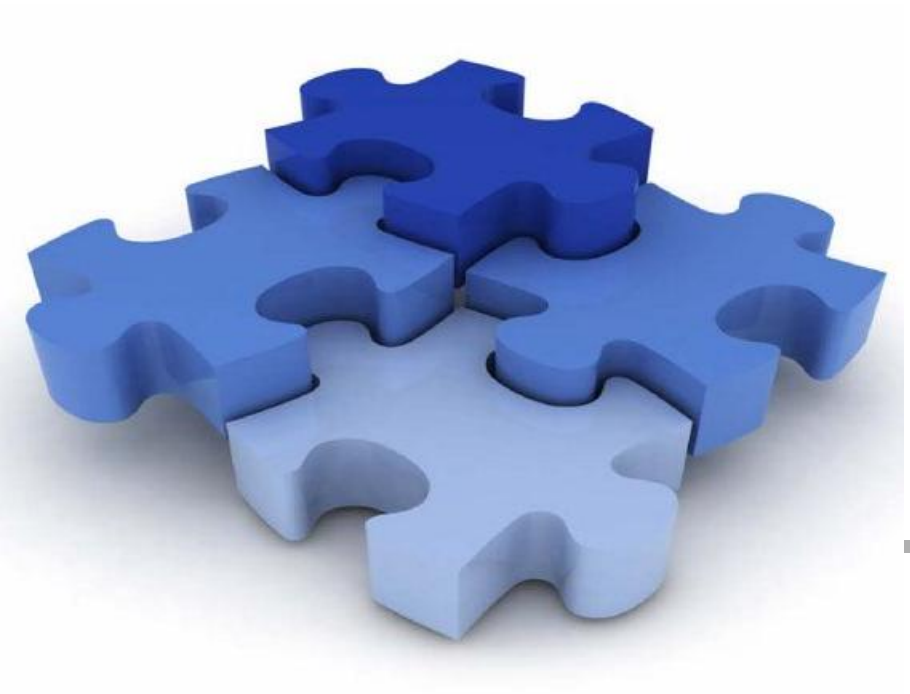
Duke
UNIVERSITY

What is environmental diversity?

- The underlying idea of Environmental diversity
  - Retry a previously faulty operation and it most likely works -- Why?
  - because of the environment where the operation was executed has changed enough to avoid the fault activation.
- The environment is understood as
  - OS resources, other applications running concurrently and sharing the same resources, interleaving of operations, concurrency, or synchronization.

Outline

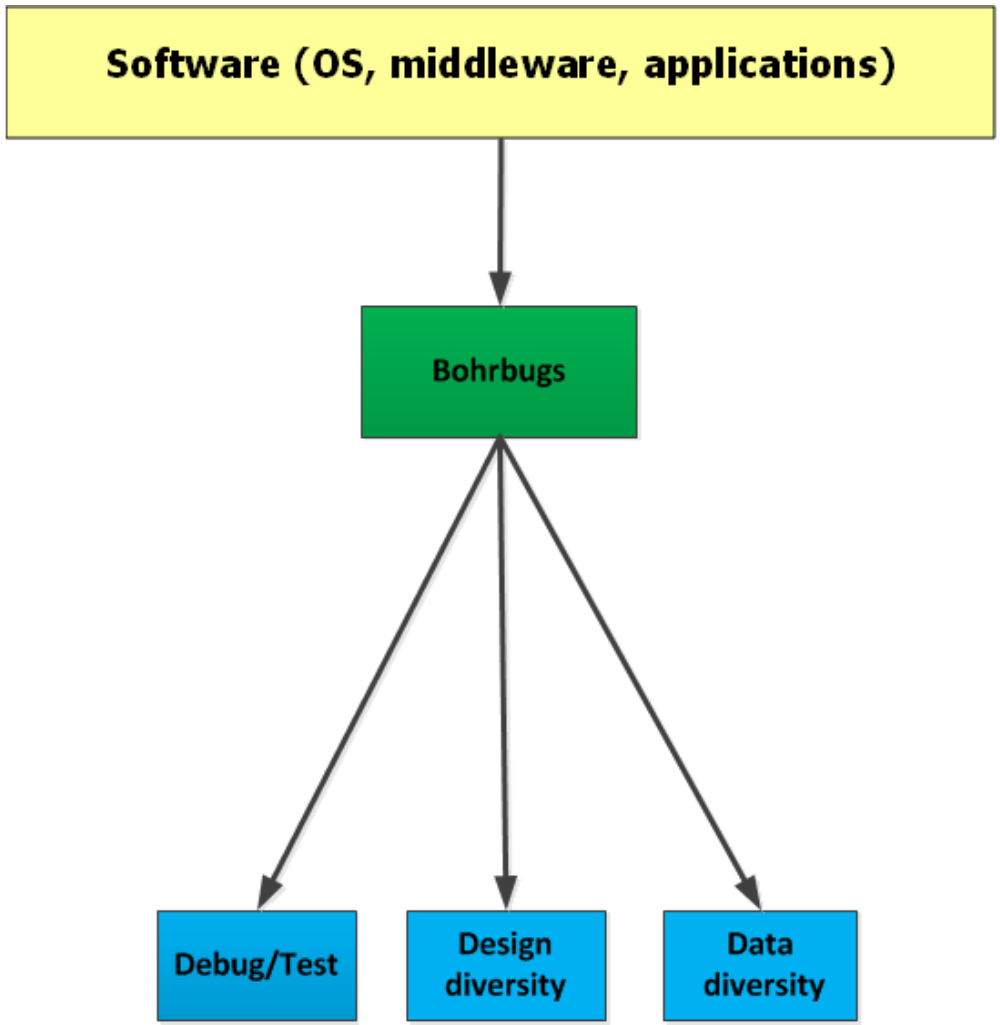
- Motivation
- A Real System
- Software Fault Classification
- Environmental Diversity
- Methods of Mitigation
- Software Aging and Rejuvenation
- Conclusions



Methods of Mitigation

Duke
UNIVERSITY

Mitigation



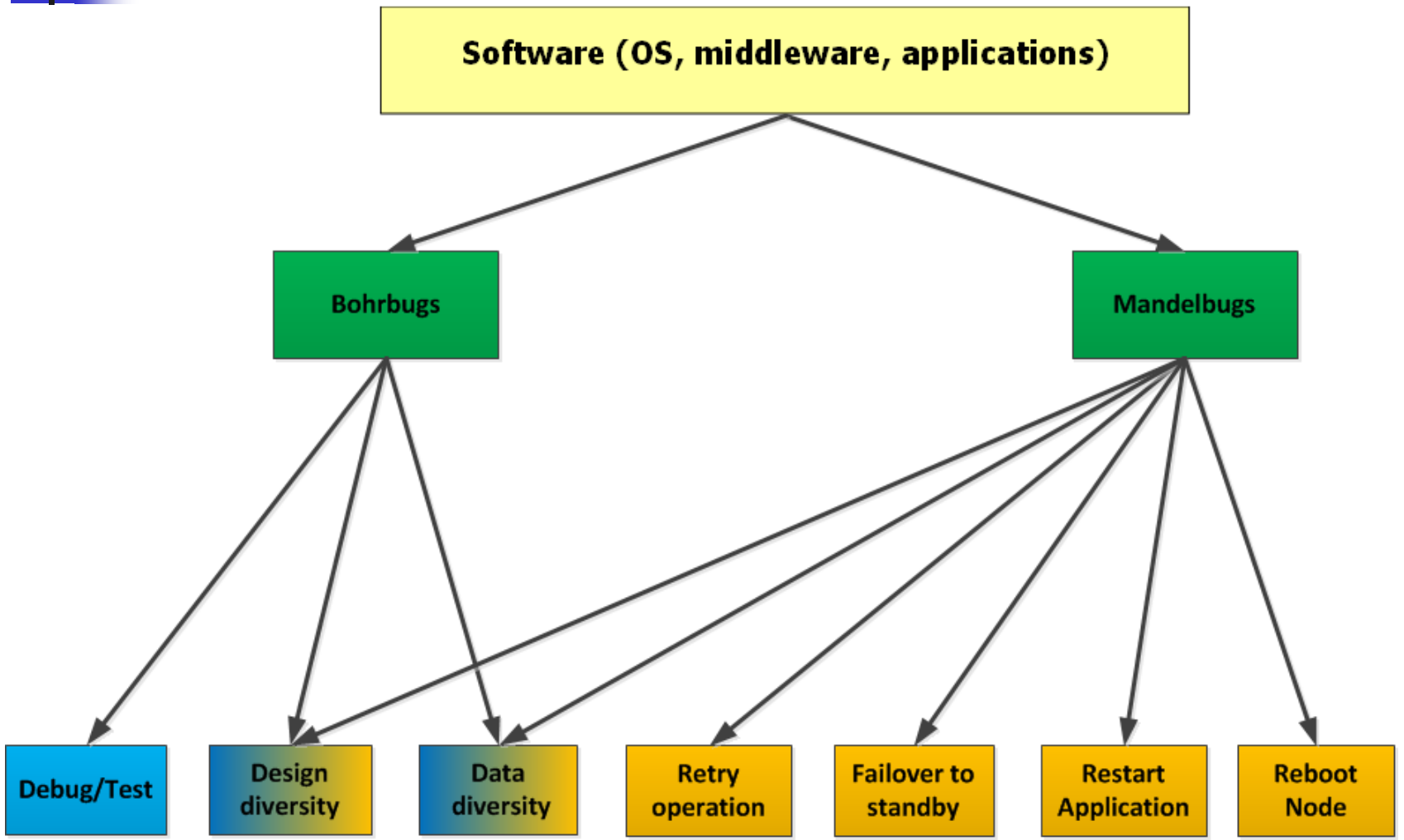
Duke UNIVERSITY

Bohrbugs: Remove

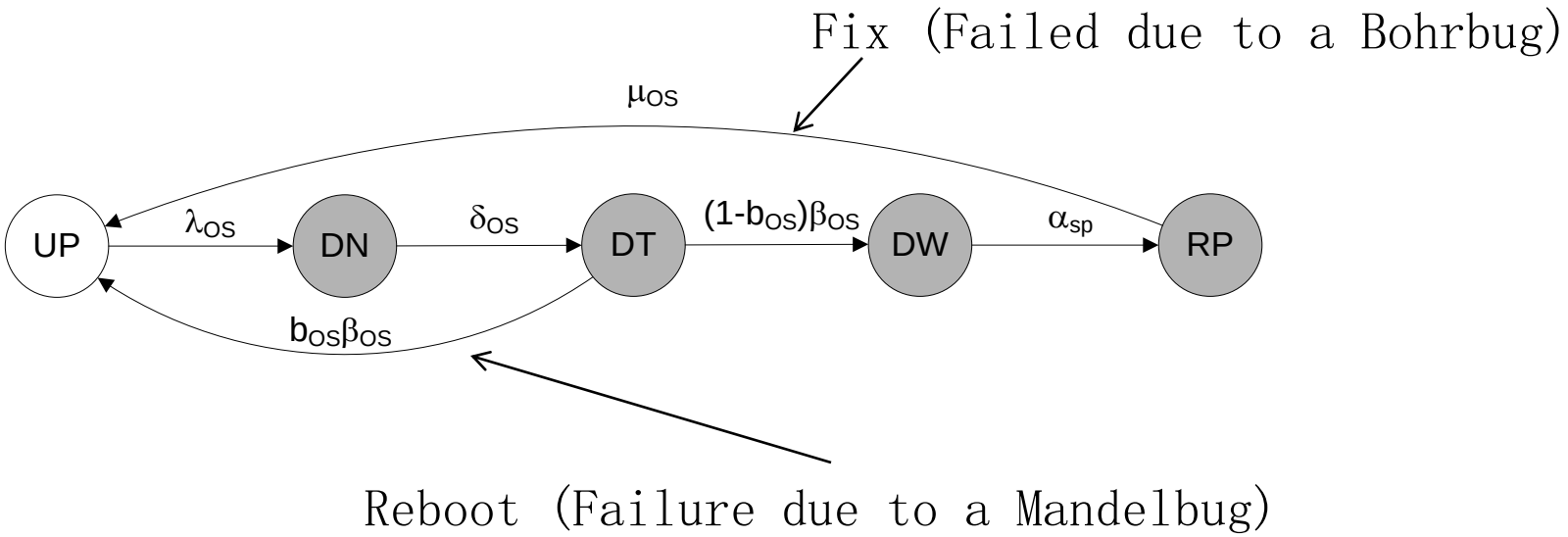
- Find and fix the bugs during testing
- Failure data collected during testing
- Calibrate a *software reliability growth model* (SRGM) using failure data; this model is then used for prediction
- Many SRGMs exist (JM,NHPP,HGRGM, etc.)
  - Books by Lyu, Musa, Cai
  - Gokhale & Trivedi, A Time/Structure Based Software Reliability Model, *Annals of Software Engineering*, 1999
- Measurements → Empirical (*statistical*) models

Duke
UNIVERSITY

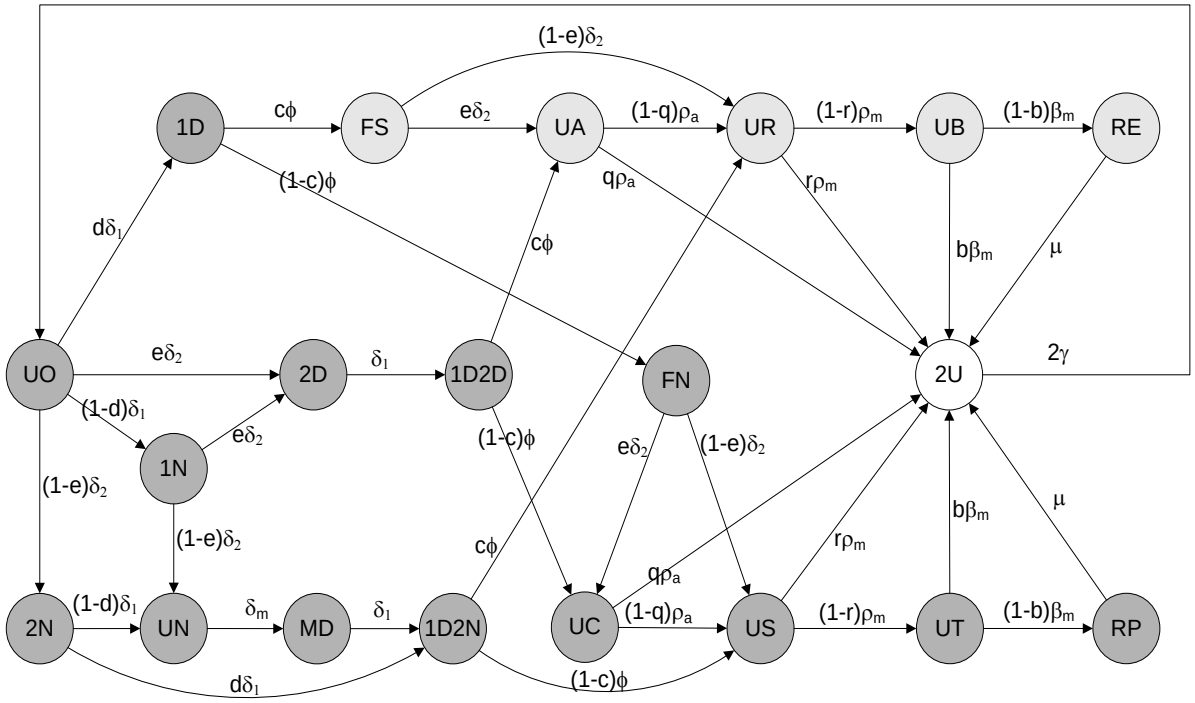
Mitigation



OS Availability Model (IBM BladeCenter)



Markov Availability Model for a Single Replication Domain



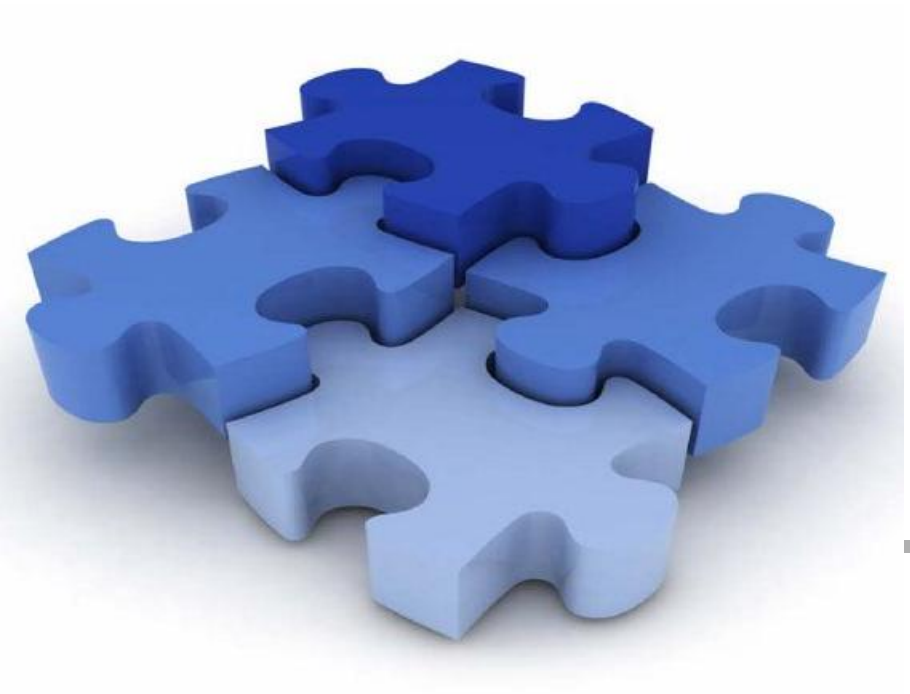
- Failure detection
  - By WLM
  - By Node Agent
  - Manual detection
- Recovery
  - WLM
    - Failover
  - Node Agent
    - Auto process restart
  - Manual recovery
    - Process restart
    - Node reboot
    - Repair

Duke UNIVERSITY

Outline

- Motivation
- A Real System
- Software Fault Classification
- Environmental Diversity
- Methods of Mitigation
- **Software Aging and Rejuvenation**
- Conclusions

Duke
UNIVERSITY



Aging Related Bugs: Replicate, Restart, Reboot, Rejuvenate

Duke
UNIVERSITY

Software Aging

Aging phenomenon

Error conditions accumulating over time

Performance degradation, system failure

Main causes of Software Aging

Memory leak, fragmentation, Unterminated threads, Data corruption,

Round-off errors, Unreleased file-locks, *etc*

Observed system

OS, Middle-ware, Netscape, Internet Explorer *etc*

Duke UNIVERSITY

Software Aging - Definition

"Software Aging" phenomenon

Long-running software tends to show an increasing failure rate.

Not related to application program becoming obsolete due to changing requirements/maintenance.

Software appears to age; no real deterioration

Duke UNIVERSITY

Software aging examples:

Oct. 2012 - Amazon Web Services Outage Caused By Memory Leak And Failure In Monitoring Alarm

Sept. 2011 - Google Docs Outage Blamed on Memory Glitch

Feb. 1991 - The Patriot Missile Software Failure

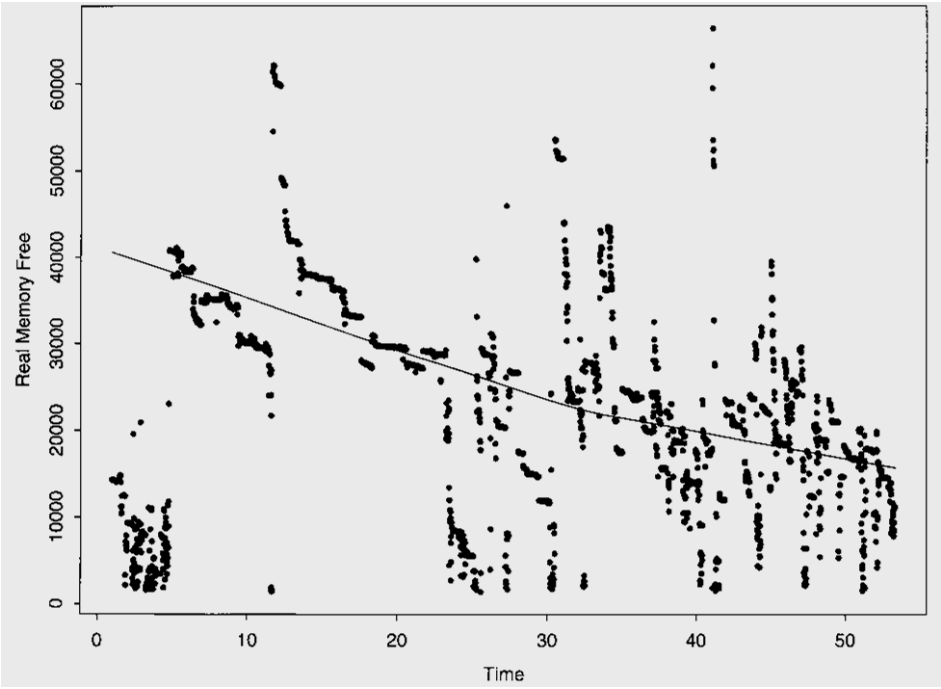
International Space Station (ISS) FC SSC memory leaks problems

Software Aging – More Examples

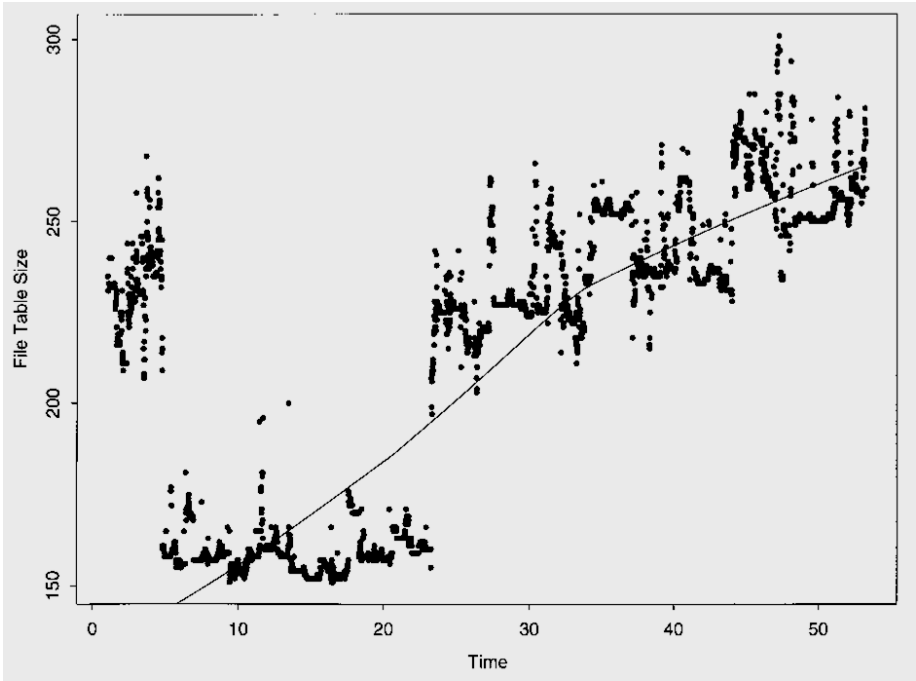
- Cisco Catalyst Switch **[Matias Jr.]**
- File system aging **[Smith & Seltzer]**
- Gradual service degradation in the AT&T transaction processing system **[Avritzer *et al.*]**
- Error accumulation in Patriot missile system's software **[Marshall]**
- Resources exhaustion in Apache **[Li *et al.*, Grottke *et al.*]**
- Physical memory degradation in a SOAP-based Server **[Silva *et al.*]**
- Software aging in Linux **[Cotroneo *et al.*]**
- Crash/hang failures in general purpose applications after a long runtime

Measurements Showing Resource Exhaustion or Depletion

Real Memory Free



File Table Size



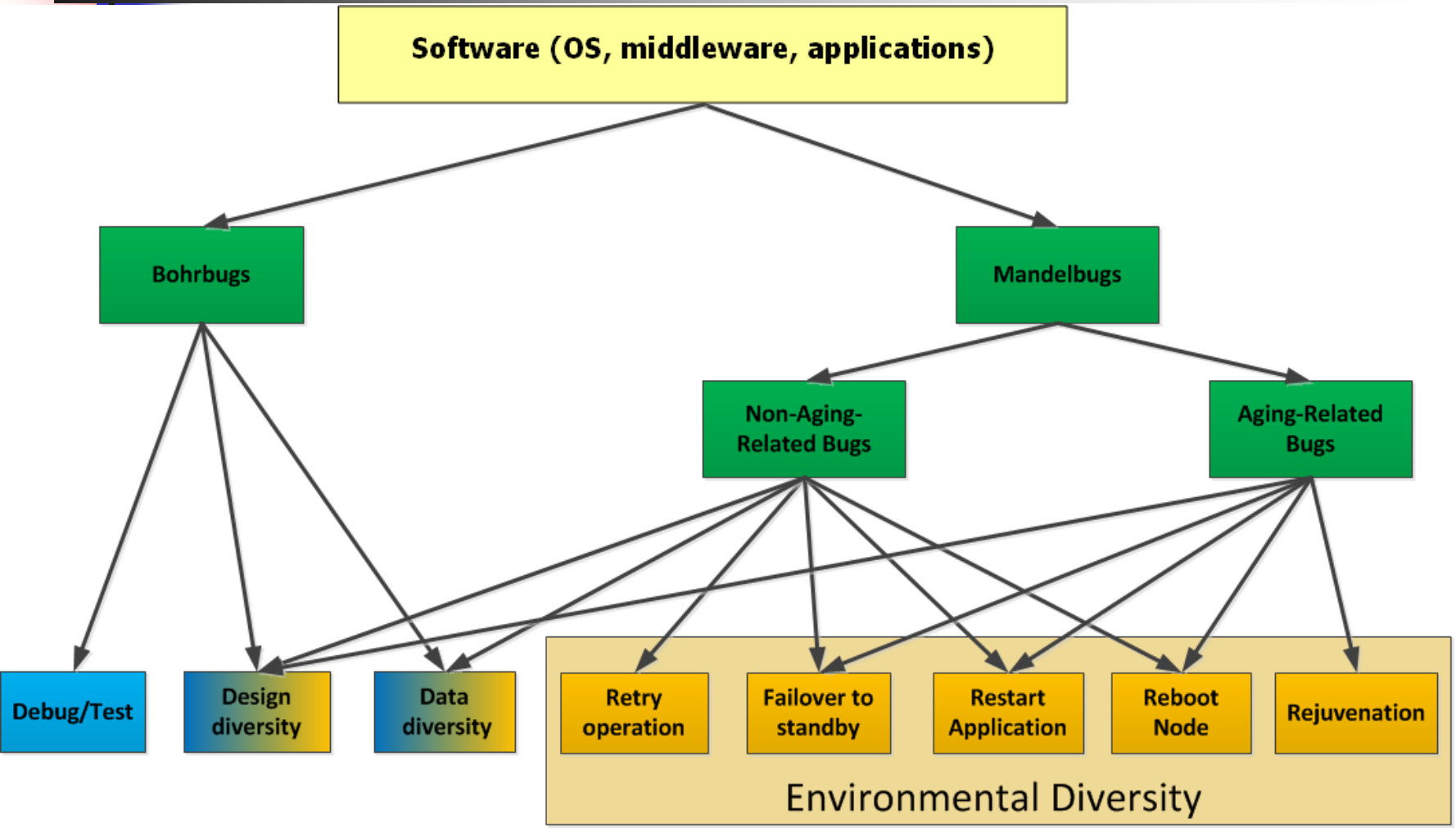
A Methodology for Detection and Estimation of Software Aging,

S. Garg, A. van Moorsel, K. Vaidyanathan and K. Trivedi.

Proc. of IEEE Intl. Symp. on Software Reliability Engineering, Nov. 1998.

Duke UNIVERSITY

Software Fault Types & Their Mitigation



Software rejuvenation

Software rejuvenation is a cost effective solution for improving software reliability by avoiding/postponing unanticipated software failures/crashes.

It allows proactive recovery to be carried either automatically or at the discretion of the user/administrator

Rejuvenation of the environment, not of software

Duke UNIVERSITY

Software rejuvenation examples

- ✓ Patriot missile system software – switch off and on every 8 hours
- ✓ ISS FS SSC (ISS File system) – switch off and on every 2 months
- ✓ Process and connections restart/recycling
- ✓ Tens of US Patents related with this technology

Software Rejuvenation More Examples

- AT&T billing applications **[Huang et al.]**
- On-board preventive maintenance for long-life deep space missions (NASA's X2000 Advanced Flight Systems Program) **[Tai et al.]**
- IBM Director Software Rejuvenation (x-series) **[IBM & Duke Researchers]**

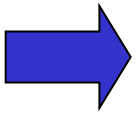
For more examples:

"**Software rejuvenation - Do IT & Telco industries use it?".** Javier Alonso, Antonio Bovenzi, Jinghui Li, Yakun Wang, Stefano Russo, and Kishor Trivedi. The 4rd International Workshop on Software Aging and Rejuvenation (WoSAR 2012) . Held in conjunction with The 23nd annual International Symposium on Software Reliability Engineering (ISSRE 2012), Dallas, USA, 2012.

Duke UNIVERSITY

Software Rejuvenation –Trade-off

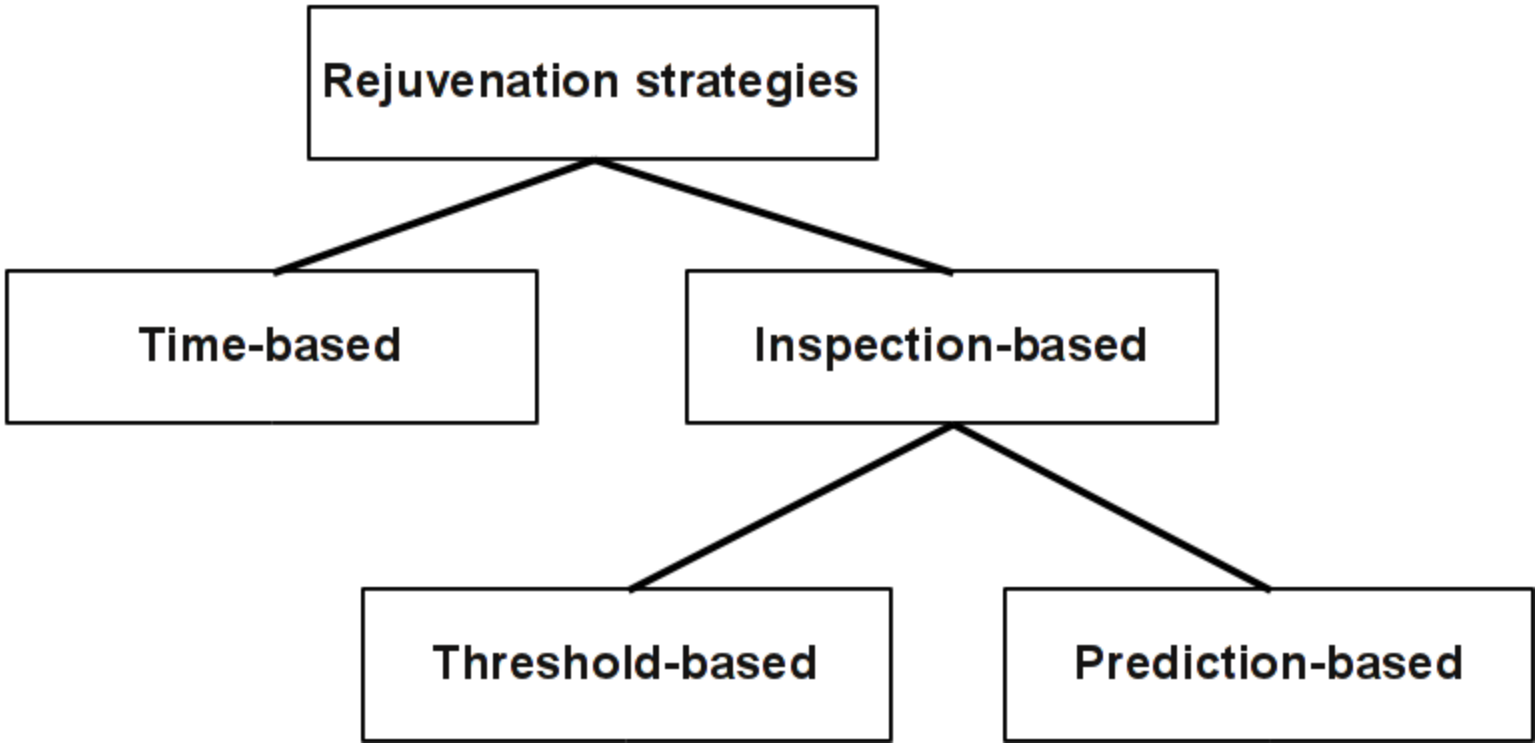
- Advantages
  - Reduces costs of sudden aging-related failures
  - Can be applied at the discretion of the user/administrator
- Disadvantages
  - Direct costs of carrying out rejuvenation
  - Opportunity costs of rejuvenation (downtime, decreased performance, lost transactions etc)



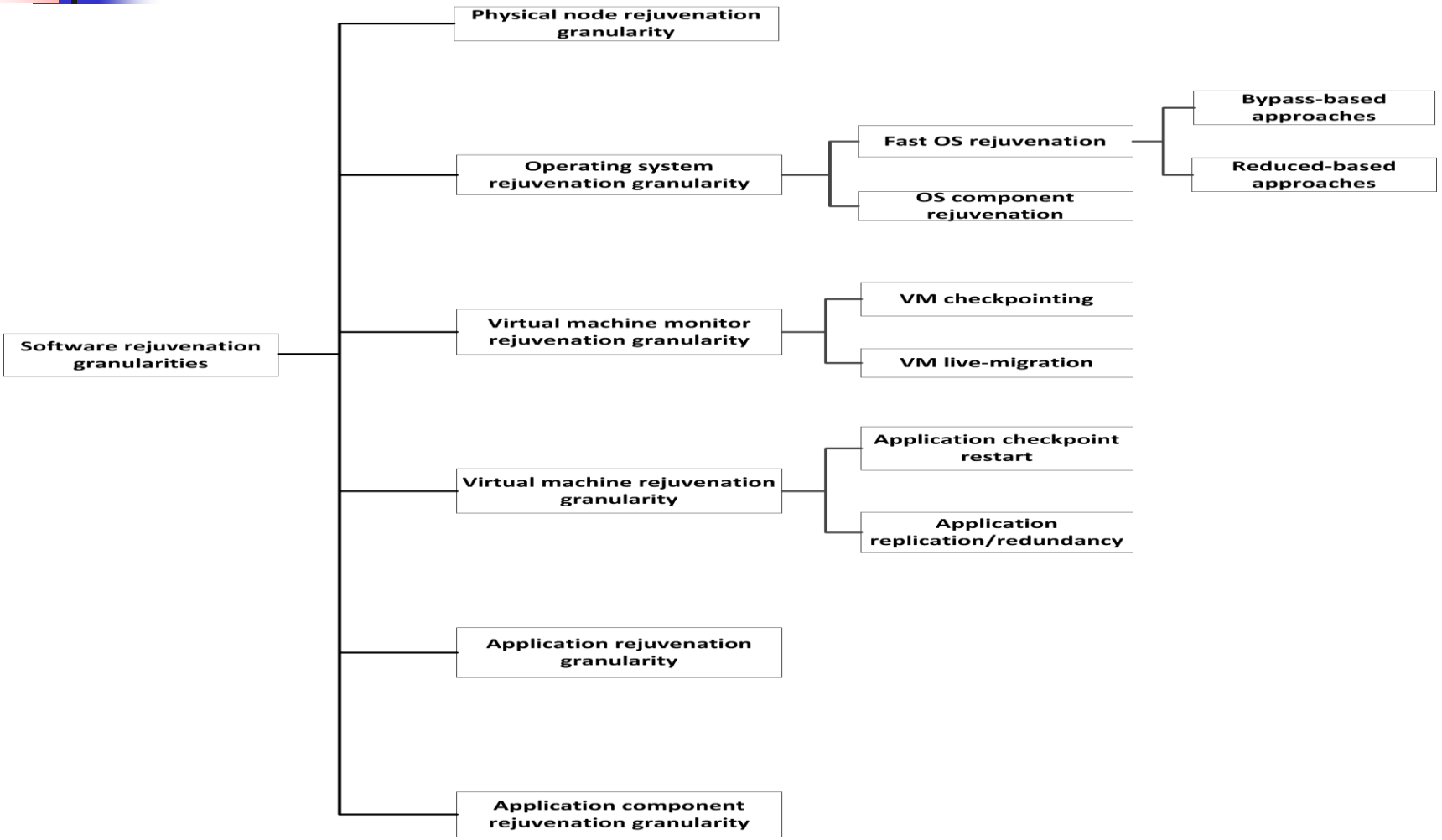
Important research issue:
Find optimal times to perform rejuvenation!

Duke UNIVERSITY

Software rejuvenation



Software Rejuvenation Granularities



IBM xSeries Software Rejuvenation Agent (SRA)

IBM Director system management tool

- Provides GUI to configure SRA
- Acts upon alerts

Two versions

- Periodic rejuvenation
- Prediction-based rejuvenation

Duke UNIVERSITY

Summary

It is possible to enhance software availability during operation exploiting environmental diversity

Multiple types of recovery after a software failure can be judiciously employed: restart, failover to a replica, reboot and if all else fails repair (patch)

Duke
UNIVERSITY

Summary

Software aging not anecdotal – real life scientific phenomenon

Rejuvenation implemented in several special purpose applications and many general purpose cluster systems

Duke
UNIVERSITY

Key References

- **Fighting Bugs: Remove, Retry, Replicate and Rejuvenate**, M. Grottke and K. Trivedi, IEEE Computer, Feb. 2007.
- **Availability Modeling of SIP Protocol on IBM WebSphere**, K. S. Trivedi, D. Wang, D. J. Hunt, A. Rindos, W. E. Smith, B. Vashaw, Proc. PRDC 2008.
- **Using Accelerated Life Tests to Estimate Time to Software Aging Failure,** MATIAS JR, R., TRIVEDI, K., Maciel, P. , ISSRE, 2010.
- **Accelerated Degradation Tests Applied to Software Aging Experiments**, Rivalino Matias, Jr., K. S. Trivedi and Paulo J. F. Filho and Pedro A. Barbetta, IEEE Transactions on Reliability, March 2010.
- **An Empirical Investigation of Fault Types in Space Mission System Software**, M.Grottke, A. P. Nikora and K. S. Trivedi, Proc. DSN, 2010.
- **Software fault mitigation and availability assurance techniques**, K. S. Trivedi, M. Grottke, and E. Andrade. International Journal of System Assurance Engineering and Management, 2011.
- **Recovery from Failures due to Mandelbugs in IT Systems**, K. Trivedi, R. Mansharamani, D.S. Kim, M. Grottke, M. Nambiar , Proc. PRDC 2011
- O. Kyas. (2001). **Network Troubleshooting**, Palo Alto California, Agilent Technologies (book)
- M. Kaaniche and K. Kanoun (1996). **Reliability of a Commercial Telecommunications System**, ISSRE 1996
- R. Cramp, M. A. Vouk, and W. Jones (1992). **On Operational Availability of a Large Software-Based Telecommunications System**, ISSRE 1992

Key References

- **Software Rejuvenation: Analysis, Module and Applications**, Y. Huang, C. Kintala, N. Kolettis and N. Fulton, In Proc. FTCS-25, June 1995.
- **A Methodology for Detection and Estimation of Software Aging**, S. Garg, A. van Moorsel, K. Vaidyanathan and K. S. Trivedi. Proc. ISSRE 1998.
- **Performance and Reliability Evaluation of Passive Replication Schemes in Application Level Fault Tolerance**, S. Garg, Y. Huang, C. M. R. Kintala, K. S. Trivedi, and S. Yajnik. In Proc. FTCS 1999.
- **Statistical Non-Parametric Algorithms to Estimate the Optimal Software Rejuvenation Schedule**, T. Dohi, K. Goseva-Popstojanova and K. S. Trivedi, Proc. PRDC 2000.
- **Proactive Management of Software Aging**, V. Castelli, R. E. Harper, P. Heidelberger, S. W. Hunter, K. S. Trivedi, K. Vaidyanathan and W. Zeggert, IBM Journal of Research & Development, March 2001.
- **A Comprehensive Model for Software Rejuvenation**, K. Vaidyanathan and K. S. Trivedi. IEEE-TDSC, April-June 2005.
- **Analysis of software aging in a web server**, M. Grottke, L. Li, K. Vaidyanathan and K. S. Trivedi, IEEE Trans. Reliability, Sept. 2006.

Duke UNIVERSITY

Key References

- **Software dependability in the Tandem GUARDIAN system**, Lee I., Iyer, R.K., In IEEE Transactions on Software Engineering, vol.21, no.5, pp.455,467, May 1995
- **Whither generic recovery from Application Faults? A fault study using Open-Source Software**, Chandra S., Chen P. M., In Proceedings of the 2000 International Conference on Dependable Systems and Networks