



James Larus
Microsoft Research
ISSTA, July 22, 2008

THE REAL VALUE OF TESTING

(OR, WHAT I'VE LEARNED IN THE PAST DECADE)

If Only We ~~Knew~~ Listened..

“The real value of tests is not that they detect bugs in the code but that they detect inadequacies in the methods, concentration, and skills of those who design and produce the code.

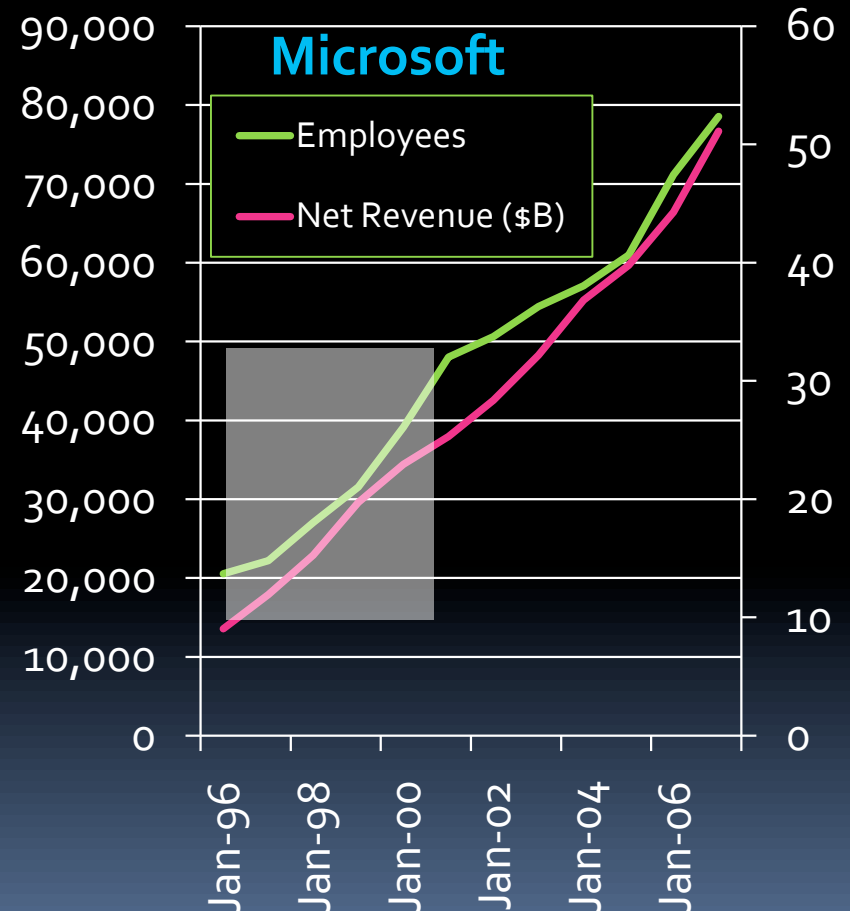
- Tony Hoare, *How did software get so reliable without proof?*, FME '96

A Bit of History

- I went to MSR on sabbatical, summer 1997
- Decided to stay, spring 1998
- Started SPT – Software Productivity Tools
 - Amitabh Srivastava joined to start PPRC – Programmer Productivity Research Center
 - Yuri Gurevich joined to start FSE – Foundations of Software Engineering

Why Fervor Around SWE?

- Microsoft struggling to ship Windows 2000
 - Company growing rapidly
 - Software growing rapidly
- Software development not evolving

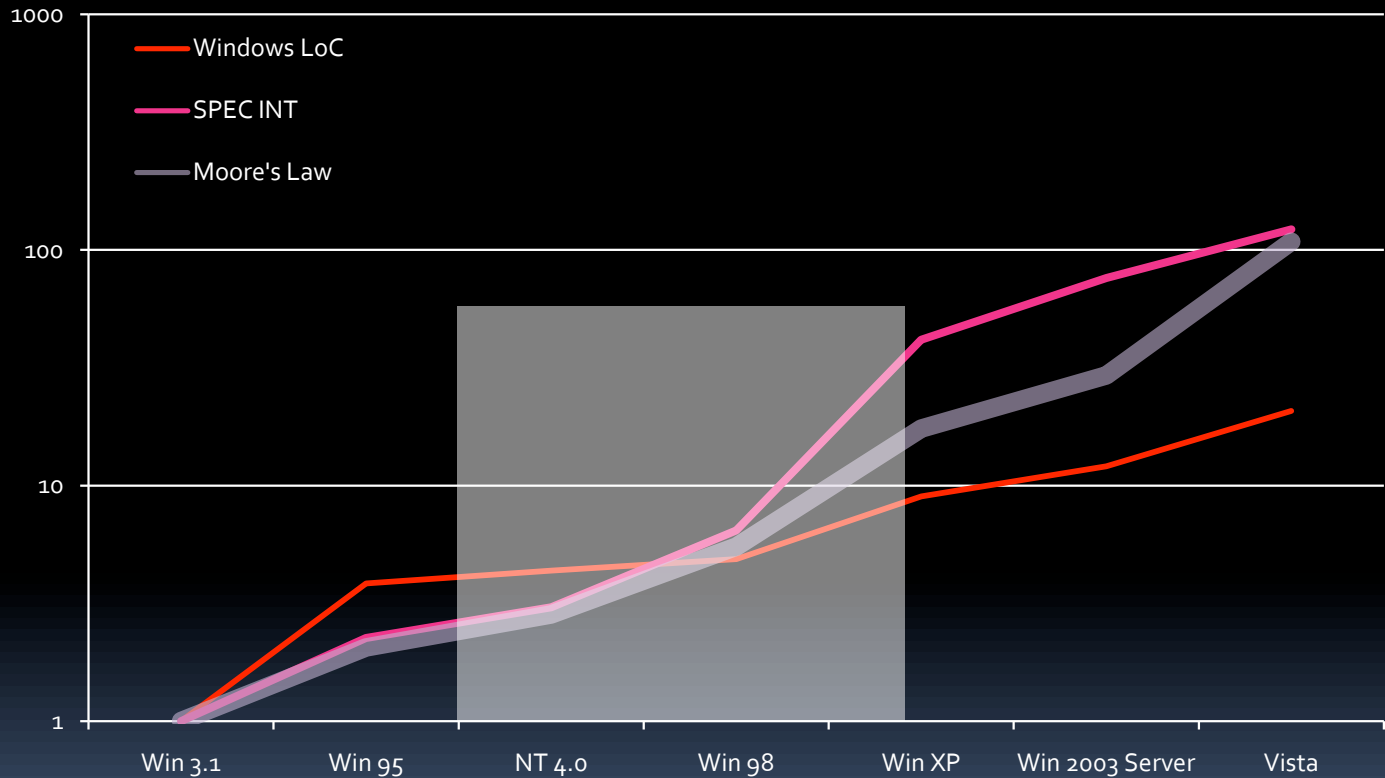


"Fast Facts About Microsoft"

http://www.microsoft.com/presspass/inside_ms.msp

• Jim Larus • Microsoft Research •

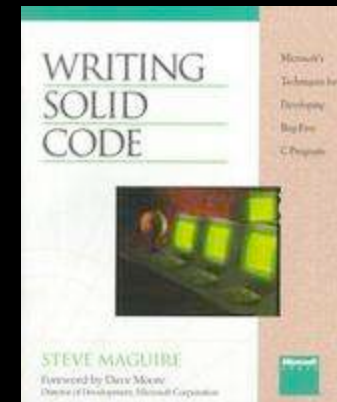
Growth in Software Size



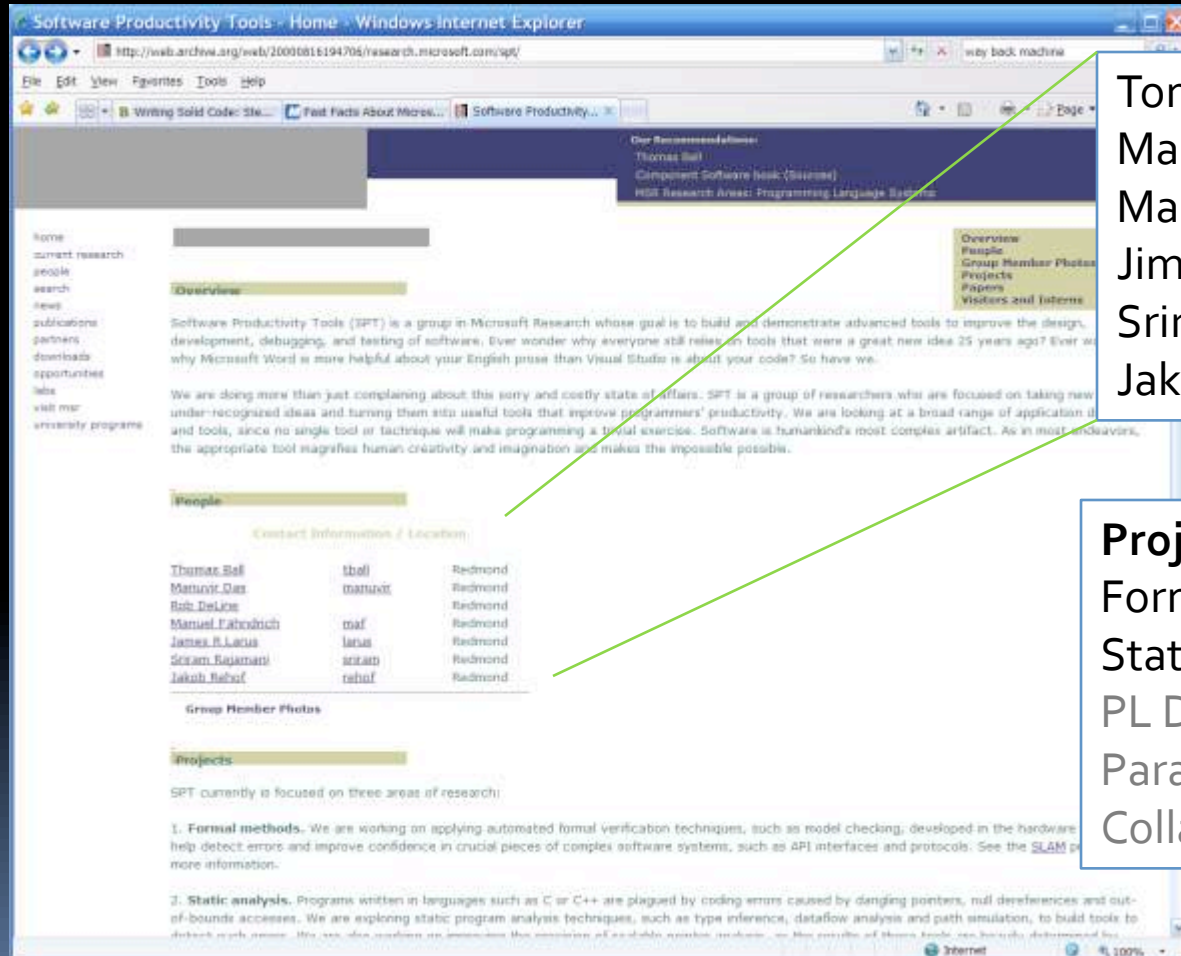
Wikipedia estimates of LoC. Does not measure code shipped to customers.
SPEC normalized between SPEC95 and SPEC2000.

Cultural Challenges

- Small company development culture under stress
 - “Why is only 25% of my time spent writing new code?” – *Developer Productivity Offsite, 1999*
 - “Stop the bugs!” – *ibid.*
 - “Do we focus on wrong things during development?” – *ibid.*
 - “Why do we write so much code during integration?” – *ibid.*



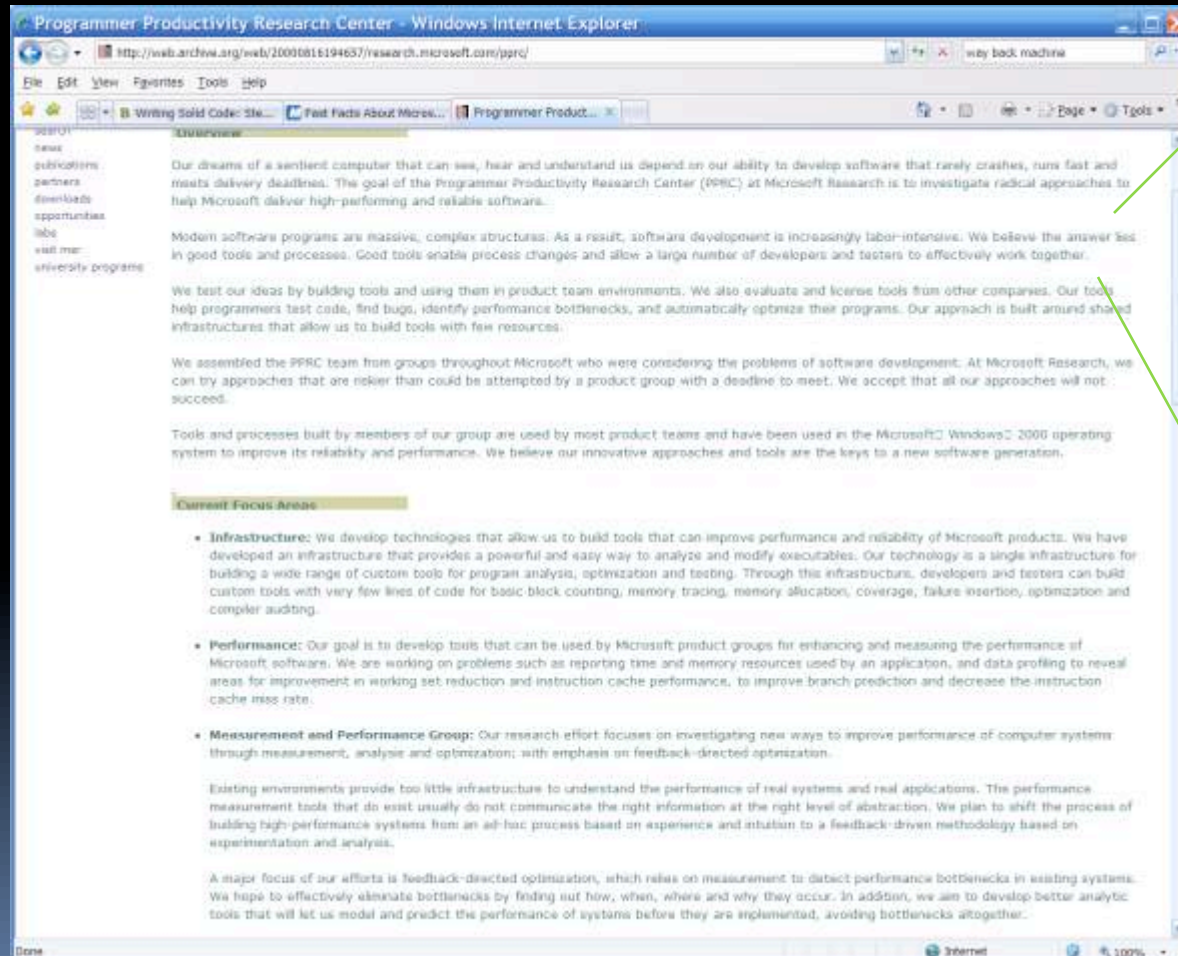
SPT, c. 1998



Tom Ball
Manuvir Das
Manuel Fahndrich
Jim Larus
Sriram Rajamani
Jakob Rehof

Projects
Formal methods
Static
PL Design
Parallel & Distributed Tools
Collaborative Programming Tools

PPRC, c. 1999



We believe the answer lies in good tools and processes. Good tools enable process changes and allow a large number of developers and testers to effectively work together.

Why Emphasis on Defects & Tools?



- Defect detection
 - Bugs seen as biggest problem in SW development
 - Enormous effort spent stabilize products (fixing bugs)
- Developers in denial about need for change
 - “Best in the world”
 - Only saw manifestations (bugs) of deeper problems
- Most tolerated tools
 - “Productivity” improvement left developers in control
 - Process change too radical
 - (Especially from pointy headed academics in MSR)

Prefix

- MS purchased Intrinsa and moved Jon Pincus and company into PPRC
- Prefix was ad-hoc, but extremely scalable, defect detection tool
 - Interprocedural analysis across 10's millions of LoC
- Found "usage" bugs
 - Null pointer, double free
 - Ignored error codes, style issues
- Batch process
 - Weekly run produced 10k's bugs
 - High false positive rate
- Identifying real and important bugs was challenge

Prefast

- Realization of earlier work by Daniel Weise (ASTToolkit)
 - Front-end of C++ compiler with clean AST
 - Plug-ins traverse AST, looking for specific patterns
 - Structural “grep”
- Enormously popular
 - Desktop checking (fast & private)
 - Easy to customize for specific problem
- Foundation for sophisticated tools
 - Full C++ parser integrated in MS build process
- Shipped in Visual Studio

ESP

- Manuvir Das's effort to put Prefix on solid analytic foundation
 - Scalable, flow-sensitive program analysis
 - Techniques were publishable research
- Enormous effort to turn into a useful tool
 - Engineering
 - "Good enough" competitors
 - Global analysis is expensive
 - C++ is messy
- Paid off for security push
 - Could demonstrate the absence of bug

SLAM

- Tom Ball & Sriram Rajamani
- Major research innovation
 - Software model checking
 - Counter-example driven refinement
- Embodied in Static Driver Verifier (SDV)
 - Checks library of intricate rules for device drivers
 - Shipped 5 years after start of work
 - Many happy third party driver writers

Spec Explorer

- Wolfram Schulte & FSE team
- Model-driven testing tool
 - Innovative research
 - Enormous amount of engineering
- Little acceptance
 - Specification is a foreign concept to most
 - Testers most enthusiastic, but least skilled
 - Did not fit development process at MS
- Successfully used for EU protocol specification

SLAMMER



- Buffer overruns! Where did they come from??
- An OMG moment for MS
 - Stopped all development for 1-2 months for security training
 - Inspected all software
 - Threat modeling became part of development
 - PPRC moved into Windows
 - Defect detection tools focused on buffer overruns
 - 4-5 years of effort culminating in Vista
- This is not productivity

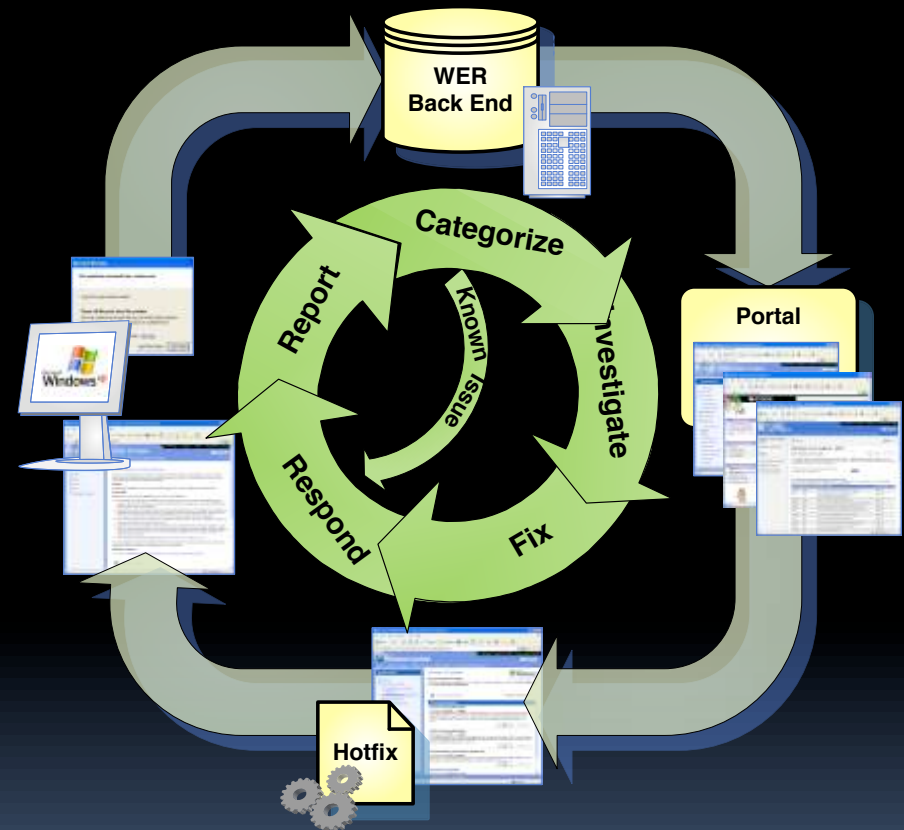
Lessons of SLAMMER



- Tools can't find or fix all fundamental flaws
 - If you don't look for it, you will not find it
- Very hard to add quality to finished product
 - Detroit in 1970's & 80's
- Education and process key to change
 - Oh, people write software?
- Crisis can change direction of a big ship
 - But, very painful way to steer

Watson

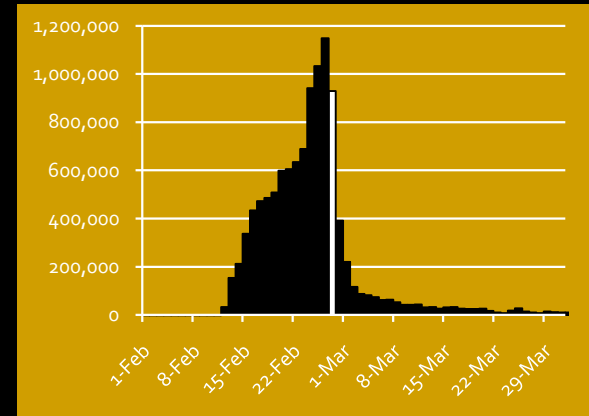
- Microsoft error reporting system
 - Report errors from user's machine to Microsoft
 - Not just MS software (7000 products)
 - 42% of computers generate report
- Greatest improvement in SW development



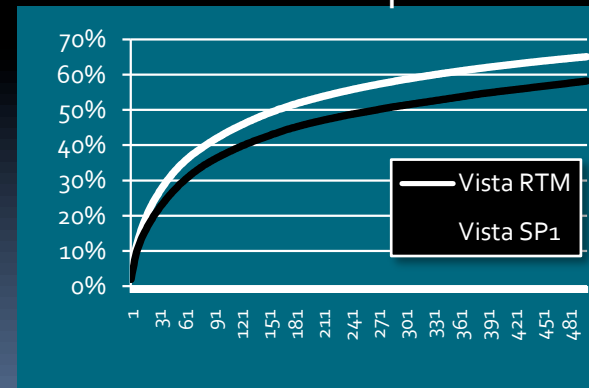
Lessons of Watson

- Find errors in real time
- Prioritize effort by customer benefit
 - Heavily skewed distributions
 - No question of importance
 - Maximum benefit from fixed budget
- Close loop by distributing patches

Renos Malware



% of Error Reports

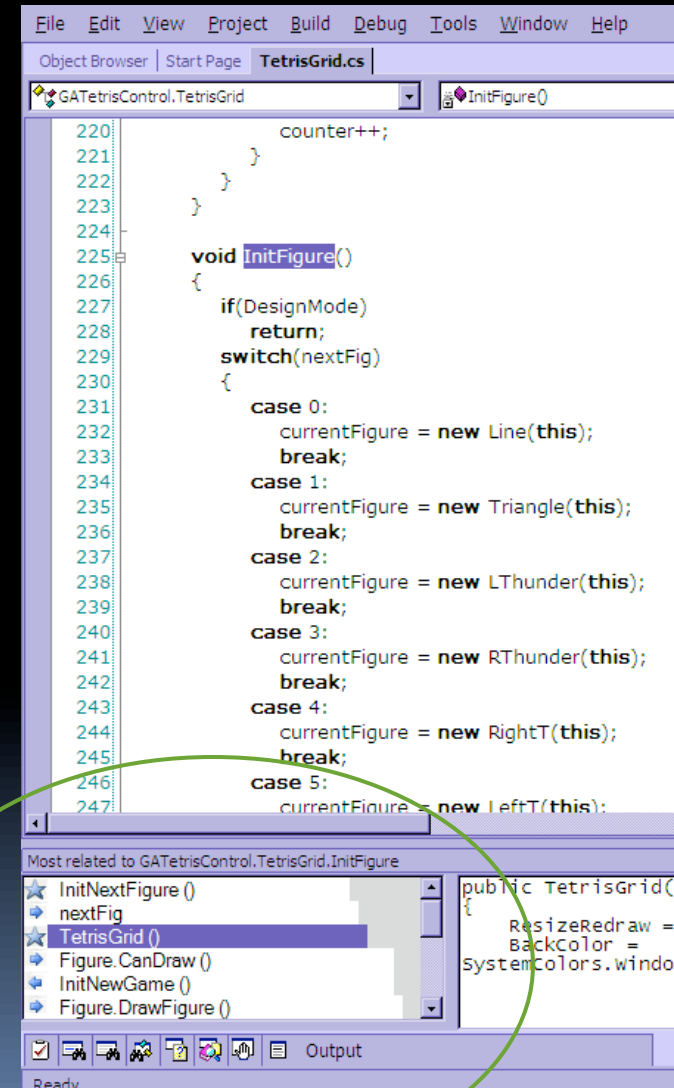


MSR Becomes HIP

- Rob DeLine started Human Interactions in Programming (HIP)
 - “We study programming, as if it was done by people”
- Different approach to software development
 - People and organization (vs. code)
 - New methodology
 - Empirical research (survey, interview, observe)
 - Tools address identified problems
 - Rigorous user testing of tools
- #1 problem is information discovery (not bugs!)

TeamTracks

- How do you find your way around big/new code base?
 - Recommender system based on what other people look at
- Improved task completion rates
 - Task 3 (localized code):
1 / 7 without
3 / 9 with Team Tracks
 - Task 4 (dispersed code):
1 / 7 without
7 / 9 with Team Tracks
- Group 2 quiz scores 18% higher



The screenshot shows the Visual Studio IDE with the file TetrisGrid.cs open. The code editor displays the following code:

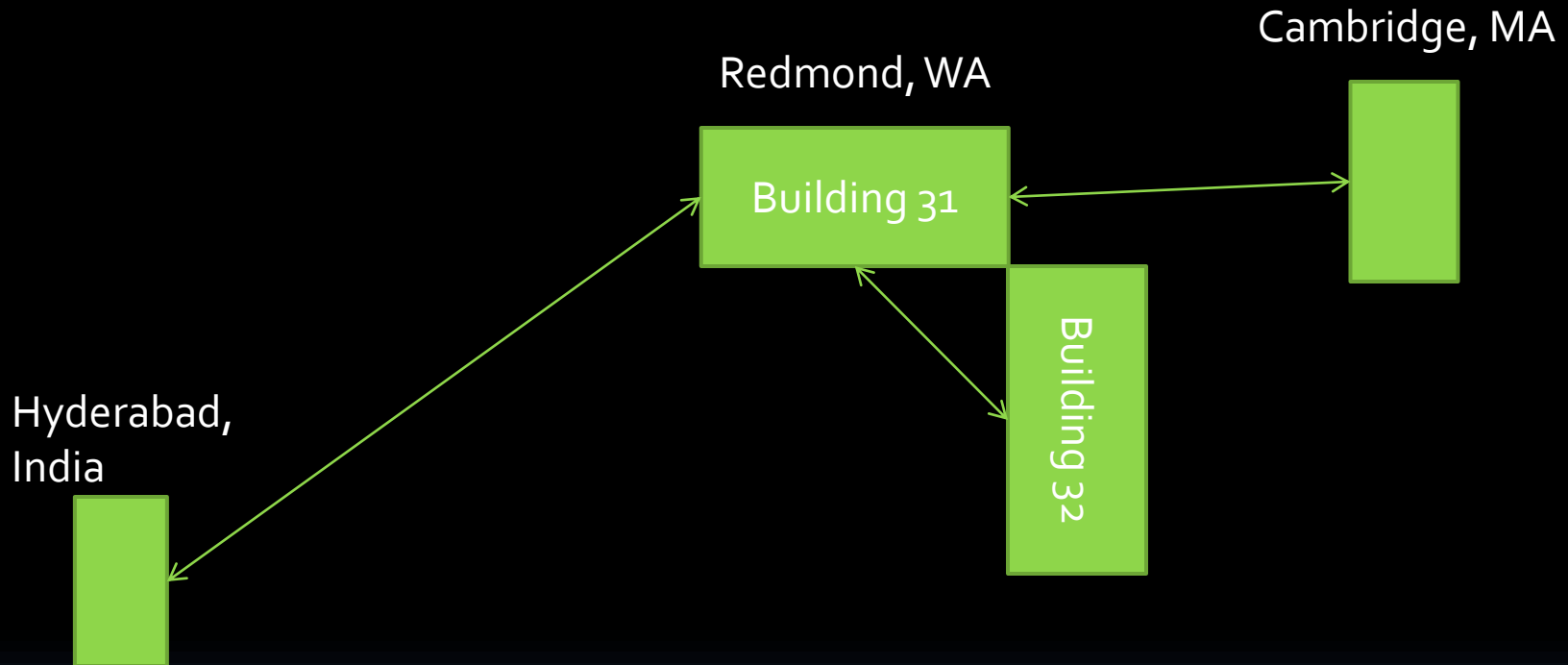
```
220         counter++;
221     }
222 }
223 }
224
225 void InitFigure()
226 {
227     if(DesignMode)
228         return;
229     switch(nextFig)
230     {
231     case 0:
232         currentFigure = new Line(this);
233         break;
234     case 1:
235         currentFigure = new Triangle(this);
236         break;
237     case 2:
238         currentFigure = new LThunder(this);
239         break;
240     case 3:
241         currentFigure = new RThunder(this);
242         break;
243     case 4:
244         currentFigure = new RightT(this);
245         break;
246     case 5:
247         currentFigure = new LeftT(this);
```

The Object Browser at the bottom shows the following methods related to GATetrisControl.TetrisGrid.InitFigure:

- InitNextFigure ()
- nextFig
- TetrisGrid ()
- Figure.CanDraw ()
- InitNewGame ()
- Figure.DrawFigure ()

A green circle highlights the Object Browser and the code lines 225-247.

Team Coordination Work



SPT → SRR

- Tom Ball took over SPT and redirected it
- Nachi Nagappan started empirical software research
 - Mine wealth of unexamined data
 - Build predictive models to guide development
 - Process change from the inside
 - Show teams what is really happening, and they adapt
- Patrice Godefroid increased testing research

Failure Prediction Models

- Early models based on conventional inputs
 - Bugs found by Prefix/Prefast
 - Code churn
 - Complexity metrics
- More recent models are people focused
 - Organizational metrics
- Enormously popular and in demand
 - Sharp contrast to defect tools

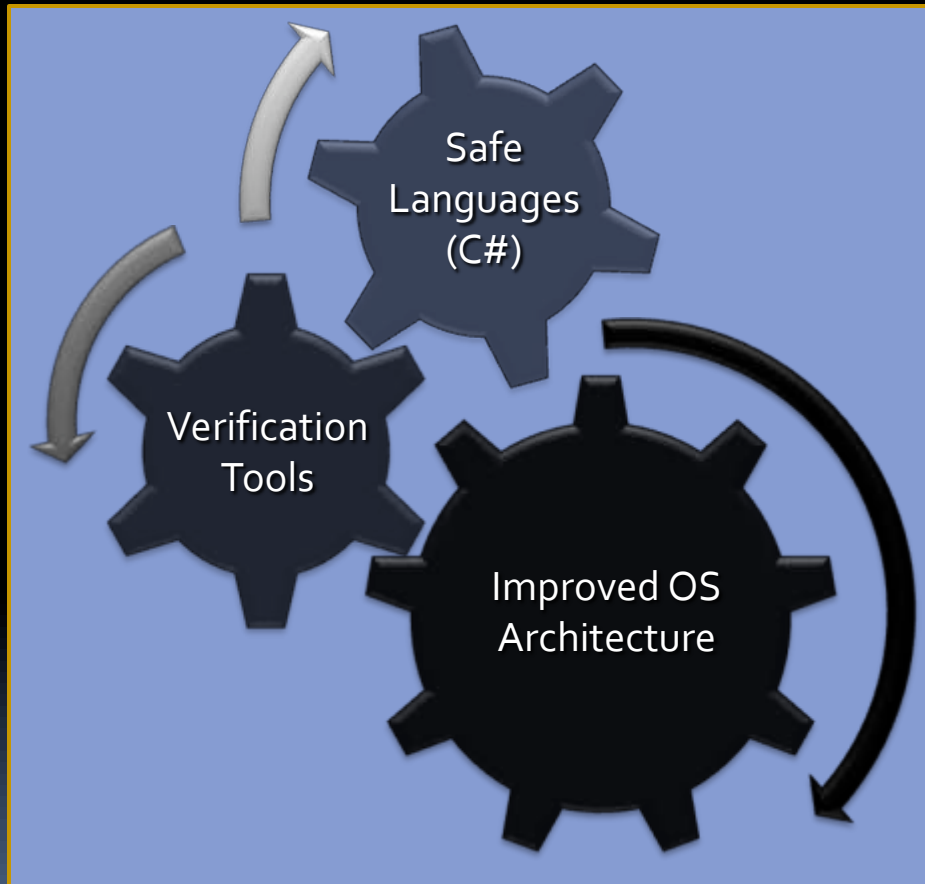
Organizational Metrics Study

| Model | Precision | Recall |
|--------------------------|-----------|--------|
| Organizational Structure | 86.2% | 84.0% |
| Code Churn | 78.6% | 79.9% |
| Code Complexity | 79.3% | 66.0% |
| Dependencies | 74.4% | 69.9% |
| Code Coverage | 83.8% | 54.4% |
| Pre-release Bugs | 73.8% | 62.9% |

Prediction of Windows Vista binaries (3404) as failure-prone (above lower confidence bound for all failures).

-- Nagappan, Murphy, Basili, "Influence of Organizational Structure on Software Quality," *ICSE 2008*.

Singularity



- Microsoft Research project with goal of more robust and reliable software
- James Larus & Galen Hunt
- Rethink the software stack
- Articulated architectural principles
 - Software will fail, system should not
 - System should be self-describing
 - Verify many aspects as possible
- No single magic bullet
 - Mutually reinforcing improvements to languages and compilers, systems, and tools

Renewed Interest in Testing

- Can't statically analyze parallel programs
 - Fundamental limitation (?)
 - Systematic testing (Chess) is very effective
- Random and fuzz testing complement people
 - Also to Black Hat community
- Combined static analysis and run-time exploration offers advantages of both
 - Concrete error traces (and no false positives)
 - Systematic coverage

Back to Tony Hoare

“The real value of tests is not that they detect bugs in the code but that they detect inadequacies in the methods, concentration, and skills of those who design and produce the code.

- Tony Hoare, *How did software get so reliable without proof?*, FME '96

Improving Software

- My original view was simplistic
 - (Not unproductive, however)
- Tools, by themselves, do not improve software
 - People improve software
 - Tools can reinforce process change
- Defect detection is inferior to defect prevention
 - Starting with architecture
 - Improve organization
 - Finally, improve practice
- Must understand people and organizations develop SW
 - Communication and information discovery central
 - Specifications not natural

Conclusion

- SW development is too complex to be reduced to a single problem or solution
 - Analysis tools will not perfect software
 - But, nor will thousands of eyeballs
- Balance the strengths of each approach
 - Unthinking enthusiasm for technology impedes solutions
- Testing is quality metric after everything else is done
 - Feedback closes development loop
 - Final opportunity to fix code should be less important