

The Role of Static Analysis in an Effective Risk Management Strategy

Steve Howard – European Technical Manager

21 November 2011 · COMPANY PROPRIETARY

Agenda

- »» What is source code analysis?
- »» How can using source code analysis manage software risk?
- »» Real-world examples

Software Risk Realities

21 November 2011 · COMPANY PROPRIETARY

The Risk in Software Development



Therac-25 Radiation Therapy Machine

Death of 3 patients due to massive overdoses of radiation



Toyota Prius

100K+ vehicle recalls



Ariane 5 Flight 501

Loss of \$370M spacecraft

Realities of Software Risk in Safety Critical Systems

- » Safety and security incidents resulting from software defects are well documented
- » Critical defects and security vulnerabilities:
 - slows development productivity and reduces time for innovation
 - impacts competitive advantage and reputation
 - Costs the U.S. economy almost \$60 billion annually
 - 80% of development costs are consumed by software developers identifying and correcting defects
- » Traditional approaches to address risk are ineffective
 - Code analysis preformed after check-in
 - 80% of defects introduced in implementation and build phases of SDLC
 - In-person code reviews
 - Difficult to schedule, time-consuming, unpleasant process

Introduction to Source Code Analysis

21 November 2011 · COMPANY PROPRIETARY

Source Code Analysis (SCA) – The Basics

- »» The analysis of computer software (source code) that is performed without actually executing programs built from that software

- »» Complete, automated view of every possible execution path, rather than an aspect of observed runtime behavior

- »» Automated code analysis technology finds weaknesses in source code
 - Logic errors and implementation defects
 - Security vulnerabilities
 - Architecture validity
 - Concurrency violations and rare boundary conditions
 - Software metrics generation and management

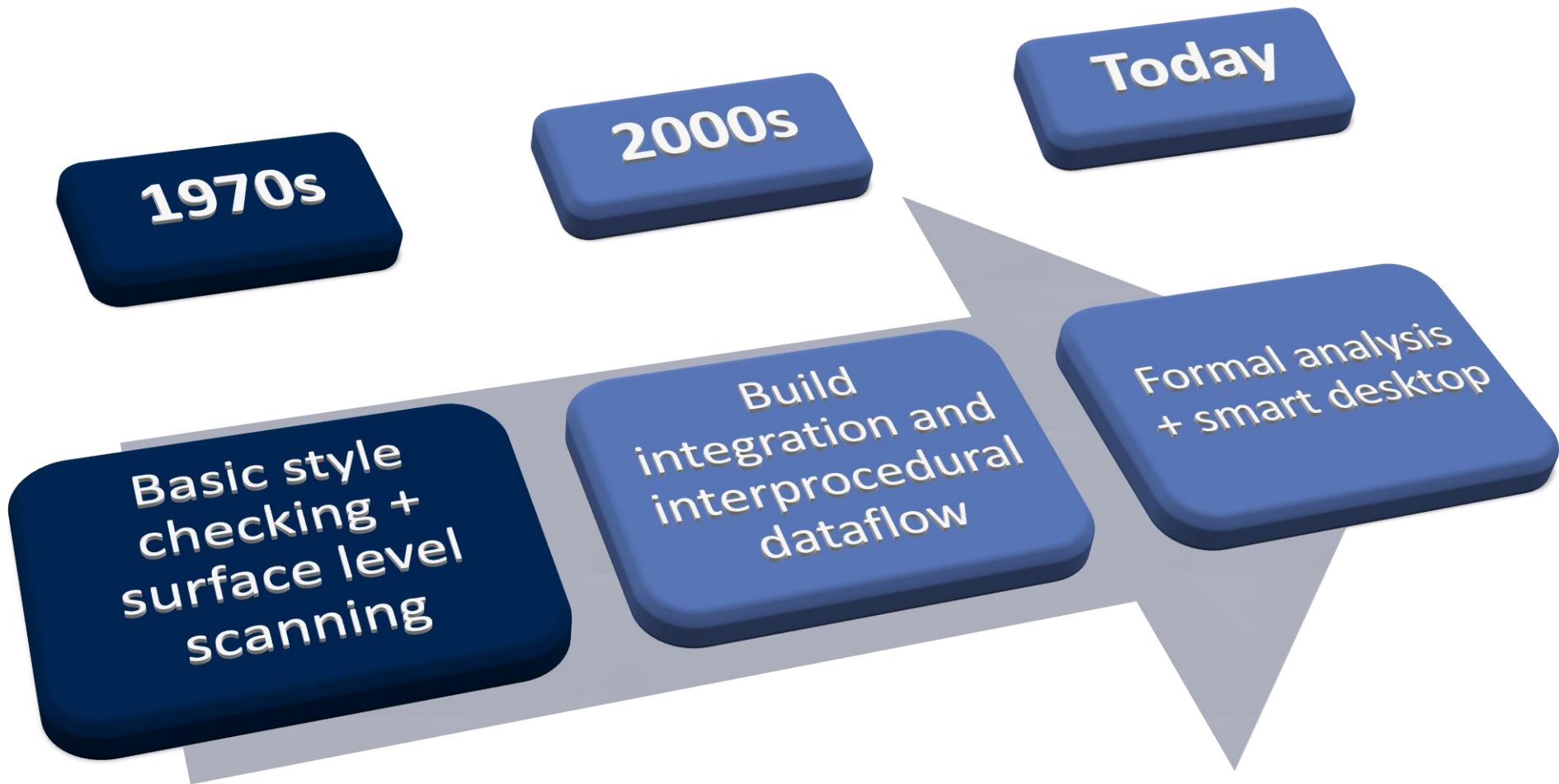
- »» Can be used at different times in the software development lifecycle

What is Source Code Analysis

- » It will significantly reduce cost of quality and compliance
 - Complements existing testing techniques and approaches
 - Analysis is automated and easily repeatable – no test cases, no stubs, no complex setup

- » Core technology is known as ‘static analysis’
 - Distinct from more traditional dynamic analysis techniques, such as unit or penetration tests
 - Used widely in mission critical software industries such as medical devices, mil/aero, telecom equipment, and others

SCA – A History Lesson



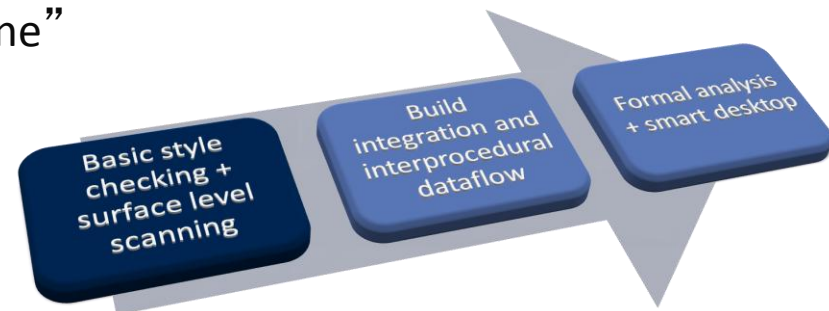
1970s... first there was Lint

»» What was Lint doing?

- Scanning, initially
- Looking for “known aberrant” problems with C
 - Missing / extra semi-colons
 - Missing curlicues
 - Potentially dangerous implicit casts

»» Lint was invented as a developer’s tool

- Lots of problems with the model
- Noise, inaccuracies, too-small a locality of reference
- But it was always intended to “just give better compiler errors to the developer”
- Seen by developers as “opt-in” and “mine”



2000s... new generation of tools start to bring something new

- » 2nd generation static analysis provided better core analysis capabilities that extended beyond syntactical and semantic analyses to include:
 - Sophisticated inter-procedural, control- and data-flow analysis
 - New approaches for pruning false paths
 - Estimating the values that variables will assume
 - Simulating potential runtime behavior
- » Moved away from being a developer tool
 - In order to produce good analysis, must do it at integration build
 - So it's not part of the developer's workflow
 - So when a bug is found it's too late
 - The developer is already on some other task and must be dragged back



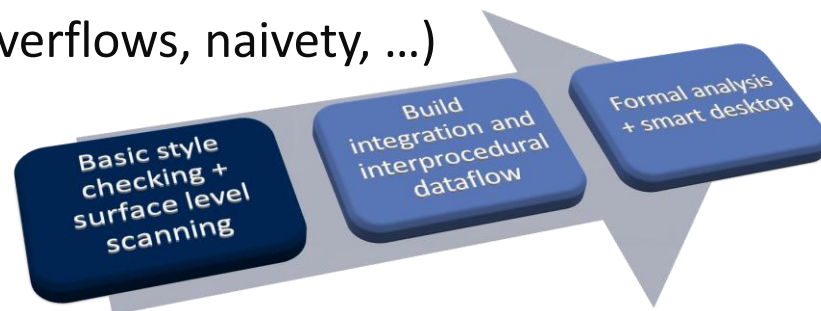
2000s... what kind of bugs?

»» Local bugs vs. inter-procedural bugs

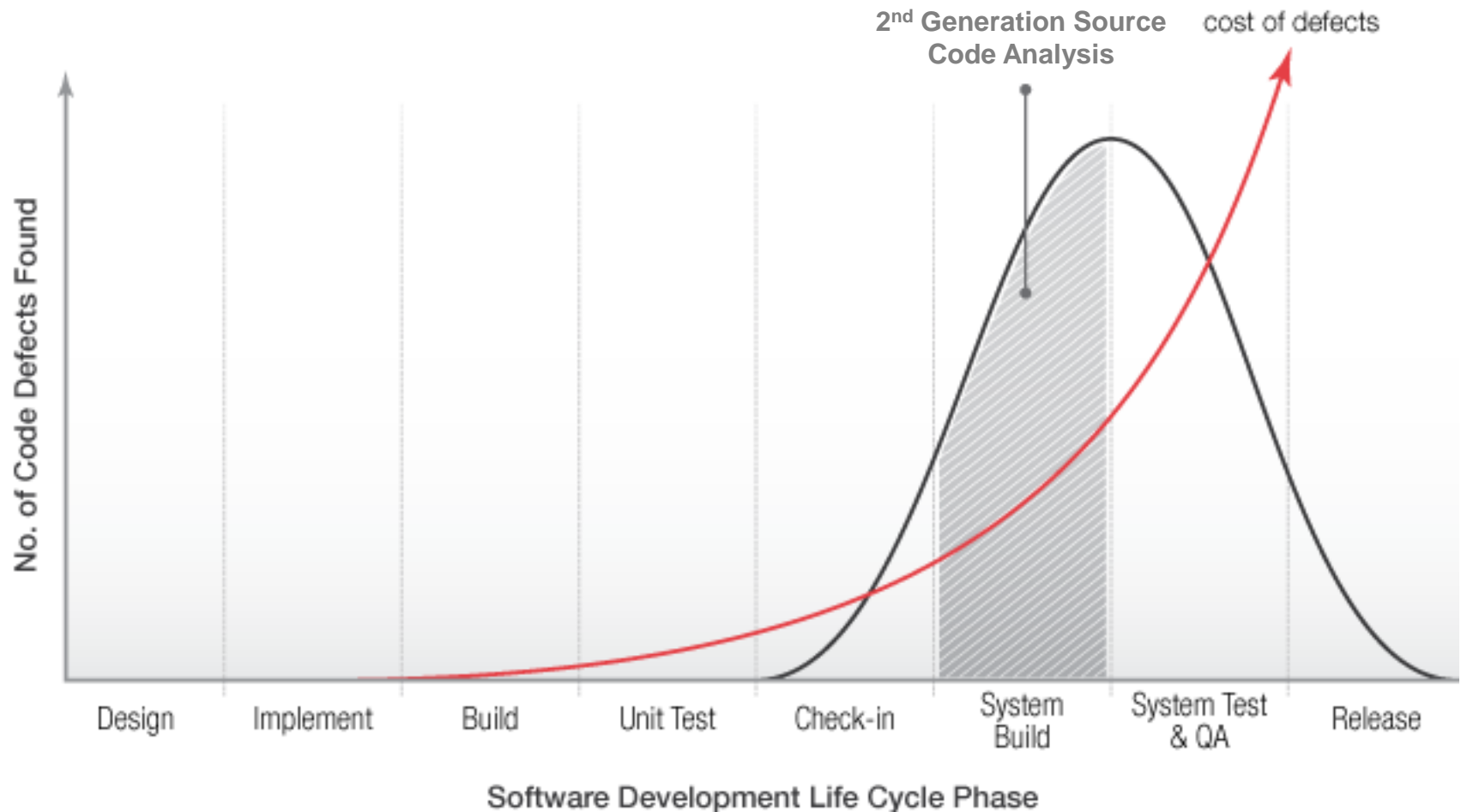
- Local bugs are “easier” to find, and certainly easier to understand
 - Picture “slapping head”
- Inter-procedural bugs are the big payoffs
 - Difficult for different developers working on projects to deal with each other’s code rationally
 - Lots of fingers in lots of pies = lots (and lots) of bugs

»» What kinds of bugs?

- Resource management
- Pointer management (and yes, this means Java too)
- Security vulnerabilities (injections, overflows, naivety, ...)
- Concurrency



2000s... good bug detection 'before testing'



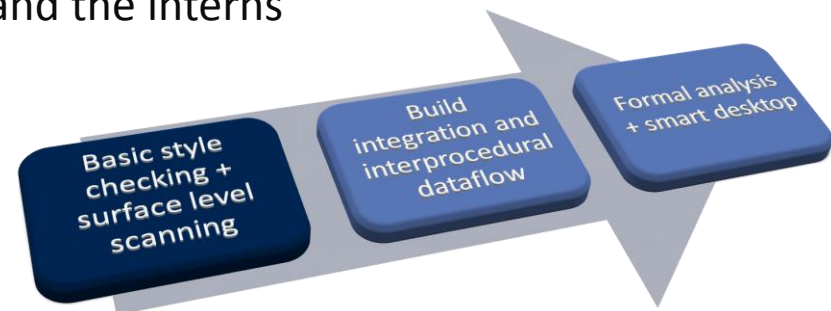
Today... deep code analysis with smart desktop

»» Deliver 2nd generation analysis capabilities right to the developer desktop

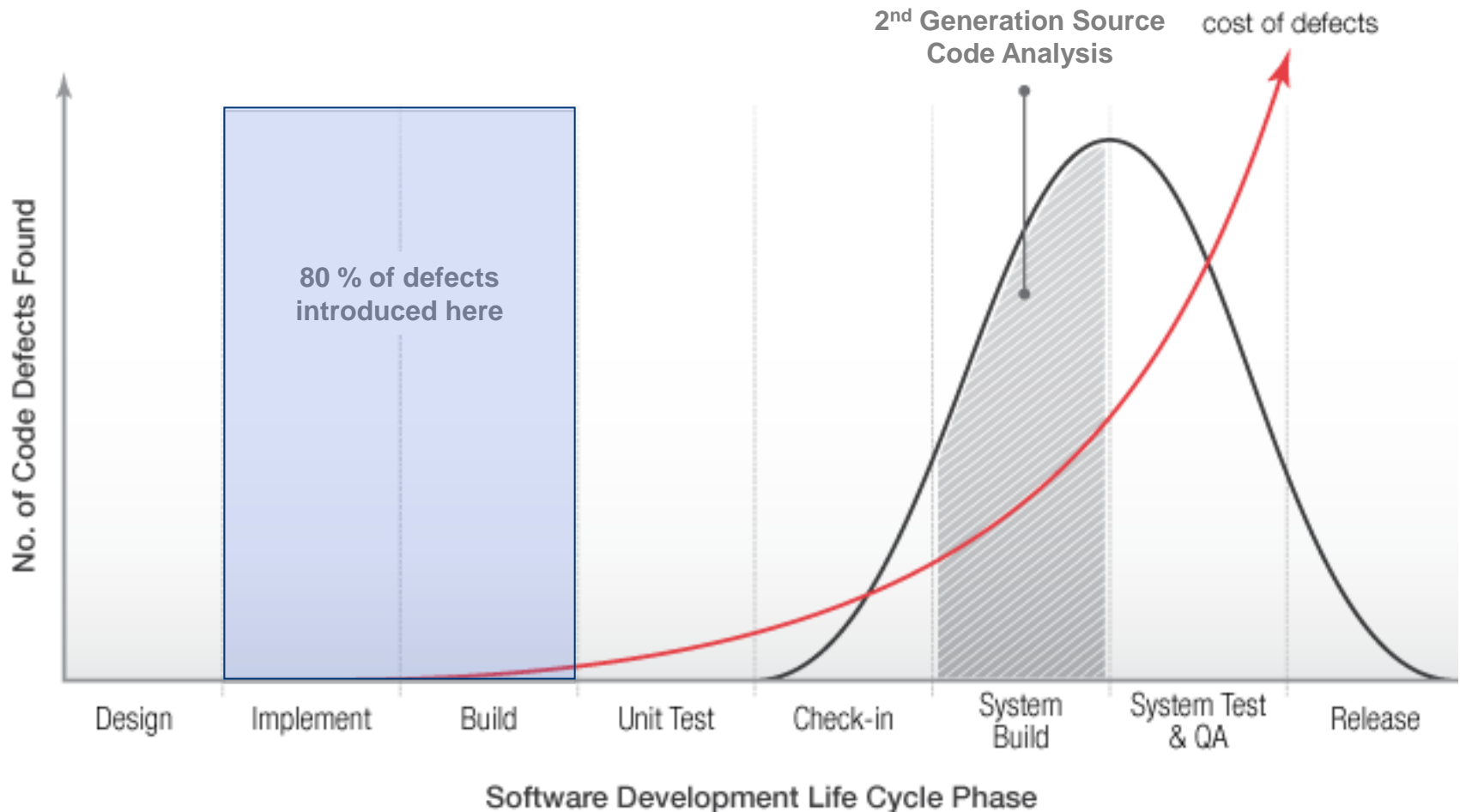
- Focus on enabling the developer, not blaming them
 - If the tool is “mine” I am more likely to use it when I am in process
 - If I use it in process, I’ll find and understand the bugs more quickly
 - If I understand the bugs, I’ll fix them faster
 - If I fix the bugs, the code stream isn’t polluted, my QA time isn’t wasted, and my product gets better

»» Bottom line

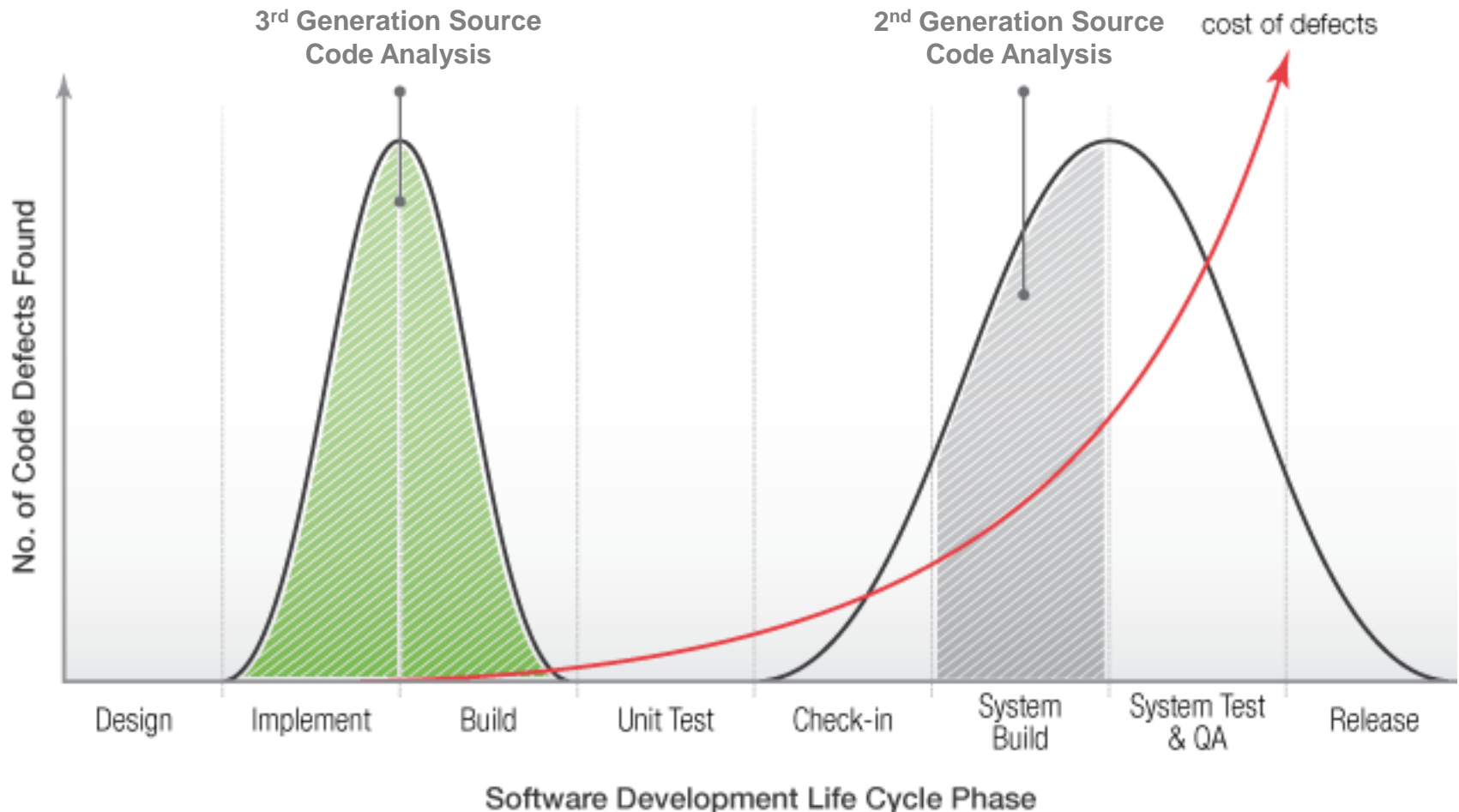
- Don’t check in code that doesn’t work!
- Build defensive coding into the organization from the ground up
- Narrow the gap between the rock stars and the interns
- Make a better product



The Costs of Bug Containment



The payoff... deep code analysis at the developer desktop

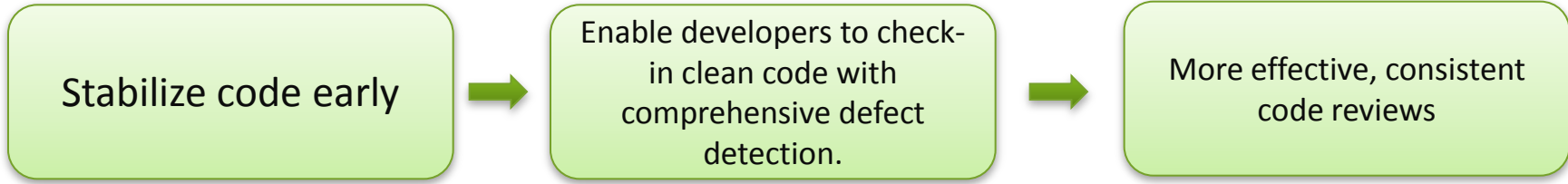


How Static Analysis Tools Can Help Manage Software Risk

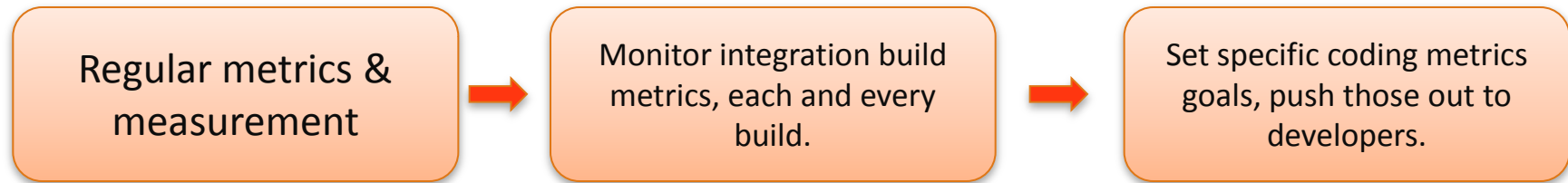
21 November 2011 · COMPANY PROPRIETARY

Software Risk Management Strategies

1



2

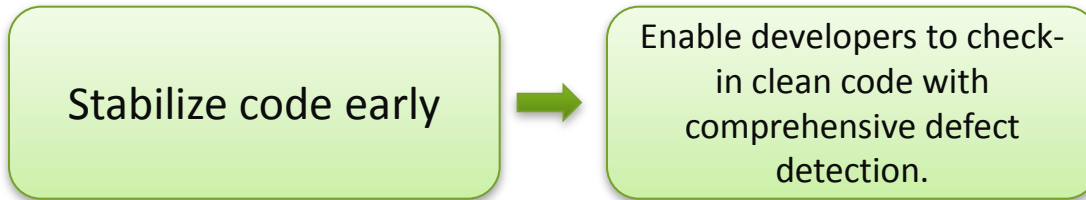


3



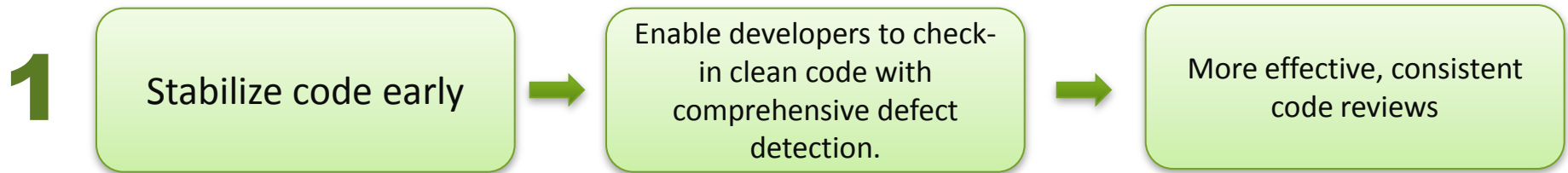
Risk Management: Check-in Clean Code

1



- »» Important to reduce the 'rinse-repeat' cycle of rework
- »» Helps limit the number of defects introduced into the code stream
 - Allows testing team to focus more on ensuring the software meets requirements
- »» Minimize product recalls
- »» An ideal solution should include:
 - Ability to find defects while coding → In-phase bug containment
 - Quality & reliability defects, security vulnerabilities, maintainability issues
 - Configuration and tuning capabilities
 - Must be a part of the regular build process
 - Work where developers work

Risk Management: Better Code Reviews

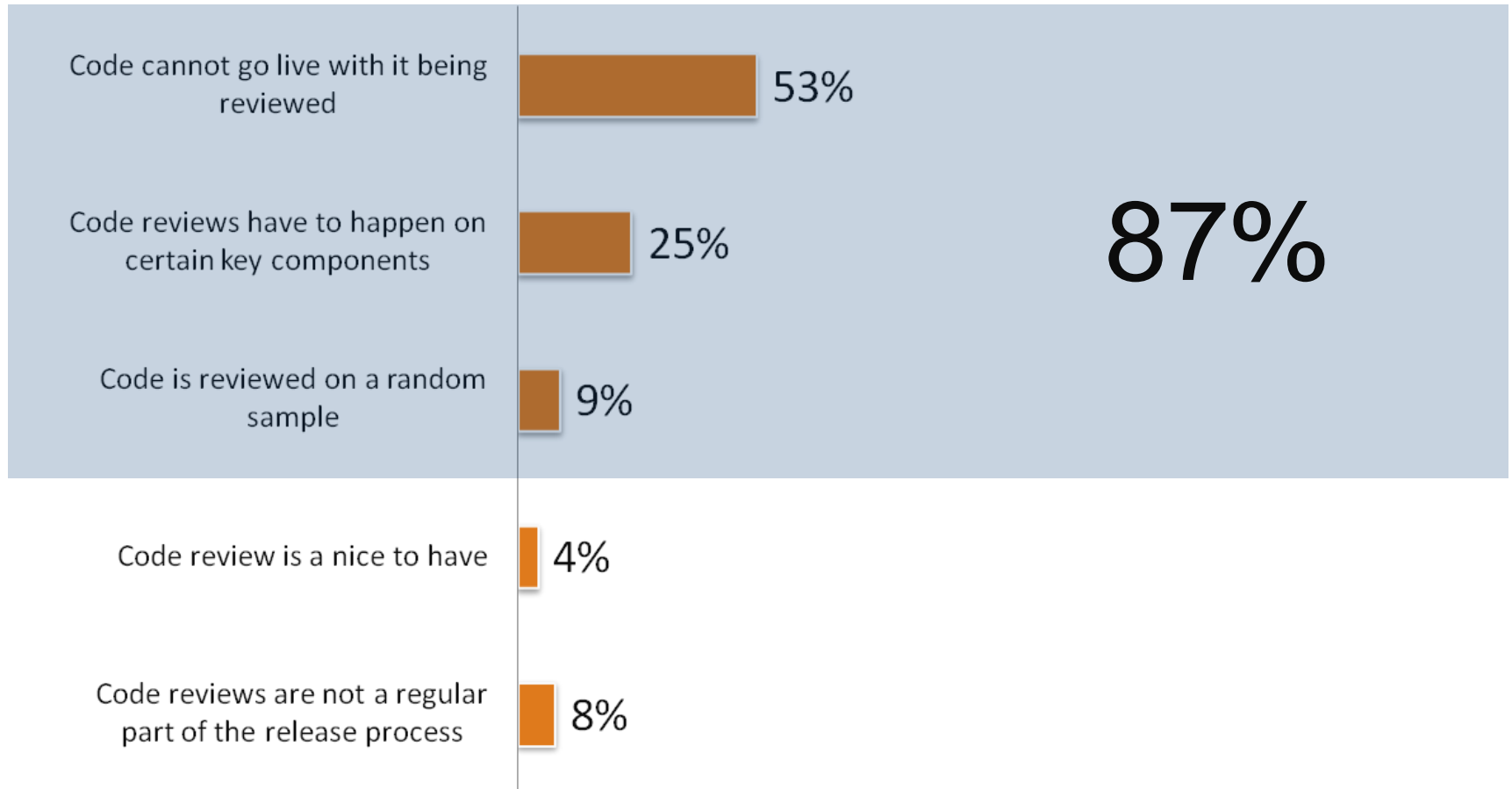


»» Code reviews are extremely valuable

- Creates consistency and a culture of quality
- Developers learn from code reviews
- Save money

Are Code Reviews Mandatory?

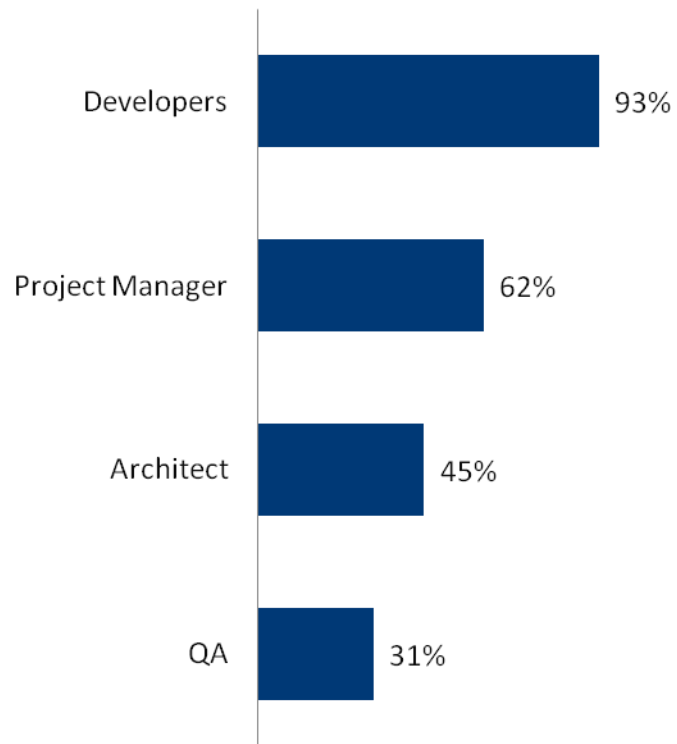
“To what extent are code reviews a part of your regular release cycle?”



Source: A commissioned study conducted by Forrester Consulting on behalf of Klocwork, February 2010

Who Participates in Code Reviews?

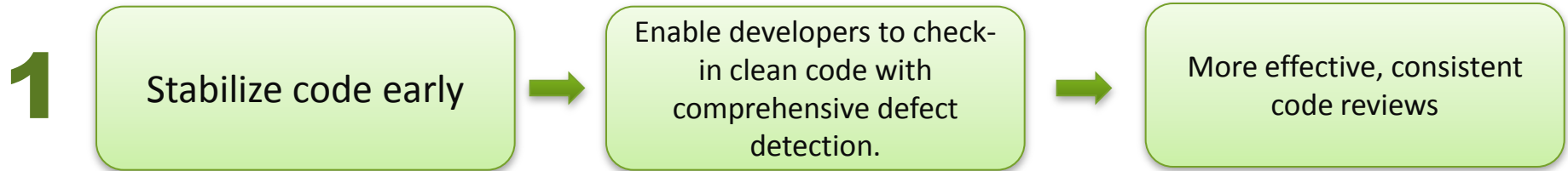
“Who is usually involved with the code review?” (select all that apply)



Base: 159 IT professionals who participate in their organization’s software development and code reviews

Source: A commissioned study conducted by Forrester Consulting on behalf of Klocwork, February 2010

Risk Management: Better Code Reviews



»» Code reviews are extremely valuable

- Creates consistency and a culture of quality
- Developers learn from code reviews
- Save money

»» However, not perfect, and usually not well-liked by developers

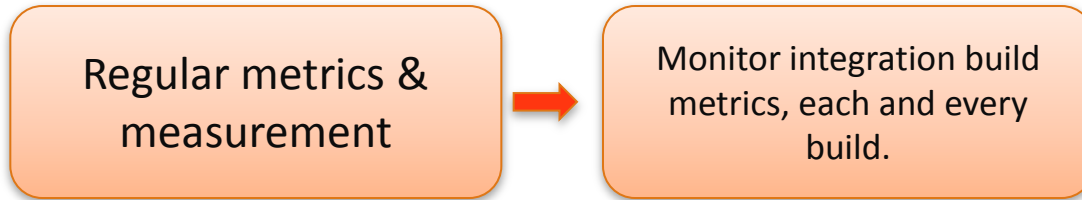
- Lots of bugs still getting through - Industry standard shows effectiveness at 55-60%
- Typical code review inefficient

»» An ideal solution should include:

- Bottom-up, rather than imposed, top-down
- Asynchronous, rather than “around the table”
- Defects found through SCA integrated into the code review

Risk Management: Monitor build metrics

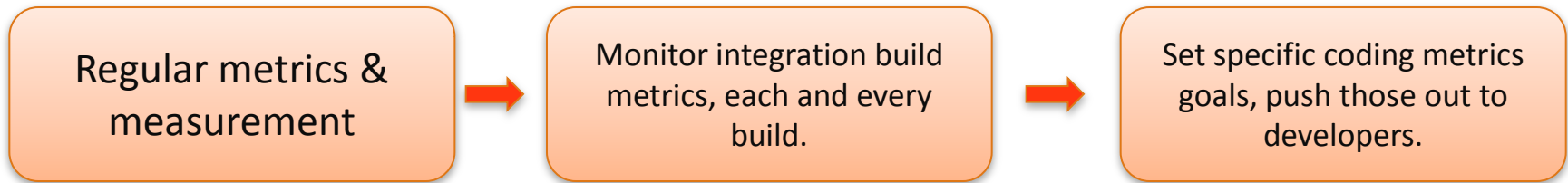
2



- Critical to be able to quickly understand and interpret data about your software
 - Trend key software metrics with every build
 - Identify areas of security and quality risk
 - Take actions to stabilize code base early
- An ideal solution should include:
 - Customizable reporting capabilities for each and every build
 - Project summary and dashboard showing key metrics and fix activity
 - List of top defects and their severity
 - Key metrics including source code size, complexity, churn, etc.

Risk Management: Push metrics out to developers

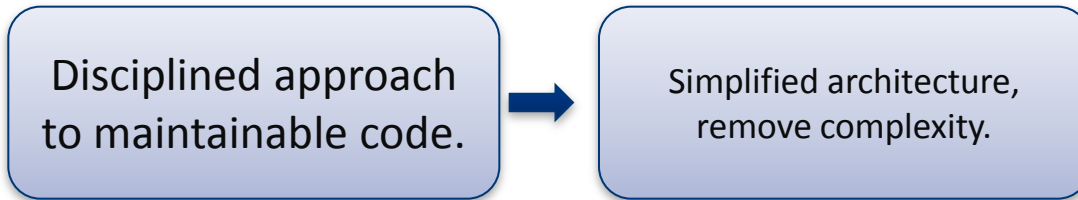
2



- » Management loves metrics...developers not so much
- » Key is to find metrics that are relevant to developers:
 - Rework
 - Code 'tightness'/reusability
- » An ideal solution should include:
 - # of defects found at the desktop
 - Bug fixes/open issues per developer
 - Compliance to coding standards
 - New issues introduced

Risk Management: Simplify architecture, remove complexity

3



» Small changes to software can have a significant impact

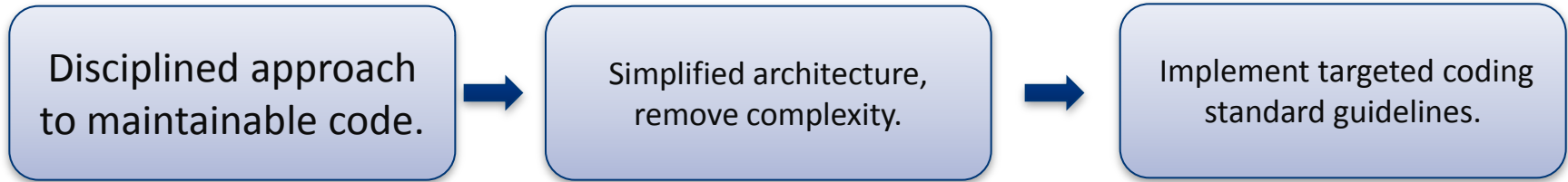
- Need to understand extent and impact changes have individually and on the entire software system

» An ideal solution should include:

- Graphical visualization of architecture and control flow
- Easily identify areas of complexity and maintainability risk
- Use of 'what-if' analysis for clean up

Risk Management: Implement targeted coding standards

3



- » Coding standards are often overlooked, but are important in software development
 - Helps improve overall quality and maintainability of code
 - Source code should be evaluated to verify its compliance with specified coding guidelines
 - Guidelines include coding conventions regarding clarity, style, complexity management, and commenting

- » An ideal solution should include:
 - Out of the box support for wide variety of coding style issues
 - Extensibility to allow users to create own rules or variants of known defect types

Real-World Examples of How Static Analysis Can Help

21 November 2011 · COMPANY PROPRIETARY

How SCA Helped Johns Hopkins

» Tasked with building the embedded software component for RP2009 (Revolutionizing Prosthetics program)

- Complex software systems translate user signals to limb motion and provide sensory feedback back to the user
- Tight project timeline
- Need for increased software quality



» Results after 5 months of use

- 225 total defects found (83 critical defects)
 - Array bounds violations, use of uninitialized data , NULL pointer references found
 - FP rate < 1%
- 900 person hours saved on team of 4 developers
- Potential security vulnerabilities also found

Ariane 5 Flight 501

- »» Ariane 5 is an expendable launch system (no crew) used to deliver payloads into low Earth orbit
- »» June 1996: 1st test flight - Unsuccessful
 - The rocket veered off its flight path 37 seconds after launch and was destroyed
- »» The Cause:
 - An error in the software design (inadequate protection from integer overflow)
- »» The Result:
 - A loss of more than US\$370 million
 - The subsequent automated analysis of the Ariane code was the first example of large-scale static code analysis by abstract interpretation



Conclusions & Summary

- » Software development in mission critical environments can be challenging
 - Regulatory requirements
 - Expensive hardware platforms that are difficult to modify
- » Automation is critical
 - Complexity and pace of change for embedded software is rapidly outpacing the ability of organizations to address these issues with manual processes
- » Source code analysis should be part of any software risk management strategy
 - View as a complement to, not a replacement for other traditional methodologies
- » Lots of cost, productivity, and overall software quality gains that can be achieved