Static Code Analysis to Find Bugs

Wright.edu CS7140 Spring 2013

(Slides collected from many sources)

Comparison of Defect-Detection Approaches

Table 20-2 Defect-Detection Rates

Removal Step	Lowest Rate	Modal Rate	Highest Rate
Informal design reviews	25%	35%	40%
Formal design inspections	45%	55%	65%
Informal code reviews	20%	25%	35%
Formal code inspections	45%	60%	70%
Modeling or prototyping	35%	65%	80%
Personal desk-checking of code	20%	40%	60%
Unit test	15%	30%	50%
New function (component) test	20%	30%	35%
Integration test	25%	35%	40%
Regression test	15%	25%	30%
System test	25%	40%	55%
Low-volume beta test (<10 sites)	25%	35%	40%
High-volume beta test (>1,000 sites)	60%	75%	85%

Source: Adapted from *Programming Productivity* (Jones 1986a), "Software Defect-Removal Efficiency" (Jones 1996), and "What We Have Learned About Fighting Defects" (Shull et al. 2002).

Inspections?

- The combination of design and code inspections usually removes 70-85 percent or more of the defects in a product (Jones 1996).
- Designers and coders learn to improve their work through participating in inspections, and inspections increase productivity by about 20 percent (Fagan 1976, Humphrey 1989, Gilb and Graham 1993, Wiegers 2002).
- On a project that uses inspections for design and code, the inspections will take up about 10-15 percent of project budget and will typically reduce overall project cost.

Best Results – Combine Approaches

- The typical organization uses a test-heavy defe ct-removal approach and achieves only about 85 percent defect removal efficiency.
- Leading organizations use a wider variety of te chniques and achieve defect-removal efficienci es of 95 percent or higher (Gones 2000).

Code Review Tools

- Advantages of Code Review Tools
  - Track suggestions
  - Allow follow up on tasks
  - Aid in comparing before and after changes
  - Source Code repository integration
- Tools
  - Crucible
  - CodeCollaborator
  - Review Board
  - Rietveld
  - Code Striker

Code Analysis Tools

- FindBugs
- PMD
- CheckStyle
- Jdepend
- Ckjm
- Cpd
- Javancss
- Cobertura
- jlint

FindBugs

- A *bug pattern* is a code idiom that is often an error.
  - Difficult language features
  - Misunderstood API methods
  - Misunderstood invariants when code is maintaine
  - typos, wrong boolean operators, …
- *static analysis* to inspect Java bytecode for bug patte rns.
  - Without executing the program
  - don't even need the program's source
- can report *false warnings*, and also miss real errors.
  - In practice, false warnings < 50%.

FindBugs Categories

- Bad practice
- Correctness
- Dodgy
- Experimental
- Internationalization
- Malicious code vulnerability
- Multithreaded correctness
- Performance
- Security

FindBugs Bug Detector Report

The following document contains the results of FindBugs Report

FindBugs Version is *1.3.8*

Threshold is *Low*

Effort is *Max*

Summary

Classes	Bugs	Errors	Missing Classes
2296	1927	20	4

Files

Class	Bugs
com. _ admin.Servlet. _AdminServlet	1
com. .DisplayChart	1
com. .EXCELServlet	2
com. .MSRAction	2

PMD scans Java source code and looks for potential problems like:

- Possible bugs - empty try/catch/finally/switch statements
- Dead code - unused local variables, parameters and private methods
- Suboptimal code - wasteful String/StringBuffer usage
- Overcomplicated expressions - unnecessary if statements, for loops that could be while loops
- Duplicate code - copied/pasted code means copied/pasted bugs

PMD RuleSets

- Android Rules: These rules deal with the Android SDK.
- Basic Rules: The Basic Ruleset contains a collection of good practices which everyone should follow.
- Braces Rules: The Braces Ruleset contains a collection of braces rules.
- Clone Implementation Rules: The Clone Implementation ruleset contains a collection of rules that find questionable usages of the clone() method.
- Code Size Rules: The Code Size Ruleset contains a collection of rules that find code size related problems.
- Controversial Rules: The Controversial Ruleset contains rules that, for whatever reason, are considered controversial.
- Coupling Rules: These are rules which find instances of high or inappropriate coupling between objects and packages.
- Design Rules: The Design Ruleset contains a collection of rules that find questionable designs.
- Import Statement Rules: These rules deal with different problems that can occur with a class' import statements.
- JavaBean Rules: The JavaBeans Ruleset catches instances of bean rules not being followed.
- JUnit Rules: These rules deal with different problems that can occur with JUnit tests.
- Java Logging Rules: The Java Logging ruleset contains a collection of rules that find questionable usages of the logger.
- Migration Rules: Contains rules about migrating from one JDK version to another.
- Migration15: Contains rules for migrating to JDK 1.5
- Naming Rules: The Naming Ruleset contains a collection of rules about names - too long, too short, and so forth.
- Optimization Rules: These rules deal with different optimizations that generally apply to performance best practices.
- Strict Exception Rules: These rules provide some strict guidelines about throwing and catching exceptions.
- String and StringBuffer Rules: Problems that can occur with manipulation of the class String or StringBuffer.
- Security Code Guidelines: These rules check the security guidelines from Sun.
- Unused Code Rules: The Unused Code Ruleset contains a collection of rules that find unused code.

PMD Rule Example

PMD Basic Rules

- EmptyCatchBlock: Empty Catch Block finds instances where an exception is caught, but not hing is done. In most circumstances, this swallows an exception which should either be acte d on or reported.
- EmptyIfStmt: Empty If Statement finds instances where a condition is checked but nothing i s done about it.
- EmptyWhileStmt: Empty While Statement finds all instances where a while statement does nothing. If it is a timing loop, then you should use Thread.sleep() for it; if it's a while loop th at does a lot in the exit expression, rewrite it to make it clearer.
- EmptyTryBlock: Avoid empty try blocks - what's the point?
- EmptyFinallyBlock: Avoid empty finally blocks - these can be deleted.
- EmptySwitchStatements: Avoid empty switch statements.
- JumbledIncrementer: Avoid jumbled loop incrementers - it's usually a mistake, and it's conf using even if it's what's intended.
- ForLoopShouldBeWhileLoop: Some for loops can be simplified to while loops - this makes t hem more concise.

CheckStyle

- Development tool to help programmers write Java c ode that adheres to a coding standard. It automates the process of checking Java code to spare humans of this boring (but important) task.
- Highly configurable and can be made to support al most any coding standard. An example configuratio n file is supplied supporting the Sun Code Conventions. Other sample configuration files are supplied for other well known conventions.

CheckStyle Example

Summary

Files	Infos	Warnings	Errors
14	0	123	12

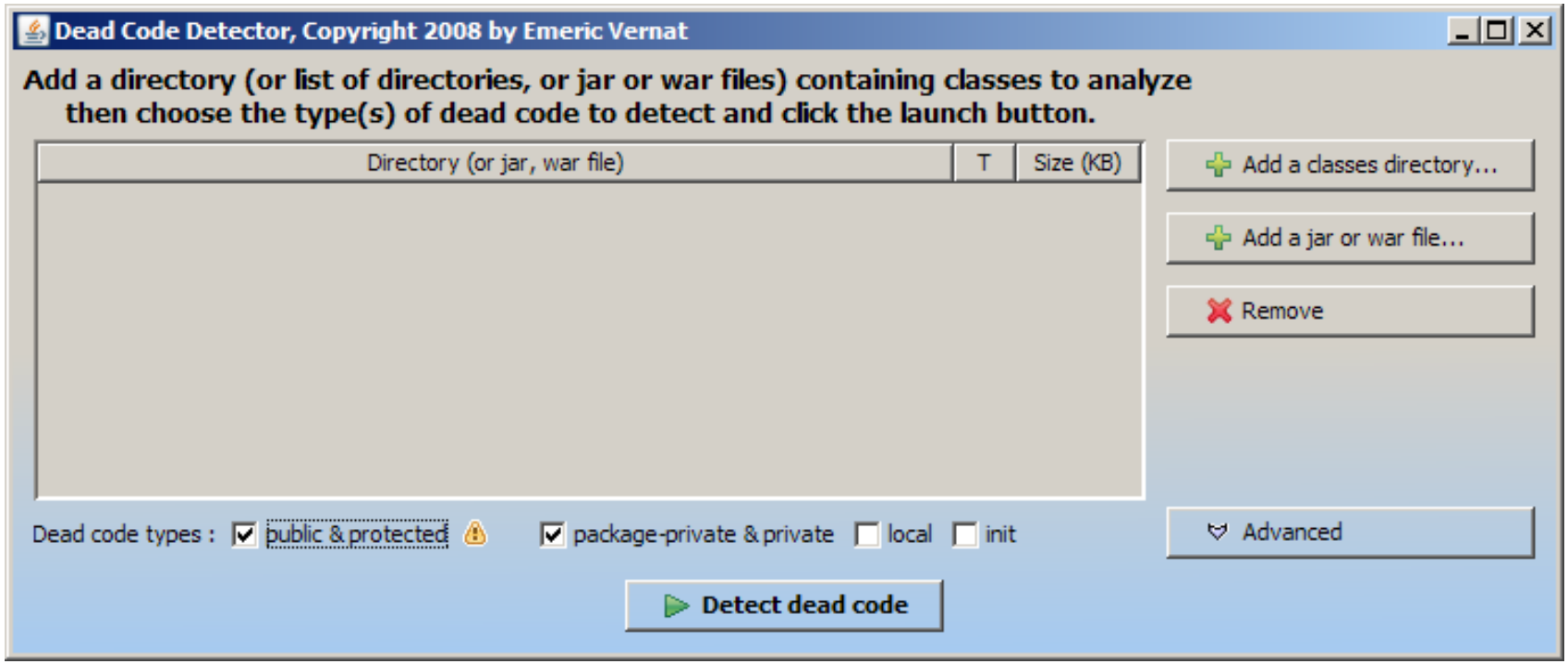
Files

Files	I	W	E
org/apache/maven/plugin/checkstyle/CheckstyleExecutor.java	0	4	0
org/apache/maven/plugin/checkstyle/CheckstyleExecutorException.java	0	4	0
org/apache/maven/plugin/checkstyle/CheckstyleExecutorRequest.java	0	60	2

org/apache/maven/plugin/checkstyle/CheckstyleExecutor.java

Violation	Message	Line
	Unused @param tag for '{@link'.	33
	Expected @param tag for 'request'.	38
	Expected @throws tag for 'CheckstyleExecutorException'.	39
	Expected @throws tag for 'CheckstyleException'.	39

Dead Code Detector



Miscellaneous Tools

- **CKJM** - Chidamber and Kemerer Java Metrics
- **Cobertura & EMMA** – Test Code Coverage
- **JavaNCSS** - A Source Measurement Suite
- **JDepend** – Package Dependencies; Efferent Coup lings (Ce) (number of other packages that the classes in the package dep end upon)
- **PMD-CPD** - Copy/Paste Detector (CPD)
- **Java2HTML** - Source Code turned into a colorize d and browseable HTML representation.

Structure Tools

- Struture101 -- For understanding, analyzing, m easuring and controlling the quality of your So ftware Architecture as it evolves over time.
- Sotoarc/Sotograph — Architecture and quality in-depth analysis and monitoring for Java,
- http://en.wikipedia.org/wiki/List_of_tools_for_static_code_analysis

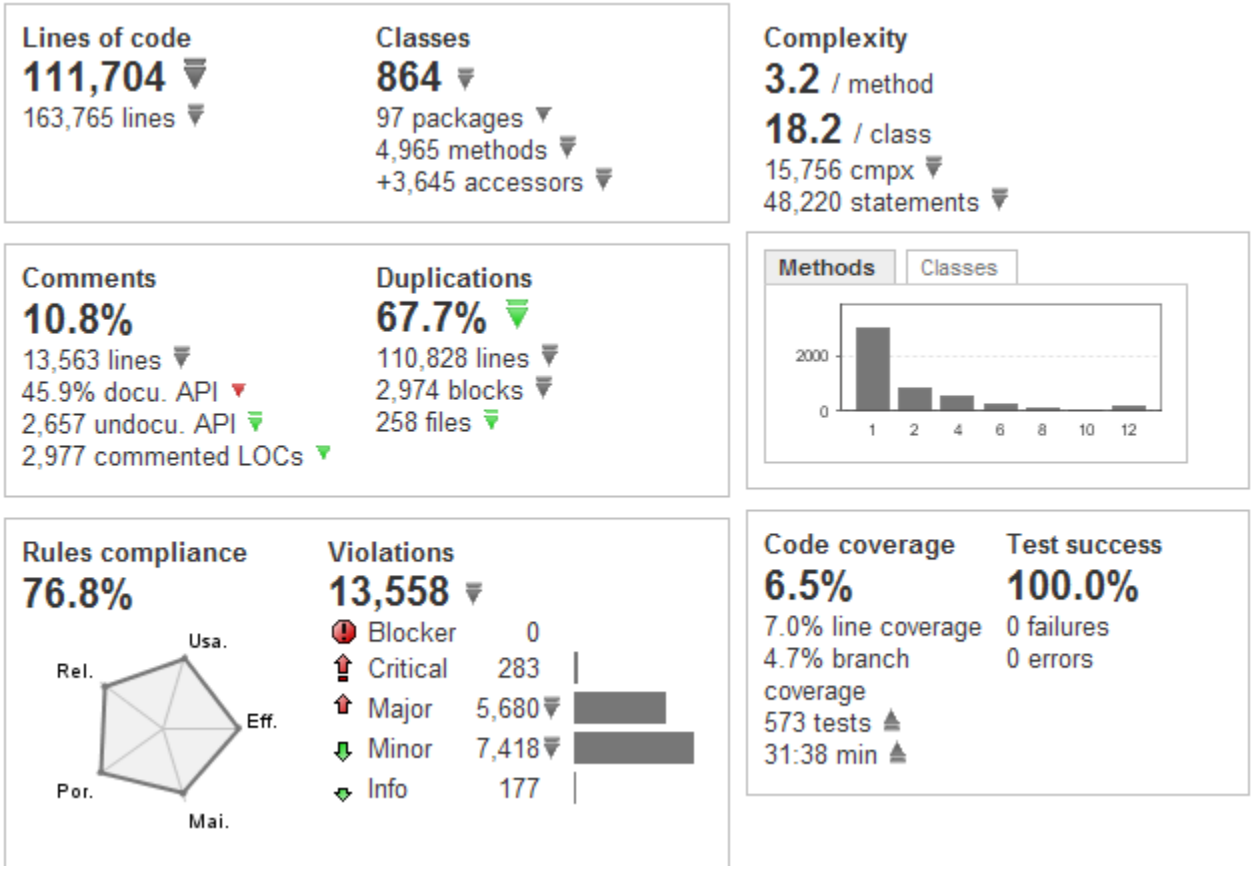
XRadar

- XRadar is an open extensible code report tool c urrently supporting all Java based systems.
- The batch-processing framework produces HT ML/SVG reports of the systems current state an d the development over time - all presented in s exy tables and graphs.

Sonar

- Dashboard to summarize Static and Dynamic a nalysis Tools.
- Conventions (Checkstyle)
- Bad Practices (PMD)
- Potential Bugs (FindBugs)

Sonar Application Dashboard



Sonar Violations Drilldown

Priority	Category
Blocker	0
Critical	283
Major	5,680
Minor	7,418
Info	177

Rule	
Security - Array is stored directly	163
Empty If Stmt	104
Empty Finally Block	9
Avoid Catching Throwable	5
Equals Hash Code	1
Broken Null Check	1

com.	:chem	544
com.	r.gui	498
com.	:dealer.detail	460

GenerateCharts	280
ChemSummaryCharts	280
ChemFillRptDAO	221

Priority	Category
Efficiency	195
Maintainability	4,034
Portability	90
Reliability	7,261
Usability	1,978

Rule	
Security - Array is stored directly	163
Empty If Stmt	104
Empty Finally Block	9
Avoid Catching Throwable	5
Equals Hash Code	1
Broken Null Check	1

com.	em	544

GenerateCharts	280

Sonar Hotspots

Most violated rules Any priority more

Design For Extension	94
Signature Declare Throws Exception	42
Magic Number	23
Visibility Modifier	14
Avoid Duplicate Literals	12

Most violated more

BudgetMaintAction	0	0	22	10	0
BudgetMaintDAO	0	0	21	12	0
BudgetMaintBO	0	0	14	12	0
BudgetMaintActionForm	0	0	7	16	0
BudgetMaintTO	0	0	0	28	0

Longest unit tests more

BudgetMaintTO_UT	266 ms
SpecieTO_UT	250 ms
TeamTO_UT	204 ms
ChemFamilyTO_UT	110 ms

Highest untested lines more

BudgetMaintDAO	269
BudgetMaintAction	120
BudgetMaintActionForm	113
BudgetMaintBO	111
BudgetUploadBO	73

Highest complexity more

BudgetMaintDAO	74
BudgetMaintBO	50
BudgetMaintActionForm	37
BudgetMaintAction	27
BudgetUploadBO	25

Highest average method complexity more

BudgetUploadBO	12.5
BudgetMaintDAO	12.3
BudgetMaintBOFactory	11.0
BudgetMaintActionForm	9.3
BudgetMaintBO	3.3

Highest duplications more

Most undocumented APIs more

IntelliJ Idea

IDE Features	Community Edition	Ultimate Edition
Code Duplicates	No	Yes
Code Coverage	No	Yes
Code Inspector	Yes	Yes
Spell Checker	Yes	Yes

- More than 600 automated Code Inspections
- Finding probable bugs
- Locating the “dead” code
- Detecting performance issues
- Improving code structure and maintainability
- Conforming to coding guidelines and stan-dards
- Conforming to specifications