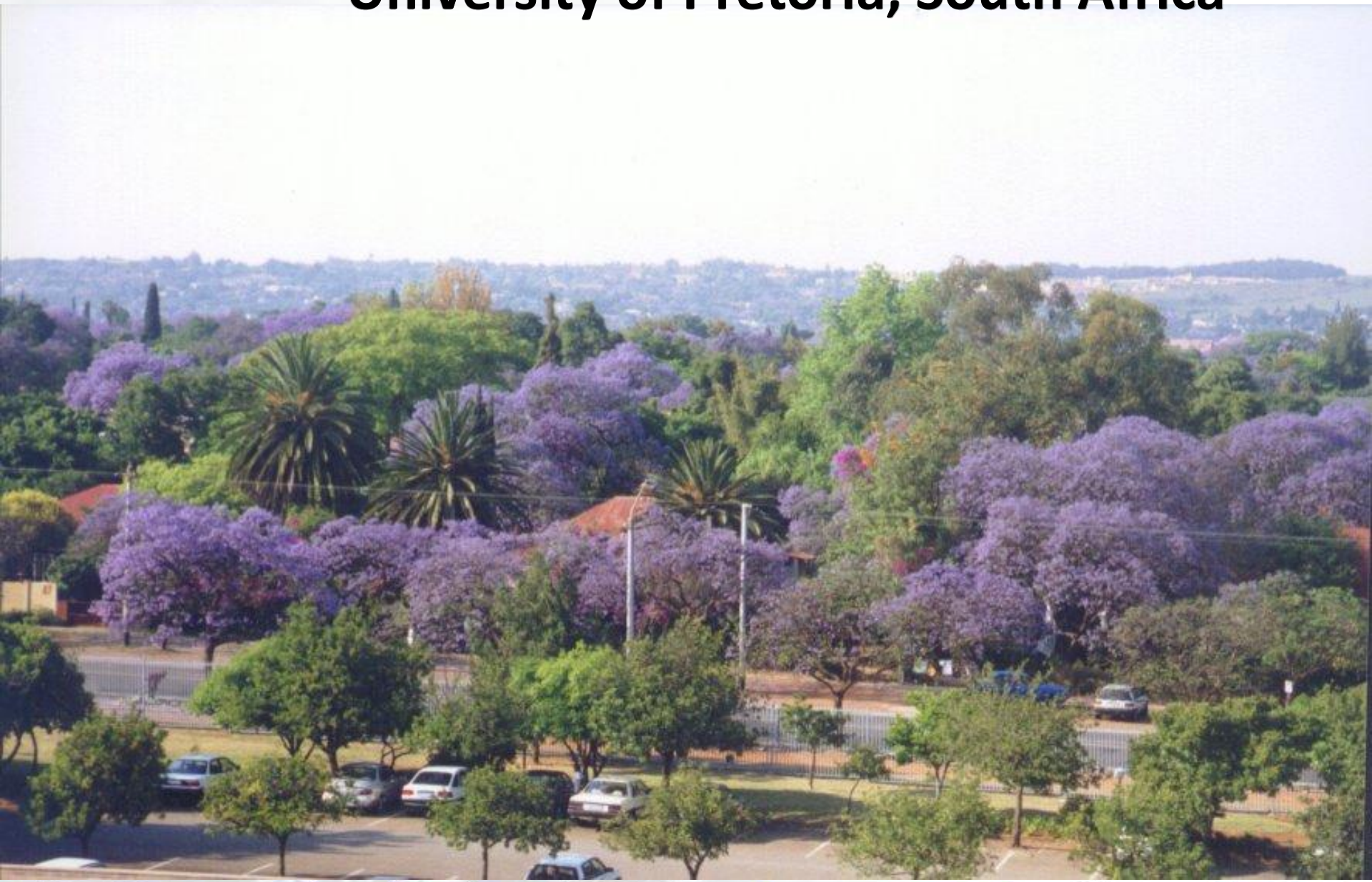
Maintaining Software in the Billions

Judith Bishop

Microsoft Research, Redmond, USA

CSMR 2010, Madrid

University of Pretoria, South Africa



Microsoft, Redmond USA

Microsoft Research

Started in 1991

Now more than 850 PhD researchers

Microsoft Research

Search Microsoft Research

Projects | Publications | People | Downloads

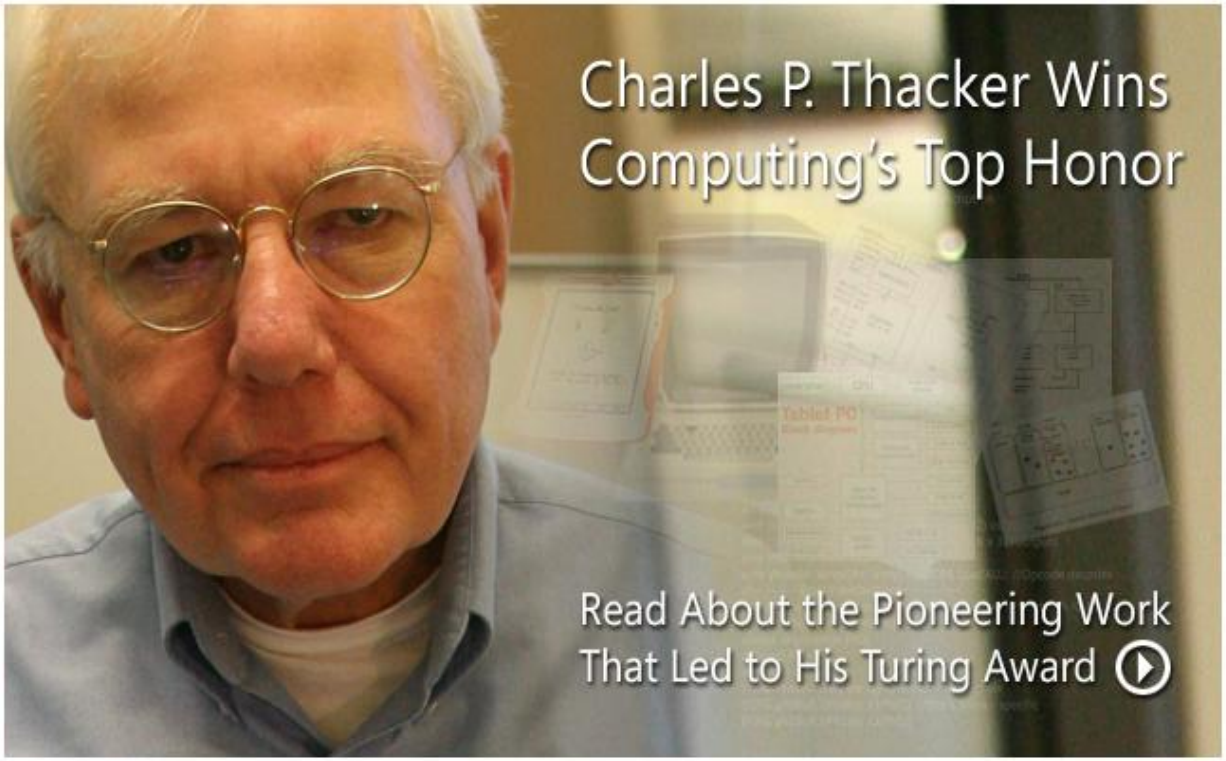
Home | Our Research | Collaboration | Careers

About Us | News | Media Resources | Events | Community

2008 2010 2009

A Timeline of Microsoft Research

Explore our research through the years



Feature Stories (1/3)

A New Way to Interact with the Cloud

Cloud computing encompasses a mind-boggling array of technologies, from databases to greener data centers. Demos on display during TechFest 2010, Microsoft Research's annual tech showcase, focus on work shaping the future of computing. Read more...

More feature stories

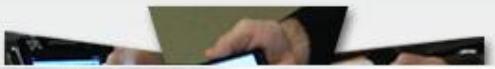
News | Microsoft's Bill Buxton on Design

Quick Links

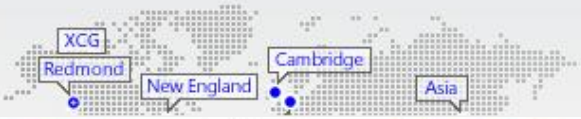
» Project Tuva: Watch the Feynman Lectures

Spotlight

Making User Interfaces Natural



Worldwide Locations



External Research Global Themes

Community and Geographic Outreach

Core Computer Science

Earth, Energy & Environment

Education & Scholarly Communication

Health & Wellbeing

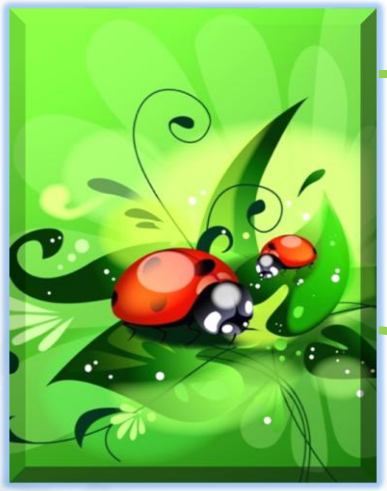
Advanced Research Tools and Services

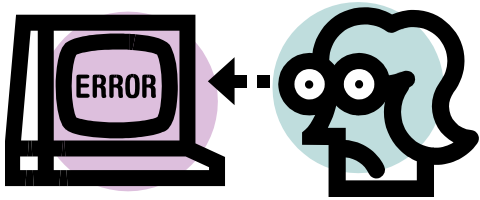
Agenda

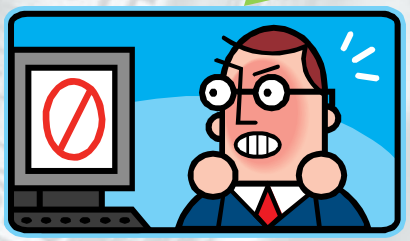
1. Experience with Error Reporting
2. Global Software Servicing
3. Concurrency bugs
4. Challenges – large and small
5. Conclusions

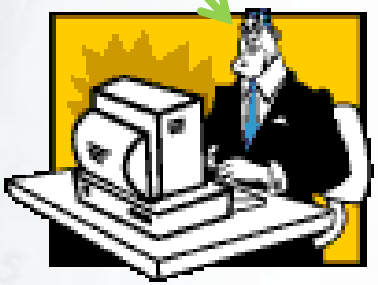
Two Definitions

- **Bug**: a **flaw** in program logic
- **Error**: a **failure in execution** caused by a bug
- One bug may cause <u>many</u> errors.









The Challenge

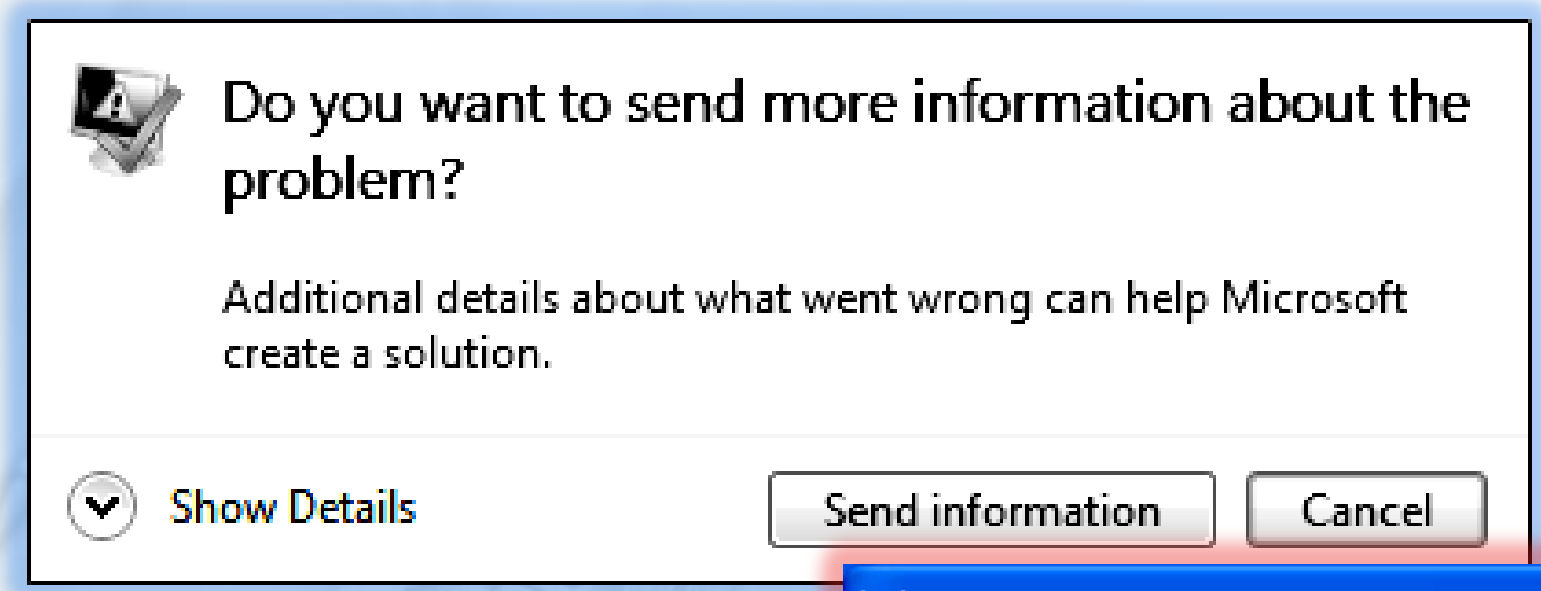
- Microsoft ships software to **1 billion** users around the world
  - How do we find out when things go wrong?

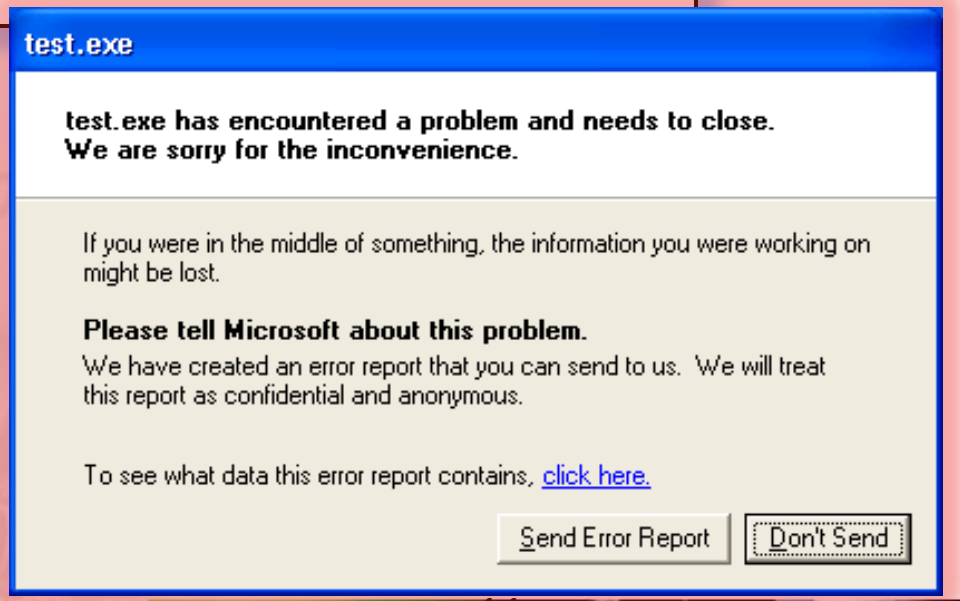
- We want to
  - fix bugs **regardless of source**
    - application or OS
    - software, hardware, or malware
  - prioritize bugs that **affect the most users**
  - **generalize the solution** to be used by any programmer
  - get the **solutions out to users** most efficiently
  - try to **prevent bugs** in the first place

Reported Bugs

Error	Reporting Trigger
Kernel Crashes	Crash dump found in page file on boot.
Application Crashes	Unhandled process exception.
Application Hangs	Failure to process user input for 5 seconds.
Service Hangs	Service thread times out.
Installation Failures	OS or application installation fails.
App. Compat. Issues	Program calls deprecated API.
Custom Errors	Program calls WER APIs to report error.
UI Delays	Timer assert takes longer than expected.
Invariant Violations	Ship assert in code fails.

Windows Error Reporting (WER)





WER's properties

- The huge data based can be mined to prioritize work
  - Fix bugs from most (not loudest) users
- Correlate failures to co-located components
  - Show when a collection of unrelated crashes all contain the same culprit (e.g. a device driver)
- Proven itself "in the wild"
  - Found and fixed 5000 bugs in beta releases of Vista after programmers had found 100 000 with static analysis and model checking tools.

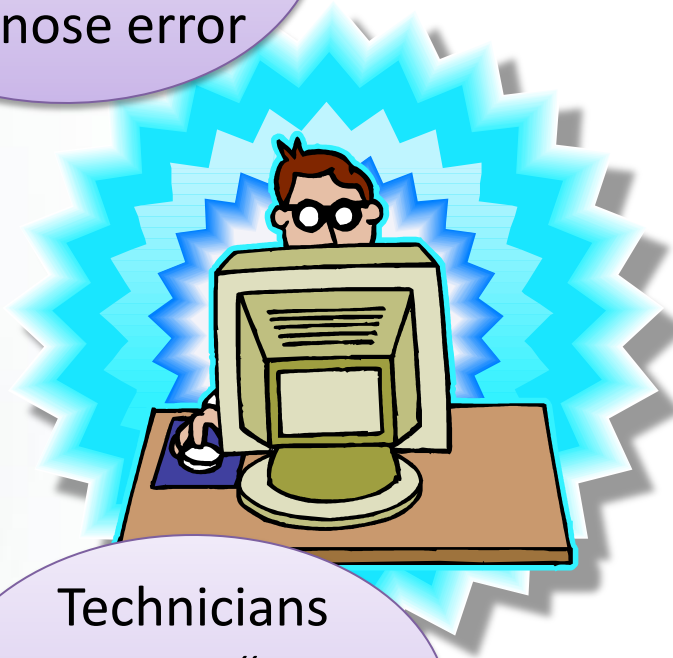
WER by the Numbers

billions	Error reports collected per year (App,OS,HW)
1 billion	Clients
100 million	Reports /day processing capacity
many 1000s	Bugs fixed
almost all	Microsoft product teams use it
over 700	Companies using WER
200	TB of Storage
60	Servers
10	Years of use
2	Servers to record every error received

Debugging in the Small…

In the Large without WER…

Support technician tries to diagnose error



User calls technical support

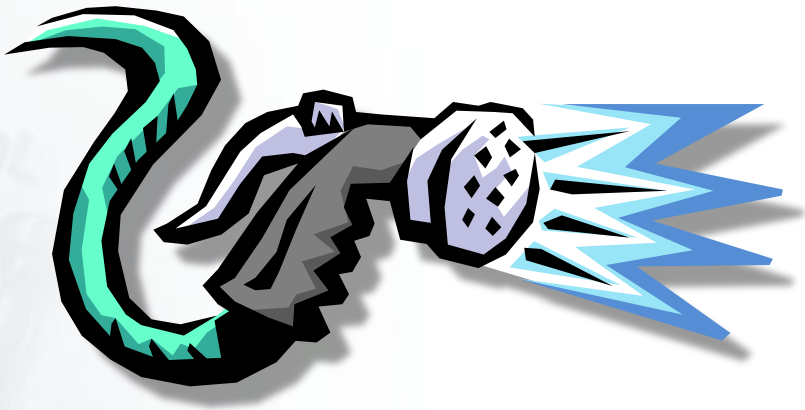
Technicians reports “top ten” issues to programmers

The Human Bottleneck

- Can’t hire enough technicians
- Data is inaccurate
- Hard to get additional data
- No “global” baseline
- Useless for heisenbugs
- **Need to remove humans**

Goal: Fix the Data Collection Problem

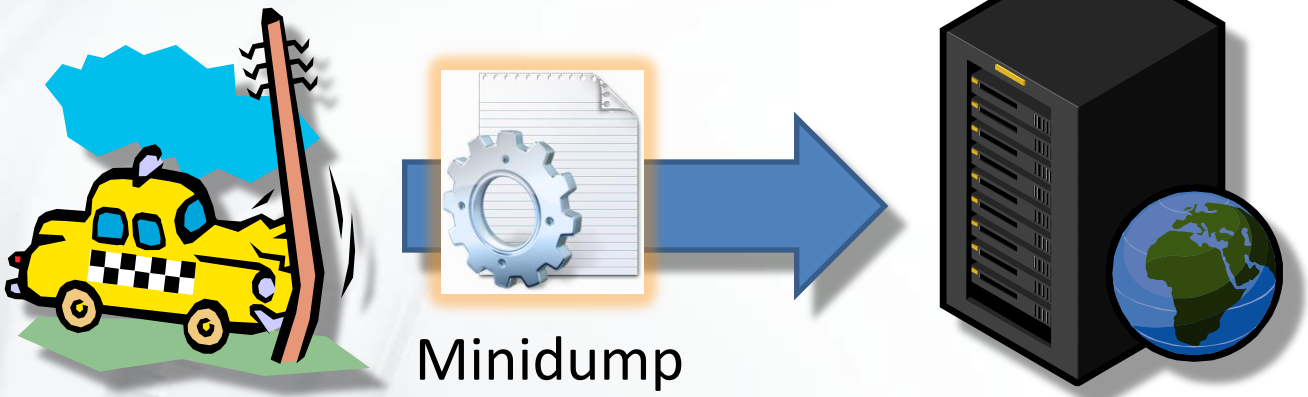
- Allow one service to record
  - **every error** (application, OS, and hardware)
  - on **every Windows system**
  - **worldwide**.

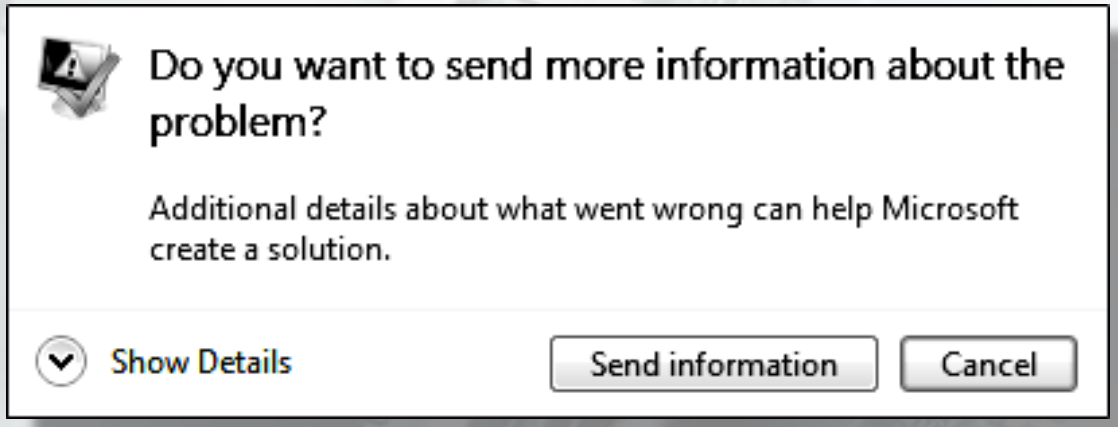


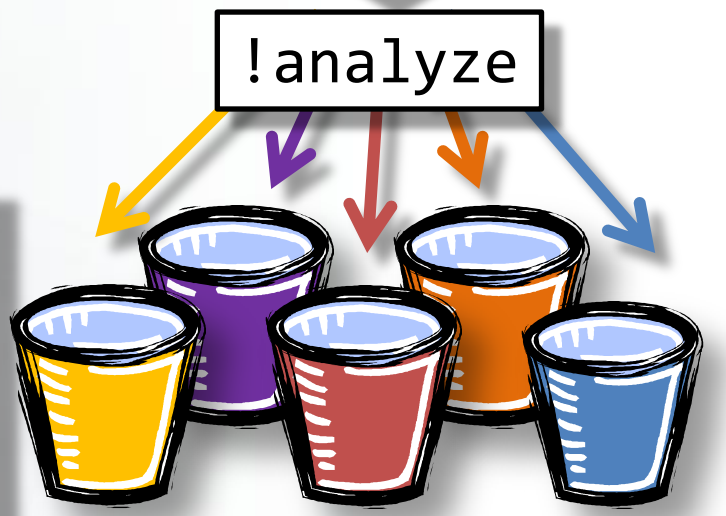
- Corollary:

- What we can measure, we can fix…

Debugging in the Large with WER…





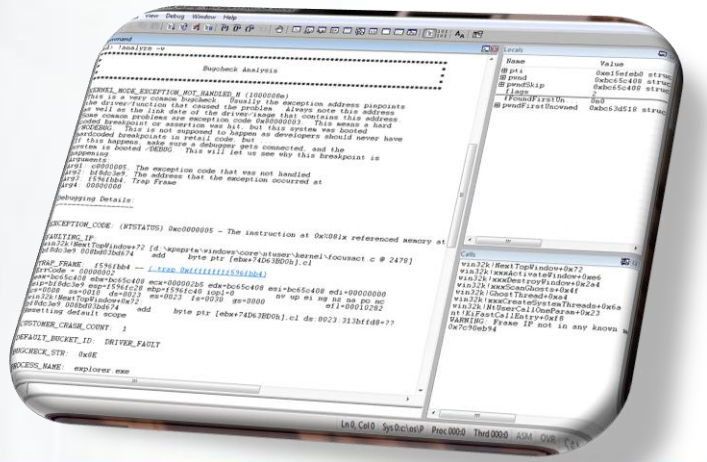


5 17 23,450,649

!analyze

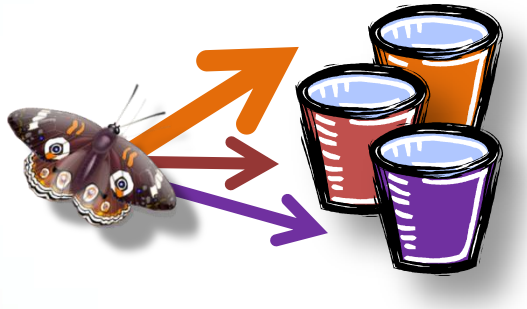
- Engine for WER bucketing heuristics
- Extension to the *Debugging Tools for Windows*
  - input is a minidump, output is bucket ID
  - runs on WER servers (and programmers desktops)
  - http://www.microsoft.com/whdc/devtools

- 500 heuristics
  - grows ~ 1 heuristic/week

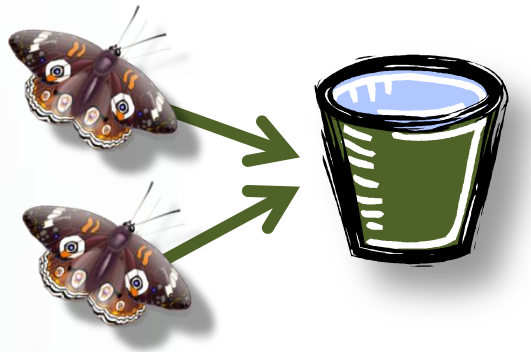


Bucketing Mostly Works

- One bug can hit multiple buckets
  - up to **40%** of error reports
  - duplicate buckets must be hand triaged



- Multiple bugs can hit one bucket
  - up to **4%** of error reports
  - harder to isolate each bug



- But if bucketing is wrong 44% of the time?
- Solution: scale is our friend
  - With billions of error reports, we can thow away a few million ☺

Improving Bug Triage

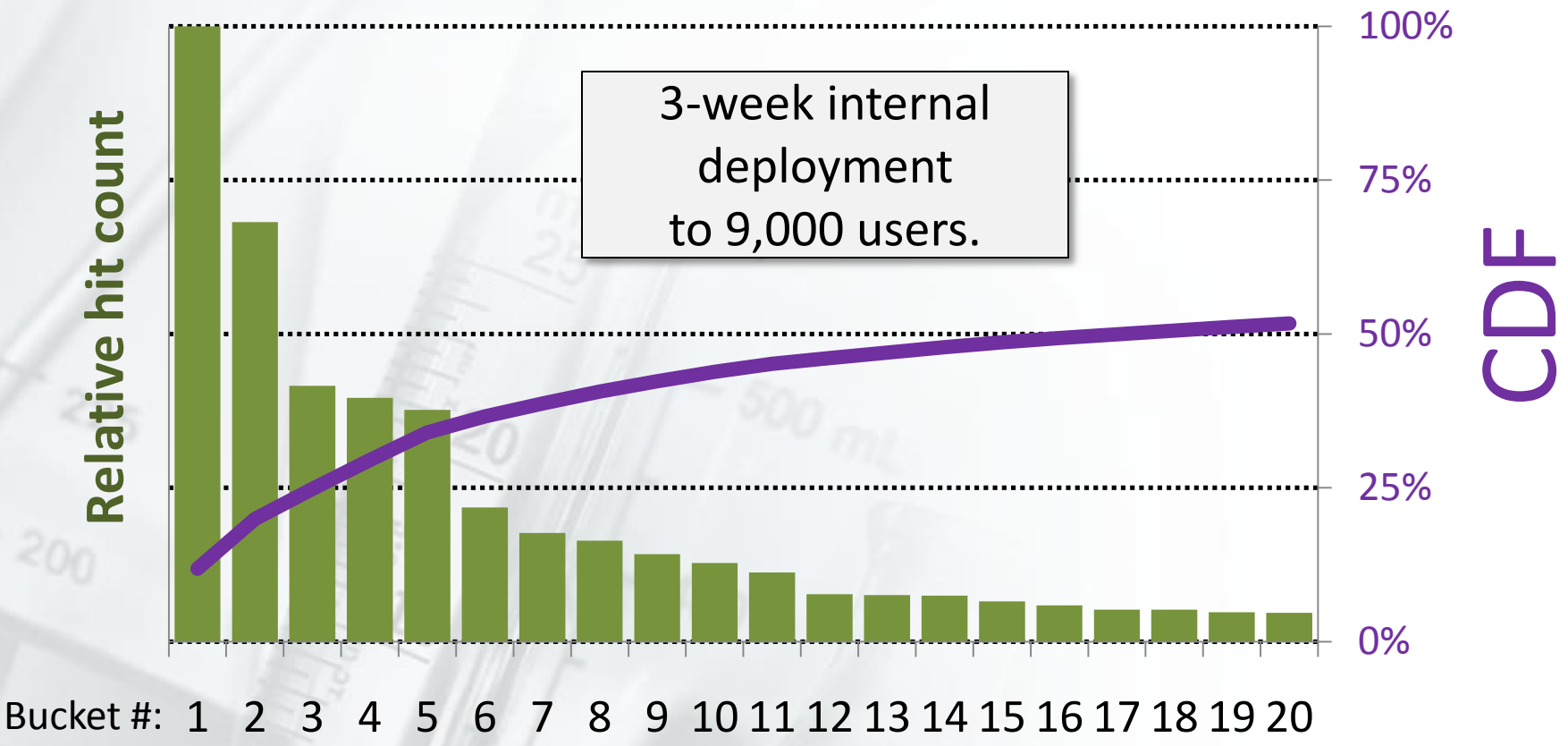
- Next stage: the bug gets to a developer
  - But it might be the wrong developer, or more skills might be needed
  - In Mozilla and Eclipse, between 37%-44% of bug reports are “tossed”
  - Tossing increases time-to-correction
- Microsoft Research teams have been able to reduce tossing by up to 72%
  - And prediction accuracy goes up by 23% points compared to traditional approaches
  - Uses a graph model based on Markov chains, which capture a bug tossing history and discover team structures

Gaeul Jeong, Sunghun Kim, Thomas Zimmermann. Improving Bug Triage with Bug Tossing Graphs. In (ESEC/FSE 2009), Amsterdam, The Netherlands, August 2009.

Watch out for

Philip J. Guo, Thomas Zimmermann, Nachiappan Nagappan, and Brendan Murphy, Characterizing and Predicting Which Bugs Get Fixed: An Empirical Study of Microsoft Windows, in *Proceedings of the 32th International Conference on Software Engineering (ICSE)*, Association for Computing Machinery, Inc., 2 May 2010

Top 20 Buckets for *MS Word 2010*

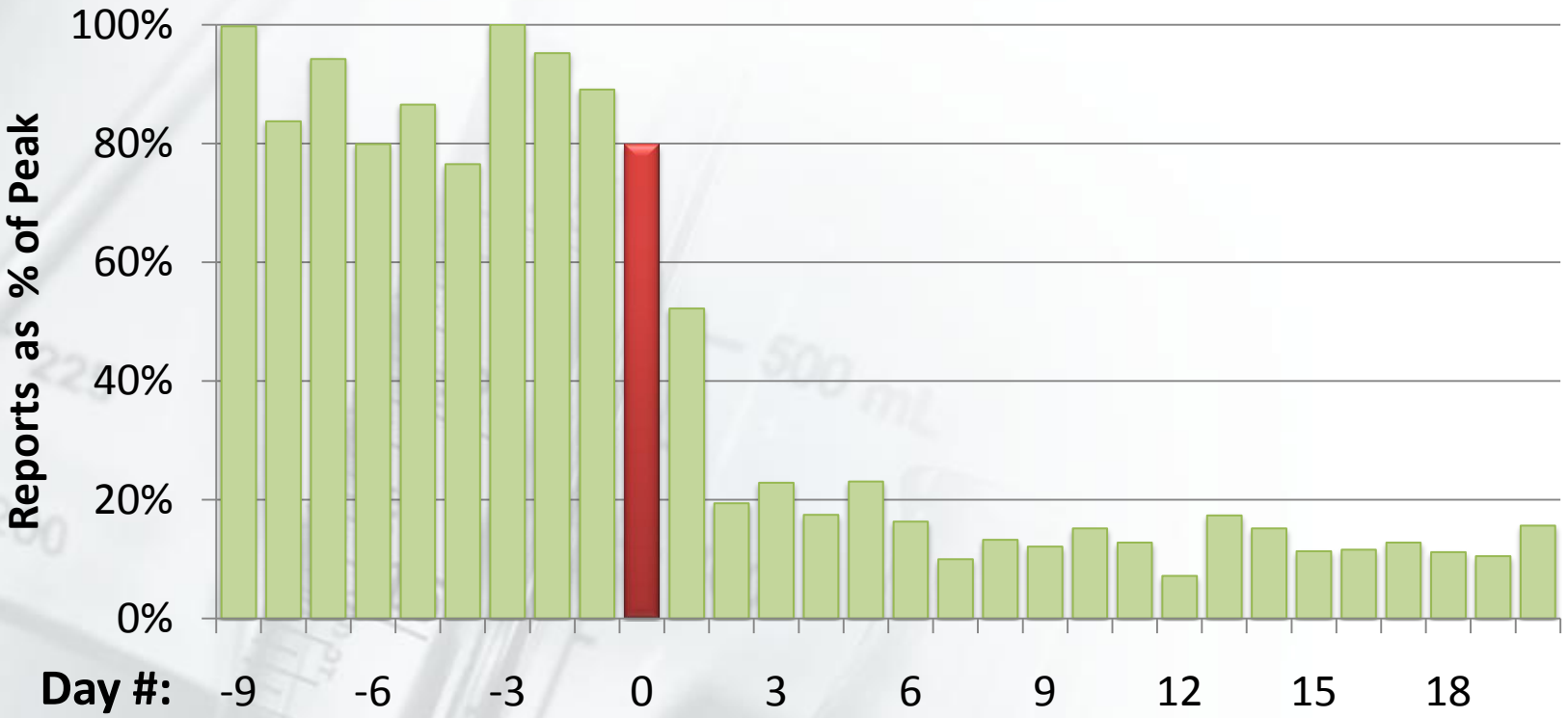


- Just 20 buckets account for **50% of all errors**
- Fixing a small # of bugs will help **many users**

Fixing bugs in software

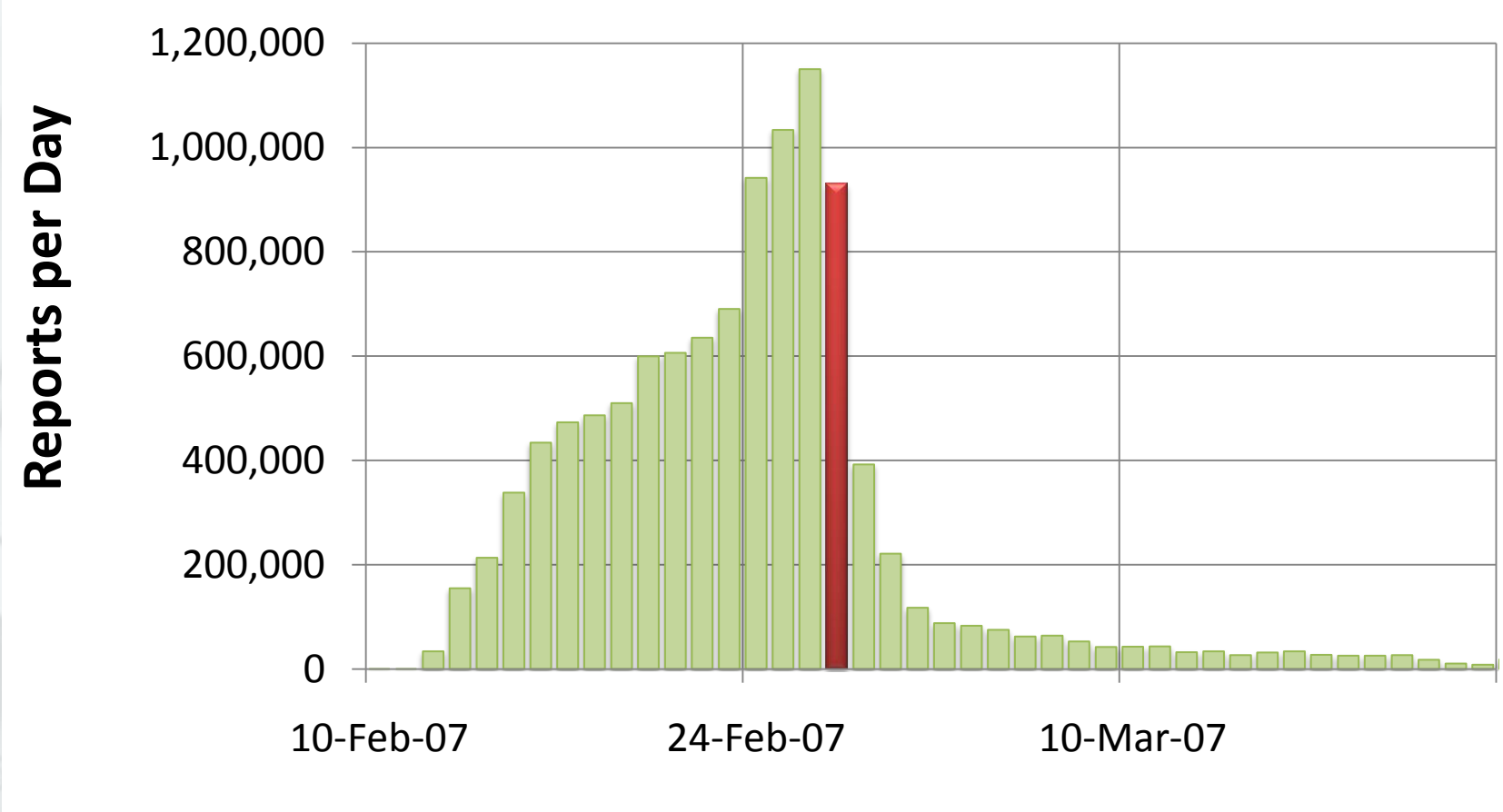
- First use found >=**5-year old** heisenbugs in Windows
- Windows Vista team **fixed 5,000 bugs** in beta
- Anti-Virus vendor fixed top 20 buckets and **dropped from 7.6% to 3.6%** of all kernel crashes
- Office 2010 team fixed 22% of reports **in 3 weeks**
- And you can fix yours…

Hardware: Processor Bug



- WER helped fix hardware error
- Manufacturer **could** have caught this earlier w/ WER

Renos Malware



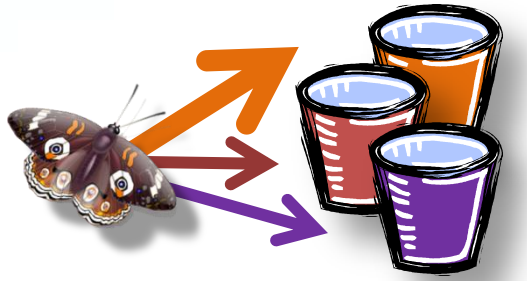
- Early detection w/o user action (renos, blaster, slammer, etc.)
- WER **scales** to handle global events

Weaknesses in the Heuristics

Problem with the heuristic

Program	In Second Bucket	In One-Hit Bucket	Combined
Excel	17.2%	4.4%	21.6%
Outlook	7.7%	10.0%	17.7%
PowerPoint	30.6%	6.2%	36.8%
Word	19.3%	8.8%	28.1%

WER works because …



- … bucketing mostly works
- Windows Error Reporting (WER) is
  - the **first** post-mortem reporting system with automatic diagnosis
  - the **largest** client-server system in the world (by installs)
  - helped 700 companies fix **1000s** of bugs and **billions** of errors
  - **fundamentally changed software development at Microsoft**

Kirk Glerum, Kinshuman Kinshumann, Steve Greenberg, Gabriel Aul, Vince Orgovan, Greg Nichols, David Grant, Gretchen Loihle, and Galen Hunt, Debugging in the (Very) Large: Ten Years of Implementation and Experience, in *SOSP '09*, Big Sky, MT, October 2009

http://winqual.microsoft.com

Other Ways, and Reporting Back

Microsoft answers.

Microsoft twitter
@MicrosoftHelps

Microsoft Support

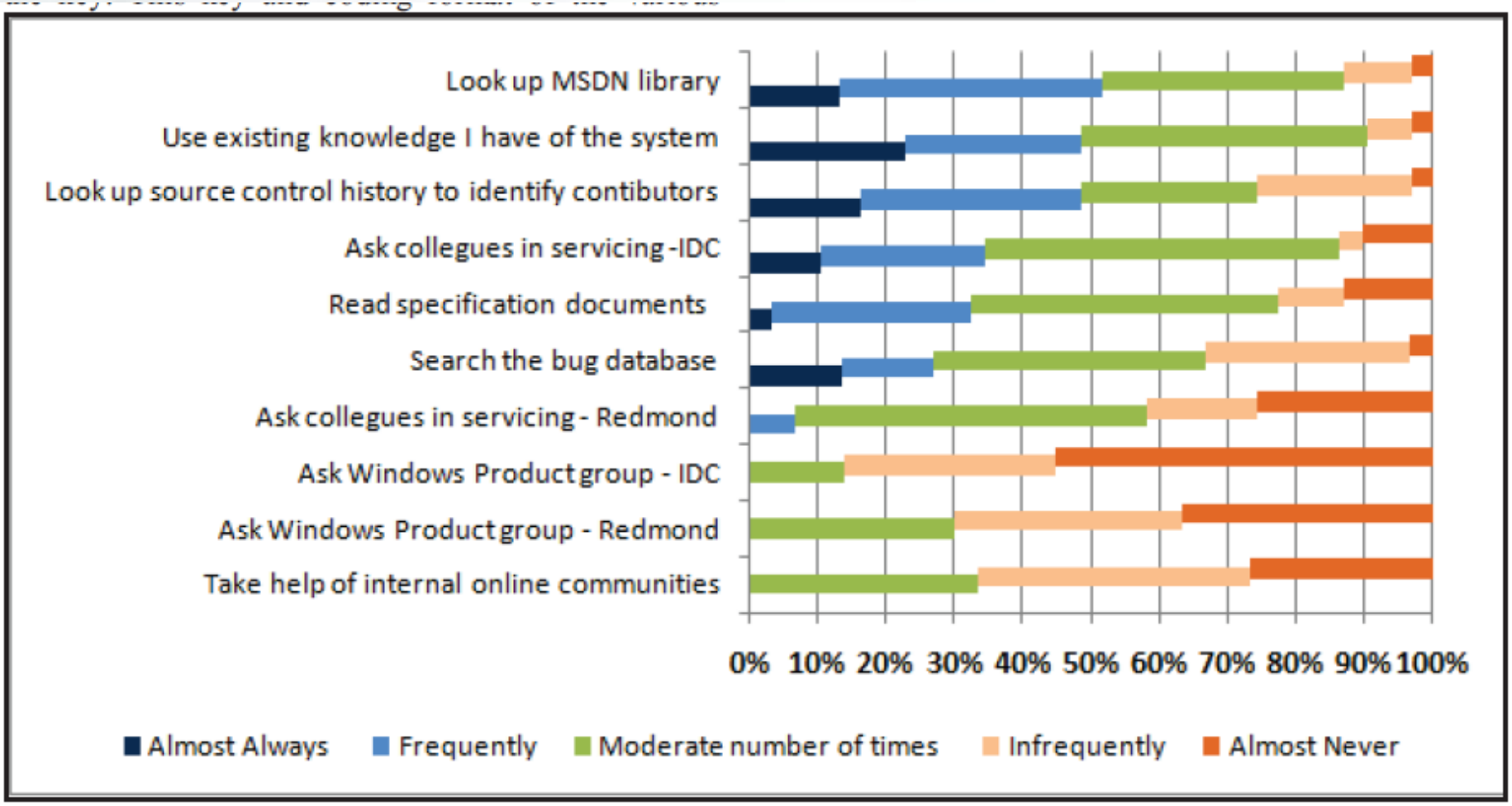
"Now, [customers who need technical support] may have an even better option…Microsoft has quietly added a "Fix it" button to a few of the thousands of help documents on its Web site. When clicked, the computer then takes all the recommended steps automatically."
CNET

2. Global Software Servicing

- Microsoft's major code is maintained and evolved by programmers who were not the developers
- Testers in e.g. China and India lack in depth knowledge and institutional memory
- Issues
  - Strategic (keep projects disjoint architecturally)
  - Cultural (face-to-face is important)
  - Inadequate communication (time zones)
  - Knowledge management (move people between zones)
  - Project and process management (synchronized deadlines)
  - Technical issues (bandwidth)

Global Software Servicing: Observational Experiences at Microsoft, Shilpa Bugde, Nachiappan Nagappan, Sriram Rajamani, G.Ramalingam, IEEE International Conference on Global Software Engineering (ICGSE 2008), Bangalore, India.

Source of satisfying info. At MS India



3. Concurrency Bugs

- Some schedules find the bugs, others do not.
- Probability of finding bug = probability that the OS follows a buggy schedule.
- Probability is low
  - Distribution of OS schedules is not particularly effective at flushing out concurrency bugs.
- **Cuzz:** Use a randomized scheduler with probabilistic guarantees of finding bugs.

Madanlal Musuvathi, Sebastian Burckhardt, Pravesh Kothari, and Santosh Nagarakatte, A Randomized Scheduler with Probabilistic Guarantees of Finding Bugs, in *Proceedings of the Fifteenth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2010)*, Association for Computing Machinery, Inc.

Cuzz

Three steps to better stress testing

1. Understand nature of most frequently occurring concurrency bugs, so define *bug depth*

2. Construct a randomized scheduler with a *guaranteed* probability of hitting bugs
   - Reasonably large probability for bugs of low depth

3. *Override OS* scheduling decisions during stress testing to focus on effectively flushing out the bugs of low depth.

Bug Depth

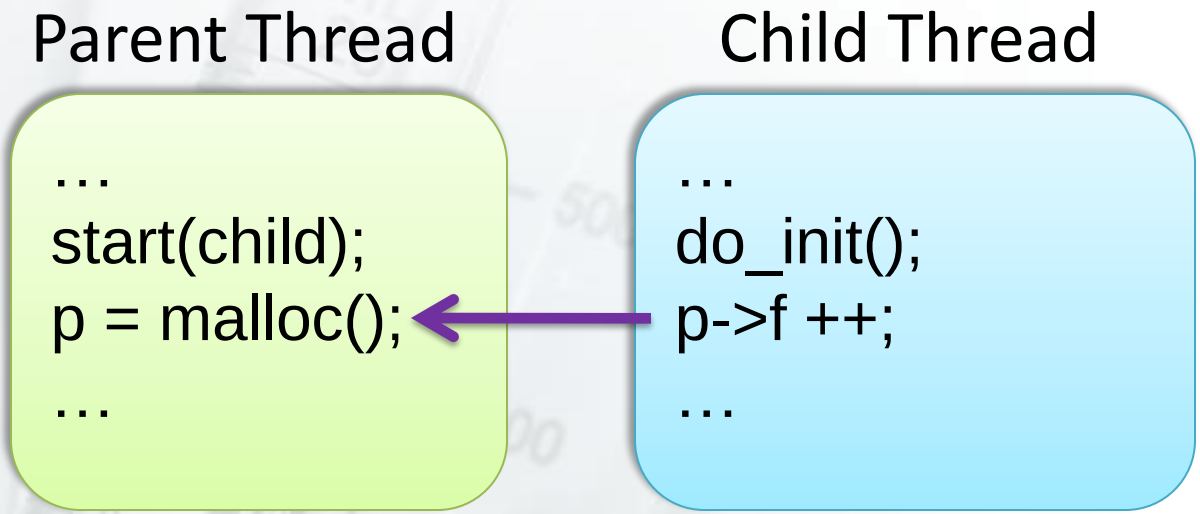
Bug Depth = the number of ordering constraints a schedule has to satisfy to find the bug.

More constraints means more things have to go “just right” to find the bug.

Conjecture: many typical bugs have low depth.
Let’s look at 3 examples.

Ordering Violation Example: A Bug of Depth 1

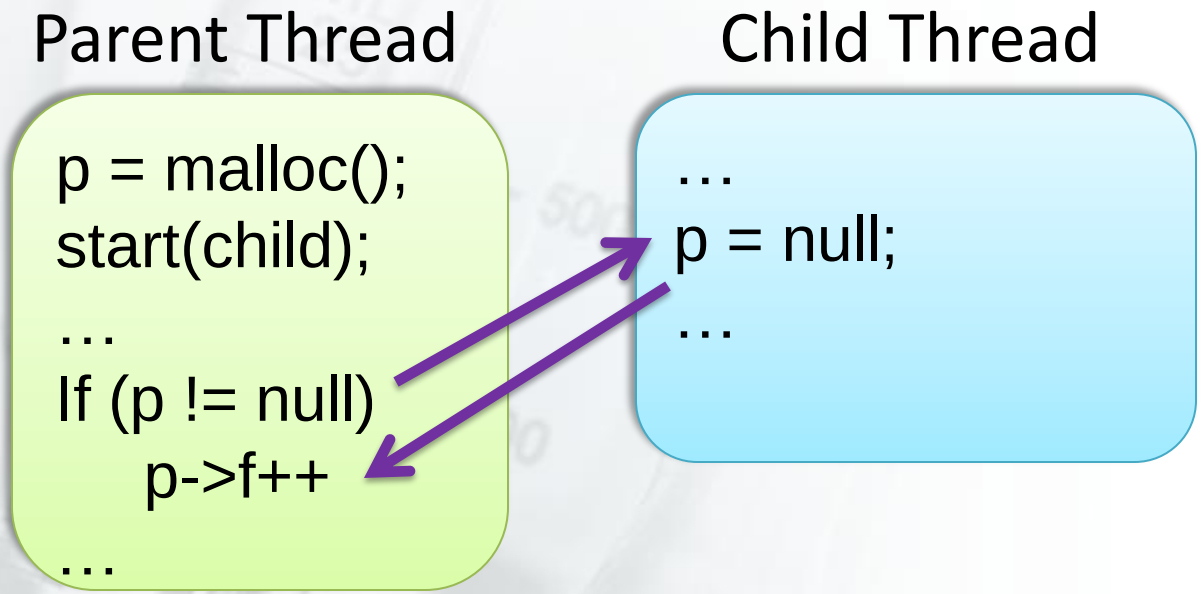
Bug depth = the number of ordering constraints sufficient to find the bug.



All schedules that satisfy the "←" find the bug.

Atomicity Violation Example: A Bug of Depth 2

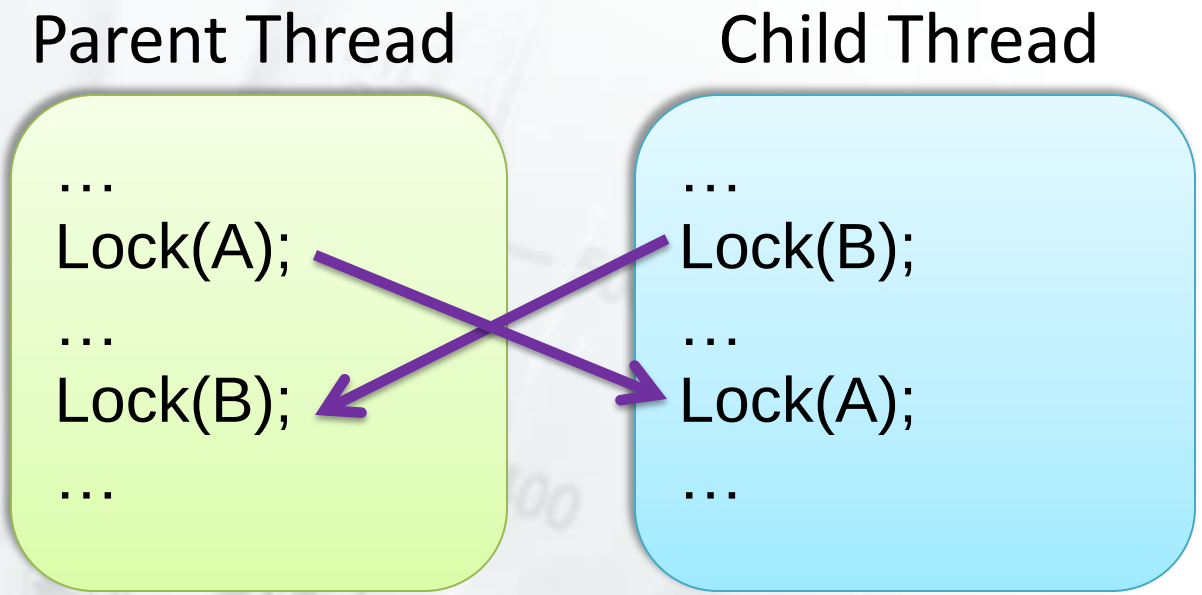
Bug depth = the number of ordering constraints sufficient to find the bug.



All schedules that satisfy both "←" find the bug.

Deadlock Example: A Bug of Depth 2

Bug depth = the number of ordering constraints sufficient to find the bug.



All schedules that satisfy both "←" find the bug.

The Cuzz Guarantee

- **Given a program with**
  - *n* threads (~tens)
  - *k* steps (~millions)
  - a bug of depth *d* (1,2)

- **Each run under the Cuzz scheduler finds the bug with probability**

$$\geq \frac{1}{n\ k^{d-1}}$$

1 day of stress testing
= 11 seconds of Cuzz testing

Experiment with Windows server team

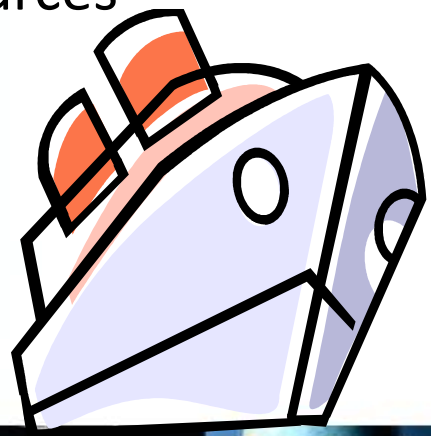
- Cuzz is robust enough to run a large server program
- Ran ~250 existing stress tests with cuzz
  - With almost no modification to the program or the test
- 22 failures (crashes + test failures)
  - 5 crashes due to a finalization bug
    - (VSTS # 352919)
  - 4 crashes due to test bugs
    - (VSTS # 351253) (VSTS # 356379) (VSTS # 356388)
  - 4 crashes + 5 test failures due to timing based sync
  - 3 crashes, being debugged
  - 1 livelock, being debugged

Current Cuzz Clients

- Windows
- Office
- SQL
- Windows Mobile
- XXX

Cuzz advantages

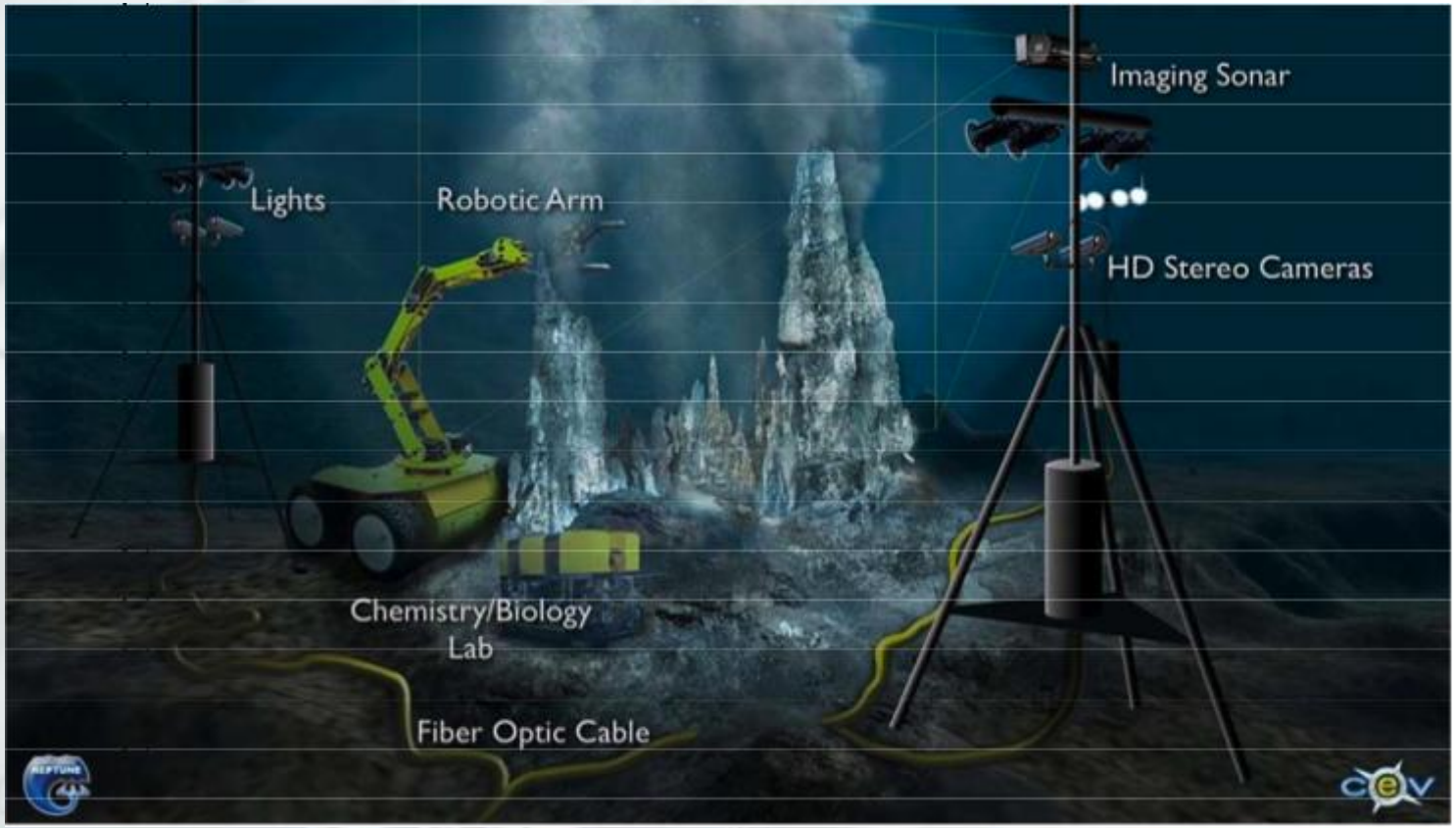
- **Whatever stress can do, cuzz can do better**
  - Effective in flushing out bugs with existing tests
  - Scales to large number of threads, long-running tests
  - Low adoption barrier
- **Probabilistic guarantees are useful**
  - Find as many bugs as possible before shipping
  - Determine the best way to allocate testing resources
  - Decide when to stop testing



4. Connected Devices and The Cloud

Smart Sensors and Data Fusion

- The NSF Ocean Observing Initiative
  - Hundreds of cabled sensors and robots exploring the sea floor
  - Data to be collected, curated, mined

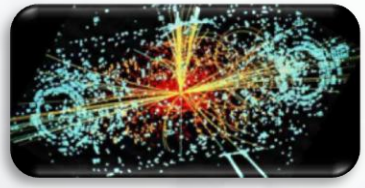


Conceptual representation of a future seafloor laboratory on the Regional Cabled Observatory network. *Credit: the NEPTUNE Project* www.neptune.washington.edu

The Future: an Explosion of Data

Experiments

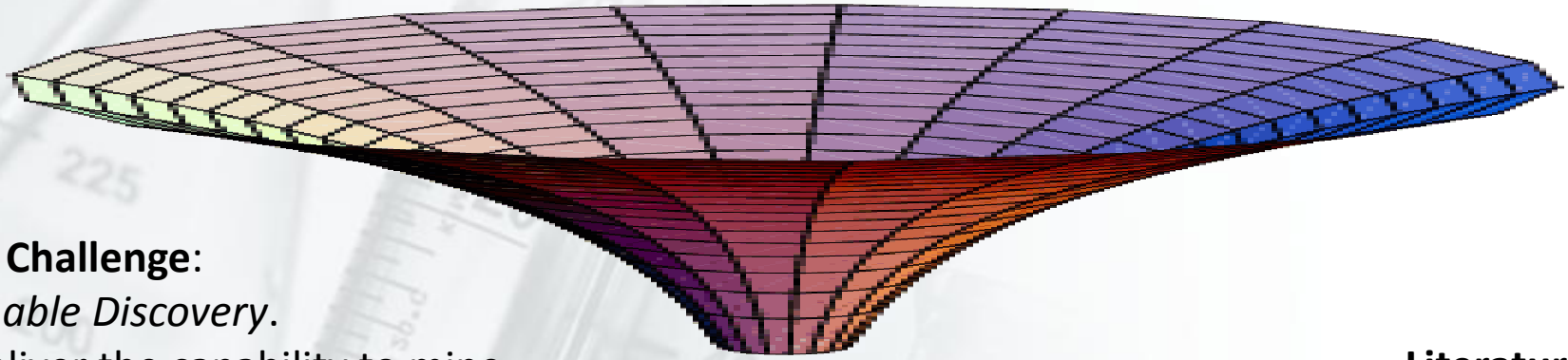
Simulations



Archives

Literature

Instruments



The Challenge:

Enable Discovery.
Deliver the capability to mine, search and analyze this data in near real time.

Enhance our Lives
Participate in our own heath care. Augment experience with deeper understanding.

Petabytes
Doubling every 2 years

Literature

The Current Cloud Challenge

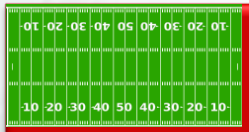
- The current driver: how do you
  - Support email for **375 million users**?
  - Store and index **6.75 trillion photos**?
  - Support **10 billion web search queries/month**?
- And
  - deliver deliver a quality response in **0.15 seconds** to millions of simultaneous users?
  - **never go down**?

Data-driven software engineering

Clouds are built on Data Centers

- Range in size from “edge” facilities to megascale.
- Economies of scale
  - Approximate costs for a small size center (1000 servers) and a larger, 100K server center.

Technology	Cost in small-sized Data Center	Cost in Large Data Center	Ratio
Network	$95 per Mbps/ month	$13 per Mbps/ month	7.1
Storage	$2.20 per GB/ month	$0.40 per GB/ month	5.7
Administration	~140 servers/ Administrator	>1000 Servers/ Administrator	7.1



Each data center is
11.5 times
the size of a football field

For small devices

- Collection has to be more transparent
- Twitter and other social ways of fixing are more acceptable
- Phones should just WORK!

Conclusions

- The law of large numbers has helped in ten years of WER usage
- It will help as move into the world of data-driven development and mobile applications on a large scale
- Maintenance has been a huge success story where research has helped product teams and rolled out to billions of users worldwide, and continues to do so

- Thanks to colleagues at Microsoft especially Galen Hunt, Barbara Gordon, Tom Ball, Nachi Naggapan, Sebastian Burckhardt, Dennis Gannon, Greg Caldwall

Microsoft Research