

Software Bugs

Debugging Your Mindset

Alisson Sol

Development Manager

Microsoft Research Cambridge Innovation Development

Overview

- Who am I?
- Conclusions
- Let's talk about bugs and debugging
- Conclusions again...
- Questions



Microsoft
BizTalk
Server 2002
December 5, 2001

Microsoft
Business
Solutions
Business Portal 1.0
April 15, 2003

Microsoft®
Office Solution Accelerator
for Proposals
Version 1.0
October 16, 2003

 Microsoft
Office
Information Bridge
Framework
Version 1.0
June 30, 2004

 Microsoft®
Office
November 3, 2006


Microsoft® Research
AutoCollage
July 25, 2008

Microsoft
Business
Solutions
Business Portal 1.0
April 15, 2003

Microsoft
Office
Information Bridge
Framework
Version 1.0
June 30, 2004


Microsoft® Research
AutoCollage
July 25, 2008

Conclusions

- Good debugging starts with good bugs
- Individual bug requires patience
- Deal with the “bug load” urgently
- Do not trade the bug that you know for the bug that you don't know
- Discipline pays off

Bug?

- Developers think just of “coding errors”
- But for users...
 - This applications doesn't do <this>
 - The application does <this> but the side-effect is <that>

The Drawing Won't Rotate...

- Scenario
 - Working as a consultant in a mining company
 - Several professionals used CAD tools
 - Called to help to “rotate a drawing”



The Main Rule of Debugging

- Every assumption may be wrong
 - It is harder to doubt than you think
 - Humans “want to believe”
 - You cannot see the facts if blinded by your assumptions
 - Main benefit of adding new testers late in a project: they don’t know about the “accepted bugs”

Bug Tracking Tool

- Bug Identification and Status
 - Title, status, dates, etc.
- Repro steps
 - Steps to arrive at issue
 - Expected results
 - What is happening
- Additional information
 - Environment, hints

Is The Customer Wrong?

- The customer may at most be misinformed
 - Who had to inform the customer?
 - There is no such thing as “intuitive user interface”
 - Software manuals: read as much as car manuals
 - Software help files: between obvious and obscure
 - Games: tutorial mode

My Simple Taxonomy

- Bugs and sizes
 - Small, medium, big (by total cost of fixing)
- Bug sources
 - Mistakes, integration, miscommunication
- Bug status
 - Active, resolved, closed

Coding Bugs

- Typically small mistakes
 - Off-by-one
 - Copy & paste
 - Uninitialized variable
- Inevitable
 - The main reason for hiring testers
- Technical interviews focus too much on this
 - Off-by-one error. So what?

Developing Together

- Scenario
 - Need to enumerate files from a folder
 - Files could be copied to folder at any time
 - No problem if file is enumerated twice
 - Request: Enumerate every file again if new one is found
 - Should not miss file recently copied

Code Review

```
FileInfo[] directoryFiles = directoryInfo.GetFiles();
do
{
    long ticksAtStart = System.DateTime.Now.Ticks;
    foreach (FileInfo fileInfo in directoryFiles)
        yield return fileInfo.FullName;

    directoryFiles = directoryInfo.GetFiles();
    long ticksAtLastUpdate = ticksAtStart;
    foreach (FileInfo fileInfo in directoryFiles)
        if (fileInfo.LastWriteTime.Ticks > ticksAtLastUpdate)
            ticksAtLastUpdate = fileInfo.LastWriteTime.Ticks;
} while (ticksAtStart < ticksAtLastUpdate);
```

Integration Bugs

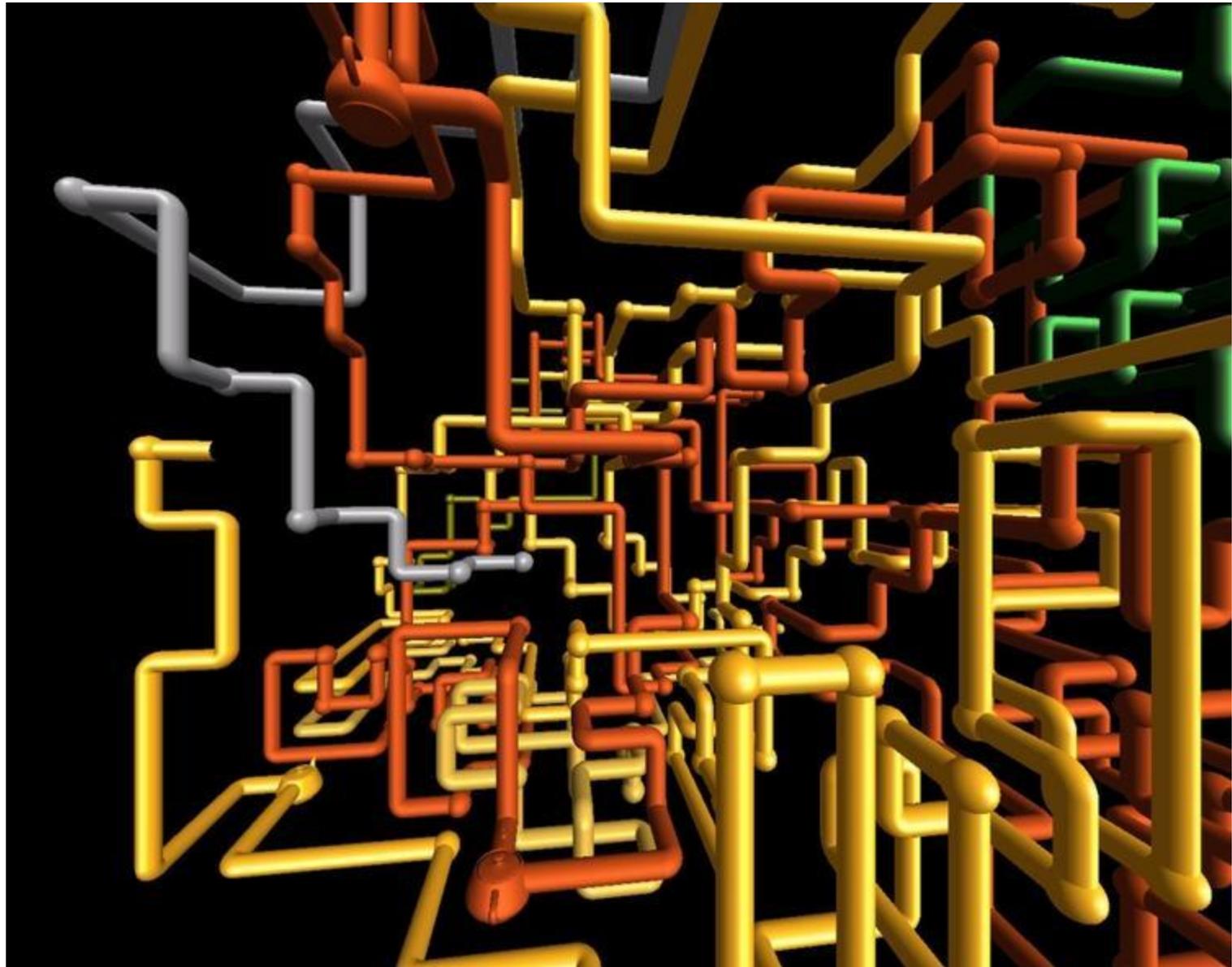
- Analogy: car tire in airplane, or vice-versa
- The reason for the stabilization period
- Depend on a lot of ad-hoc testing
 - Mitigation: alphas, betas, release candidates
- Integration is not only in the present
 - Anticipate future deployment issues

APIs... They Work, and Then...

- Scenario
 - Application with two buttons
 - Button1 moves square 100 pixels to the right
 - Button2 animates square 100 pixels to the left

Observer Effect

- Scenario
 - Client-server application (2-tier)
 - Client: Windows client application
 - Server: Database server on Windows NT
 - Performance deteriorated after a few minutes
 - Performance would be normal when debugging



The Importance of External Testing

- Scenario
 - Add-in for Excel
 - Managed code (dependency on .NET 2.0)
 - Works perfectly within team
 - Several “test add-ins” developed to test “compatibility”
 - Fails on internal alpha

A Bit About Process

- Each individual bug is important
- The trend is extremely important
 - The trend defines stability
- Stable software may still be bad software
 - Unstable software is guaranteed to be bad!
- Almost never it is “The End”
 - Almost always “To be continued...”

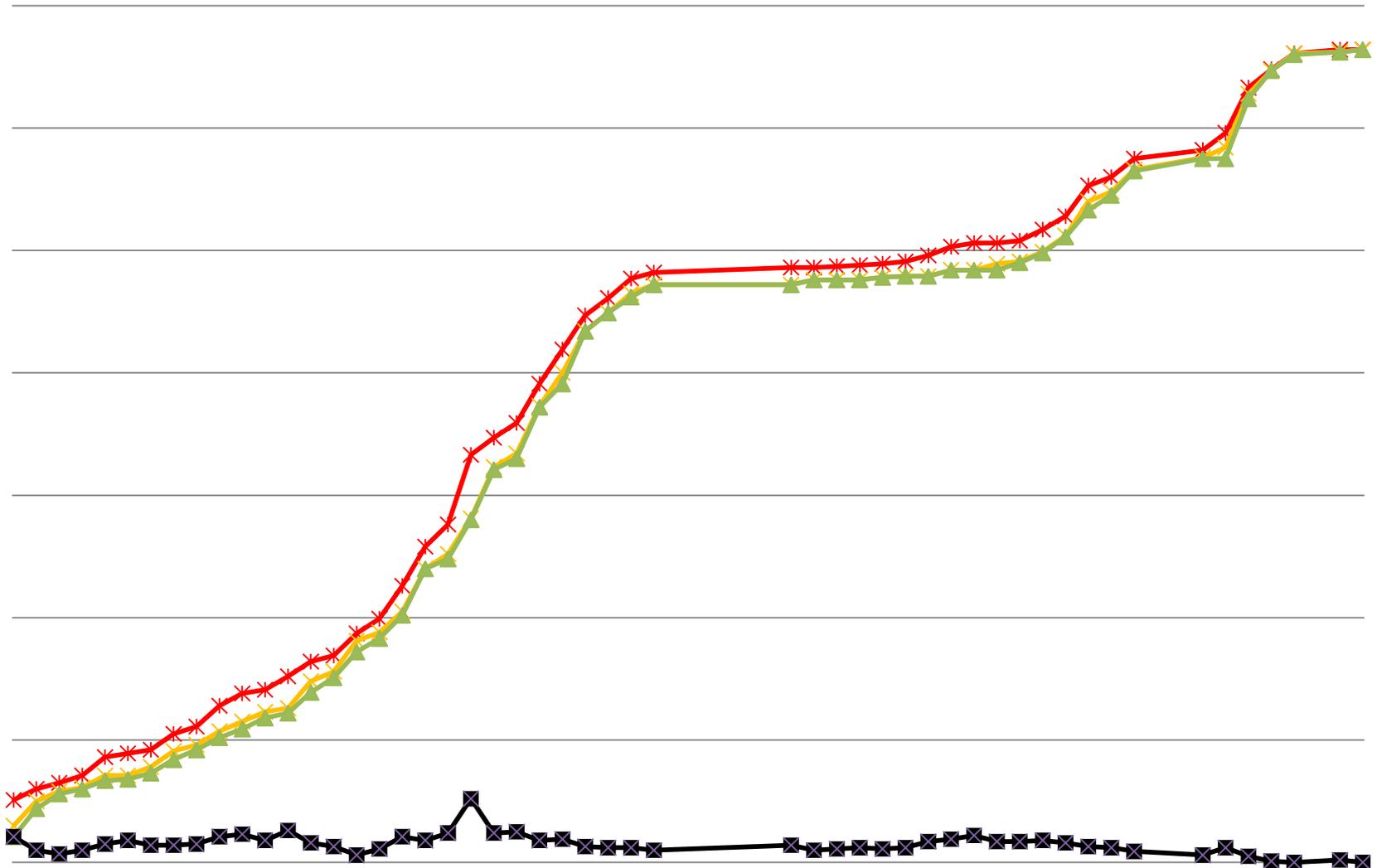
Individual Bug Process

- Opened
 - Opened by, bug details
- Triage
 - Severity x Priority
- Resolved
 - Counted: Fixed, External, Postponed, Won't Fix
 - Not counted: By Design, Duplicate, Not Repro
- Closed
 - Resolution verified by test team

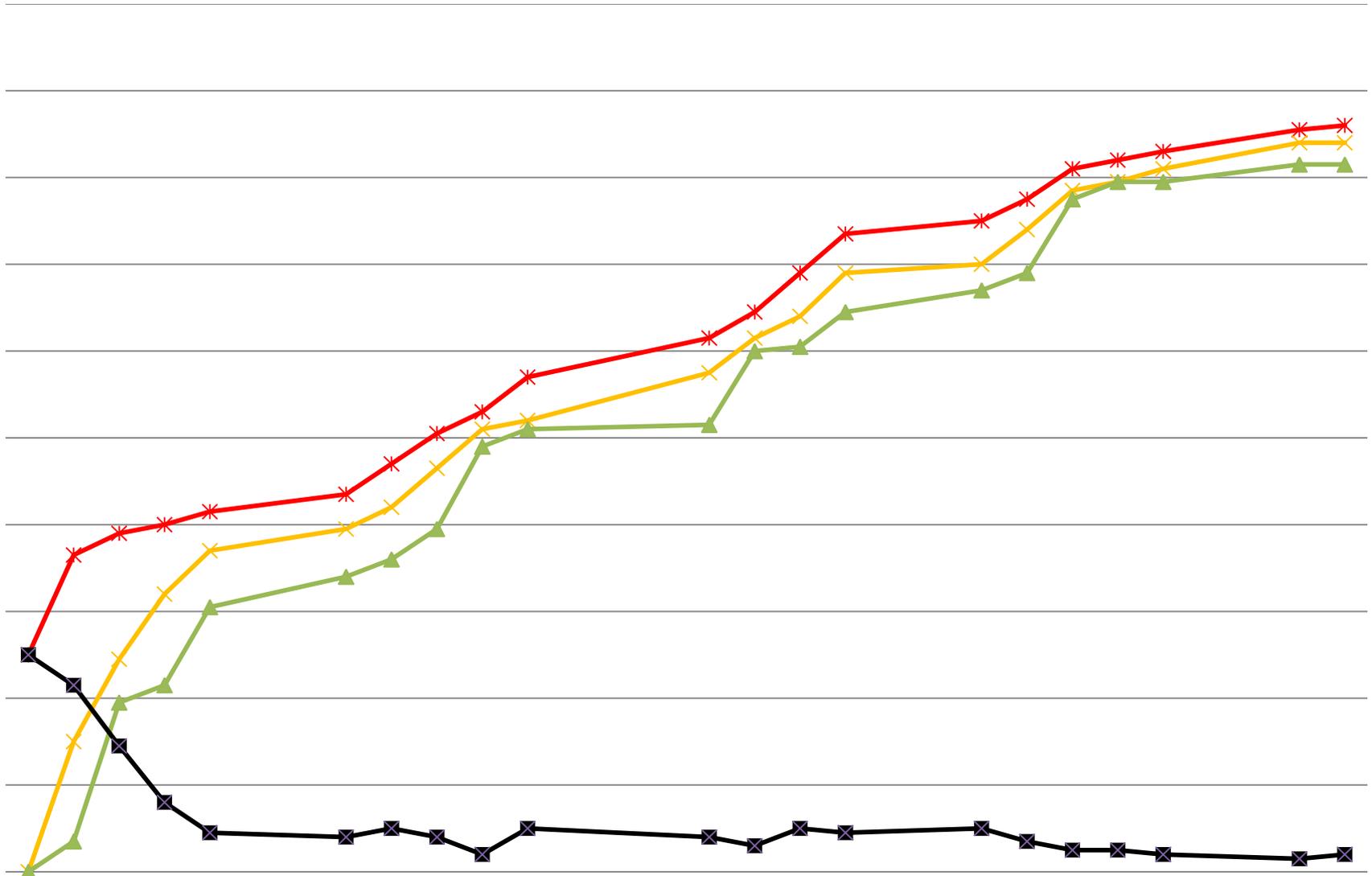
Bug Queries

- Single Bug Status
 - Active
 - Resolved
 - Closed
- “Bug load”
 - Active + Resolved
- Quantitative Analysis
 - Opened 
 - Resolved 
 - Closed 
 - Active 

Bugs x Time: Recent Project 1



Bugs x Time: Recent Project 2



Addressing the “Bug Load”

- Promptly
 - “Active” bugs should be assigned to “individual”
 - Exception: tracking issues, like time bomb removal
- Bug jail
 - Forecast time to zero bugs, and work backwards
- High level strategy
 - Cut features
 - Rework the component
 - Delay or cancel release

Discipline is Freedom

- Lack of discipline, and the problems
 - Application shipped with a time bomb
 - Last minute change added virus to sample
 - Country code fixed on the binary
 - Developer added a “minor feature”
 - Test harness used undocumented API

Conclusions Again...

- Good debugging starts with good bugs
- Individual bug requires patience
- Deal with the “bug load” urgently
- Do not trade the bug that you know for the bug that you don't know
- Discipline pays off

Thank You!

- Questions?