

# The Business Case for Static Analysis

Ilya Dreytser

Software Engineer, Technical Account Manager



# Agenda

- Building Security into the SDLC
- Where do Static Analysis Tools Fit in
- Evaluating Static Analysis
- Measuring the ROI of Static Analysis tools
- Adopting Static Analysis – How to do it right
- Case Studies

# Build Security into the SDLC

- *“The software development group is dealing with application security since they are building the applications. I just need to worry about other aspects of security.”*
- *“The security group is dealing with security of our applications since they are the experts.”*

**The reality is that both are required!**

Source: The Seven Deadly Myths of Software Security

# Dual Perspectives on Security

	Security Auditors	Developers
Goal	Find <i>all</i> potential security issues; Assess risk to applications & application infrastructure	Deliver software that meets customers needs with a minimal risk of failure
Strategy	Flag all security violations and potential violations; make sure nothing slips through	Prioritize and resolve security-related issues based on severity, available resources, and schedule
Challenge	Leverage tools to analyze complex applications and complex operating environments	Build security into all phases of the development process at minimal incremental cost

# Build Security into the SDLC

- What do you mean?
  - Security can not be an afterthought – it must be built into the SDLC
- Why is that?
  - Reducing software defects (including security flaws) throughout the SDLC provides significant cost benefits

**Ok, how do we do that?**

# When Does Security Start to Analyze It in?

DESIGN

CODE

TEST

DEPLOY

# How do Static Analysis Tools Fit in?

- Treat Security Vulnerabilities as Software Defects
- Why?
  - Developers understand defects
  - Developers work with the Bug Tracking System
  - Developers are the ones that write (and make changes to) the source code
- So what do Static Analysis tools provide us with?

**Potential Defects!**

What kinds of Static Analysis Tools are there?

**Coding-Style Checkers**

**Defect Detection Analysis Tools**

# Faster, Better, Cheaper - Pick Two

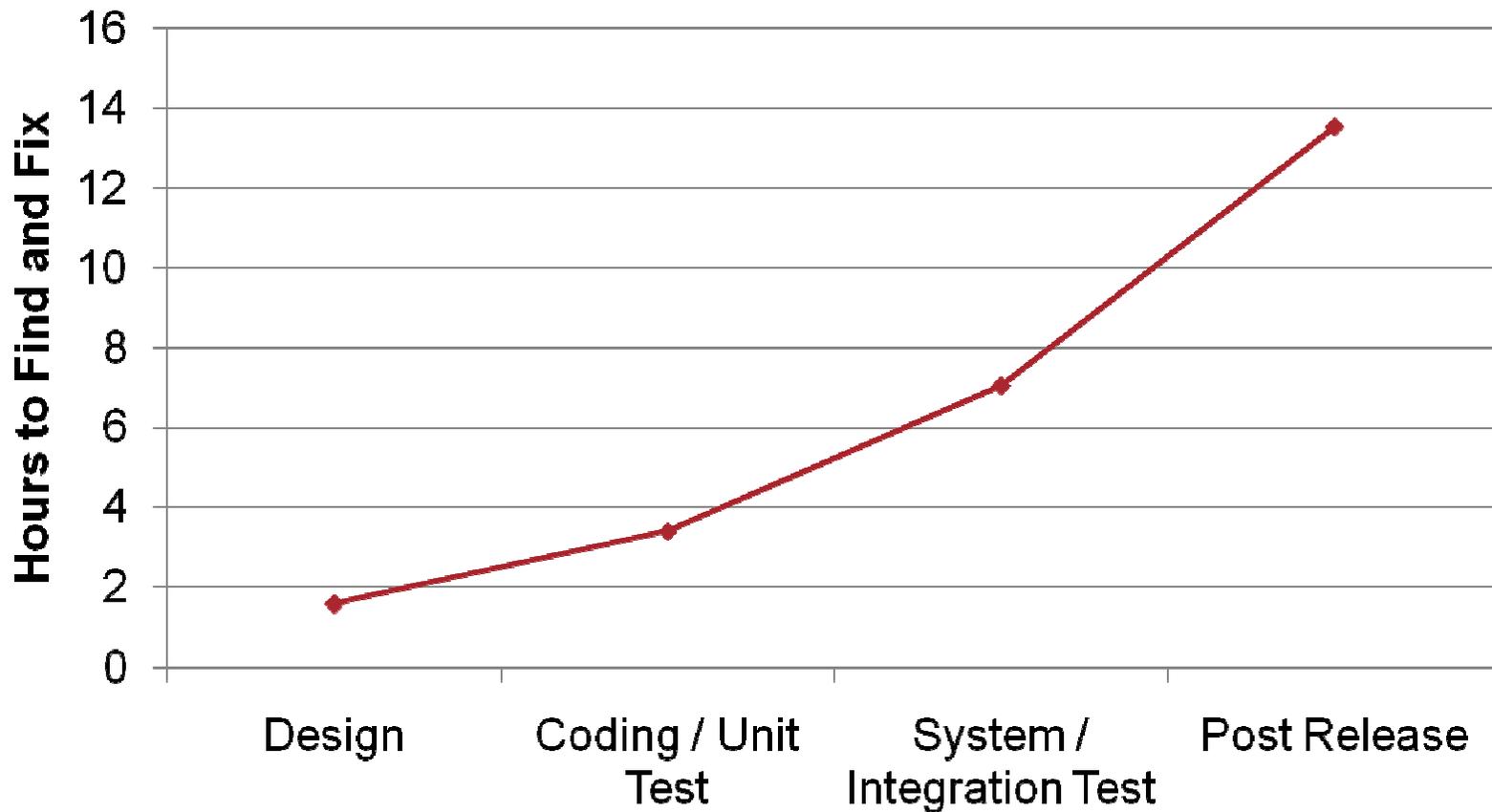


# ~~Faster, Better, Cheaper - Pick Two~~



# Find and Fix Rises Later in Development

## Hours to Fix By Development Phase



# Simple Math

Let's say we have 100 real defects:

- 25 were found during Code/Unit Test
  - NIST data shows it takes 3.4 hours per defect in Code/Unit Test
- 50 were found during Integration/Testing
  - NIST data shows it takes 7 hours per defect in Integration/Test
- The rest are in the product today. Let's say 1 of them causes a Security Failure
  - NIST data shows it takes 13.5 hours to find a defect after release

**TOTAL: 448.5 hours** + 24 defects still in product + collateral damage

# Static Analysis Math

- Let's say a Static Analysis tool reports 100 potential defects...
- Q: How long does it take to review 1 defect, determine root cause, and fix the defect?
- Q: How many False Positives are reported?
  - How much time does it take to identify them?
  - Is the status persistent?
- Q: How many defects are “significant” or “must be fixed” as opposed to “low risk” or “no risk” issues?

# How to Evaluate Static Analysis

<u>Issue</u>	<b>Tool A</b>	<b>Tool B</b>
Ability to detect serious defects	<ul style="list-style-type: none"> <li>• Null Pointer Dereferences</li> <li>• Out Of Bounds Array Access</li> <li>• Uninitialized Variables</li> <li>• Memory Leaks</li> </ul>	<ul style="list-style-type: none"> <li>• Null Pointer Dereferences</li> <li>• Out Of Bounds Array Access</li> <li>• Uninitialized Variables</li> <li>• Memory Leaks</li> <li>• Multi-Thread Support</li> <li>• 64-bit OS Support</li> </ul>
False positive rate	<p><b>58%</b></p> <ul style="list-style-type: none"> <li>• Identifies 60% of the errors identified by Tool B but also reports 2.5 times the total number of errors</li> </ul>	<p><b>25%</b></p>
Defects that are “worth fixing”	<p><b>17% Fix Rate</b></p>	<p><b>40% Fix Rate</b></p>

# How to Evaluate Static Analysis

<b><u>Cost</u></b>	<b>Tool A</b>	<b>Tool B</b>
Reported Defects	<b>250 Defects</b>	<b>100 Defects</b>
Inspect Time	<b>10 minutes</b>	<b>10 minutes</b>
Fix Time	<b>50 minutes</b>	<b>50 minutes</b>
False positive impact	<b>58%</b> 145 False Positives x 10 min ~ <b>24.1 hours</b>	<b>25%</b> 25 False Positives x 10 min ~ <b>4.1 hours</b>
Defects that are “worth fixing”	<b>17% Fix Rate</b> ~18 defects fixed x 1 hours ~ <b>18 hours</b>	<b>40% Fix Rate</b> 30 defects fixed x 1 hour <b>30 hours</b>
Defects not being fixed	~87 defects inspected x 10 min ~ <b>14.5 hours</b>	45 defects inspected x 10 min <b>7.5 hours</b>

# So what's the ROI already?

Tool A: 18 defects = 56.6 hours at Code/Unit Test

Manual Inspection: 18 defects = 139.05 hours

- 4.5 defects x 3.4 hours
- 9 defects x 7 hours
- 4.5 defects x 13.5 hours

**ROI: 82.45 hours**

Tool B: 30 defects = 41.6 hours at Code/Unit Test

Manual Inspection: 30 defects = 231.75 hours

- 7.5 defects x 3.4 hours
- 15 defects x 7 hours
- 7.5 defects x 13.5 hours

**ROI: 190.15 hours**

# Select a Code Base for Evaluation

- Pick a codebase and run the tool.
- The selected code base should be:
  - As large as possible
  - Actively developed

# Review the Results Generated by the Tool

- Review the results.
- Quality not quantity - actionable, relevant defects
  - Evidence based defect reports
  - Low rate of false positives
  - High rate of severe issues leading to fixes

# Evaluate East of Adoption

- Find barriers to adoption
  - Classification and annotation
  - Ownership
  - Integration with your SDLC systems
    - No changes to existing build system
    - Defects can be exported to your Bug Tracking tool

# Questions



# How to NOT Drive Adoption

To: Developers

From: Management

Dear developers, we purchased a new tool which finds mistakes that you made. You don't know how to use it, you don't know what exactly it does, and you didn't participate in the evaluation or purchase decision.

If you want to use it – you can get it here:

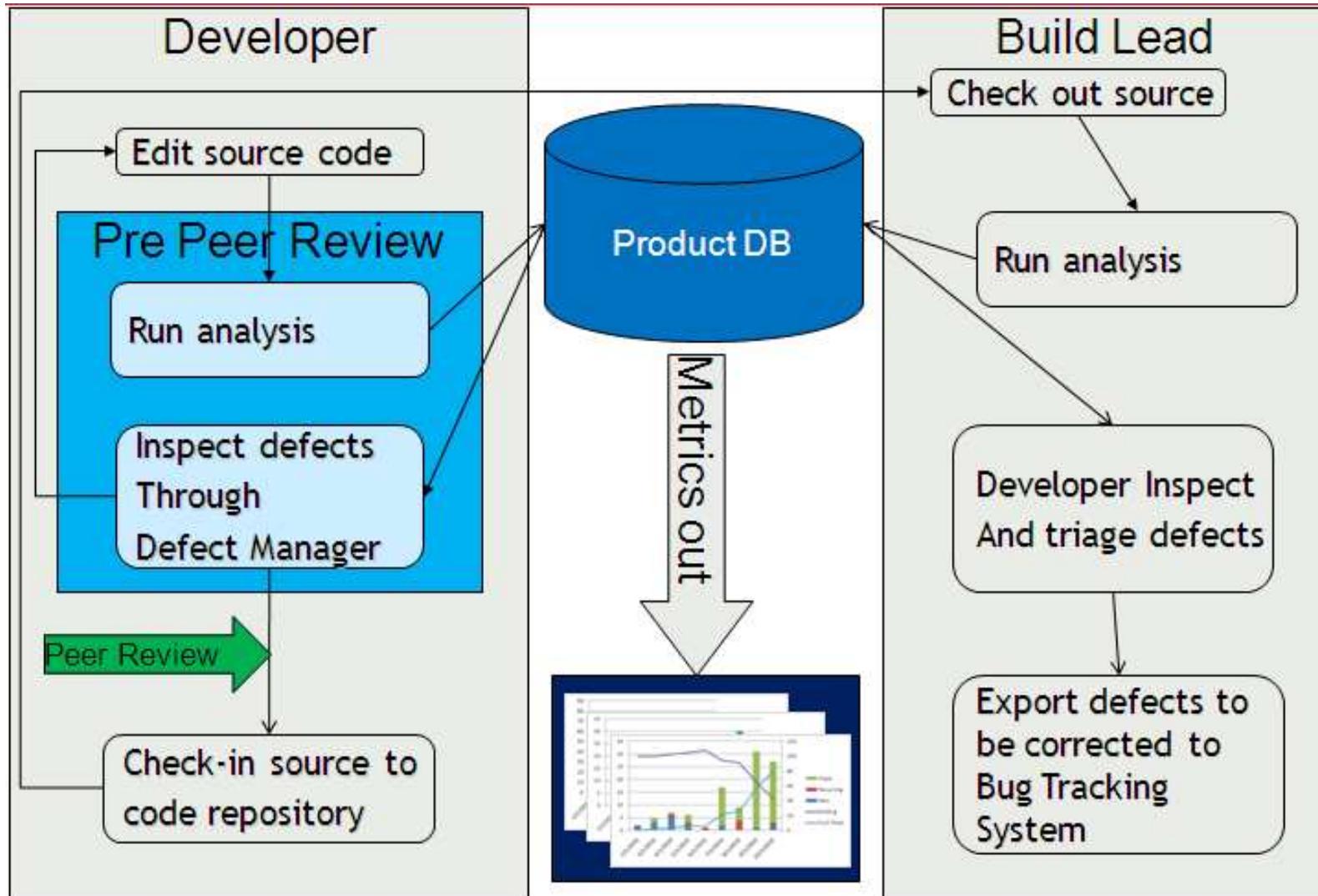
<http://intranet.download.new.tool>

- Your boss.

# Plan the Deployment – Drive Adoption

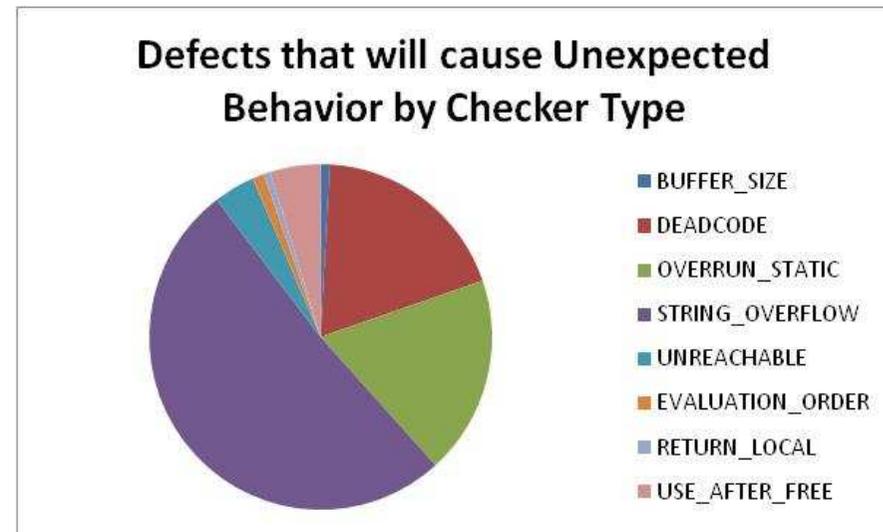
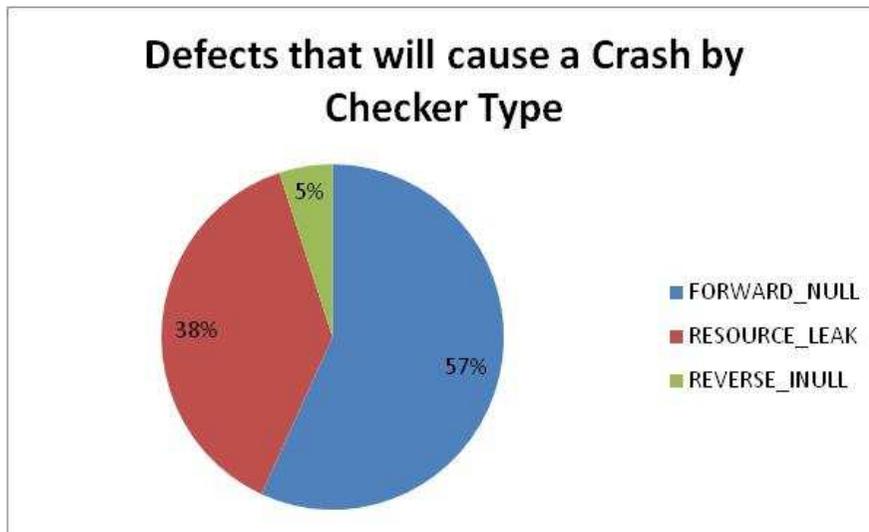
- Coordinated between metrics group, process group, IT, and development org managers
- Dev Managers selected a member from each dev team to install the tool and review the initial set of defects
- Developers are trained
- IT provided central machine for shared Defect Manager Database
- Process group defined where in the process the tool will fit in
- Metrics group decided on which metrics to measure to show ROI

# The Big Picture



# The Results?

- 4 groups and 20 projects/code bases were up and running within 4 weeks
- Metrics collection was automated
- Process group modified the process documents to include running the static analysis tool, and to mandate it prior to code review

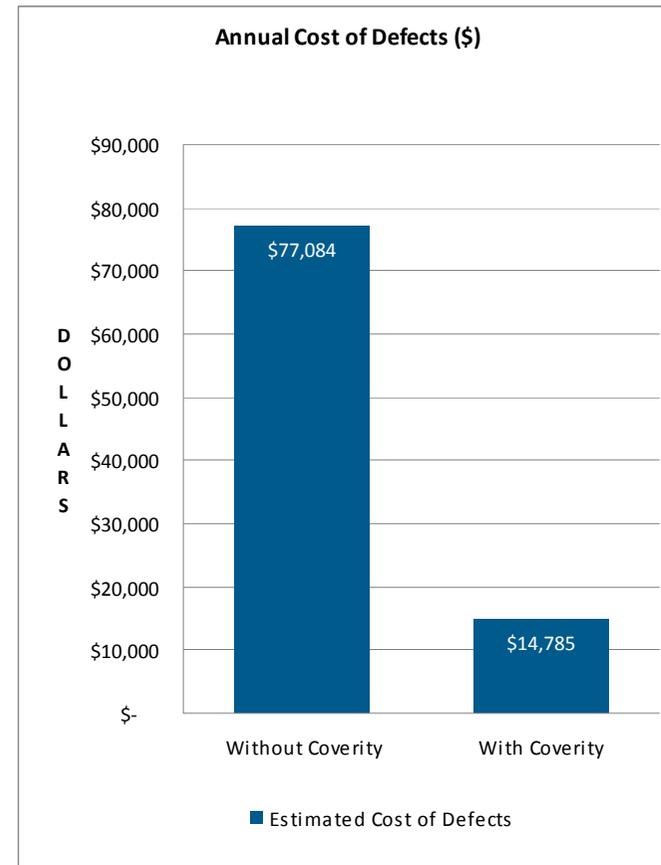
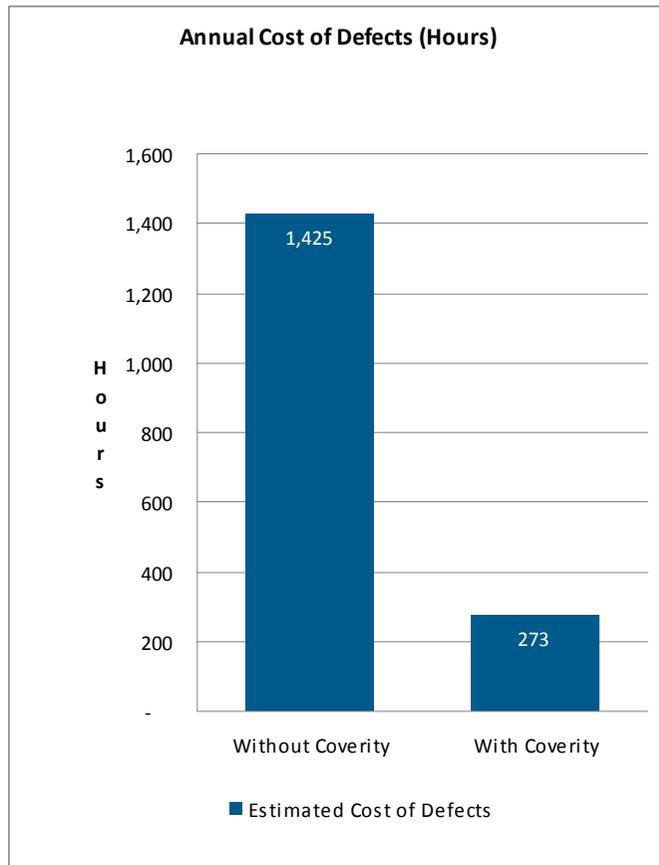


# Coverity Customer Case Study

*We use Coverity Prevent to realize the following productivity improvements:*

- A savings of approximately thirty (30) percent of development time that would be consumed finding/repairing defects,
- Easier defect detection and removal (closer to their point of introduction),
- Significant defect removal costs savings (that would grow significantly, if detected later in the development process).

# Coverity Customer Case Study



# Coverity Customer Case Study

- Type: Embedded Firmware Product
- Language: C
- Size: ~500 KLOC of proprietary code
- Static Analysis time frame (setup, build, scan, analysis, rescan, manual review): ~4 weeks
- Real Security Vulnerabilities (confirmed through analysis of Coverity output and manual code review):

**High: 17**

**Medium: 6**

**Low: 9**

# Coverity Scan Project

## DHS Sponsored Open Source Initiative

- <http://scan.coverity.com>
- Over 280 commonly used open source packages
- Over 60 Million LOC analyzed nightly on standard hardware
- Maintainers fixed over 12,000 bugs and security violations to date

The screenshot displays the Scan.Coverity.com website interface. At the top, it features the site logo and navigation links such as 'MAIN', 'ABOUT SCAN', 'FAQ', and 'DEVELOPER FAQ'. A central section titled 'ACCELERATING OPEN SOURCE QUALITY' describes the project's collaboration with Stanford University and the Department of Homeland Security. A red banner indicates 'TOTAL NUMBER OF DEFECTS FIXED (SINCE 03/06/2006): 6,035'. Below this, a bar chart titled 'Number of references to defects in Amanda, resulting in fix, colored by defect type' shows the distribution of fixes. The chart has a y-axis labeled 'Number of Views' ranging from 0 to 12 and an x-axis with values from 0 to 140. To the right of the chart, there is a 'NEWS' section with articles like 'Happy First Birthday, Scan' and 'Coverity Names David Maxwell as Open Source Strategist'. The bottom right corner of the screenshot contains logos for Coverity, Stanford University, the U.S. Department of Homeland Security, and Symantec.

# Questions

