Software Fault Tolerance via Environmental Diversity

Keynote Talk

QRS 2020

Dec. 11, 2020

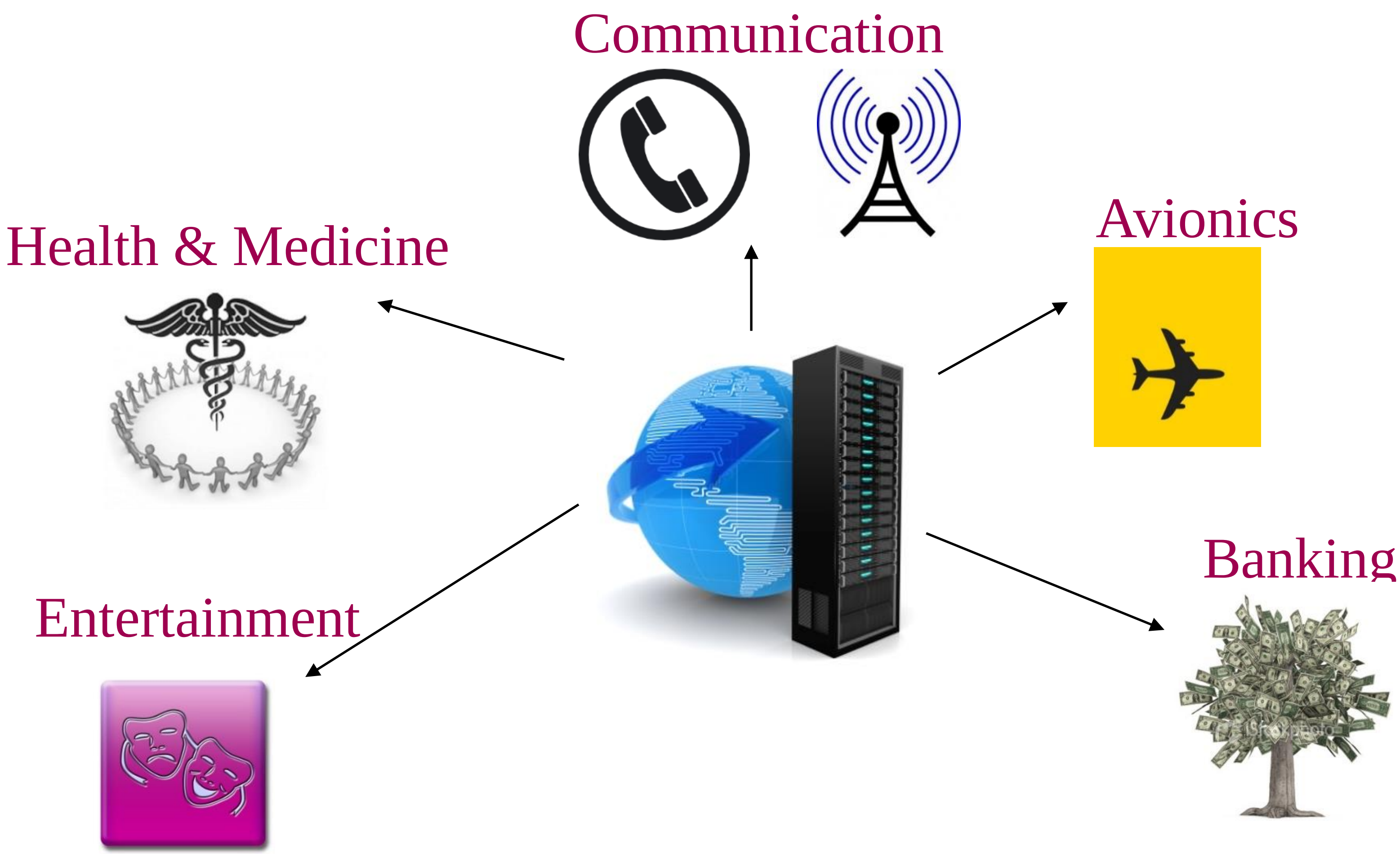
Kishor Trivedi

Department of Electrical and Computer Engineering,
Duke University, Durham, USA

Outline

- Motivation/Definitions
- Real System Examples
- Software Fault Classification
- Environmental Diversity
- Methods of Mitigation
- Conclusions

Pervasive Dependence on Computer Systems Implies the Need for High Reliability/Availability



3

Dependability– An umbrella term

- Laprie: 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 Prevention/Avoidance
    - Fault Removal
    - Fault Tolerance
    - Fault Forecasting
  - Threats
    - Faults
    - Errors
    - Failures

Two of the Attributes of Dependability

- Reliability
  - Continuity of service, how long does system work w/o system failure
- Availability
  - Readiness of service, how frequently it fails and how quickly can it be repaired/restored/recovered

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** (or **bug**) is adjudged or hypothesized cause of an error.

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

.... Fault ⟶ Error ⟶ Failure ....

Example Failures 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 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

These examples indicate that even the most advanced tech companies are not offering high levels of dependability

Duke UNIVERSITY

More Recent Examples

- 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
- Air India’s passenger service system software, which looks after check-in, baggage and reservation, was down for more than 5 hours on April 27, 2019.

Duke UNIVERSITY

Software is a big 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

Ensuring Software Reliability: Known Means

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

Reliable Software

- **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, PRISM)
- **Bug free code not yet possible for large scale software systems**
- **Yet there is a strong need for failure-free system operation**

System outages and software

- *The unstoppable cost increase of software failures*
  - Brokerage $6,450,000 / h
  - Credit card authorization $2,600,000 / h
  - eCommerce $225,000 / h
  - Airline reservation $89,000 / h
  - ...
- Failures must be avoided through rigorous **testing and fault removal** as well as by **fault tolerance** against residual faults

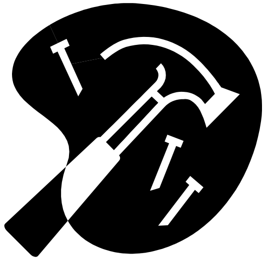
Duke UNIVERSITY

Ensuring Software Reliability: Known Means

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

Reliable Software

- **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(SRGM) to predict when to stop testing



- Impossible to fully test and verify if software is fault-free

"Testing shows the presence, not the absence, of bugs" - E. W. Dijkstra

- Software is still delivered with many bugs either because of inadequate budget for testing, very difficult to reproduce/detect/localize/correct bugs or inadequacy of techniques employed/known

Ensuring Software Reliability: Known Means

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

High Reliability/Availability:

Software is a big problem

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

Software Fault Tolerance:

Classical Techniques

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

Design diversity

Expensive → not used much in practice!

Yet there are stringent requirements for failure-free operation

Challenge: *Affordable* Software Fault Tolerance

A possible answer: Environmental Diversity

TAKE AWAY MESSAGE

- Complex systems (e.g., SDN, CPS, IoT) have a large amount of software. **Software failures are a major cause of undependability**.

- **Software failures during operation are a fact that we need to learn to deal with**. Traditional method of software fault tolerance based on design diversity is expensive and hence does not get used extensively.

- Software fault tolerance based on ***inexpensive environmental diversity* should be exploited**.

- The focus so far has been on **software faults**; we need to pay attention also to **failures** caused by software bugs and the recovery from these failures.

- Or, focus so far has been on software reliability; we need to pay attention to **software availability** as well.

Outline

- Motivation
- A Real System Example
- Software Fault Classification
- Environmental Diversity
- Methods of Mitigation
- Conclusions

REAL SYSTEM: SIP ON WEBSPHERE

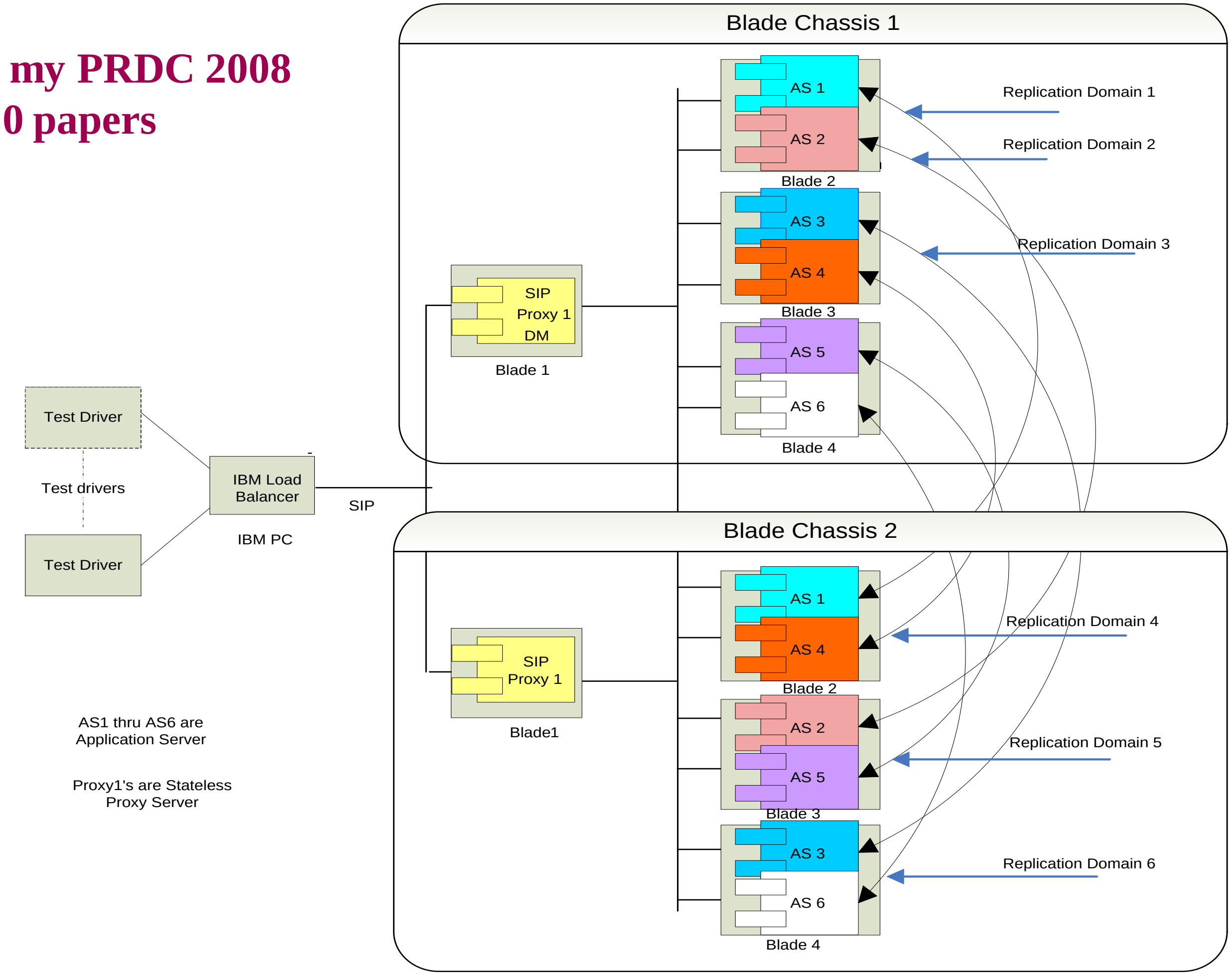
IBM Implementation (around 2007)

Duke
UNIVERSITY

Real System Example

High availability SIP Application Server Configuration on IBM WebSphere

More details in my PRDC 2008 and ISSRE 2010 papers



High availability SIP Application Server Configuration on IBM WebSphere

- Hardware configuration:
  - Two BladeCenter chassis; 4 blades (nodes) on each chassis
  - **1 chassis is sufficient from performance perspective**
- Software configuration:
  - 2 copies of SIP/Proxy servers (**1 sufficient for performance**)
  - 12 copies of WebSphere Application Server (WAS or AS)
  - **6 copies sufficient for performance**
  - Each WAS instance forms a redundancy pair (**replication domain**) with WAS installed on another node on a different chassis
- Fault Tolerance:
  - The system has both **hardware redundancy**
  - and **software redundancy.**

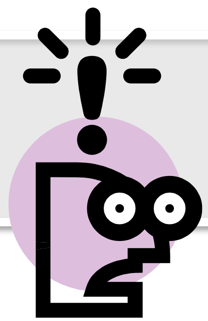
High availability SIP Application Server Configuration on IBM WebSphere

- **Software Redundancy**
  - 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 – uses time redundancy**
    - ✓ Restart software, reboot node or fail-over to a software replica; only when all else fails, a "software repair" is invoked

Software Fault Tolerance: New Thinking

Does it help?

If yes, why?



Thirty years ago this would be considered crazy!

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

Both have the same bugs

Software Fault Tolerance: New Thinking

1. Have been Known to help in dealing with hardware transient faults

2. Do they help in dealing with failures caused by software bugs? Without fixing those bugs?

3. If yes, why?

RETRY

RESTART

REBOOT!

Bugs are not all equal !

- **Fault triggers** make the difference
- Some bugs are "trivial", and failures caused by them can be easily reproduced. So it is relatively easy to remove these bugs
- Others are "subtle", and reproducing the failures caused by these bugs is challenging
  - Concurrency bugs
  - *Race conditions*
  - *Memory leaks*
  - *Hardware-related bugs affecting software*
  - ...
  - These bugs have a significant impact in terms of the number of software failures and the resultant losses

Duke UNIVERSITY

Outline

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

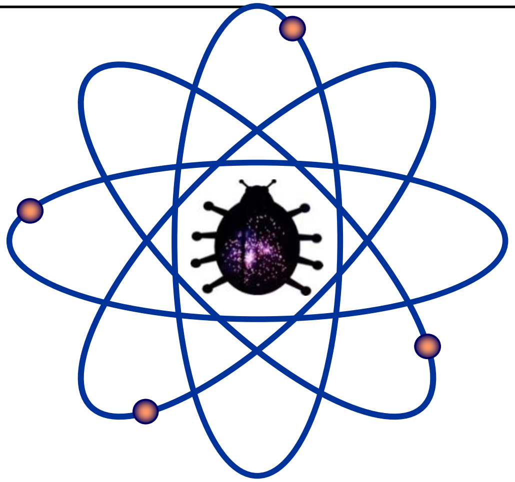
IFIP Working Group 10.4 (Laprie)

- **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** (or **bug**) is adjudged or hypothesized cause of an error

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

…….. **Fault** ⟶ **Error** ⟶ **Failure** ………

A New Classification of Software Faults



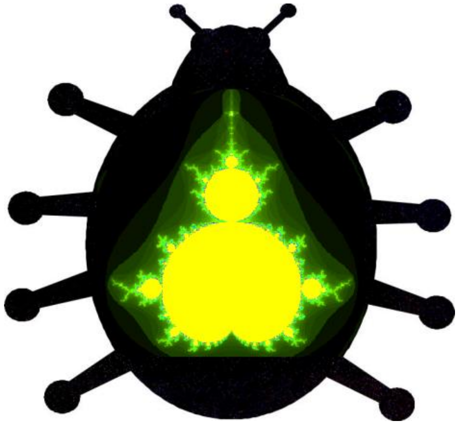
Bohrbug:= A fault that is easily isolated and that manifests *consistently* under a well-defined set of conditions, because its activation and error propagation lack complexity.

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

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

A New Classification of Software Faults

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 bugs or non-deterministic bugs, soft bugs or environment-dependent bugs; failures resulting from these bugs are sometimes called transient failures

Mandelbug Complexity Factors

- Besides workload and internal state of the software system, its system-context (or operating) environment participates in determining whether a failure due to such a bug will occur

- So a fault is a Mandelbug if its manifestation as a failure is subject to the following complexity factors
  - **Long time lag** between fault activation and failure appearance
  - **Operating environment dependence** (OS resources, other applications running concurrently, hardware, network…)
  - **Timing** among submitted operations
  - **Sequencing or ordering** of operations

- A failure due to a Mandelbug thus may not recur upon the resubmission of the same workload if the operating *environment* has changed enough

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.

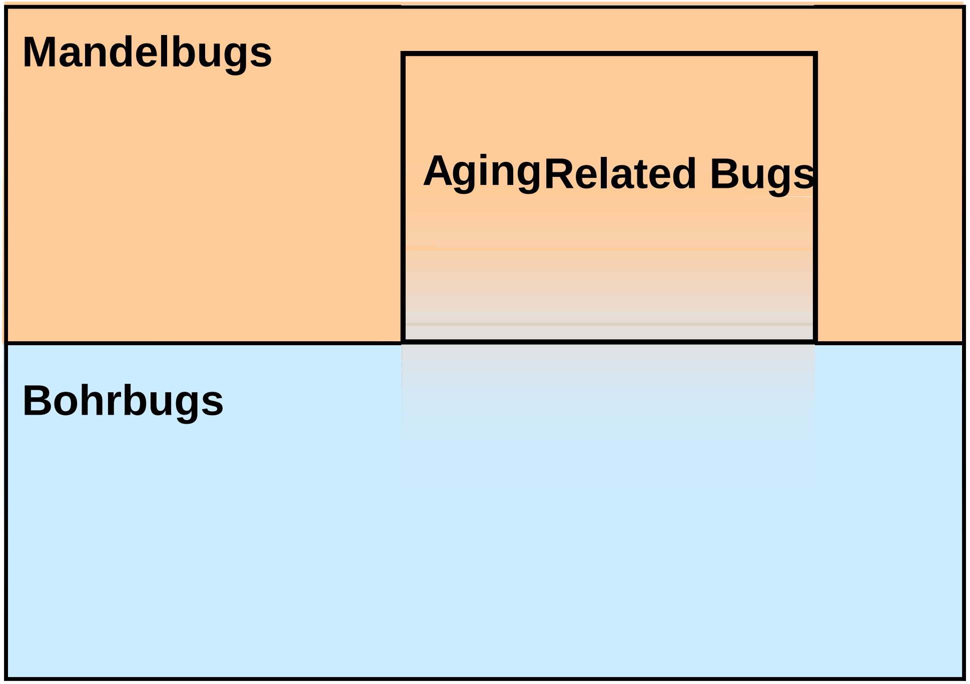
Example:

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

Relationships of the Bug Types

- Bohrbug and Mandelbug are complementary antonyms.

Aging-related bugs are a subtype of Mandelbugs



Dealing with Mandelbugs

- Depending on the bugtype, appropriate strategies are needed
- **Traditional testing** tends to be ineffective for Mandelbugs; more suitable verification strategies are
  - Model checking
  - Combinatorial testing
  - Ratliff, Kuhn, Kacker, Lei & Trivedi, "The Relationship between Software Bug Type and Number of Factors Involved in Failures," *IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), 2016*
- Failures due to Mandelbugs can be **tolerated** by
  - Retrying failed operation, Restarting a process or Rebooting the VM
  - Failover to an identical replica
- Failures due to Aging-related bugs can be prevented by
  - Rejuvenation
  - **Handbook on Software Aging and rejuvenation**, Dohi, Trivedi & Avritzer (eds.), World scientific, 2020

Dealing with Software Failures

- 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 (environment-dependent bugs)** as opposed to the traditional software bugs now known as **Bohrbugs**.

Examples of Mandelbugs in IT Systems

- Mandelbugs in IT Systems: "***Recovery from failures due to Mandelbugs in IT systems**,*" Trivedi, Mansharamani, Kim, Grottke, Nambiar. *PRDC 2011; IEEE TR*, 2016 (Roberto Natella was added co-author for IEEE-TR paper)
- The selected TCS (Tata Consultancy Services) projects ranged across a number of business systems in the banking, financial, government, IT, pharmacy, and telecom sector

Duke UNIVERSITY

Mandelbug “Reproducibility”

- **(Failures due to) Mandelbugs** are really hard to reproduce
  - Conducted a set of experiments to study the environmental factors that affect the reproducibility of Mandelbugs in MySql
    - disk usage,
    - memory occupancy
    - Concurrency level
  - High usage levels of environmental factors increases significantly failure occurrences due to Mandelbugs
- **Reproducibility of Environment-Dependent Software Failures: An Experience Report**, Cavezza, Pietrantuono, Alonso, Russo, Trivedi, *ISSRE*, 2014.

Duke UNIVERSITY

Important Questions about these Bugs

- What fraction of bugs in real software systems are Bohrbugs, Mandelbugs and aging-related bugs
  - How do these fractions vary
    - ✓ over time
    - ✓ over projects, languages, application types,…
  - Need of Real Data

Fault Types in NASA Software

- Bug types in JPL/NASA flight software - "**An empirical investigation of fault types in space mission system software**," Grottke, Nikora, and Trivedi. *DSN,* 2010.
- This papers won the Test of Time Award at DSN 2020

Project	LoC	% BOH	% NAM	% ARB	% UNK
JPL/NASA		61.4	32.1	4.4	2.1

Fault Types in Open-Source Systems

- Bug types in Linux, MySQL, Apache AXIS, HTTPD - "**Fault triggers in open-source software: An experience report**," D. Cotroneo, M. Grottke, R. Natella, R. Pietrantuono, and K. Trivedi. *ISSRE*, 2013.

Project	LoC	% BOH	% NAM	% ARB	% UNK
JPL/NASA		61.4	32.1	4.4	2.1
Linux	1.31M	42.2	41.9	8.3	7.6
MySQL	453K	56.6	30.3	7.7	5.4
HTTPD	145K	81.1	10.5	7.0	1.4
AXIS	80K	92.5	3.5	4.0	0.0

Bug Types in Open-Source Systems: NAM classification

- **LAG**: *there can be a time lag between the activation of the fault and the occurrence of a failure*

- **ENV**: *the activation and/or error propagation is influenced by the interactions of the software application with its system-internal environment*

- **TIM**: *the activation and/or error propagation is influenced by the timing of inputs and operations*

- **SEQ**: *the activation and/or error propagation is influenced by the sequencing (i.e., the relative order) of operations*

Duke UNIVERSITY

Bug Types in Open-Source Systems: ARB classification

- **MEM**: ARBs causing the accumulation of errors related to memory management
- **STO**: ARBs causing the accumulation of errors that affect disk storage space
- **LOG**: ARBs causing leaks of “other logical resources”, that is, system-dependent data structures
- **NUM**: ARBs causing the accumulation of numerical errors
- **TOT**: ARBs in which the increase of the fault activation/error propagation rate with the total system run time is not caused by the accumulation of internal error states

Examples of ARB/NAM

Project	Type	Description
MySQL	NAM/ SEQ	“if you ‘alter table .. rename to ..’ on a table that has an active transaction open and UNIV DEBUG is defined, mysqld crashes”
Linux	NAM/ LAG	”[The e1000 network driver at suspend/resume does not] explicitly free and allocate irq [...] Restarting the network solved the problem”
HTTPD	NAM/ ENV	“The error only occurs intermittently [...] It behaves as if requests are being distributed (via round-robin or the like) and handled sometimes by a worker thread that is not properly initialized”
Axis	ARB/ MEM	“Strings and char[]s are being leaked”
Linux	ARB/ LOG	“In 2.6.35 and earlier, shutdown(2) will fully remove a socket. This does not appear to be true any more and is causing software to misbehave.”
HTTPD	ARB/S TO	“Apache child processes will die trying to write logs which have reached 2GB in size.”

Duke UNIVERSITY

Fault Types in Android

- Bug types in Android operating system- "**An Empirical investigation of fault triggers in Android operating system**," F. Qin, Z. Zheng, X. Li, Y. Qiao, and K. Trivedi. *PRDC*, 2017.

Project	LoC	% BOH	% NAM	% ARB	% UNK
JPL/NASA		61.4	32.1	4.4	2.1
Android		65.2	27.0	4.4	3.4

Example of Mandelbug in Android

- **ENV:** On certain Android devices, performing the following operations in sequence could lead to a crash

Open camera → Set flash ON → Take a picture → Set flash OFF → Take another picture.

(Caused by Environments)

Duke UNIVERSITY

Fault Types in Linux Revisited

- Bug types in Linux - "**Fault Triggers in Linux Operating System: From Evolution Perspective**," G. Xiao, Z. Zheng, B. Yin, and K. Trivedi. *ISSRE*, 2017 (all the bug reports in Linux)

Project	LoC	% BOH	% NAM	% ARB	% UNK
JPL/NASA		61.4	32.1	4.4	2.1
Linux2		55.8	31.7	7.8	4.7

Fault Types in Several Systems

- Bug types in JPL/NASA flight software - "**An empirical investigation of fault types in space mission system software**," M. Grottke, A. Nikora, and K. Trivedi. *DSN*, 2010.
- Bug types in Linux, MySQL, Apache AXIS, HTTPD - "**Fault triggers in open-source software: An experience report**," D. Cotroneo, M. Grottke, R. Natella, R. Pietrantuono, and K. Trivedi. *ISSRE*, 2013.
- Bug types in Android operating system - "**An Empirical investigation of fault triggers in Android operating system**," F. Qin, Z. Zheng, X. Li, Y. Qiao, and K. Trivedi. *PRDC*, 2017.
- Bug types in Linux - "**Fault Triggers in Linux Operating System: From Evolution Perspective**," G. Xiao, Z. Zheng, B. Yin, and K. Trivedi. ISSRE, 2017 (all the bug reports in Linux)

Project	LoC	% BOH	% NAM	% ARB	% UNK
JPL/NASA		61.4	32.1	4.4	2.1
Linux	1.31M	42.2	41.9	8.3	7.6
MySQL	453K	56.6	30.3	7.7	5.4
HTTPD	145K	81.1	10.5	7.0	1.4
AXIS	80K	92.5	3.5	4.0	0.0
Android		65.2	27.0	4.4	3.4
Linux2		55.8	31.7	7.8	4.7

Software Faults and Mitigation Types

- The fault classification is not only theoretical, it has also practical implications

- Each type of software fault may require different type of approach during development, testing, as well as during operations

Outline

- Motivation
- Real System Examples
- Software Fault Classification
- Environmental Diversity
- Methods of Mitigation
- Conclusions

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**) may well work since failures due to **Mandelbugs** are not negligible. We thus have an affordable software fault tolerance technique that we call **Environmental Diversity**

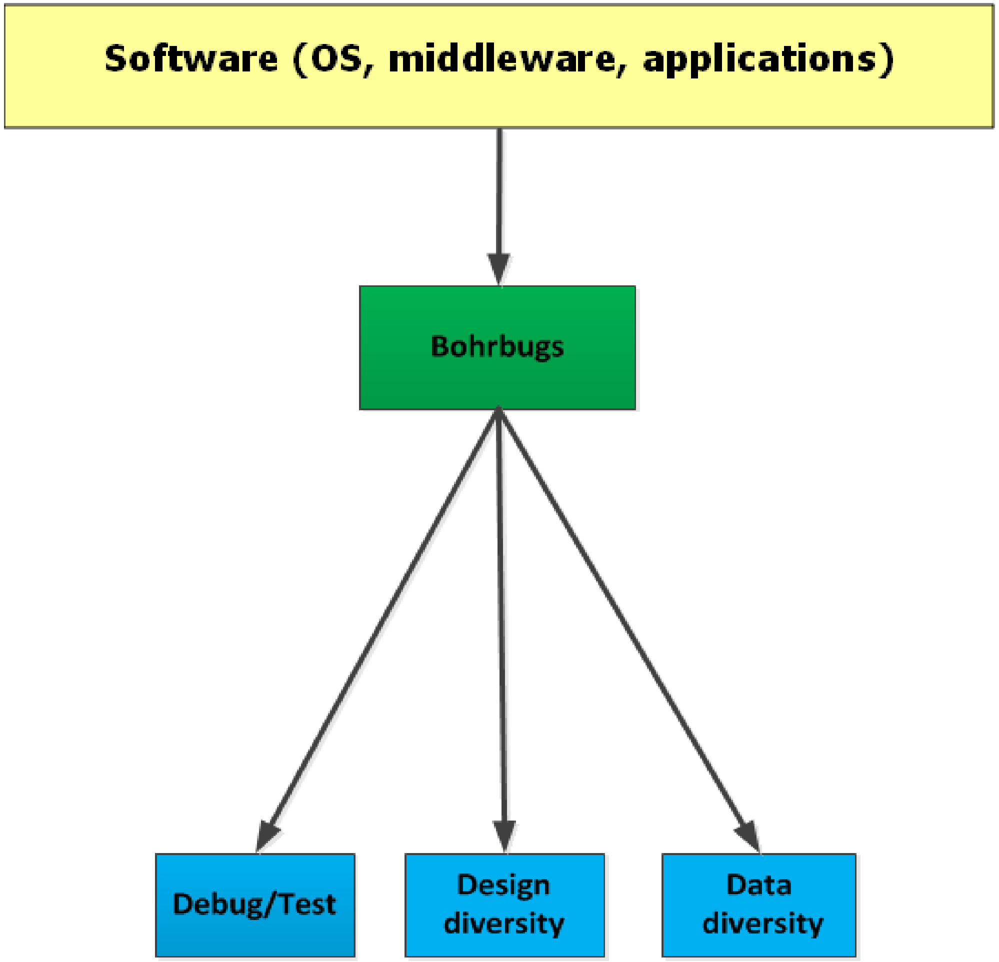
What is Environmental diversity?

- The underlying idea of Environmental diversity
  - Restart an application (without fixing the bus) and it most likely works -- Why?
  - because of the environment where the application is 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.
- This is Fault Tolerance since we do not necessarily fix the fault; fault caused a failure but this failure is dealt with by using time redundancy hence the user may not experience the failure again on retry

Outline

- Motivation
- Real System Examples
- Software Fault Classification
- Environmental Diversity
- Methods of Mitigation
- Conclusions

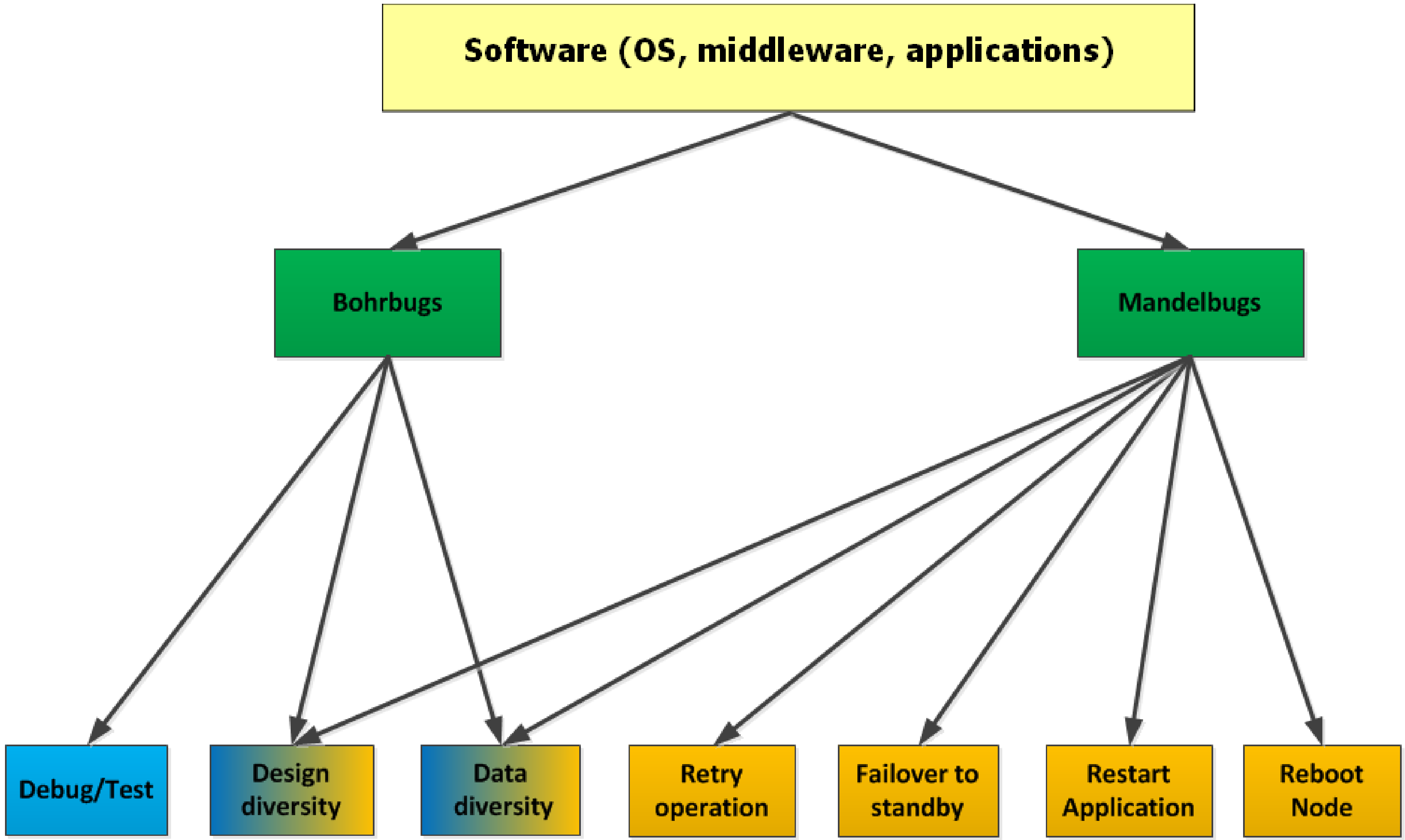
Bohrbugs: Remove



Bohrbug: 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
  - Books by Lyu, Musa and several others

Methods of Mitigation: Mandelbugs



Implications of Mandelbugs

- Can measure/model software ***availability***
- Combined of software and hardware availability
- Need:
  - Develop methods of debugging and testing for environment-dependent bugs
  - Methods to determine environmental factors and their effects
  - Run-time control of environmental factors to avoid failure occurrences
  - Optimal recovery sequence after failure occurrence
  - Experimental methods to determine the nature software failure times including use of ALT

Duke UNIVERSITY

Determine Environmental Factors

- **There are two steps:**
  - Step 1: List all the possible environmental factors.
  - Step 2: Determine the critical environmental factors that can affect the times to failure through:
    - Either ***logically***, according to the failure mechanism
    - Or by ***experimental method***, need Design of Experiments.

Determine Environmental Factors Step 1

- **Five Categories of Candidate Environmental Factors:**
  - **Hardware resources**
    - Physical memory, CPU, disk, network, I/O devices, buses, etc.
    - Connected firmware
  - **Operating System kernel's subsystems**
    - OS memory management, device drivers, file-system, networking, process management, etc.
  - **Concurrent software**
    - Utility software, daemon processes, etc.
    - Application-level interacting software, middleware, etc.
  - **Interfaces**
    - Third-party library, open-source library, etc.
  - **Others**

Duke UNIVERSITY

Determine Environmental Factors Step 2

- **Using logic.** Examples as follows:
  - **Data Race problem.** According to the OS theory, **smaller the physical Resident-Set-Size memory, larger the number of concurrent users, and larger the number of CPU cores,** larger the context switch frequency among threads; thereby increasing the race's activation process. These three environmental factors are therefore critical ones. Details are in [Kun et al. TR 2019].

Qiu, Zheng, **Trivedi**, et al. *Stress Testing With Influencing Factors to Accelerate Data Race Software Failures.* IEEE T. Reliab., 2019.

Duke UNIVERSITY

Outline

- Motivation
- A Real System Example
- Software Fault Classification
- Environmental Diversity
- Methods of Mitigation
- Conclusions

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)

TAKE AWAY MESSAGE

- Complex systems (e.g., SDN, CPS, IoT) have a large amount of software. **Software failures are a major cause of undependability**.

- **Software failures during operation are a fact that we need to learn to deal with**. Traditional method of software fault tolerance based on design diversity is expensive and hence does not get used extensively.

- Software fault tolerance based on ***inexpensive environmental diversity* should be exploited**.

- The focus so far has been on **software faults**; we need to pay attention also to **failures** caused by software bugs and the recovery from these failures.

- Or, focus so far has been on software reliability; we need to pay attention to **software availability** as well.

Thank you for your attention

Kishor Trivedi
Dept. of Electrical & Computer Engineering
Duke High Availability Assurance Lab (DHAAL)
Duke University
ktrivedi@duke.edu
www.ee.duke.edu/~ktrivedi

Key References

Motivation

- **On Operational Availability of a Large Software-Based Telecommunications System**, R. Cramp, M. A. Vouk, and W. Jones, Proc. ISSRE 1992.
- **Software dependability in the Tandem GUARDIAN system**, I. Lee, R. Lyer,, IEEE Transactions on Software Engineering, 1995.
- **Reliability of a Commercial Telecommunications System**, M. Kaaniche and K. Kanoun, Proc. ISSRE 1996.
- **Network Troubleshooting**, O. Kyas, Agilent Technologies , 2001
- **Lessons Learned From the Analysis of System Failures at Petascale: The Case of Blue Waters**, C. D. Martino, F. Baccanico, J. Fullop, W. Kramer, Z. Kalbaczyk, and R. Lyer, Proc. DSN 2014.

Key References

Real System

- **Availability Modeling of SIP Protocol on IBM WebSphere**, K. S. Trivedi, D. Wang, D. J. Hunt, A. Rindos, W. E. Smith, and B. Vashaw, Proc. PRDC 2008.
- **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. Proc. FTCS 1999.

Fault Classification

- **Fighting Bugs: Remove, Retry, Replicate and Rejuvenate**, M. Grottke and K. Trivedi, IEEE Computer, 2007.
- **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 – IEEE TR 2015
- R. Chillarege, et al. "**Orthogonal Defect Classification – A Concept for In-process Measurements**", IEEE Trans. on Software Engineering, 1992
- F. Frattini, R. Ghosh, M. Cinque, A. Rindos, and K. Trivedi. "**Analysis of Bugs in Apache Virtual Computing Lab**", IEEE/IFIP DSN Workshop, 2013.
- **An Empirical investigation of fault triggers in Android operating system Android operating system,** F. Qin, Z. Zheng, X. Li, Y. Qiao, K. S. Trivedi, PRDC 2017;
- **Fault Triggers in Linux Operating System: From Evolution Perspective** , G. Xiao, Z. Zheng, B. Yin, K. S. Trivedi", ISSRE 2017.

Key References

Environmental Diversity and Methods of Mitigation

- **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. Proc. FTCS 1999.
- **Whither generic recovery from Application Faults? A fault study using Open-Source Software**, Chandra S., Chen P. M., Proc. CDSN 2000.
- Vibhu S. Sharma, Kishor S. Trivedi, "**Reliability and Performance of Component Based Software Systems with Restarts, Retries, Reboots and Repairs**", 17th Int. Symp. on Software Reliability Engineering (ISSRE 2006)
- **An Empirical Investigation of Fault Repairs and Mitigations in Space Mission System Software** J. Alonso, M. Grottke, A. Nikora, and K. Trivedi. Proc. DSN 2013.
- **Software fault mitigation and availability assurance techniques**, K. 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
- **Fault triggers in open-source software: An experience report**, Cotroneo, Grottke, Natella, Pietrantuono, Trivedi, ISSRE 2013.
- **Reproducibility of environment-dependent software failures: an experience report**, Cavezza, Pietrantuono, Alonso, Russo, Trivedi , ISSRE 2014.
- **Understanding the impacts of influencing factors on time to a datarace software failures**, K. Qiu, Z. Zheng, K.S. Trivedi, B. Yin, ISSRE 2017.
- **Stress Testing With Influencing Factors to Accelerate Data Race Software Failures**, K. Qiu, Z. Zheng, K.S. Trivedi, B. Yin, TR 2020.
- **Availability Analysis of Systems Deploying Sequences of Environmental-Diversity-Based Recovery Methods**, K. Qiu, Z. Zheng, K.S. Trivedi, I. Mura, TR 2020.

Extra Slides if Needed

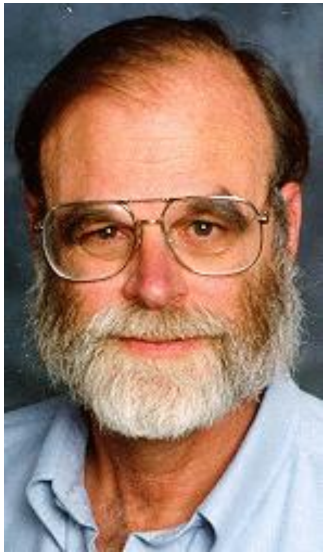
- To answer questions about Heisenbugs vs. Mandelbugs

Jim Gray's Definitions

- The terms "Bohrbug" and "Heisenbug" were first used in print by Jim Gray in 1985.

- "Bohrbugs, like the Bohr atom, are solid, easily detected by standard techniques, and hence boring."

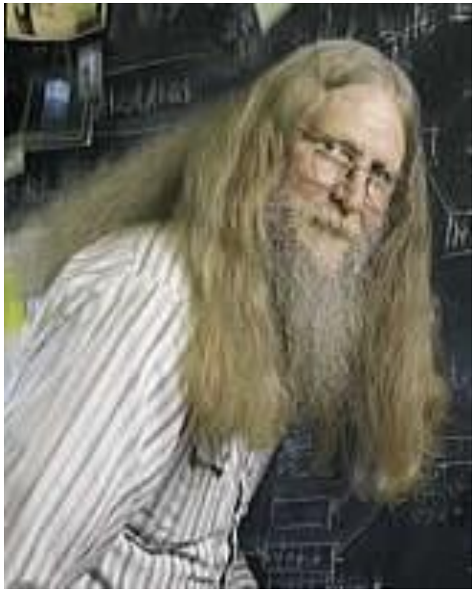
- "Most production software faults are soft. If the program state is reinitialized and the failed operation is retried, the operation will not fail a second time. … The assertion that most production software bugs are soft – Heisenbugs that go away when you look at them – is well known to systems programmers." (Gray, 1985)



J. Gray

Bruce Lindsay's Definition

- Based on Gray's paper, researchers have often equated Heisenbugs with soft faults.
- However, when Bruce Lindsay originally coined the term in the 1960s (while working with Jim Gray), he had a more narrow definition in mind.
- "Heisenbugs as originally defined … are bugs in which clearly the system behavior is incorrect, and when you try to look to see why it's incorrect, the problem goes away." (Lindsay, 2004)
- The term alludes to the physicist Werner Heisenberg and his Uncertainty Principle.



B. Lindsay, photo by T. Upton

Heisenbug – Our Definition

Heisenbug := A fault that stops causing a failure or that manifests differently when one attempts to probe or isolate it.

- How can probing affect the bug?
  - Some debuggers initialize unused memory to default values, thus preventing failures due to improper initialization.
  - Trying to investigate a failure can influence process scheduling in such a way that a scheduling-related failure does not occur again.