

# Adhering to Coding Standards and Finding Bugs in Code Automatically



## **Presenters:**

Muhammad Wasim  
Hammad Ali Butt



# Agenda

---

- ❑ Introduction
- ❑ Software Quality Assurance
- ❑ Code: The base for Quality
- ❑ Code: Writing it
- ❑ Code: Reviewing it
- ❑ How Review tools help?
- ❑ Selecting a Code Review Tools
- ❑ How they compare to each other?
- ❑ Conclusion





# Introduction

---

- Software Quality
- Quality Attributes
  - Maintainability
  - Testability
  - Reusability
  - Correctness
  - Efficiency
  - Usability
  - Flexibility
  - Portability
  - Interoperability
  - Reliability
  - Integrity
- Code: The Base for Quality Software Quality comes with a lot of benefits





# SQA and Benefits

---

## □ **Development/Technical:**

- Increase Readability
- Easy to locate problem area
- Performance Enhancements
- Compliance between Specs & Code
- Criteria for software acceptance

## □ **Management:**

- Better Progress visibility
- Better decision criteria
- Reduced Maintenance cost





# Code: The Base for Quality

---

- ❑ Better Coding Styles (Structured to OOP)
- ❑ Commenting of Code
- ❑ Coding Conventions
- ❑ Design Patterns
- ❑ Low Coupling
- ❑ High Cohesiveness
- ❑ Resource Management
- ❑ Remove Repetition of Code





# Code: Writing it

---

- ❑ Well Managed Code
  - ❑ Well Written Code
  - ❑ Standardized Code
  - ❑ Defensive Coding
- 
- ❑ An ideal programmer won't leave any bugs for compiler or QA team.





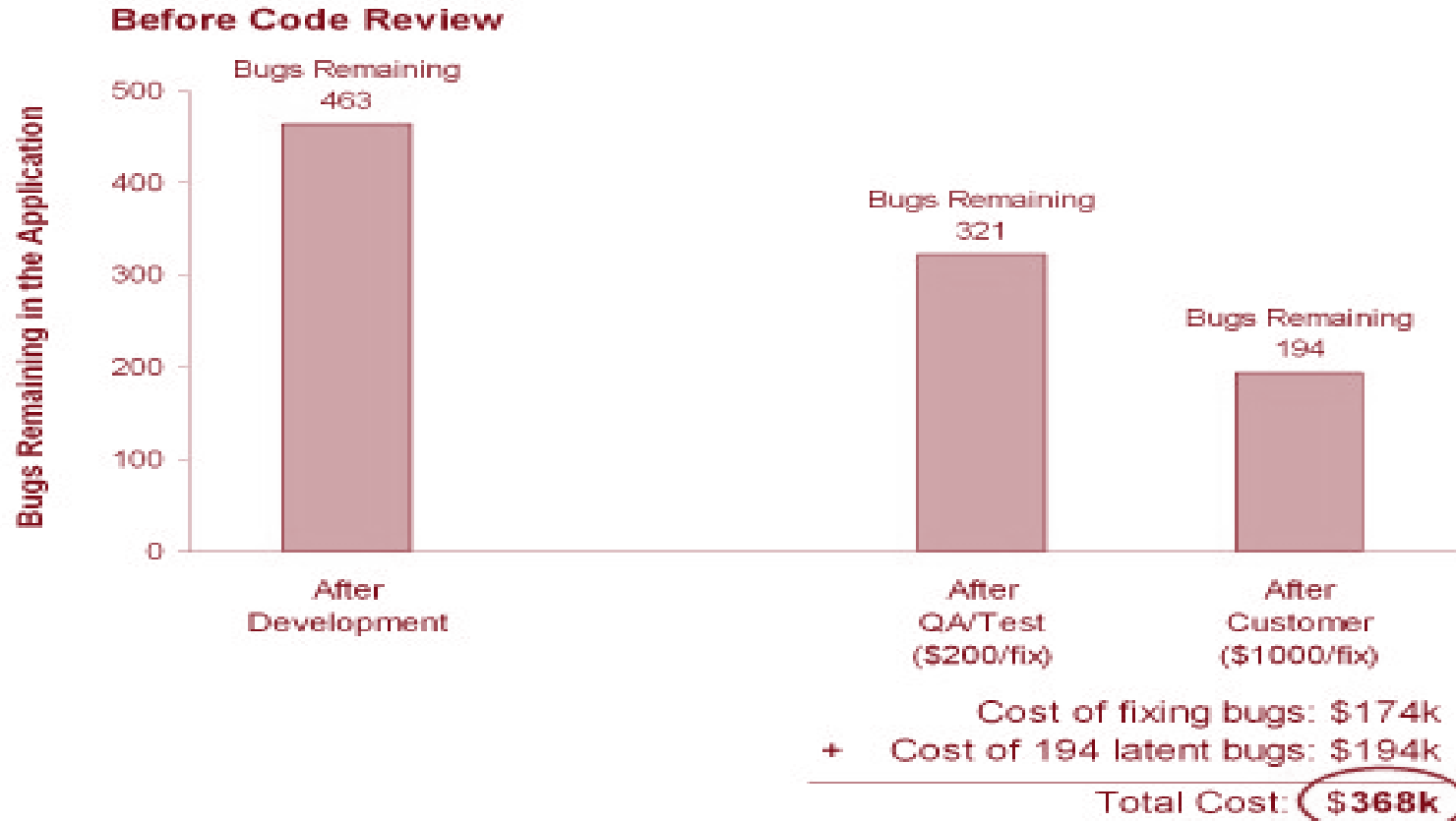
# Code: Reviewing It

---

- Review Meetings
- Peer Review
- **Benefits:**
- Code Optimization
- Reduced Cost
- Programmer Improvement
- Static Code Analysis is used for Code Review.



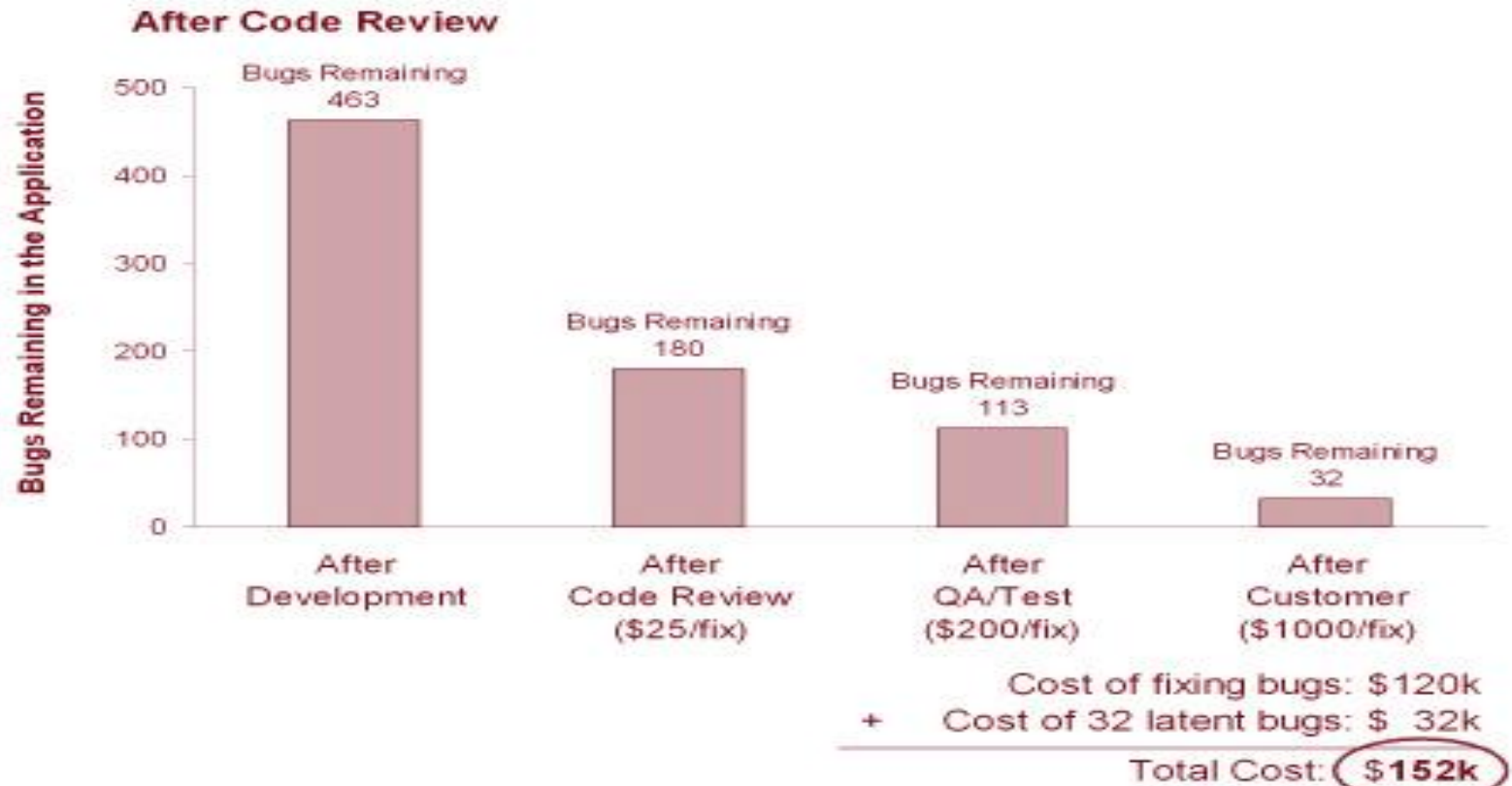
# Code Review on a Typical Project



<http://smartbear.com/docs/articles/before-code-review.jpg>



# Code Review on a Typical Project



<http://smartbear.com/docs/articles/after-code-review.jpg>





# Code Review & Issues

---

- ❑ Human is error prone.
- ❑ Tedious Job
- ❑ Slow Process
  
- ❑ Solution

## **“Automatic Code Review”**





# How Review tools help?

---

- ❑ The use of continuous **automatic** static code **reviews** is a really helpful for quality code
- ❑ We all make **mistakes** and the more you code the more that will happen
- ❑ *Static code reviews* aimed at **eating bugs** are unbiased and neutral
- ❑ Static code reviews are viable. You can get rid of quite a lot of near surface defects quickly.





# What's the benefit?

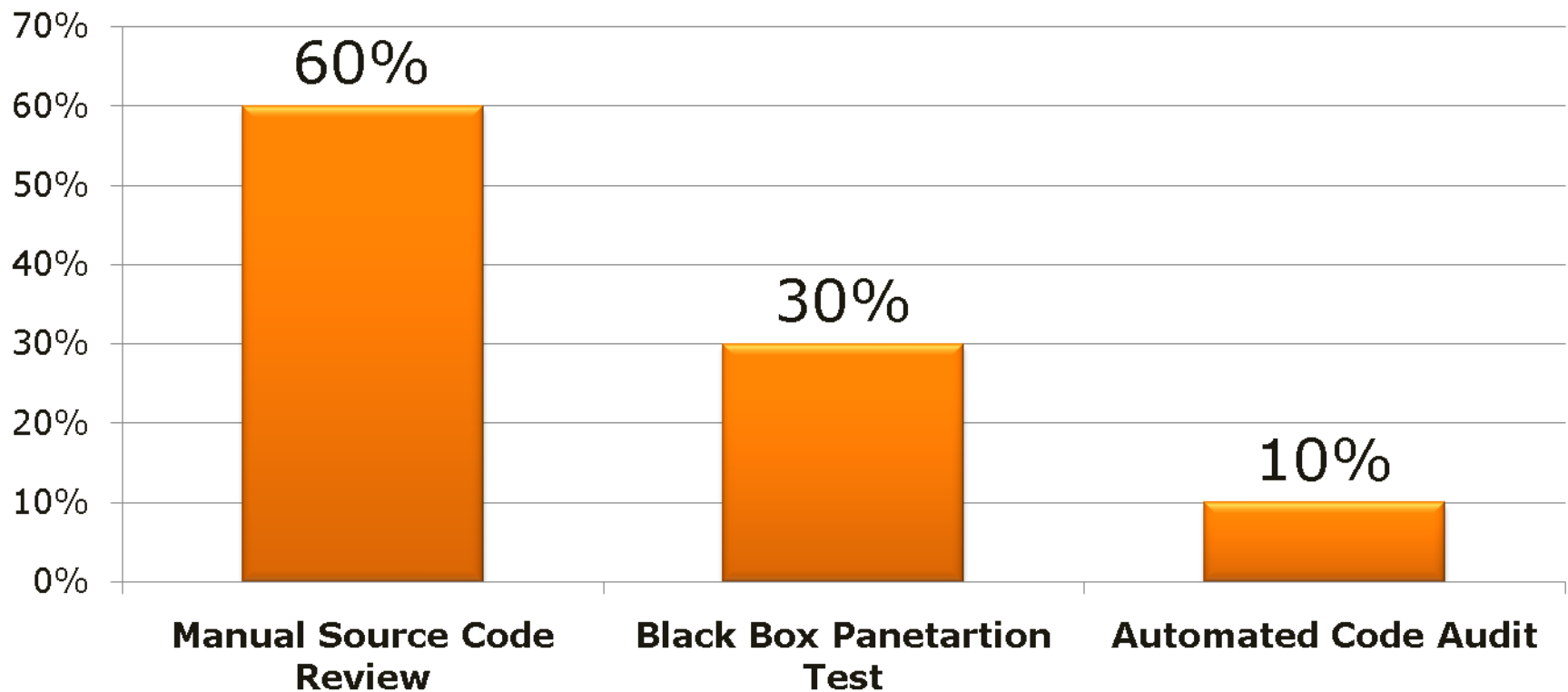
---

- ❑ Enforce coding convention
- ❑ Uniform coding structure
- ❑ Ensure 100% checking
- ❑ Reduce review time
- ❑ Developer development
- ❑ Cost effective



# Automated Review Tools

## Vulnerabilities discover after these test





# Popular Code Analysis Tools for Java

---

## □ Checkstyle

- Besides some static code analysis, it can be used to show **violations** of a configured **coding standard**

## □ FindBugs

- An open-source static **bytecode analyzer** for Java (based on Jakarta BCEL) from the University of Maryland.

## □ PMD

- A static rule set based Java **source code analyzer** that identifies potential problems.





# Popular Code Analysis Tools for Java

---

## □ Hammurapi

- (Free for non-commercial use only) versatile code review solution.

## □ Sonar

- A platform to manage source code quality (**checkstyle**, **PMD** and **find bug**)





# How a Review Tool can be Selected?

---

- ❑ Ability to modify rules
- ❑ Ability to implement coding standard at various levels
- ❑ Importing/Exporting rules
- ❑ Integration with the IDE





# Configuration of tools

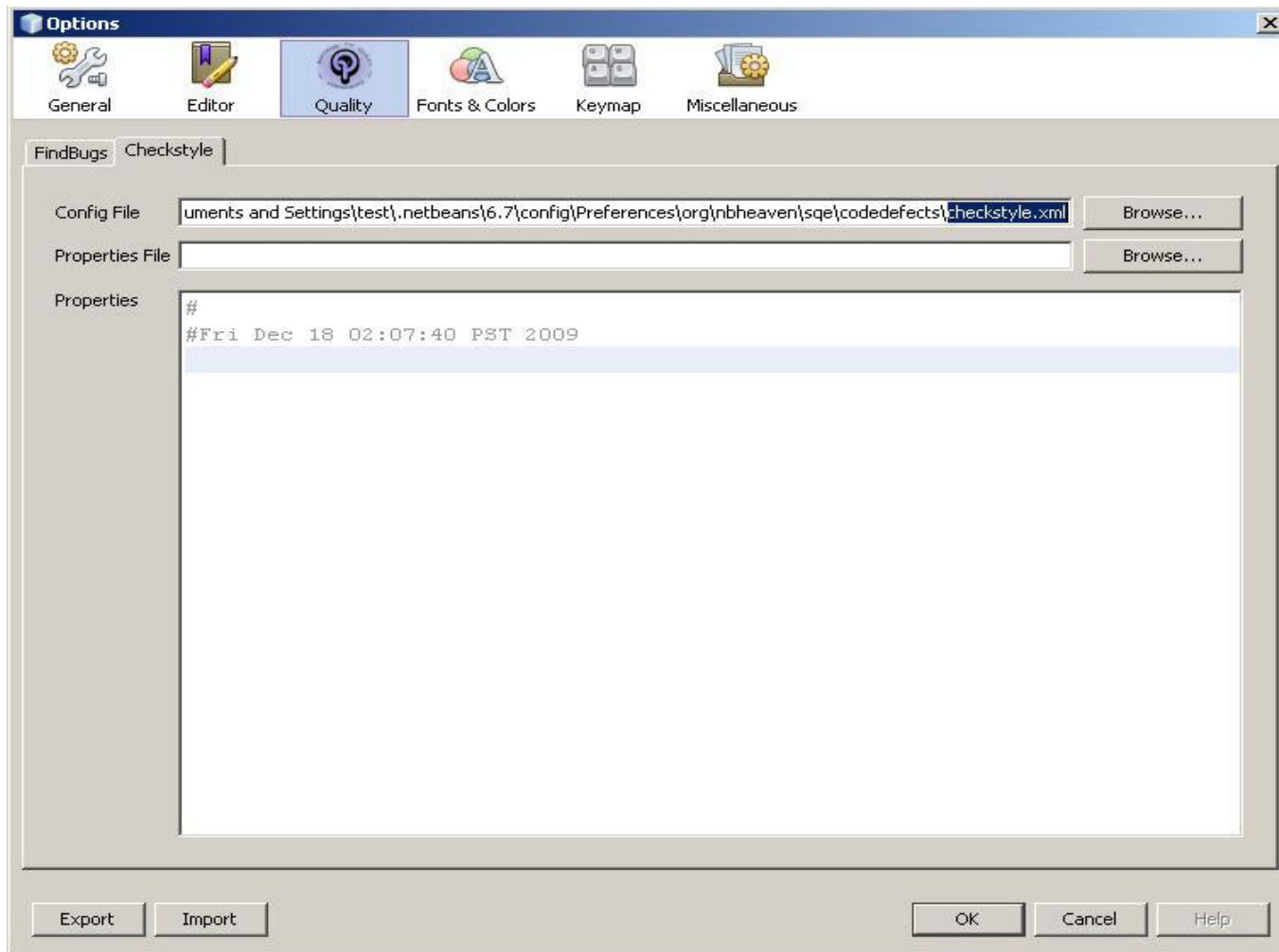
---

- ❑ Customization of two commonly used tool on a popular IDE
  
- ❑ Such that:
  - ❑ CheckStyle
  - ❑ FindBug





# Configuration CheckStyle (NetBeans)



# Configuration FindBug (NetBeans)

Options

General Editor Quality Fonts & Colors Keymap Miscellaneous

FindBugs | Checkstyle

Plugin Id: FindBugs project(edu.umd.cs.findbugs.plugins.core)

Bug Detector	Speed	Enabled
AppendingToObjectOutputStream	fast	<input checked="" type="checkbox"/>
BadAppletConstructor	fast	<input type="checkbox"/>
BadResultSetAccess	fast	<input checked="" type="checkbox"/>
BadSyntaxForRegularExpression	fast	<input checked="" type="checkbox"/>
BadUseOfReturnValue	fast	<input checked="" type="checkbox"/>
BadlyOverriddenAdapter	fast	<input checked="" type="checkbox"/>
BooleanReturnNull	fast	<input checked="" type="checkbox"/>
CallToUnsupportedMethod	fast	<input type="checkbox"/>
CheckImmutableAnnotation	fast	<input checked="" type="checkbox"/>
CheckTypeQualifiers	slow	<input checked="" type="checkbox"/>
CloneIdiom	fast	<input checked="" type="checkbox"/>
ComparatorIdiom	fast	<input checked="" type="checkbox"/>
ConfusedInheritance	fast	<input checked="" type="checkbox"/>

Restore Defaults Disable All Detectors

Export Import OK Cancel Help





# Configuration CheckStyle (xml)

---

- ❑ A Standard XML file that defines a set of modules that are used to verify source code.
- ❑ The Checks which want enable are define in module tags
- ❑ In hierarchy of module root module called the *Checker* module.





# Configuration CheckStyle

---

- ❑ Modules can also contain sub-modules.
- ❑ TreeWalker module parses individual Java source code files.
- ❑ The majority of modules must be nested in the TreeWalker module





# Customizing CheckStyle (Example)

---

A simple Checkstyle configuration is probably as

```
<?xml version="1.0"?>
<!DOCTYPE module PUBLIC
  "-//Puppy Crawl//DTD Check Configuration 1.2//EN"
  "http://www.puppycrawl.com/dtds/configuration_1_2.dtd">

<module name="Checker">
  <module name="TreeWalker">
    <property name="tabWidth" value="4"/>
    <property name="charset" value="UTF-8"/>
    <module name="JavadocMethod"/>
    <module name="JavadocVariable"/>
      <property name="scope" value="protected"/>
    </module>
    <module name="AvoidStarImport"/>
  </module>
  <module name="PackageHtml"/>
</module>
```





# Customizing CheckStyle (Example)

---

## Line Length

```
<module name="LineLength">  
  <property name="severity" value="ignore"/>  
  <property name="max" value="70"/>  
</module>
```

## Documentation

```
<module name="JavadocType">  
  <property name="authorFormat" value="\S"/>  
</module>
```

## Package Name Convention

```
<module name="PackageName">  
  <property name="format" value="^[a-z]+(\.[a-z][a-z0-9]*)*$"/>  
</module>
```



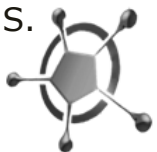


# 1- Comparison of some Bug Tools

---

Name	Version	Input	Interfaces
<b>Bandera</b>	0.3b2	Source	CL, GUI
<b>ESC/Java</b>	2.0a7	Source	CL, GUI
<b>FindBugs</b>	0.8.2	Bytecode	CL, GUI, IDE, Ant
<b>JLint</b>	3.0	Bytecode	CL
<b>PMD</b>	1.9	Source	CL, GUI, Ant, IDE

Reference: *A Comparison of Bug Finding Tools for Java* : Nick Rutar, Christian B. Almazan, Jeffrey S. Foster





# 2- Comparison of some Bug Tools

Bug Category	Example	ESC/ Java	FindBugs	JLint	PMD
General	Null dereference	*✓	*✓	*✓	✓
Concurrency	Possible deadlock	*✓	✓	*✓	✓
Exceptions	Possible unexpected exception	*✓			
Array	Length may be less than zero	✓		*✓	
Mathematics	Division by zero	*✓		✓	
Conditional, loop	Unreachable code due to constant guard		✓		*✓
String	Checking equality using == or !=		✓	*✓	✓
Object overriding	Equal objects must have equal hashcodes		*✓	*✓	*✓
I/O stream	Stream not closed on all paths		*✓		
Unused or duplicate statement	Unused local variable		✓		*✓
Design	Should be a static inner class		*✓		
Unnecessary statement	Unnecessary return statement				*✓
<b>Total</b>		5	<b>8</b>	6	7

Reference: *A Comparison of Bug Finding Tools for Java* : Nick Rutar, Christian B. Almazan, Jeffrey S. Foster





# DEMO

---





# Conclusion

---

- ❑ SQA is an important task for developing error free code.
- ❑ Few errors, although not caught by IDE, may create problems later.
- ❑ Using SCA tools makes it easy to apply the boring task of code review.
- ❑ Organizations should motivate SQA department and even to programmers to use such kind of tools to improve the performance, quality and maintenance of a software



Q & A

