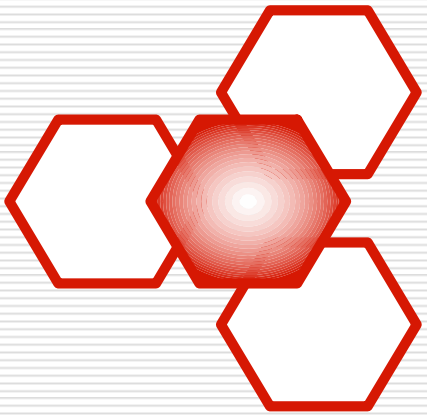


mVerify™

A Million Users in a Box ™

ISSRE 2008 Trip Report

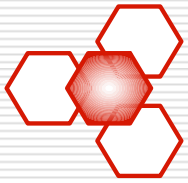
Presented to the

Chicago Software Process Improvement Network (CSPIN)

January 7, 2009

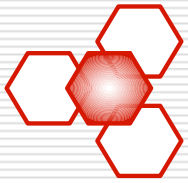
Robert V. Binder bob_binder@mverify.com

www.mVerify.com



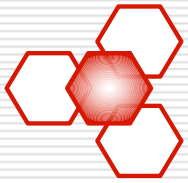
Overview

- Software Reliability Engineering (SRE) concepts
- Case Study, SRE applied to a Trading System
- ISSRE 2008 Highlights
- Applying SRE
- Q & A



What is SRE?

- Basic Concepts
- Predictive Reliability Modeling
- Profile-based Testing
- Release Management
- Corollary Techniques

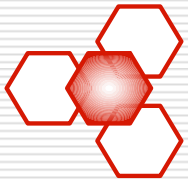


Musa's Observation

Testing driven by an operational profile is very efficient because it identifies failures (and hence the faults causing them) on average, in order of how often they occur.

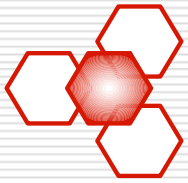
This approach rapidly increases reliability … because the failures that occur most frequently are caused by the faulty operations used most frequently.

IEEE Software, March 1993



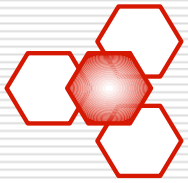
Behavioral Focus

- Can a buggy system be reliable?
  - *It depends on how it is used*
- Operational Profile
  - A list of interactions, similar to "use cases"
  - Relative frequency assigned to each operation
    - Three ops: Op1: 60%, Op2, 30%, Op3, 10%.
    - Represents expected field usage
  - For a good reliability estimate
    - Must include all field operations
    - Accurate *rank order* critical, magnitude less so
- Test cases generated/applied accordingly
  - Optimal allocation of testing resources, wrt reliability
  - High criticality operations tested separately
  - Can overlay most system testing process/technology



Reliability Arithmetic

- Reliability: probability of non-failure
- MTTR: mean time to
  - recover, repair, restart …
- Availability: percent up-time
  - Availability = 1 / 1 + (MTTR × Reliability)
  - 99.999% availability = 5 min unscheduled downtime per year
  - “Five Nines”



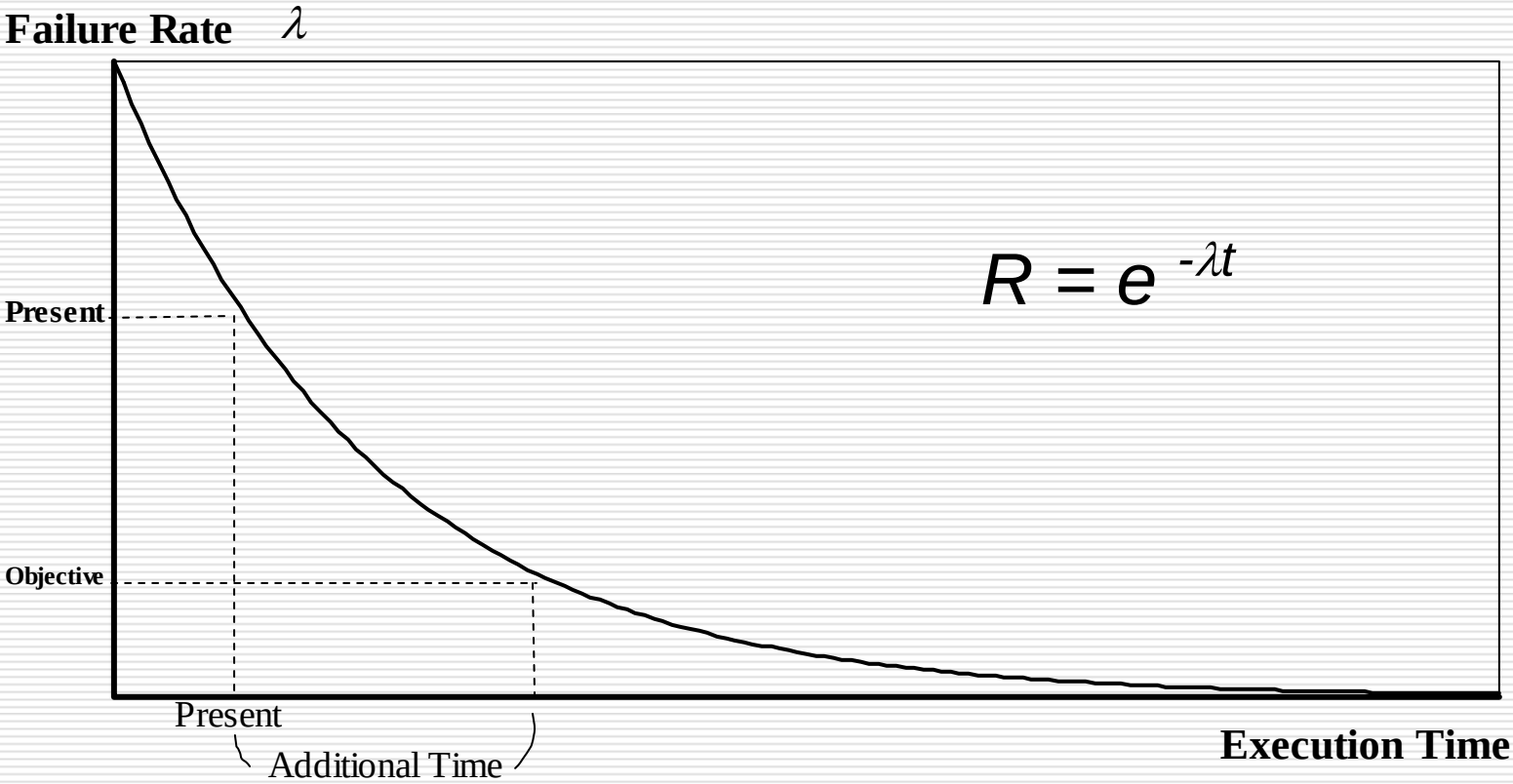
Some Reliability Data Points

	Reliability (Failures/million hours)	Availability, 6 min MTTR
NT 4.0 Desktop	82,000	0.999000000
Windows 2K Server	36,013	0.999640000
Common Light Bulb	1,000	0.999990000
Stepstone OO Framework	5	0.999999500
Telelabs Digital Cross Connect	3	0.999999842

Predictive Reliability Modeling

- Input
  - Observed number of failures
  - Elapsed time or Execution units
- Model
  - Curve fitting
  - Extensive research and application
  - Many variations
- Output
  - CASRE tool
  - Estimated Field Failure Intensity (rate), Reliability

Reliability: Failures versus Time

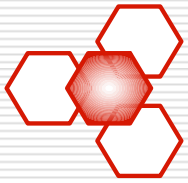


Lyu. *Software Reliability Engineering: A Roadmap*

Many Models

Data Requirements and Assumptions	Relative Importance	Geometric	Jelinski-Moranda	Littlewood-Verrall	Musa basic	Musa-Okumoto	Goel-Okumoto	Shick-Wolverton	Schneidewind	Yamada S-shaped	Duane
Data may be exact failure times (ungrouped data)	2	x	x	x	x	x	x	x	x	x	x
Data may be grouped failure times (interval count data)	2	x	x	x	x	x	x	x	x	x	x
Testing intervals may be of different length	3	x	x	x	x	x	x	x		x	x
Failures need not occur equally likely	2	x		x		x	x		x	x	x
Detection of faults may be dependent of each other	2			x	x	x	x		x	x	x
Failures need not be of the same severity	1			x	x	x	x		x	x	x
Detection rate depends on time (testing effort)	3			x	x	x	x		x	x	x
Detection rate depends on number of remaining defects	3	x	x					x			
Failures need not be repaired instantaneously	3		x		x		x	x	x	x	x
Imperfect repair of defects allowed	2										
Infinite number of errors allowed	2					x					x

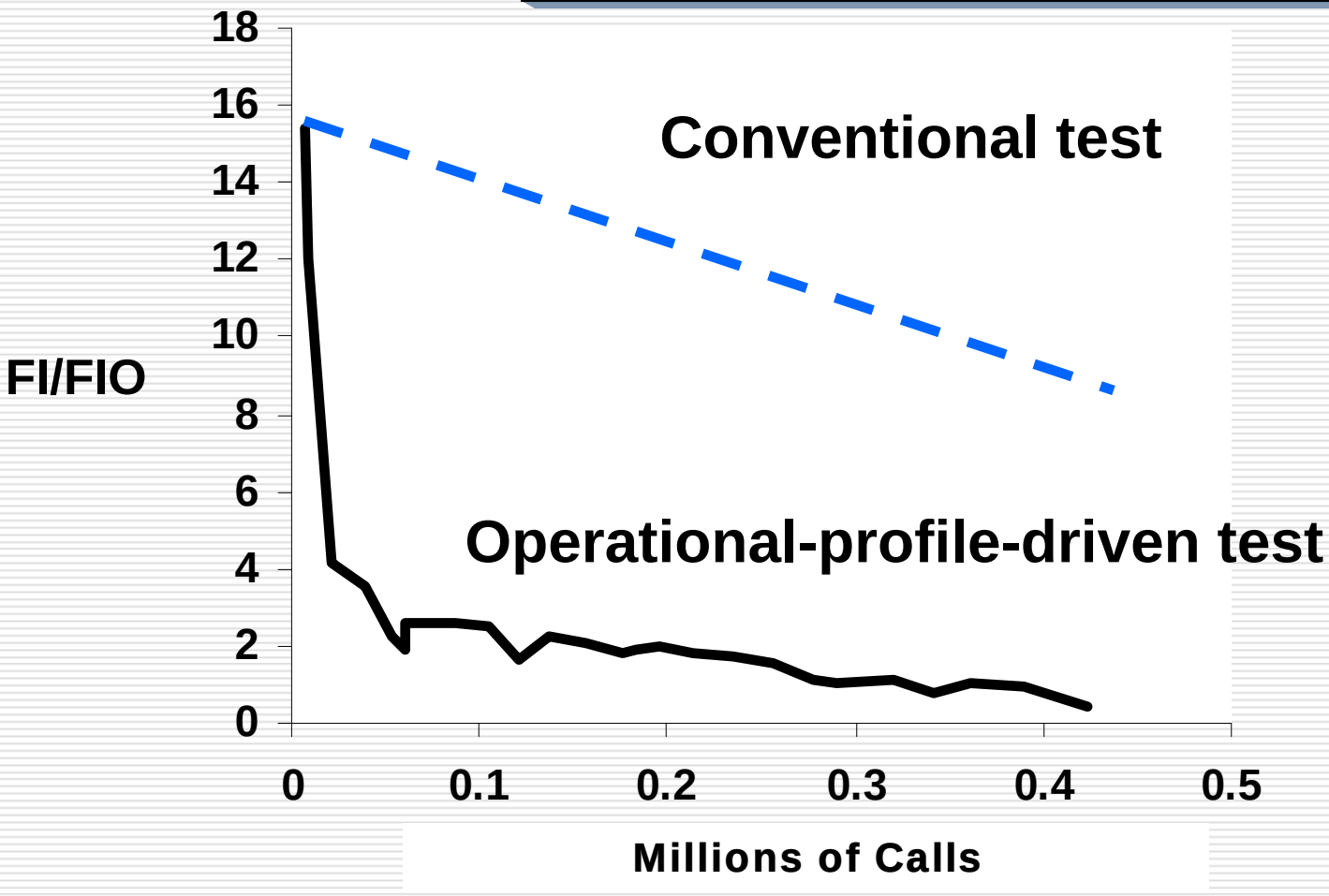
Broon et al. *A New Statistical Software Reliability Tool*



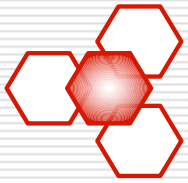
Release Management

- FIO -- *Failure Intensity Objective*
  - Maximum acceptable failure rate for release
  - Estimated from failures observed in system test
  - Application and industry specific
- FI -- *Failure Intensity*
  - Rate of failures observed
- FI/FIO over time
  - *Reliability Growth* curve
  - At 1.0, release goal is achieved

Reliability Growth

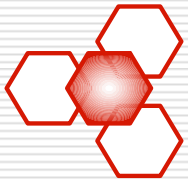


John Musa. *More Reliable Software Faster and Cheaper – An Overview*



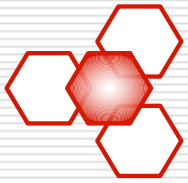
Corollary Techniques

- "Statistical" Testing Strategies
  - Randomized selection of test inputs
  - Randomized selection of test case values
- Orthogonal Defect Classification
  - Process for collecting and analyzing defect data
  - Designed to support reliability growth analysis
- Fault-Tolerant Architecture
  - Large-grain strategies
  - HW/SW focus

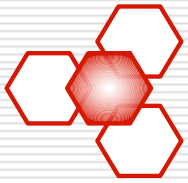


Promise of Profile-Based Testing

- Tester's point of view versus reliability analyst
  - Maximize reliability within fixed budget
  - Measurement of reliability not primary goal
- Profile-Based Testing is optimal when
  - Available information already used
  - Must allocate resources to test complex SUT
- Many significant practical obstacles

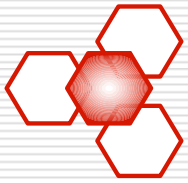


Trading System Case Study



Case Study – Trading System

- E-commerce/securities market, screen-based trading over private network
- 3 million transactions per hour (peak)
- 15 billion dollars per day (peak)

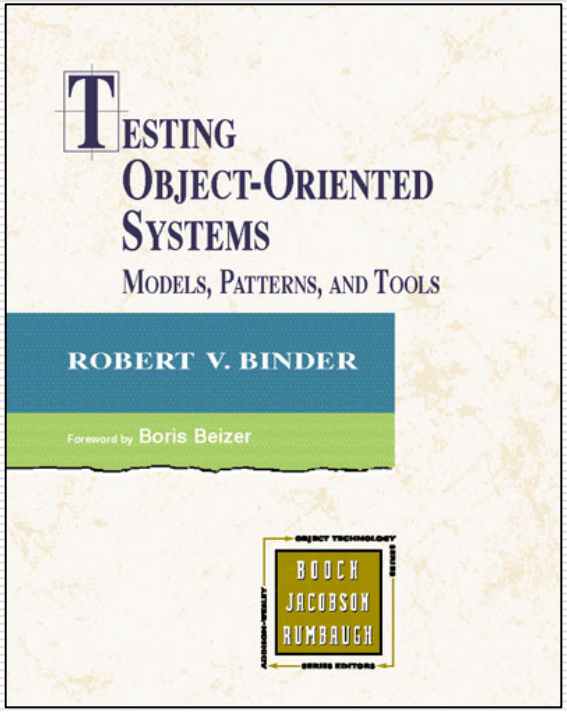


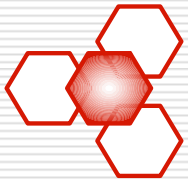
Profile-based Testing Approach

- Executable operational profile
- Simulator generates realistic unique test suites
- Loosely coupled automated test agents
- Oracle/Comparator automatically evaluate
- Support integration, functional, and stress test

Model-based Testing

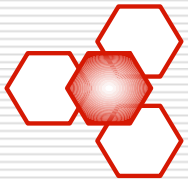
- Profile alone insufficient
- *Extended Use Case*
  - RBSC test methodology
  - Defines feature usage profile
  - Input conditions, output actions
- *Mode Machine*
- *Invariant Boundaries*





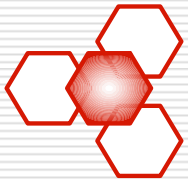
Simulator

- Generate inputs according to OP
- Discrete event simulation
  - Generate any distribution with pseudo-random
- Prolog implementation (50 KLOC)
  - Rule inversion
- Load Profile
  - Time domain variation
  - Orthogonal to operational profile
- Each event assigned a "port" and submit time



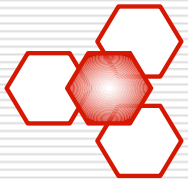
Automated Run Evaluation

- Oracle accepts output of simulator
- About 500 unique rules
- Verification
  - Splainer – result/rule backtracking tool
  - Rule/Run coverage analyzer
- Comparator
  - Extract transaction log
  - Post run database state
  - end-to-end invariant
- Stealth requirements engineering



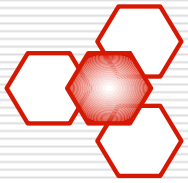
Test Process

- Six development increments
  - 3 to 5 months
  - Test design/implementation parallel with app dev
- Plan each day's test run
  - Load profile
  - Total volume
  - Configuration/operational scenarios



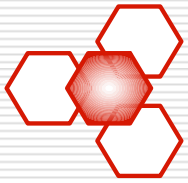
Daily Test Runs

- Run Simulator
  - 100,000 events per hour
  - FTP event files to test agents
- Start SUT
- Test agents automatically start at scheduled time
- Extract results
- Run Oracle/Comparator
- Prepare bug reports



Results

- Revealed about 1,500 bugs over two years
  - 5% showstoppers
- Five person team, huge productivity increase
- Achieved proven high reliability
  - Last pre-release test run: 500,000 events in two hours, no failures detected
  - No production failures



International Symposium on Software Reliability Engineering (ISSRE) 2008

Trip Report

General

- Nineteenth ISSRE
  - 200+ attended
  - ~1/3 outside US
  - ~1/3 practitioners
  - ~2/3 researchers/students
- Conference focus is broadening
- Incremental research progress
- Practitioner community about the same
- Student presentations impressive
- No new tool support

Presentations, by type

	Accept'd	Submit'd
Research	29	116
Industry	25	32
Posters	14	31
Student	11	30
Workshops	~25	?

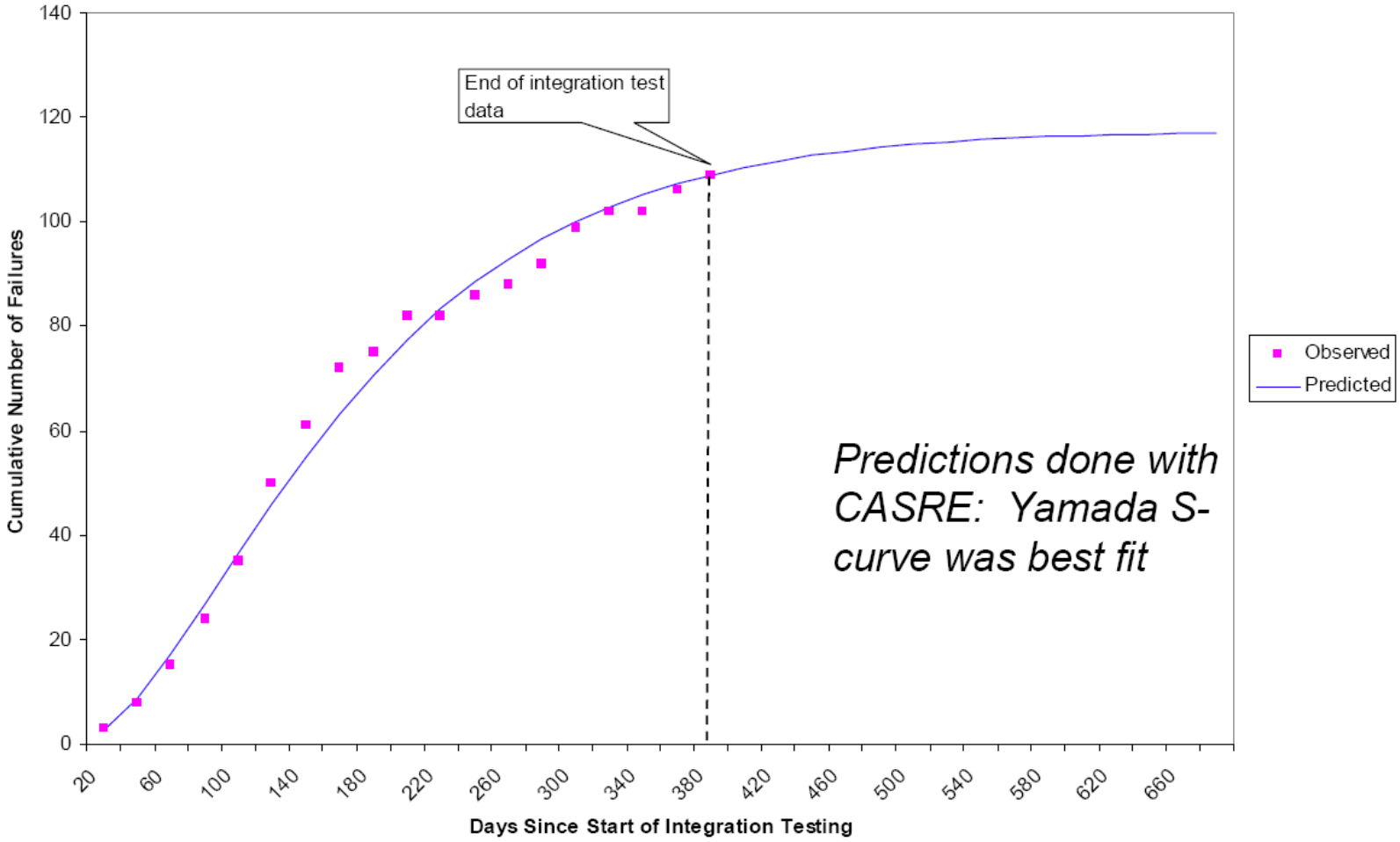
- 2009 at Mysore India, housing included in registration
- Over 100 presentations -- a few snapshots follow

Presentation Snapshots

- Test results from hardware/software integration testing of space vehicle components
  - Composite HW/SW model
  - Stochastic Activity Networks
    - Blends Petri nets and Markov chain
  - Used established SRE reliability and availability models for analysis and estimation

Hecht, McAdams, and Lam. *Use of Test Data for Integrated Hardware/Software Reliability and Availability Modeling of a Space Vehicle*

Presentation Snapshots



Hecht, McAdams, and Lam. *Use of Test Data for Integrated Hardware/Software Reliability and Availability Modeling of a Space Vehicle*

Presentation Snapshots

- Server farm reliability: Service availability is most effectively improved by server replication, but at the margin, failure prevention is more effective.

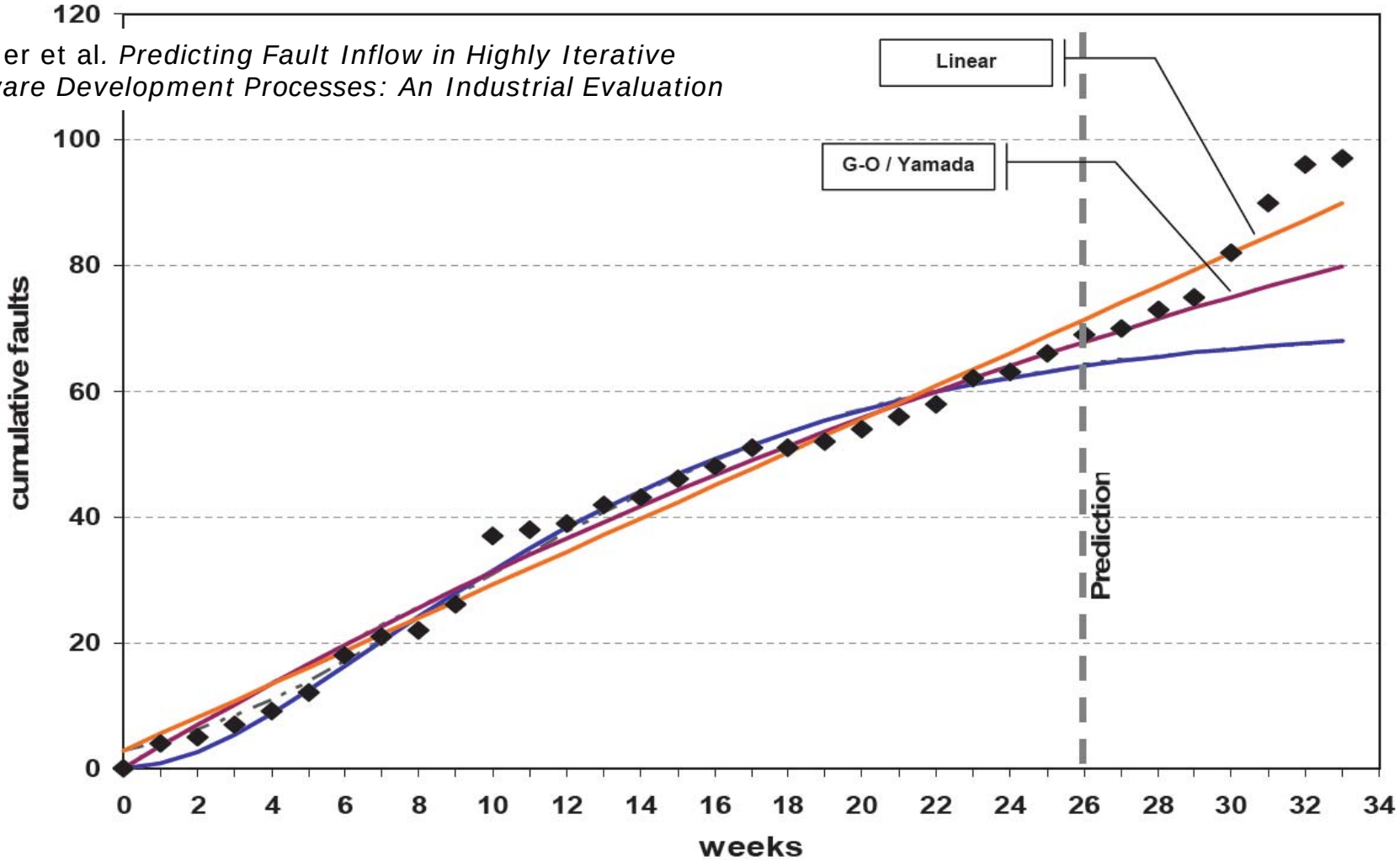
  Salfner & Wolter. *Replication vs. Failure Prevention: How to Boost Service Availability?*

- Best reliabilty growth model for an "agile" process is linear – faults revealed at a steady rate. Classic S curve corresponds to waterfall process.

  Baumer, Seidler, Torkar & Feldt. *Predicting Fault Inflow in Highly Iterative Software Development Processes: An Industrial Evaluation*

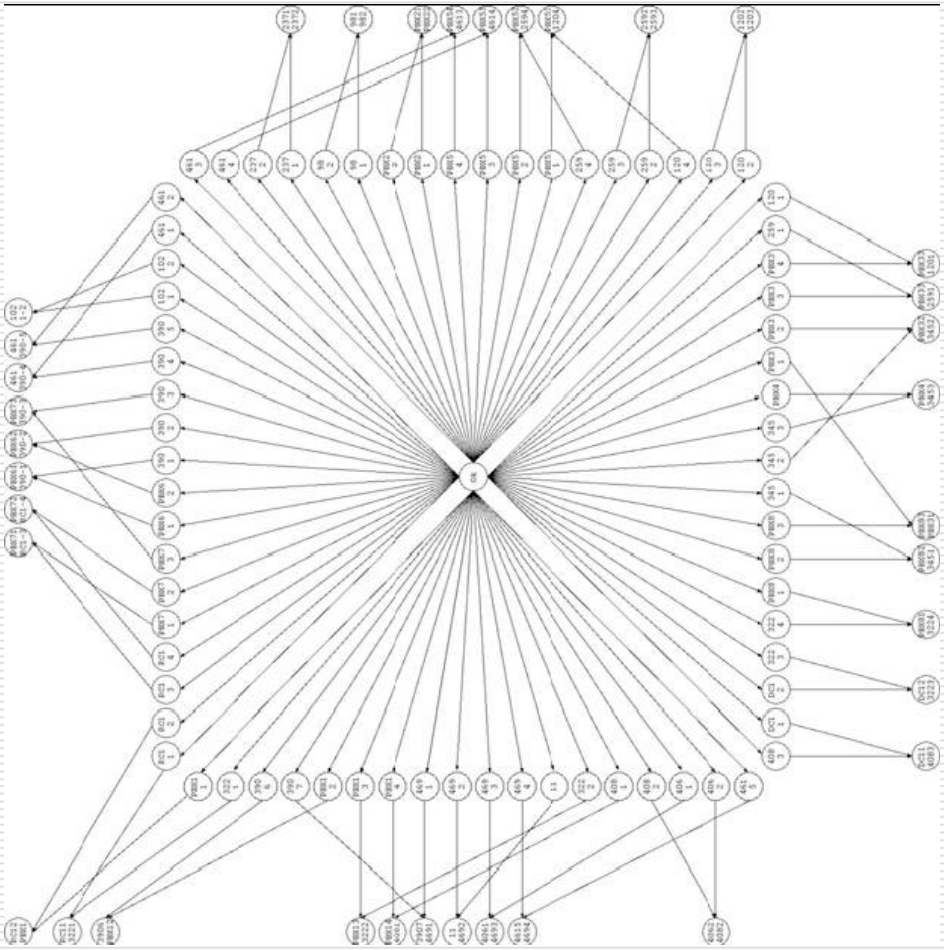
Presentation Snapshots

Baumer et al. *Predicting Fault Inflow in Highly Iterative Software Development Processes: An Industrial Evaluation*



Presentation Snapshots

- Reliability model for large campus building control system
  - Markov model corresponds to Poisson process, i.e., reliability growth.
  - Abstraction necessary: "If you wants everything, you can't have nothing."

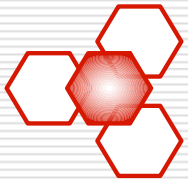


Avritzer. *Applying Markovian Models to the Reliability Estimation of Large Distributed Software Systems.*

Presentation Snapshots

- Improving safety in a new NASA system
  - Analyzed process metrics, product metrics to identify safety-related hazards
  - Metrics don't tell you if its safe, but can indicate risks
  - You can't measure "safety" per se, but you can measure its proxies
  - Approach applicable for similar characteristics, e.g., security

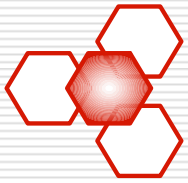
Basili. *Using Measures and Risk Indicators for Early Insight into Software Product Characteristics such as Software Safety.*



Presentation Snapshots

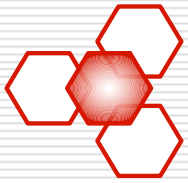
- Microsoft Crash Telemetry
  - How to use feedback from 800 million Windows hosts
  - Auto collect app crash signatures, dump, registry, etc.
  - 100,000+ hosts for Vista Customer Experience
  - 390,000 unique devices (drivers) in Vista SP1
  - Drivers extend the Windows kernel
    - 25 new daily, 100 patches
    - 000s changes to the windows kernel
  - 30x reliability differences between hardware
  - Variation by locale: 4x between worst, best

Li et al. *Reliability Assessment of Mass-Market Software: Insights from Windows Vista.*



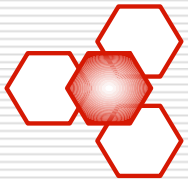
SRE Concerns

- Speed bumps
  - Operational profile can be hard to develop
  - Skepticism about cost/benefit
  - Too many models
  - Restrictive model assumptions
- Testing with an operational profile
  - No integrated representation of constraints
  - Modes not integrated
  - Hand-crafted test suites too brittle
  - Scalability
- Not much tool support
  - With better automation, applicable to broader class of apps/environments



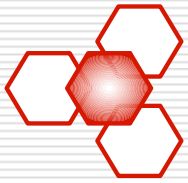
What I'm Working On

- Automatic generation of behavior model from an operational profile
  - Generate constrained test suites
  - Overcome monolithic model obstacle
  - Automatic mode partitioning
- Test generation by environment simulation
  - Statistical accuracy needed for reliability prediction
  - No brittleness
- Tight integration of failure capture and reliability dashboard
  - Actionable real-time metrics



Summary

- SRE is credible, well-established
  - Compatible with almost any process
  - Quant models refined with 20 years of peer-reviewed research and application
  - Proven best practice, 100s of firms
  - Sustainable for typical IT workers
  - Proven ROI
- Very effective in its sweet spot



Resources

- ISSRE Web Site
  - http://www.issre.org
- ISSRE Proceedings
  - http://ieeexplore.ieee.org/servlet/opac?punumber=4700288
- Musa's Web Site
  - http://softwarereliabilityengineering.com
- Definitive Industry Reference
  - *Software Reliability Engineering: More Reliable Software Faster and Cheaper.* John D. Musa
  - http://www.authorhouse.com/BookStore/ItemDetail.aspx?bookid=26806
- Survey of the State of the Art
  - *Software Reliability Engineering: A Roadmap.* Michael R. Lyu
  - http://csse.usc.edu/classes/cs589_2007/Reliability.pdf