Deploying Enterprise-Wide Program Analysis Tools:

Challenges and Opportunities

John Penix
Google

Fun with Java

```
i = i++;
int x = Math.ceiling(i / 100);

String date = getDate();
date.trim();

if (user.country().equals(“USA”))
   (what is the return type of country()?)
```

Fun with Java

```
Map<int, String> myMap;
myMap.put(“one”, “apple”); // error!
myMap.get(“one”);          // ok!?

if (x == null)
  new NoRecordException("no ” + x.name());


System.out.println(“int: ” + myInt);
System.out.println(“array: ” + myArray);
```

FindBugs

Open source static analysis tool for Java

Developed at U. Maryland by Bill Pugh and others

Guiding philosophy:

- Find a mistake developers are making in practice
- Encode it in a rule
- Try for zero false positives

Technology:

- linear byte code scanning
- minimal interprocedural null tracking

Goals and Objectives

Static Analysis Team Vision:

"Analyze every version of every file in every context."

"Everywhere there is code, there are warnings."

Success factors to impact the development process:

- Timely analysis
- Precise analysis (and false positive suppression)
- Integrate the warnings into the workflow

Software Development at Google

Paradigms

Continuous integration – no surprises downstream

No binary incompatibilities in production

Encourage reuse and SOA

Implementation

One open code repository for all projects

No binary releases (always build from “head”)

Many dependencies throughout entire code base

Goals and Objectives

Static Analysis Team Vision:

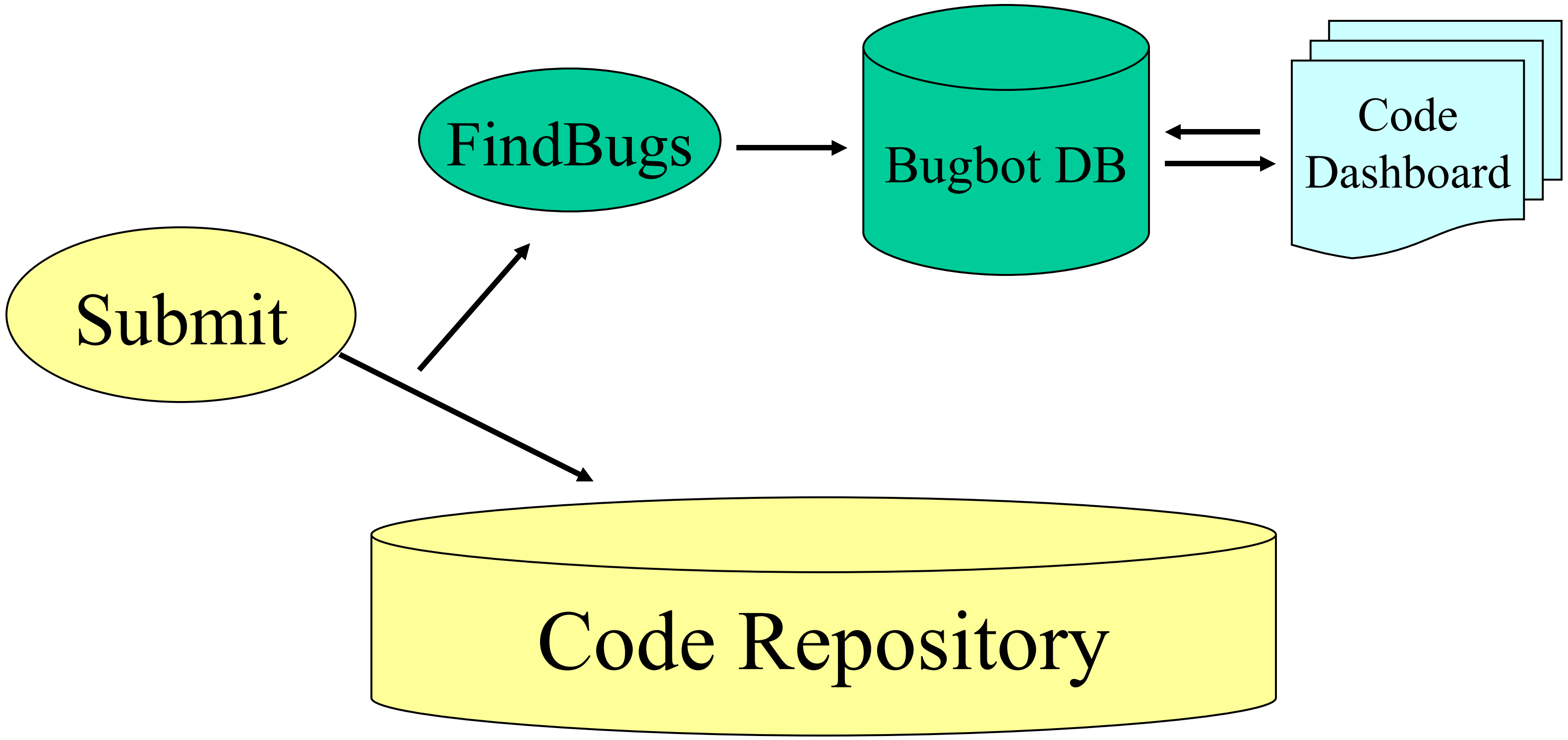
"Analyze every version of every file in every context."

"Everywhere there is code, there are warnings."

Success factors to impact the development process:

- Timely analysis
- Precise analysis (and false positive suppression)
- Integrate the warnings into the workflow

BugBot 2006



Cost Analysis

Usage Cost = Analyst Time x Labor Rate

Analyst Time = Number of Warnings x **Triage Time per Warning**

Number of Warnings: Estimate based on code base size and the number of warnings seen during a tool evaluation:

Lines of Code x Observed Warning Density

Triage Time per Warning: **Time ranges from less than 1 minute to up to 20-30 minutes in rare cases**. We estimated the average as 5 minutes.

Triage of false-positives drives up the cost of finding a real bug.

Comparing Actual Costs to Estimated Costs

Metric	**Estimate**	**Actual**
Lines of Code	X million	**XX million**
Interesting Warnings	1500	**5000**
Time to Evaluate	5 min/wrn	~8 min/wrn
Time to Triage	125 hours	650 hours
False Positive Rate	75-80%	**~55%**
True Positives	250 - 400	**~1200-1500**
Cost/True Positive	~$55 – 70	~$66-80

Static Analysis Service

Ran as a service for 6 months:

- Triaged several thousand warnings.
- Filed over 1000 bugs in bug tracker
- Over 700 bugs fixed
- Developed a ranking scheme based on
  - false positive rate
  - likely hood of being fixed when filed

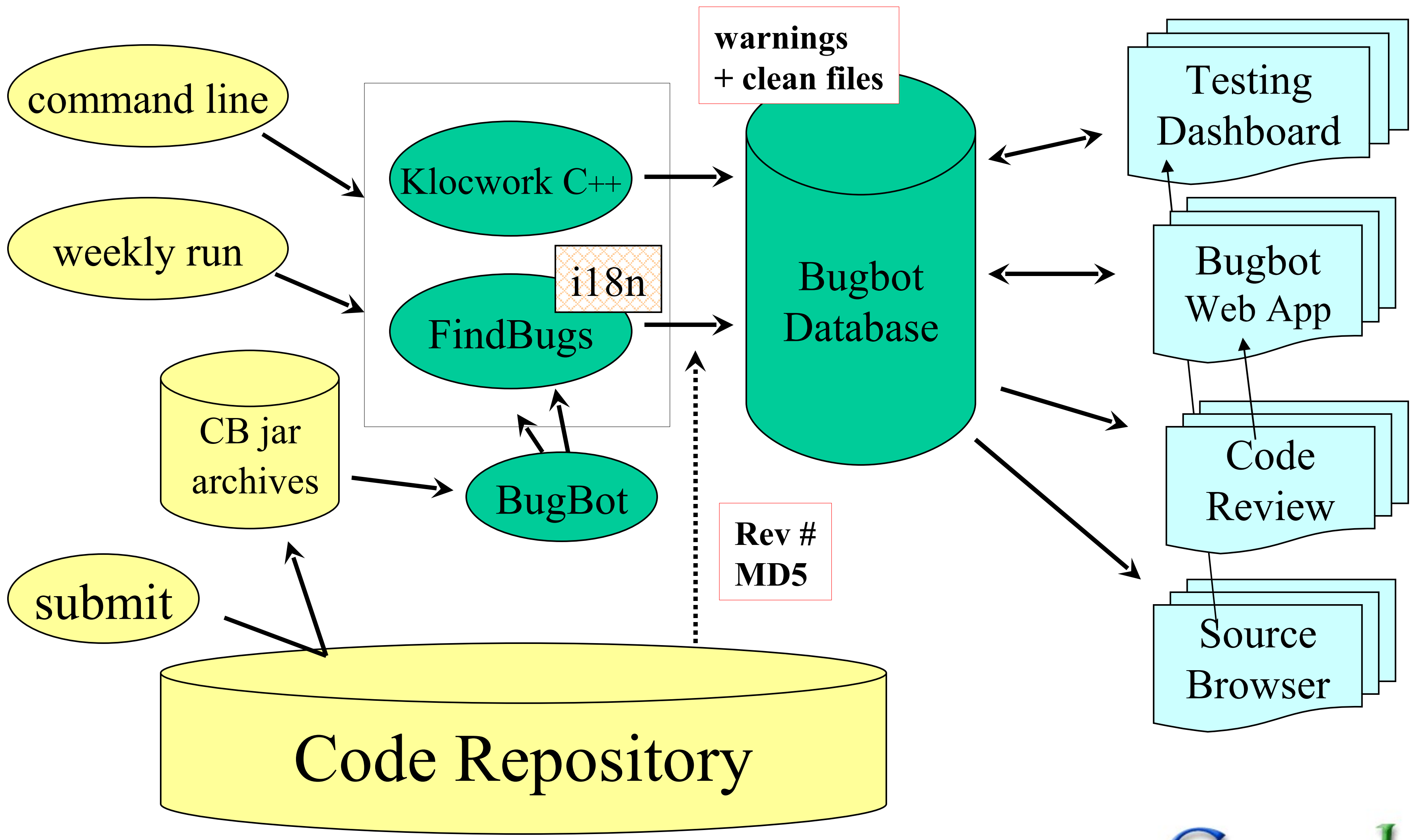
“Evaluating Static Analysis Defect Warnings on Production Software”, N. Ayewah, W. Pugh, J.D. Morgenthaler, J. Penix, Y. Zhou, Proceedings of the 7th ACM SIGPLAN-SIGSOFT Workshop on Program Analysis for Software Tools and Engineering, 2007, pp. 1-8.

Why don't developers fix bugs?

1. Didn't assign the bug to the right person.
2. The results are stale.
3. Bug has little impact: old code, logging code, testing code
4. False positives make people discount or ignore the tools
5. Developers don’t understand the bug – think it is esoteric (unlikely) or hard to fix.

“Experiences Using Static Analysis to Find Bugs”, N. Ayewah, D. Hovemeyer, J.D. Morgenthaler, J. Penix, W. Pugh, IEEE Software, vol. 25 (2008), pp. 22-29.

BugBot 2008


command line
weekly run
CB jar archives
submit
Klocwork C++
i18n
FindBugs
BugBot
warnings + clean files
Bugbot Database
Rev # MD5
Code Repository
Testing Dashboard
Bugbot Web App
Code Review
Source Browser

Why don't developers fix bugs?

1. Developers don't know the bug is there.
2. The results are stale
   - integration with build system
   - “whole program analysis” needs to be incremental
3. The bug has little impact: old code, logging code, testing code
4. False positives make people discount or ignore the tools.
5. Developers don’t understand the bug – think it is esoteric (unlikely) or hard to fix.

Why don't developers fix bugs?

1. Developers don't know the bug is there.
2. The results are stale
3. The bug has little impact: old code, logging code, testing code
4. False positives make people discount or ignore the tools.
   - "I don't care" == "false positive" to a developer
   - separate "style" checks from defect checks
   - start with the best warnings to get buy-in
5. Developers don't understand the bug – think it is esoteric (unlikely) or hard to fix.
   - Sometime the bugs are subtle: http://www.cs.umd.edu/~pugh/java/memoryModel/DoubleCheckedLocking.html

Statistics and Metrics

1 year

47 Million Reported Warnings

497,295 Unique Warnings

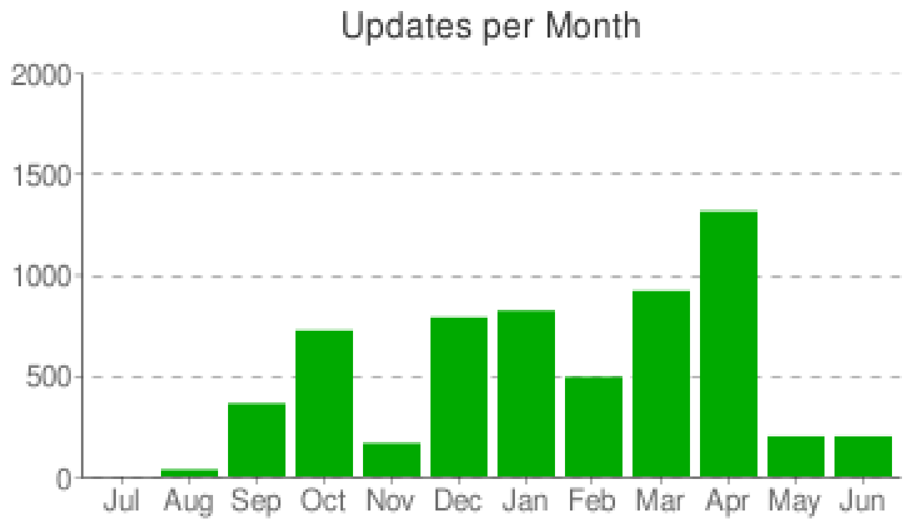
317,374 Tool Runs

930,606 File Revisions

263,298   with Warnings

6352 Status Updates from

431 Users (not us)



Working with Google

- We're hiring
- Summer internships
- Faculty visits
- Tech talks
- Research Grants: $10K - $150K - 3 page proposals

  http://research.google.com/university/relations/research_awards.html
- Summer of Code