Darryl Parks

Code Analysis Tools and Tips
(How to make your code ROCK!)

This Presentation is About

About Code Analysis, not Run-Time monitoring

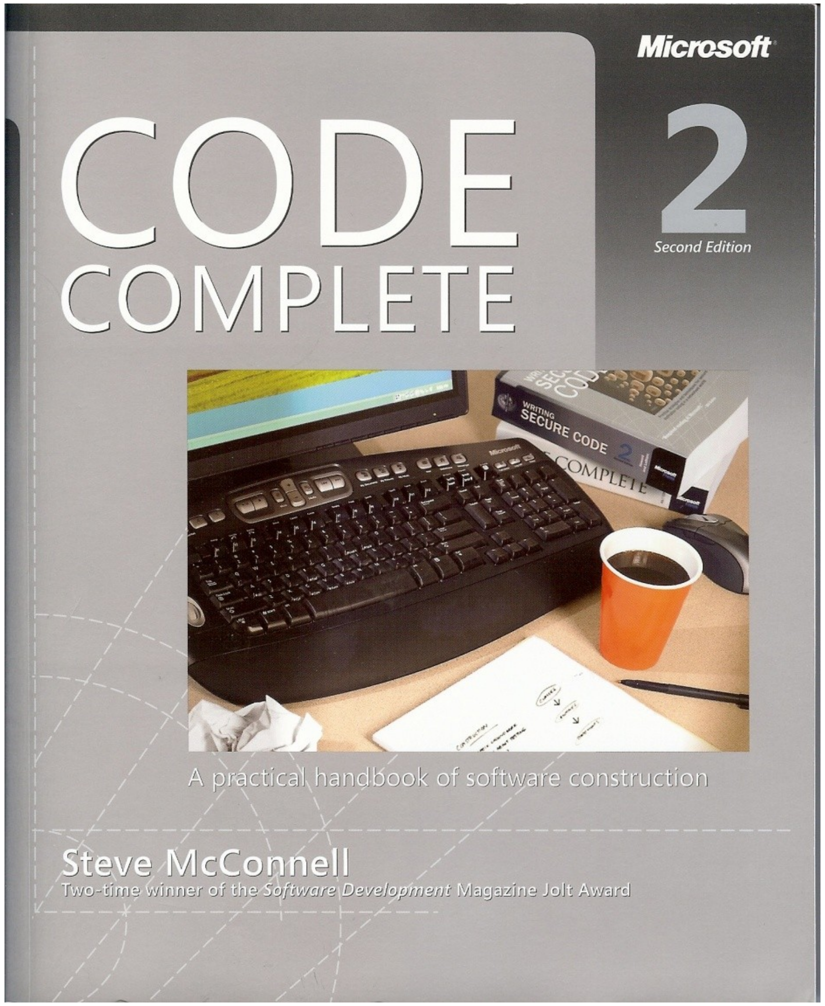
This Presentation is NOT about Performance Analysis Tools

Profiling

Jconsole or other Dynamic Memory Monitoring

Debugging Tools

Main Source of Information for Studies


Microsoft
CODE COMPLETE
2
Second Edition
A practical handbook of software construction
Steve McConnell
Two-time winner of the Software Development Magazine Jolt Award

First edition honored by Software Development Magazine’s Jolt Award for product excellence.

Praised by Martin Fowler, Grady Booch, Alan Cooper and many others.

Comparison of Defect-Detection Approaches

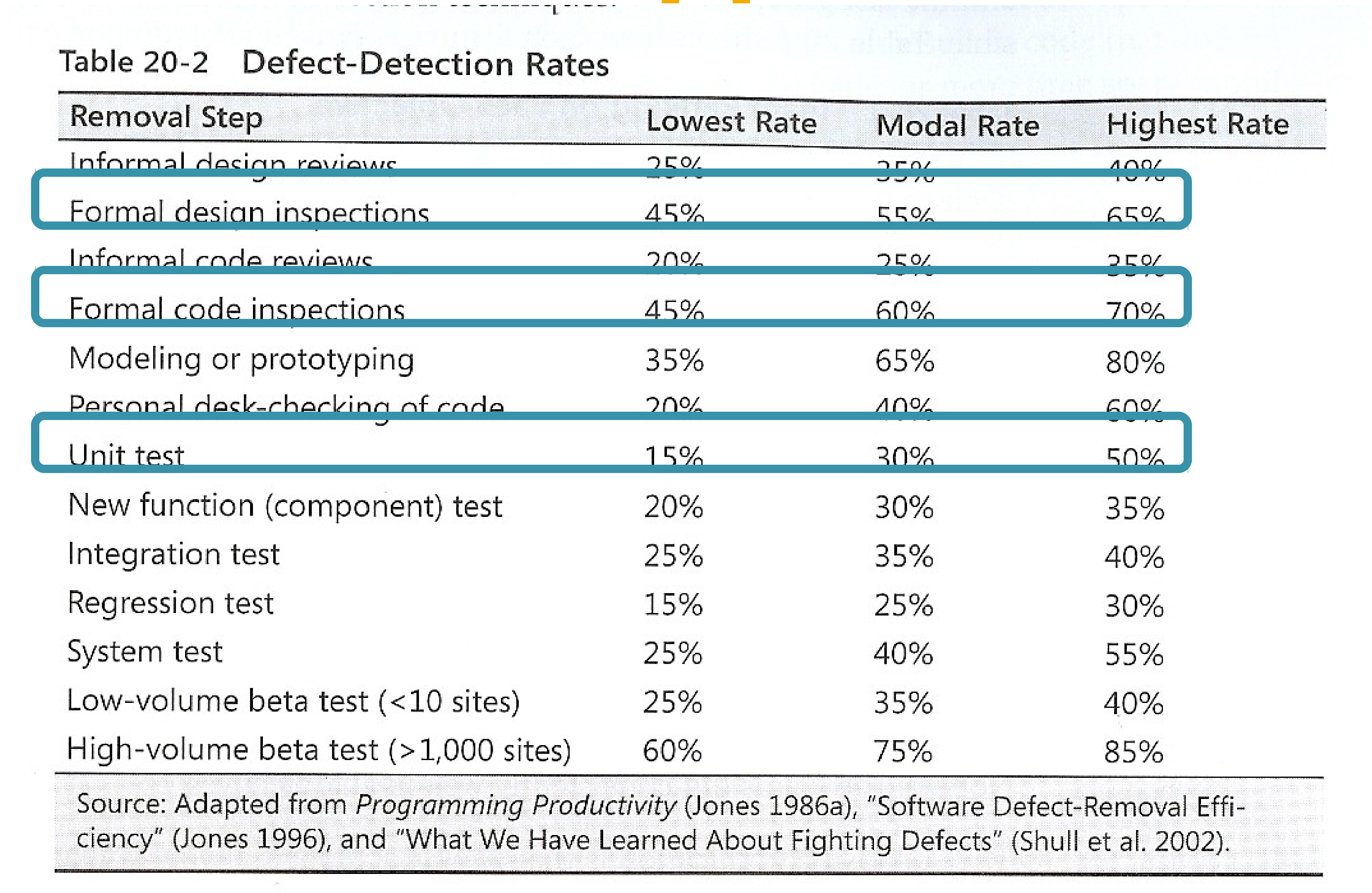


Table 20-2 Defect-Detection Rates

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).

Cost of Finding Defects

Most studies have found that inspections are cheaper than testing. A study at the Soft-ware Engineering Laboratory found that code reading detected about 80 percent more faults per hour than testing (Basili and Selby 1987).

Another organization found that it cost six times as much to detect design defects by using testing as by using inspections (Ackerman, Buchwald, and Lewski 1989).

A later study at IBM found that only 3.5 staff hours were needed to find each error when using code inspections, whereas 15-25 hours

What Results Can You Expect from 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 defect-removal approach and achieves only about 85 percent defect-removal efficiency.

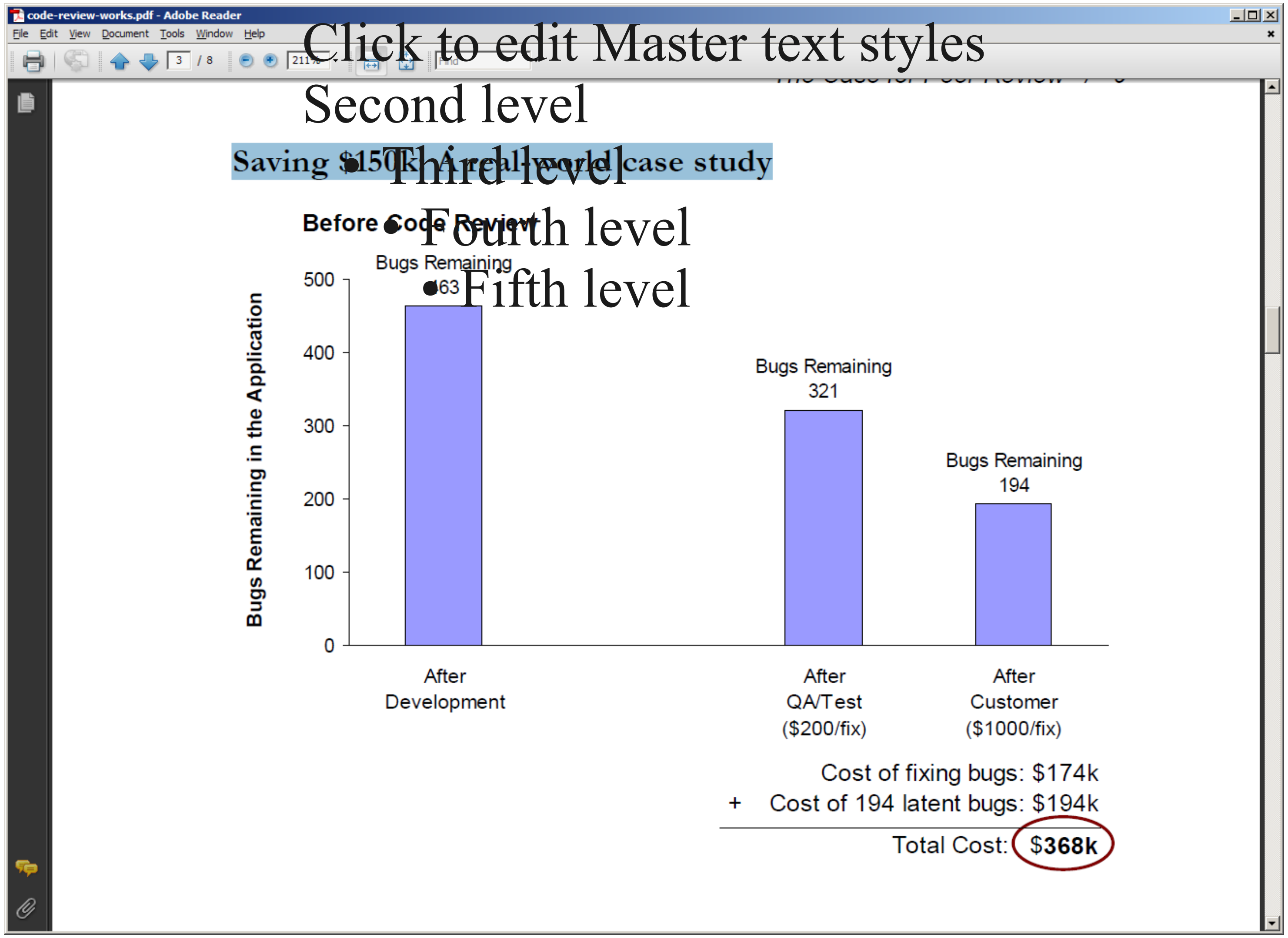
Leading organizations use a wider variety of techniques and achieve defect-removal efficiencies of 95 percent or higher (Gones 2000).

Saving $150k: A real-world case study

Click to edit Master text styles

Second level

- Third level
  - Fourth level
    - Fifth level



code-review-works.pdf - Adobe Reader
File Edit View Document Tools Window Help
3 / 8
211%
Find

Saving $150k: A real-world case study

Before Code Review

Bugs Remaining in the Application
0
100
200
300
400
500
Bugs Remaining 463
Bugs Remaining 321
Bugs Remaining 194
After Development
After QA/Test ($200/fix)
After Customer ($1000/fix)

Cost of fixing bugs: $174k

+ Cost of 194 latent bugs: $194k

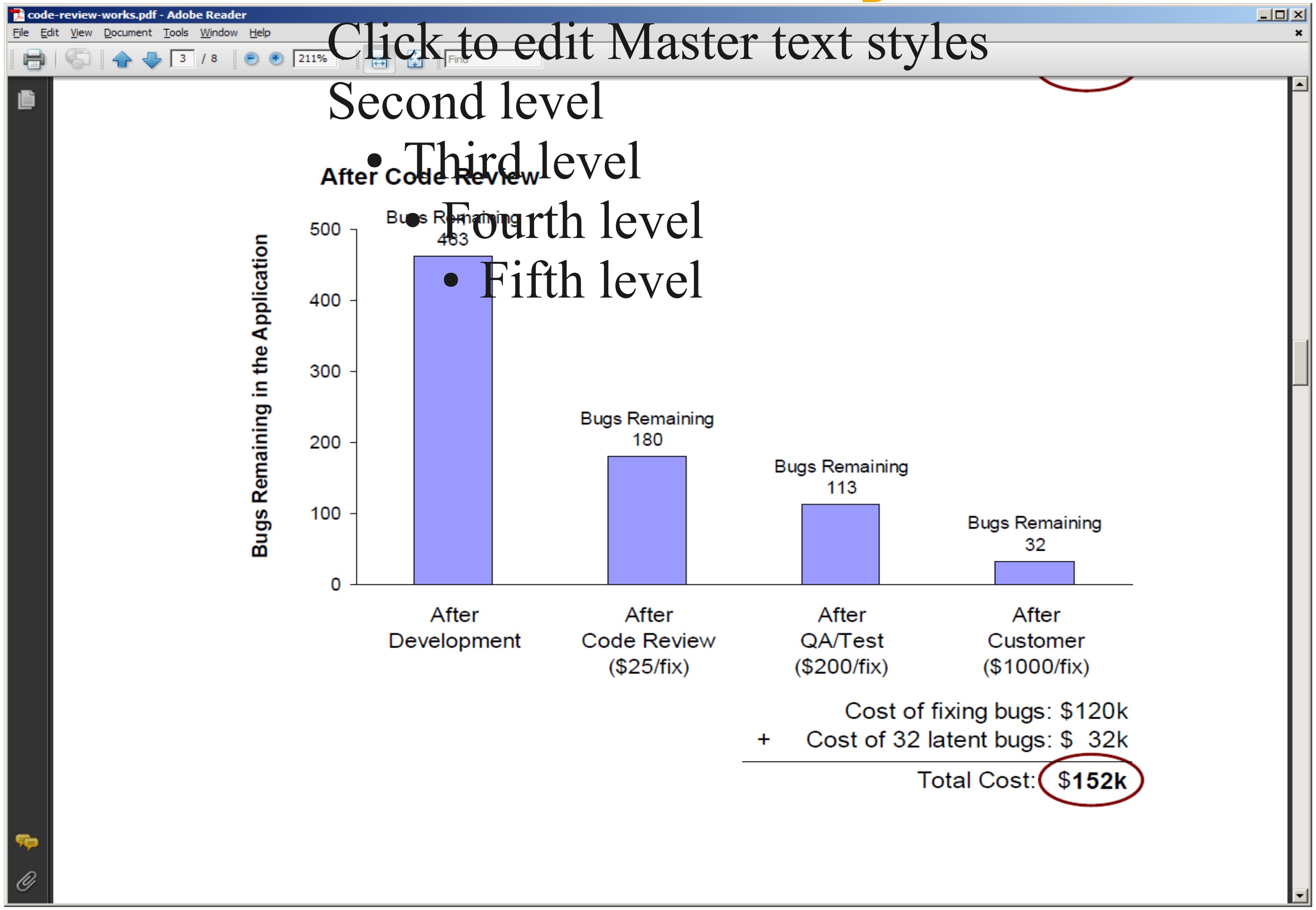
Total Cost: **$368k**

Saving $150k: A real-world case study

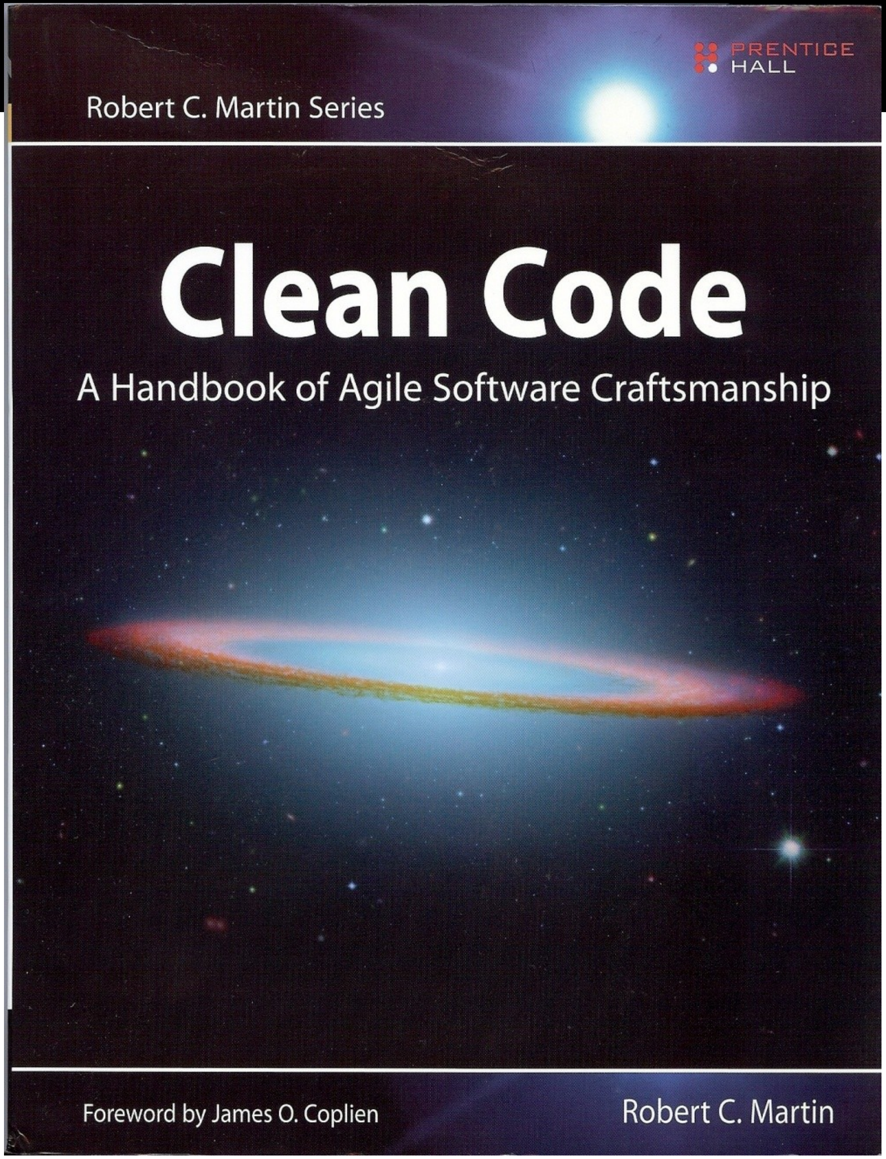
Click to edit Master text styles

Second level

- Third level
  - Fourth level
    - Fifth level



	Cost of fixing bugs:	$120k
+	Cost of 32 latent bugs:	$ 32k
	Total Cost:	**$152k**


PRENTICE HALL
Robert C. Martin Series
Clean Code
A Handbook of Agile Software Craftsmanship
Foreword by James O. Coplien
Robert C. Martin

Purpose:

Is code up to quality standards?

A forum to discuss and learn from everyone.

http://www.objectmentor.com/resources/publishedArticles.html

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

List of available tools:

Crucible

CodeCollaborator ([illegible])

Code Review Issues

Time Consuming

Belittling

Boring

Embarrassing

Maybe “Rubber Stamping”

Code Analysis (Automated Code Reviews)

FindBugs

PMD

CheckStyle

Jdepend

Ckjm

Cpd

Javancss

Cobertura

Jxr - JXR is a source cross reference

FindBugs

Based on the concept of *bug patterns*. A bug pattern is a code idiom that is often an error.

Difficult language features

Misunderstood API methods

Misunderstood invariants when code is modified during maintenance

Garden variety mistakes: typos, use of the wrong boolean operator

FindBugs uses *static analysis* to

FindBugs Categories

Bad practice

Correctness

Dodgy

Experimental

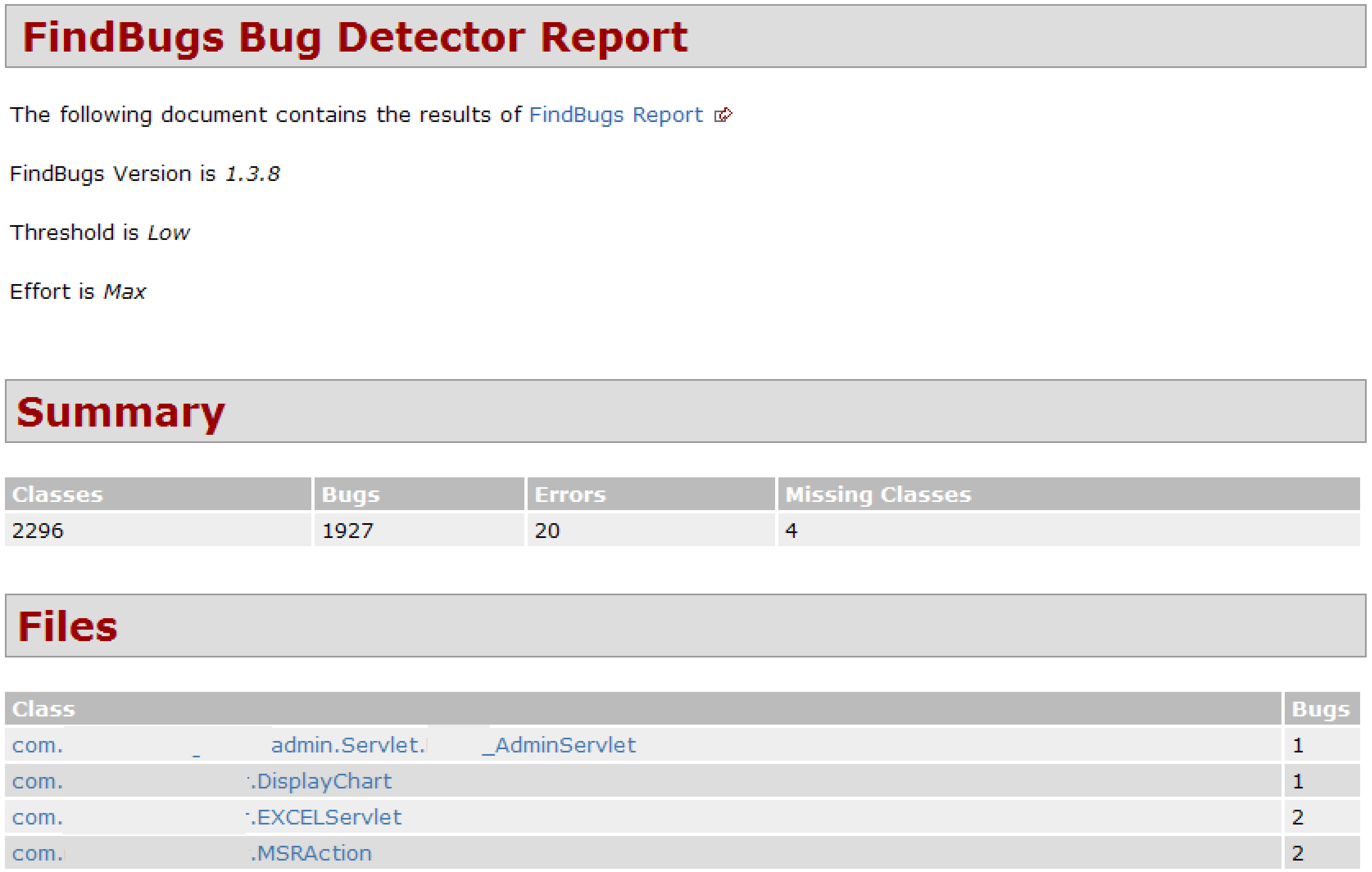
Internationalization

Malicious code vulnerability

Multithreaded correctness

Performance

FindBugs Report



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

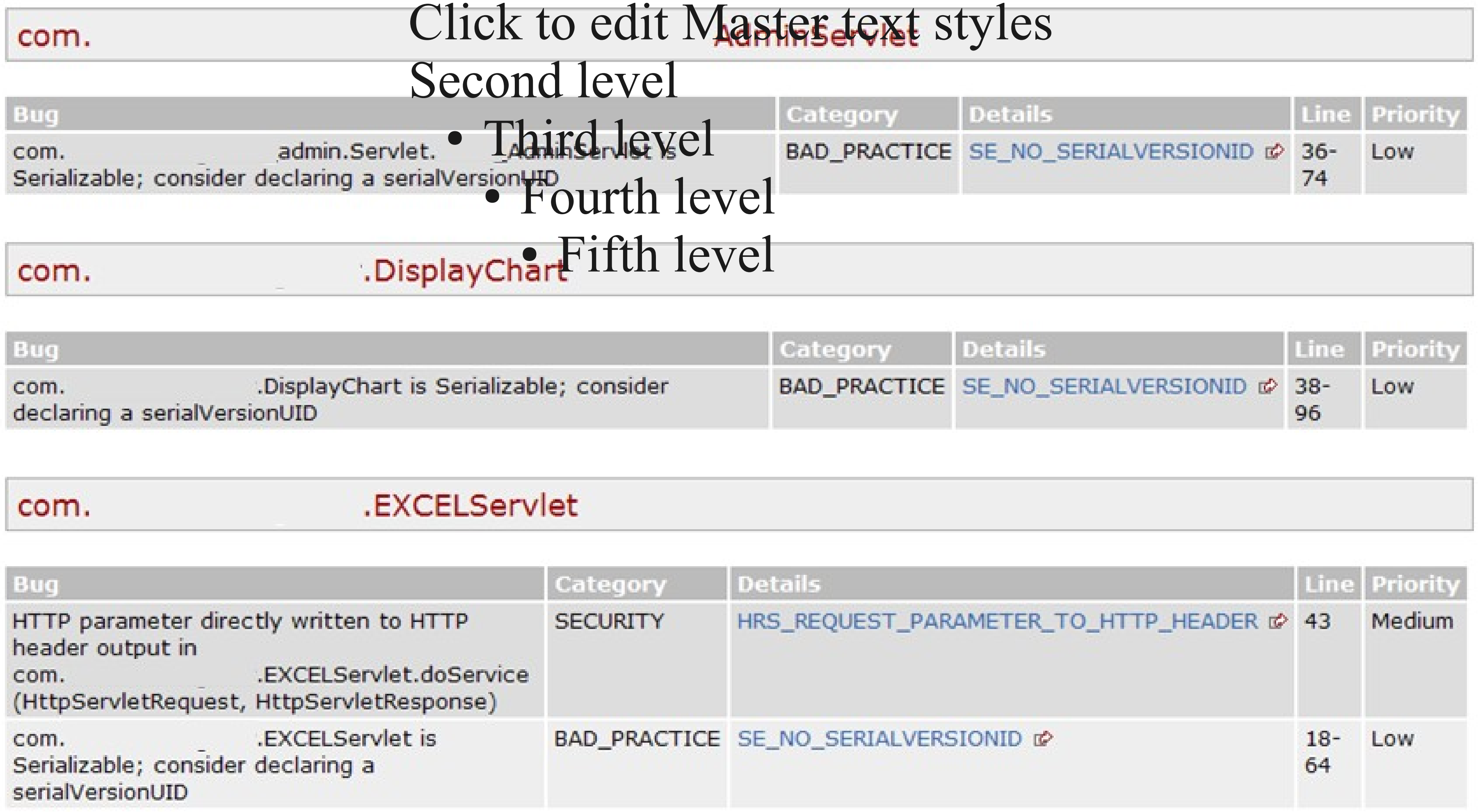
Class	Bugs
com.[redacted]admin.Servlet.[redacted]_AdminServlet	1
com.[redacted].DisplayChart	1
com.[redacted].EXCELServlet	2
com.[redacted].MSRAction	2

FindBugs Detail

Click to edit Master text styles

Second level

- Third level
  - Fourth level
    - Fifth level



com. AdminServlet

Bug	Category	Details	Line	Priority
com. admin.Servlet. AdminServlet is Serializable; consider declaring a serialVersionUID	BAD_PRACTICE	SE_NO_SERIALVERSIONID	36-74	Low

com. .DisplayChart

Bug	Category	Details	Line	Priority
com. .DisplayChart is Serializable; consider declaring a serialVersionUID	BAD_PRACTICE	SE_NO_SERIALVERSIONID	38-96	Low

com. .EXCELServlet

Bug	Category	Details	Line	Priority
HTTP parameter directly written to HTTP header output in com. .EXCELServlet.doService (HttpServletRequest, HttpServletResponse)	SECURITY	HRS_REQUEST_PARAMETER_TO_HTTP_HEADER	43	Medium
com. .EXCELServlet is Serializable; consider declaring a serialVersionUID	BAD_PRACTICE	SE_NO_SERIALVERSIONID	18-64	Low

PMD

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 -

PMD RuleSets

Android Rules: These rules deal with the Android SDK.

Basic JSF rules: Rules concerning basic JSF guidelines.

Basic JSP rules: Rules concerning basic JSP guidelines.

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.

J2EE Rules: These are rules for J2EE

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.

Jakarta Commons Logging Rules: Logging ruleset contains a collection of rules that find questionable usages.

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.

Type Resolution Rules: These are rules which resolve java Class files for comparisson, as opposed to a String

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 nothing is done. In most circumstances, this swallows an exception which should either be acted on or reported.

EmptyIfStmt: Empty If Statement finds instances where a condition is checked but nothing is done about it.

EmptyWhileStmt: Empty While Statement finds all instances where a while statement

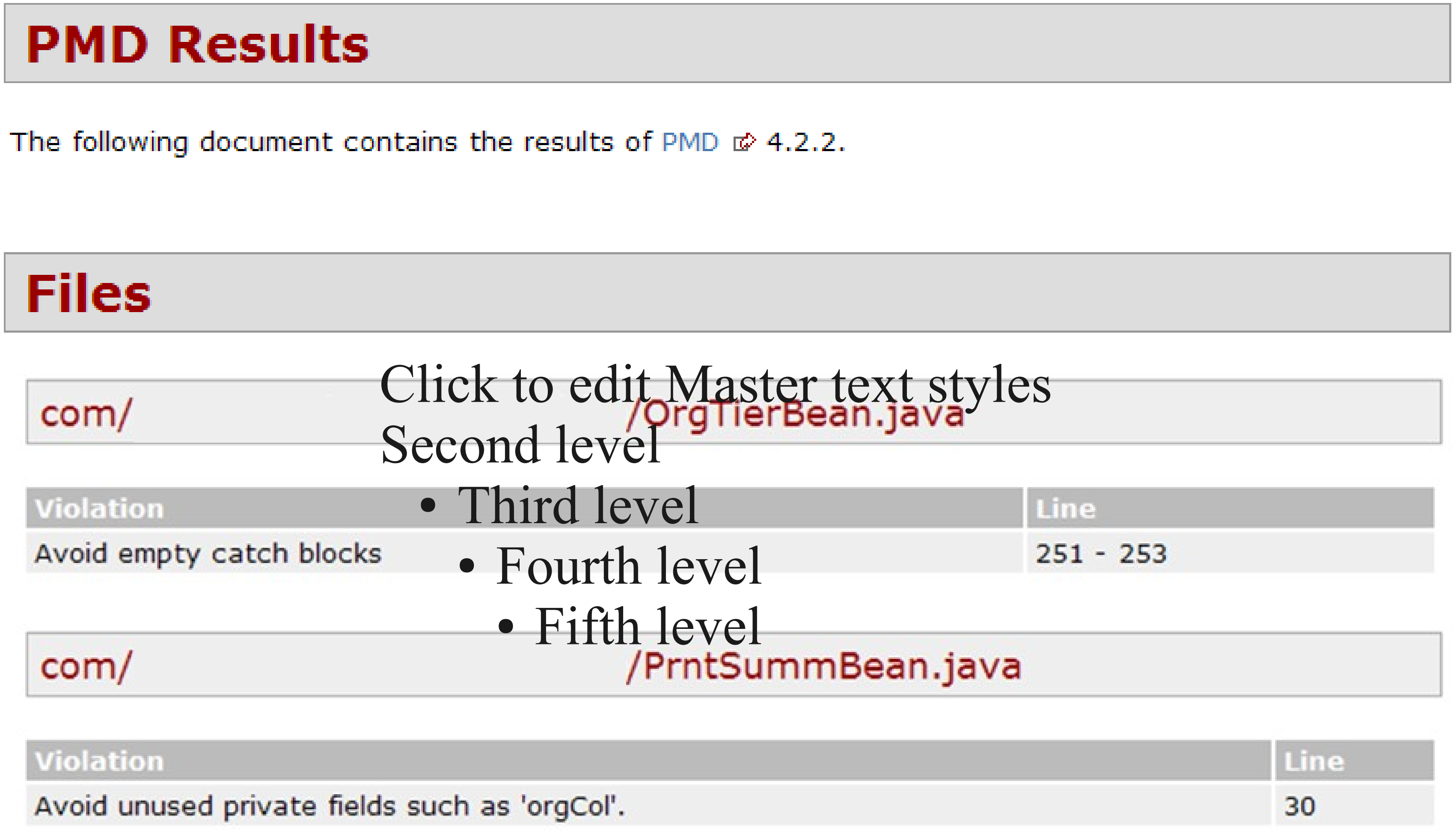
Maven PMD Configuration

```
<project>
 ...
   <reporting>
      <plugins>
        <plugin>

<groupId>org.apache.maven.plugins<
/groupId>      <artifactId>maven-pmd-
plugin</artifactId>
```

PMD Configuration

```
<reporting>
   <plugins>
   <plugin>

<groupId>org.apache.maven.plugins</groupI
d>
   <artifactId>maven-pmd-plugin</artifactId>
   <configuration>
   <rulesets>
   <ruleset>/rulesets/braces.xml</ruleset>
```

PMD Example Report



PMD Results

The following document contains the results of PMD 4.2.2.

Files

com/ /OrgTierBean.java

Violation	Line
Avoid empty catch blocks	251 - 253

com/ /PrntSummBean.java

Violation	Line
Avoid unused private fields such as 'orgCol'.	30

Click to edit Master text styles

Second level

- Third level
  - Fourth level
    - Fifth level

CheckStyle

Development tool to help programmers write Java code that adheres to a coding standard. It automates the process of checking Java code to spare humans of this boring (but important) task.

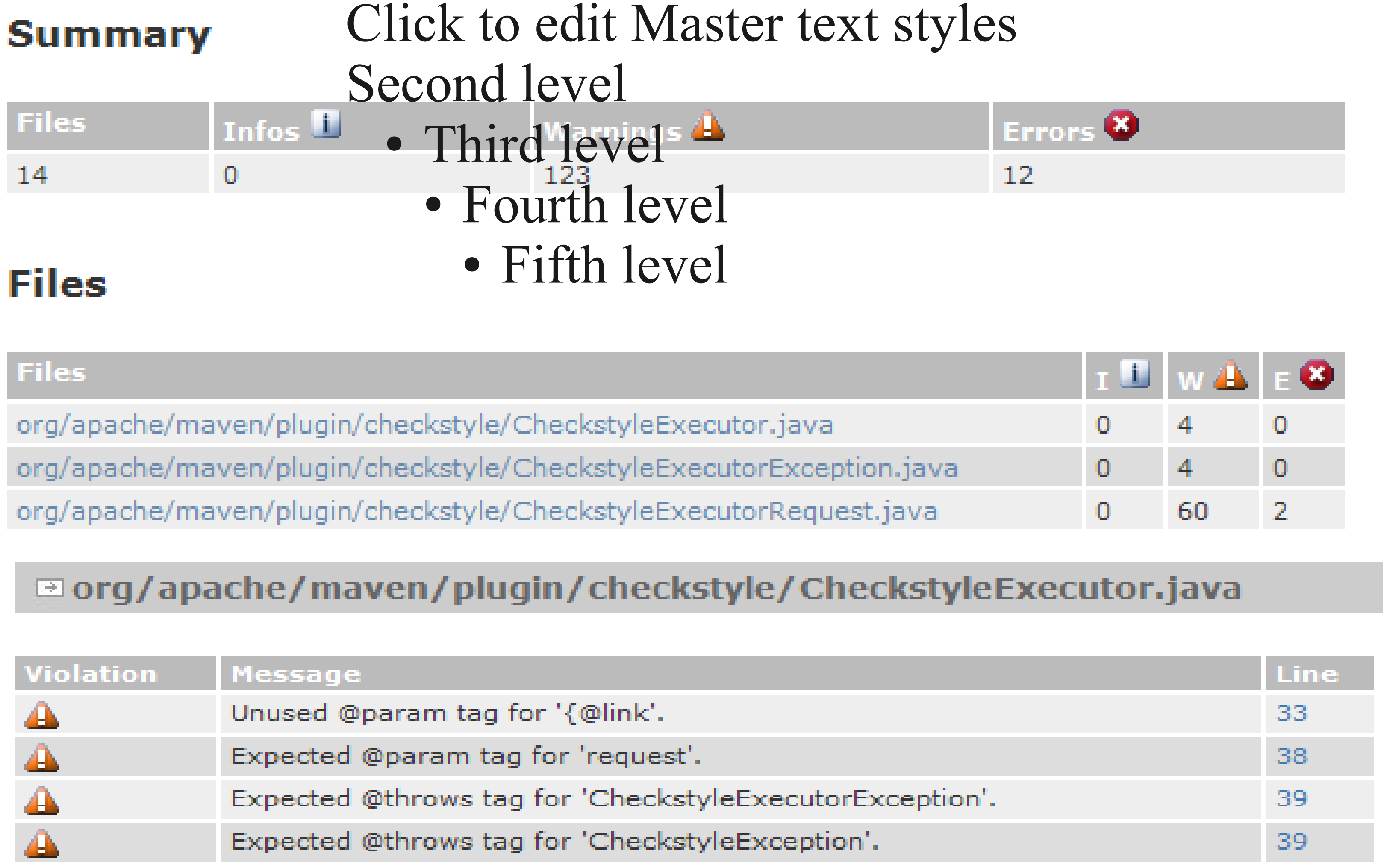
Highly configurable and can be made to support almost any coding standard. An example configuration file is supplied supporting the

CheckStyle Example

Click to edit Master text styles

Second level

- Third level
  - Fourth level
    - Fifth level



Summary

Files	Infos	Warnings	Errors
14	0	123	12

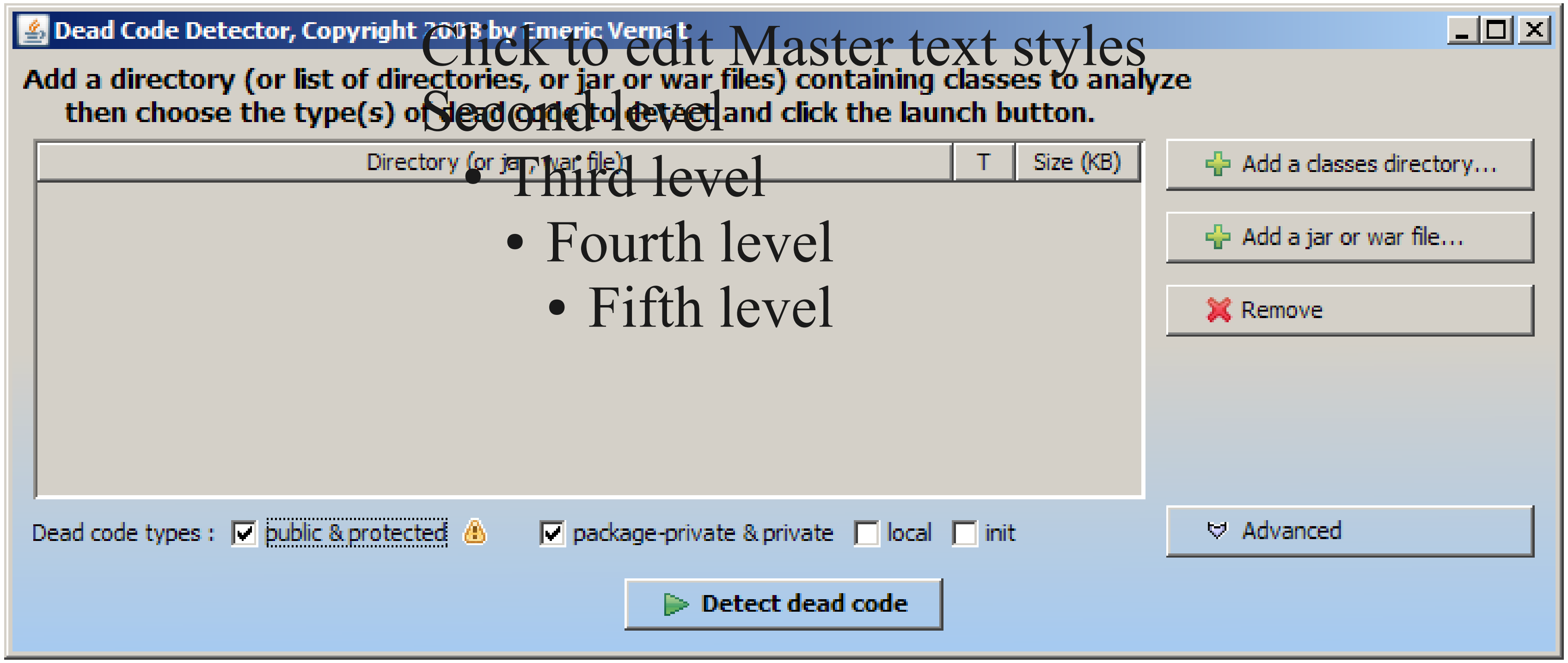
Files

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



Dead Code Detector, Copyright 2008 by Emeric Vernat

Add a directory (or list of directories, or jar or war files) containing classes to analyze
then choose the type(s) of dead code to detect and click the launch button.

Dead code types : [✓ public & protected] [✓ package-private & private] local init

Click to edit Master text styles

Second level

- Third level
  - Fourth level
    - Fifth level

Miscellaneous Tools

CKJM - Chidamber and Kemerer Java Metrics

Cobertura & EMMA – Test Code Coverage

JavaNCSS - A Source Measurement Suite

JDepend – Package Dependencies; Efferent Couplings (Ce) (number of other packages that the classes in the package depend upon)

PMD-CPD - Copy/Paste Detector (CPD)

Structure Tools

Struture101 -- For understanding, analyzing, measuring and controlling the quality of your Software 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_fo

XRadar

XRadar is an open extensible code report tool currently supporting all Java based systems.

The batch-processing framework produces HTML/SVG reports of the systems current state and the development over time - all presented in sexy tables and graphs.

It gets results from several brilliant open source projects and a couple of in house grown projects and presents the

Xradar – MVN Site

```
<reporting>
   <plugins>
      <plugin>
         <groupId>net.sf.xradar</groupId>
         <artifactId>maven-xradar-
plugin</artifactId>
         <version>1.2.2</version>
      </plugin>
   </plugins>
```

DEMO

Sonar

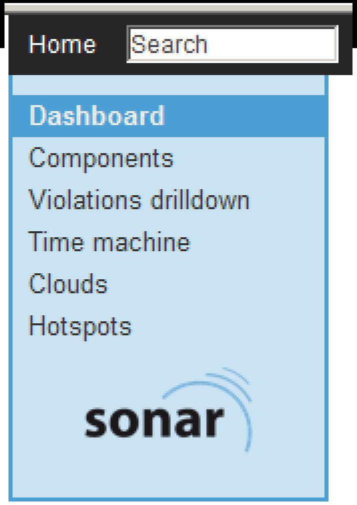
Dashboard to summarize Static and Dynamic analysis Tools.

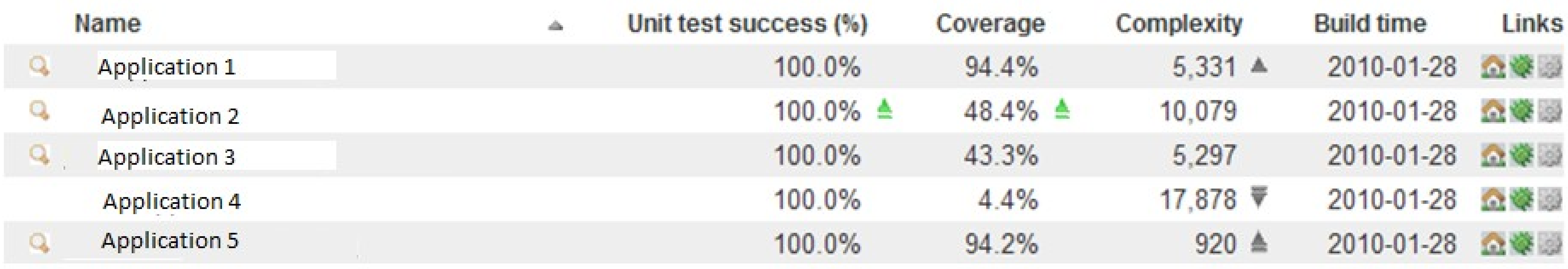
Conventions (Checkstyle)

Bad Practices (PMD)

Potential Bugs (FindBugs)

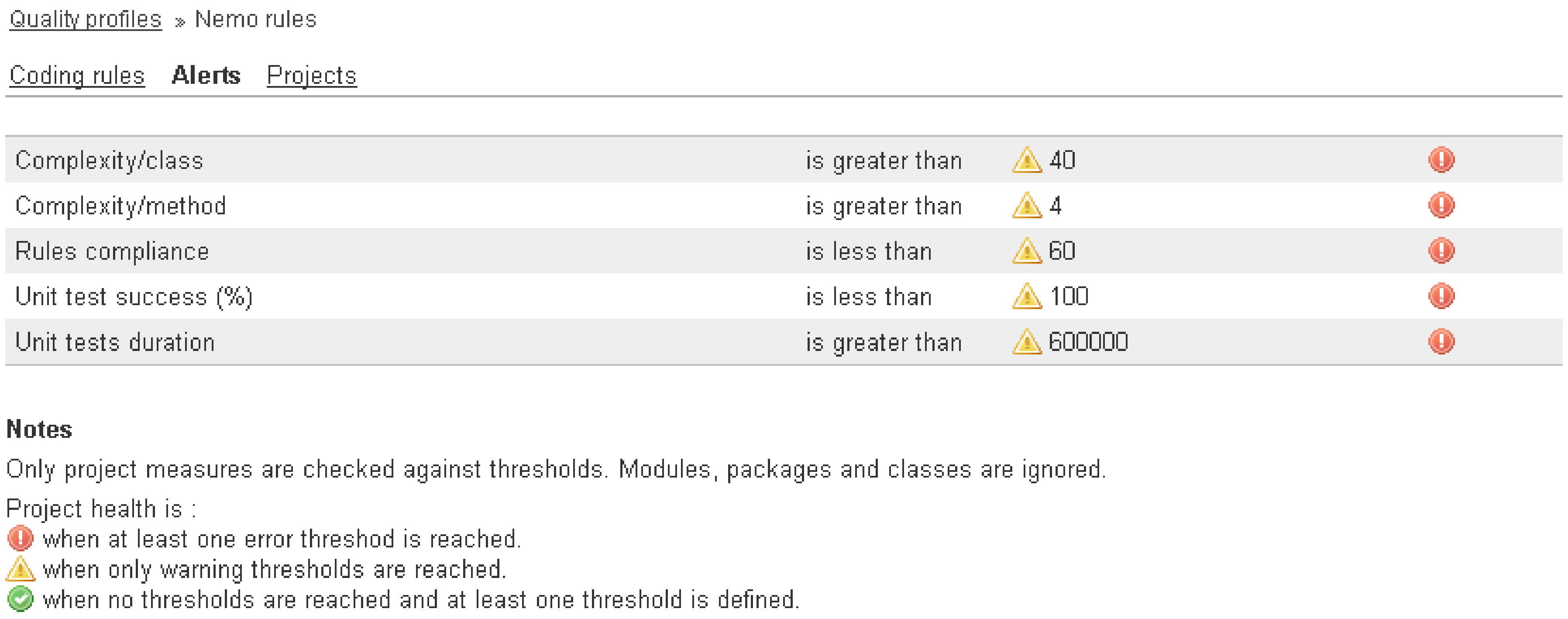
Sonar Example – Front Dashboard





Name	Unit test success (%)	Coverage	Complexity	Build time	Links
Application 1	100.0%	94.4%	5,331 ▲	2010-01-28	
Application 2	100.0% ▲	48.4% ▲	10,079	2010-01-28	
Application 3	100.0%	43.3%	5,297	2010-01-28	
Application 4	100.0%	4.4%	17,878 ▼	2010-01-28	
Application 5	100.0%	94.2%	920 ▲	2010-01-28	

Sonar Setting Alerts



Quality profiles » Nemo rules

Coding rules **Alerts** Projects

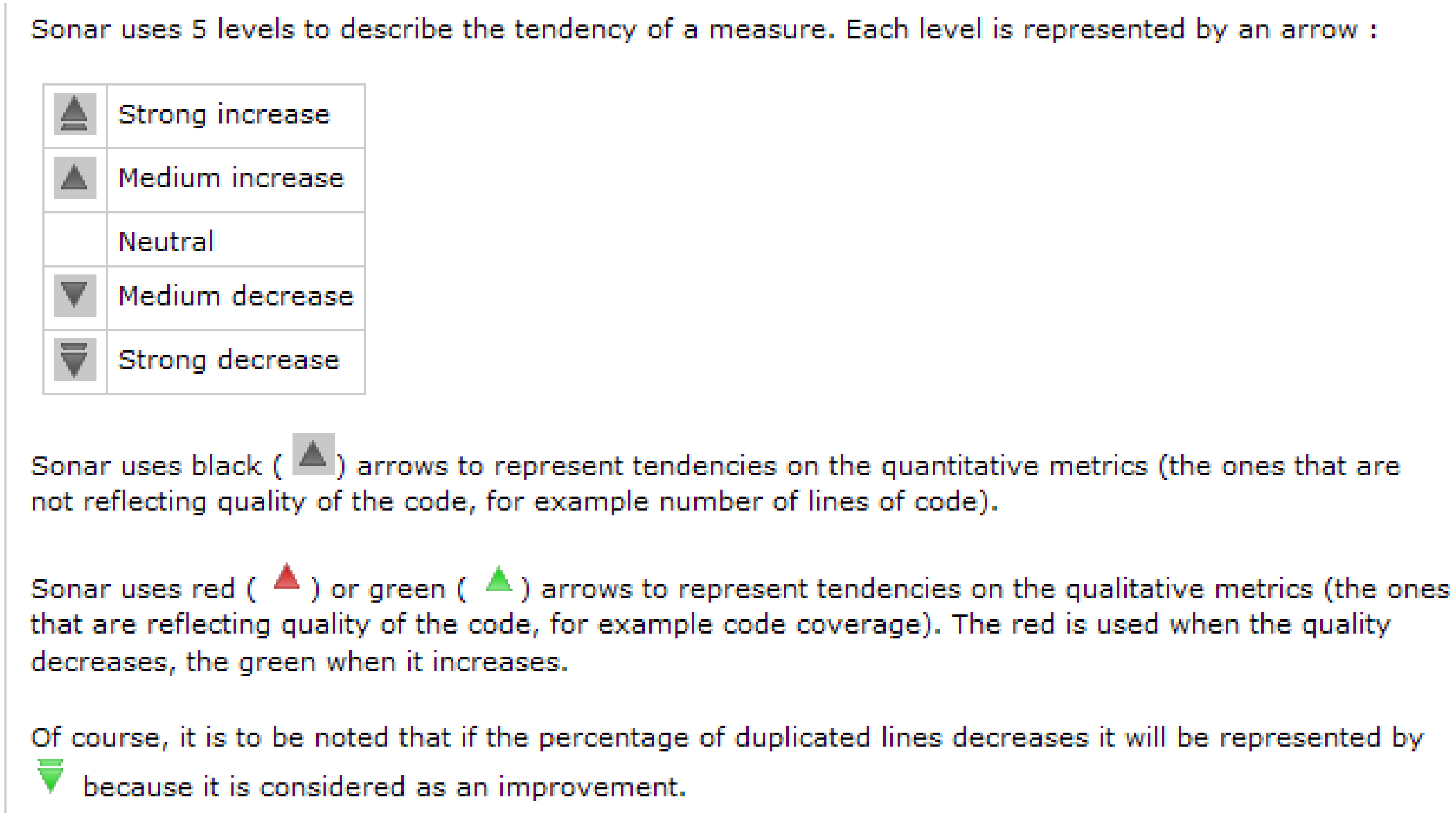
Complexity/class	is greater than	⚠ 40	●
Complexity/method	is greater than	⚠ 4	●
Rules compliance	is less than	⚠ 60	●
Unit test success (%)	is less than	⚠ 100	●
Unit tests duration	is greater than	⚠ 600000	●

Notes

Only project measures are checked against thresholds. Modules, packages and classes are ignored.

Project health is :
● when at least one error threshod is reached.
⚠ when only warning thresholds are reached.
✓ when no thresholds are reached and at least one threshold is defined.

Reading Sonar Tendencies



Sonar uses 5 levels to describe the tendency of a measure. Each level is represented by an arrow :

⏶	Strong increase
▲	Medium increase
	Neutral
▼	Medium decrease
⏷	Strong decrease

Sonar uses black (▲) arrows to represent tendencies on the quantitative metrics (the ones that are not reflecting quality of the code, for example number of lines of code).

Sonar uses red (▲) or green (▲) arrows to represent tendencies on the qualitative metrics (the ones that are reflecting quality of the code, for example code coverage). The red is used when the quality decreases, the green when it increases.

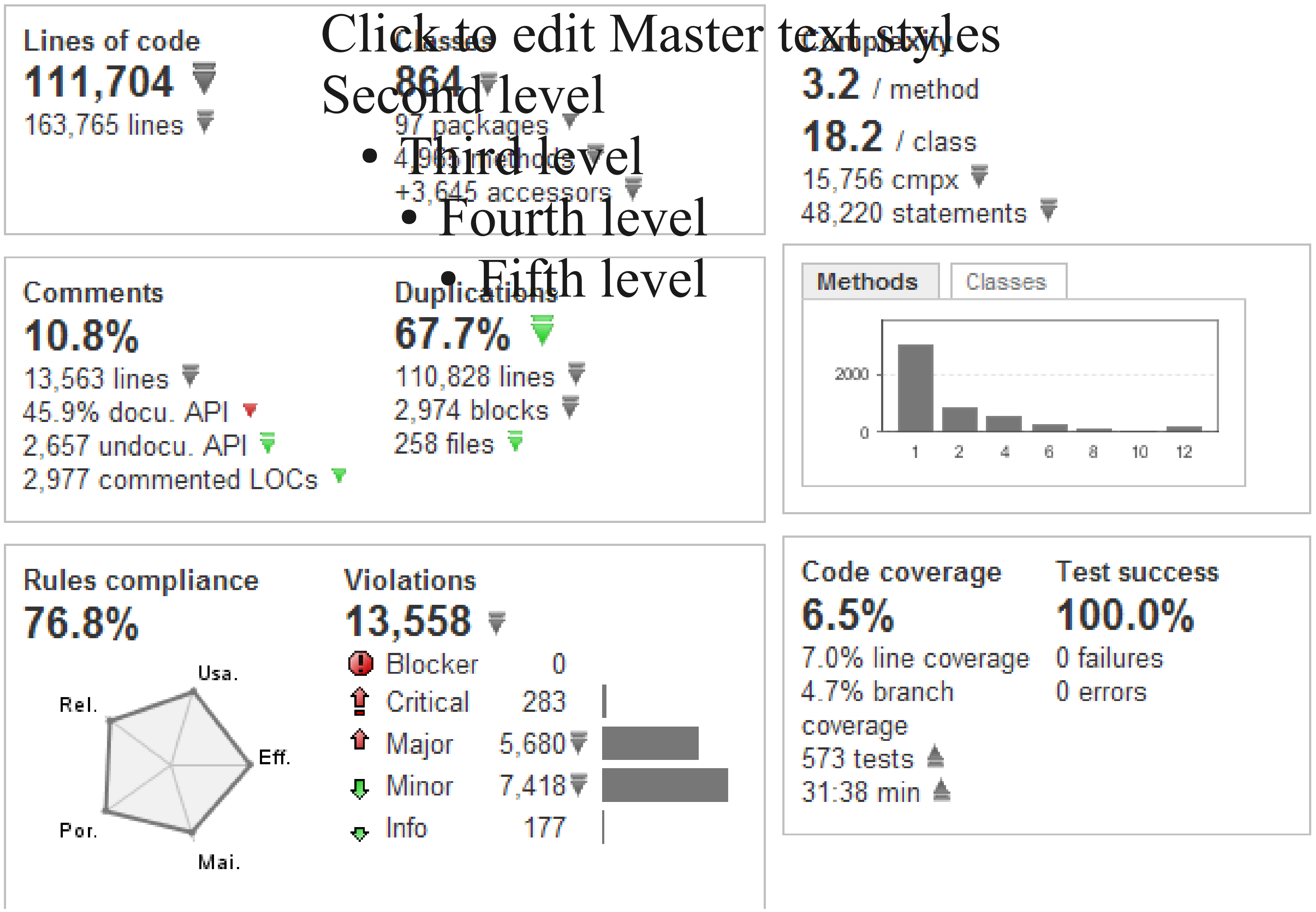
Of course, it is to be noted that if the percentage of duplicated lines decreases it will be represented by ⏷ because it is considered as an improvement.

Sonar Application Dashboard

Click to edit Master text styles

Second level

- Third level
  - Fourth level
    - Fifth level



Lines of code
111,704
163,765 lines

Classes
864
97 packages
4,965 methods
+3,645 accessors

Complexity
3.2 / method
18.2 / class
15,756 cmpx
48,220 statements

Comments
10.8%
13,563 lines
45.9% docu. API
2,657 undocu. API
2,977 commented LOCs

Duplications
67.7%
110,828 lines
2,974 blocks
258 files

Rules compliance
76.8%

Violations
13,558

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

Code coverage
6.5%
7.0% line coverage
4.7% branch coverage
573 tests
31:38 min

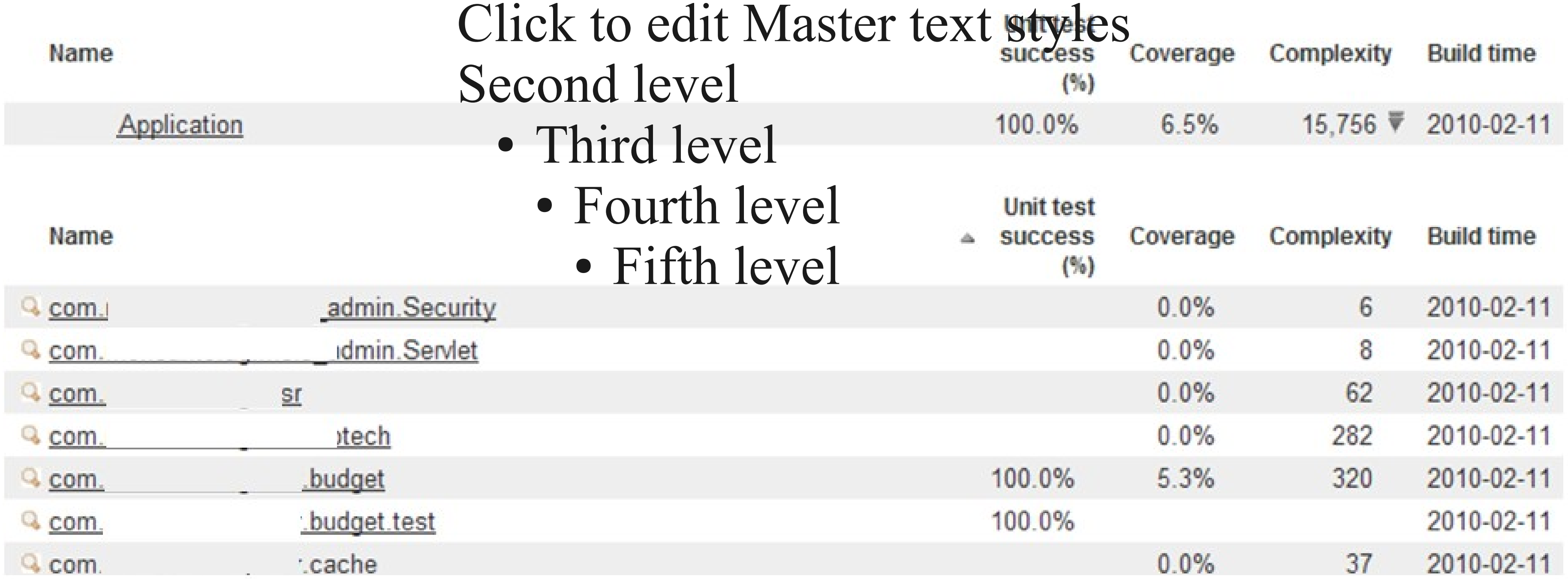
Test success
100.0%
0 failures
0 errors

Sonar Components

Click to edit Master text styles

Second level

- Third level
  - Fourth level
    - Fifth level



Name	Unit test success (%)	Coverage	Complexity	Build time
Application	100.0%	6.5%	15,756 ▼	2010-02-11

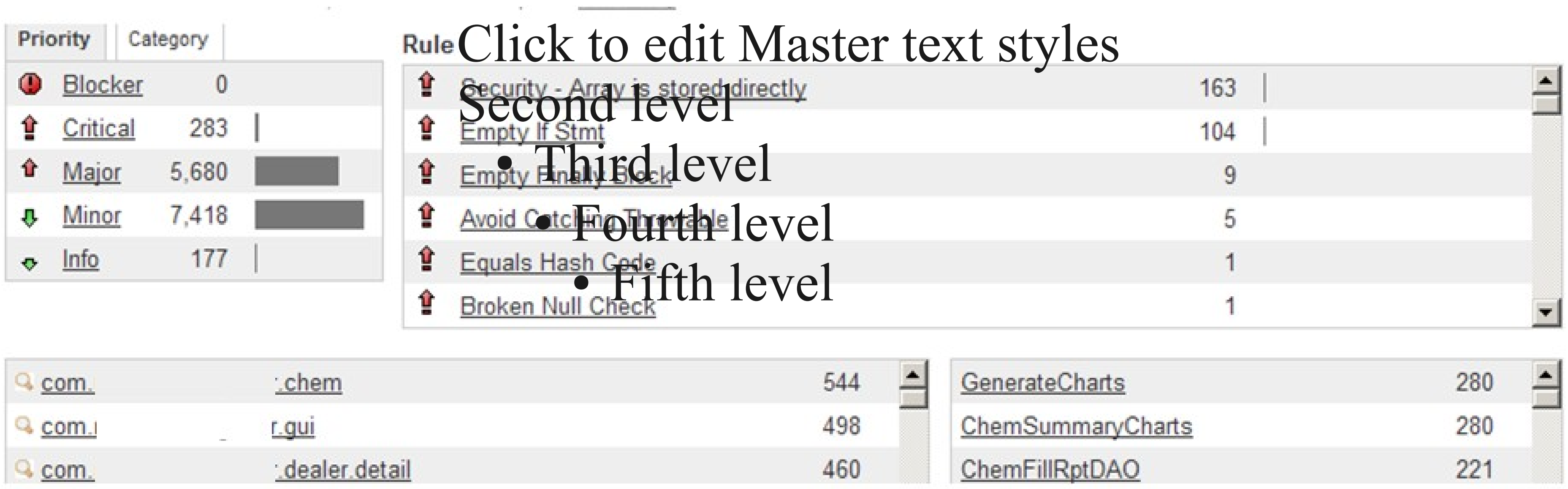
Name	Unit test success (%)	Coverage	Complexity	Build time
com. admin.Security		0.0%	6	2010-02-11
com. dmin.Servlet		0.0%	8	2010-02-11
com. sr		0.0%	62	2010-02-11
com. tech		0.0%	282	2010-02-11
com. .budget	100.0%	5.3%	320	2010-02-11
com. .budget.test	100.0%			2010-02-11
com. .cache		0.0%	37	2010-02-11

Sonar Violations Drilldown

Click to edit Master text styles

Second level

- Third level
  - Fourth level
    - Fifth level



Priority | Category

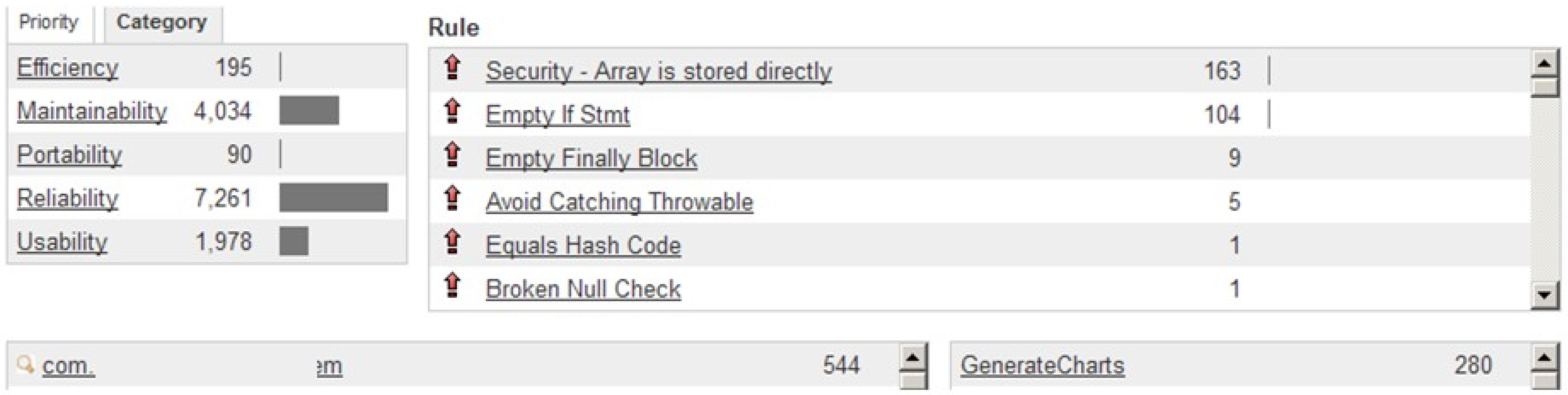
Blocker	0
Critical	283
Major	5,680
Minor	7,418
Info	177

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.i 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

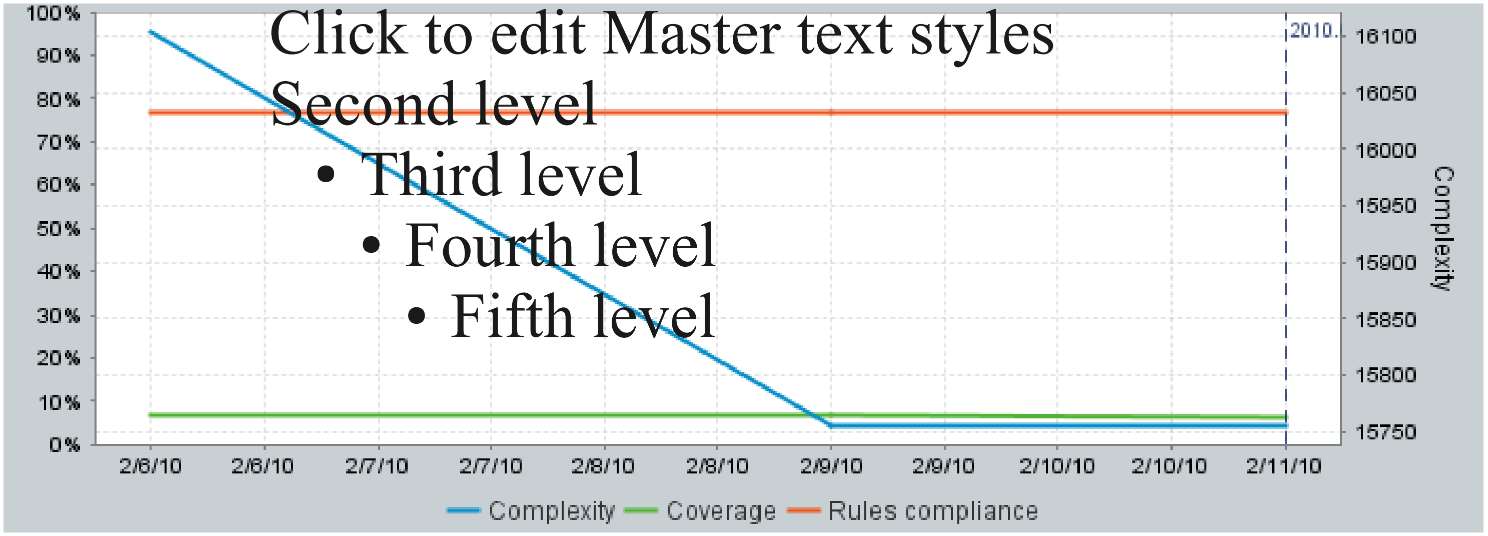
GenerateCharts	280

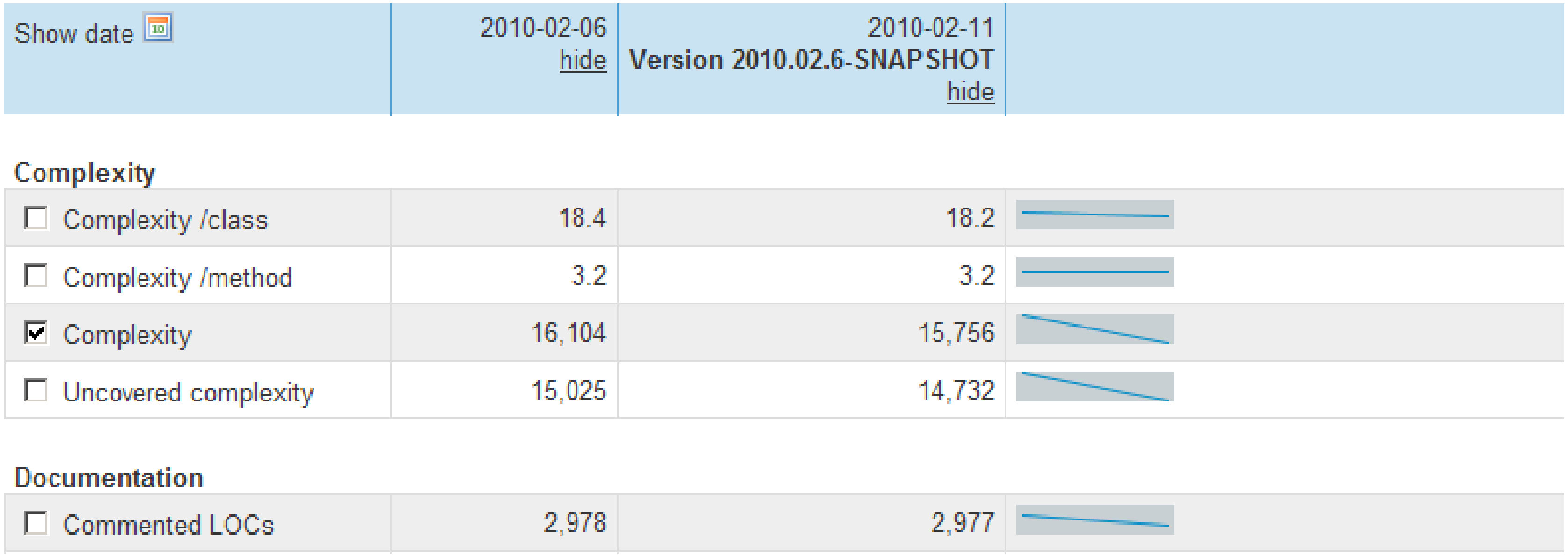
Sonar Time Machine

Click to edit Master text styles

Second level

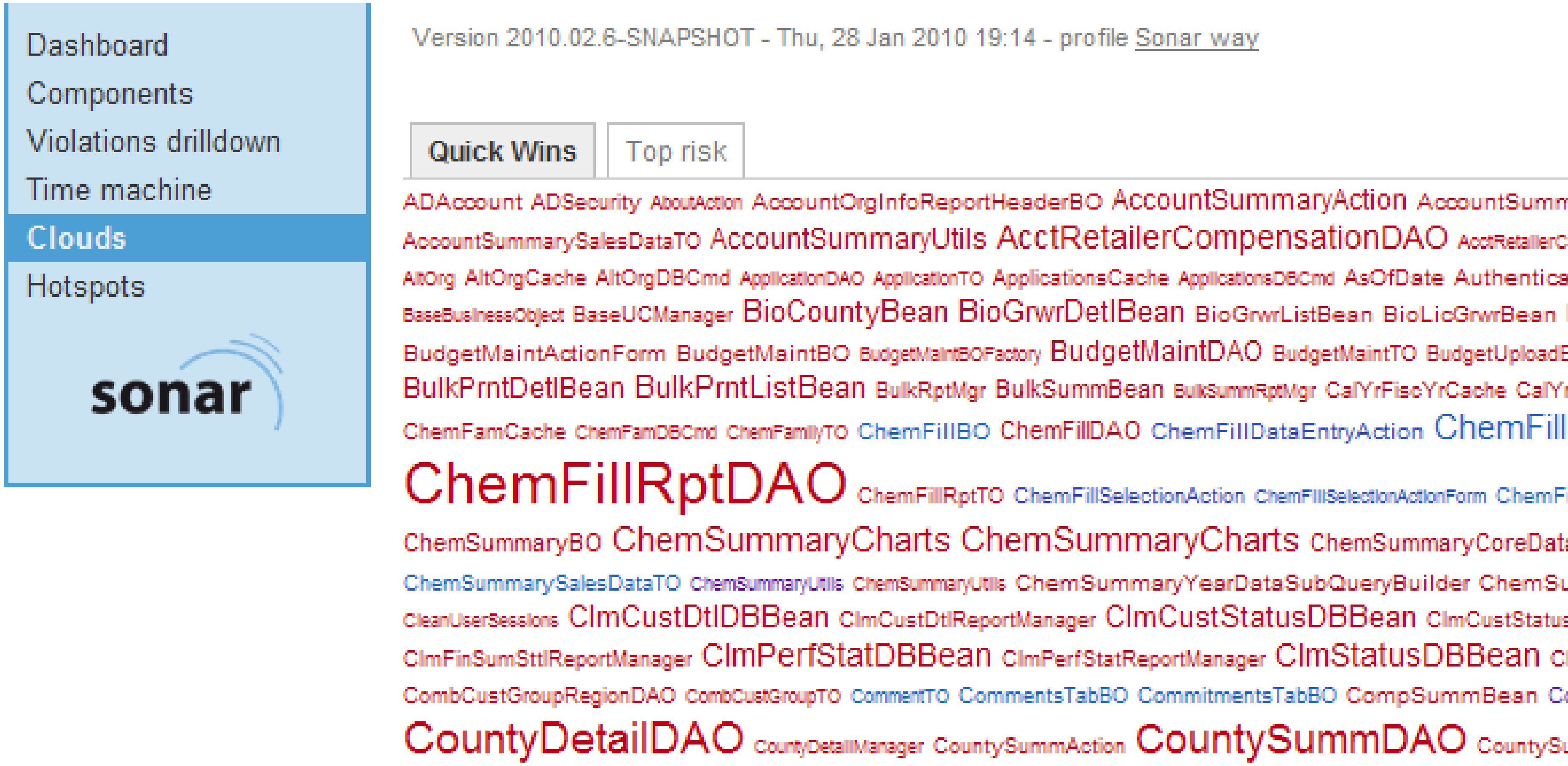
- Third level
  - Fourth level
    - Fifth level





Show date	2010-02-06 hide	2010-02-11 **Version 2010.02.6-SNAPSHOT** hide	
Complexity			
☐ Complexity /class	18.4	18.2	
☐ Complexity /method	3.2	3.2	
☑ Complexity	16,104	15,756	
☐ Uncovered complexity	15,025	14,732	
Documentation			
☐ Commented LOCs	2,978	2,977	

Sonar - Clouds



Version 2010.02.6-SNAPSHOT - Thu, 28 Jan 2010 19:14 - profile Sonar way

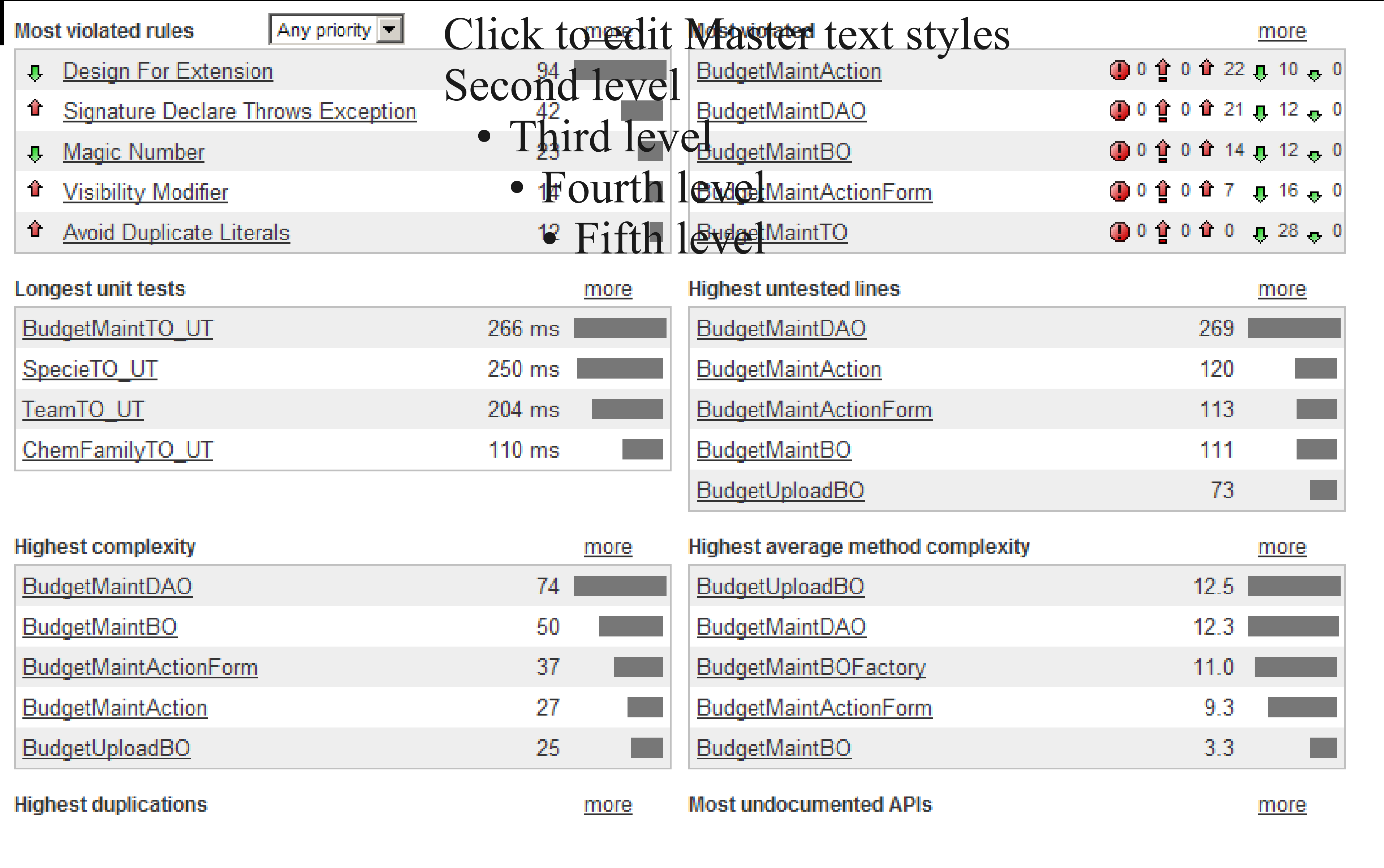
ADAccount ADSecurity AboutAction AccountOrgInfoReportHeaderBO AccountSummaryAction AccountSumm
AccountSummarySalesDataTO AccountSummaryUtils AcctRetailerCompensationDAO AcctRetailerC
AltOrg AltOrgCache AltOrgDBCmd ApplicationDAO ApplicationTO ApplicationsCache ApplicationsDBCmd AsOfDate Authentica
BaseBusinessObject BaseUCManager BioCountyBean BioGrwrDetlBean BioGrwrListBean BioLicGrwrBean
BudgetMaintActionForm BudgetMaintBO BudgetMaintBOFactory BudgetMaintDAO BudgetMaintTO BudgetUploadE
BulkPrntDetlBean BulkPrntListBean BulkRptMgr BulkSummBean BulkSummRptMgr CalYrFiscYrCache CalYr
ChemFamCache ChemFamDBCmd ChemFamilyTO ChemFillBO ChemFillDAO ChemFillDataEntryAction ChemFill
ChemFillRptDAO ChemFillRptTO ChemFillSelectionAction ChemFillSelectionActionForm ChemF
ChemSummaryBO ChemSummaryCharts ChemSummaryCharts ChemSummaryCoreData
ChemSummarySalesDataTO ChemSummaryUtils ChemSummaryUtils ChemSummaryYearDataSubQueryBuilder ChemSu
CleanUserSessions ClmCustDtlDBBean ClmCustDtlReportManager ClmCustStatusDBBean ClmCustStatus
ClmFinSumSttlReportManager ClmPerfStatDBBean ClmPerfStatReportManager ClmStatusDBBean C
CombCustGroupRegionDAO CombCustGroupTO CommentTO CommentsTabBO CommitmentsTabBO CompSummBean C
CountyDetailDAO CountyDetailManager CountySummAction CountySummDAO CountySu

Sonar Hotspots

Click to edit Master text styles

Second level

- Third level
  - Fourth level
    - Fifth level



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

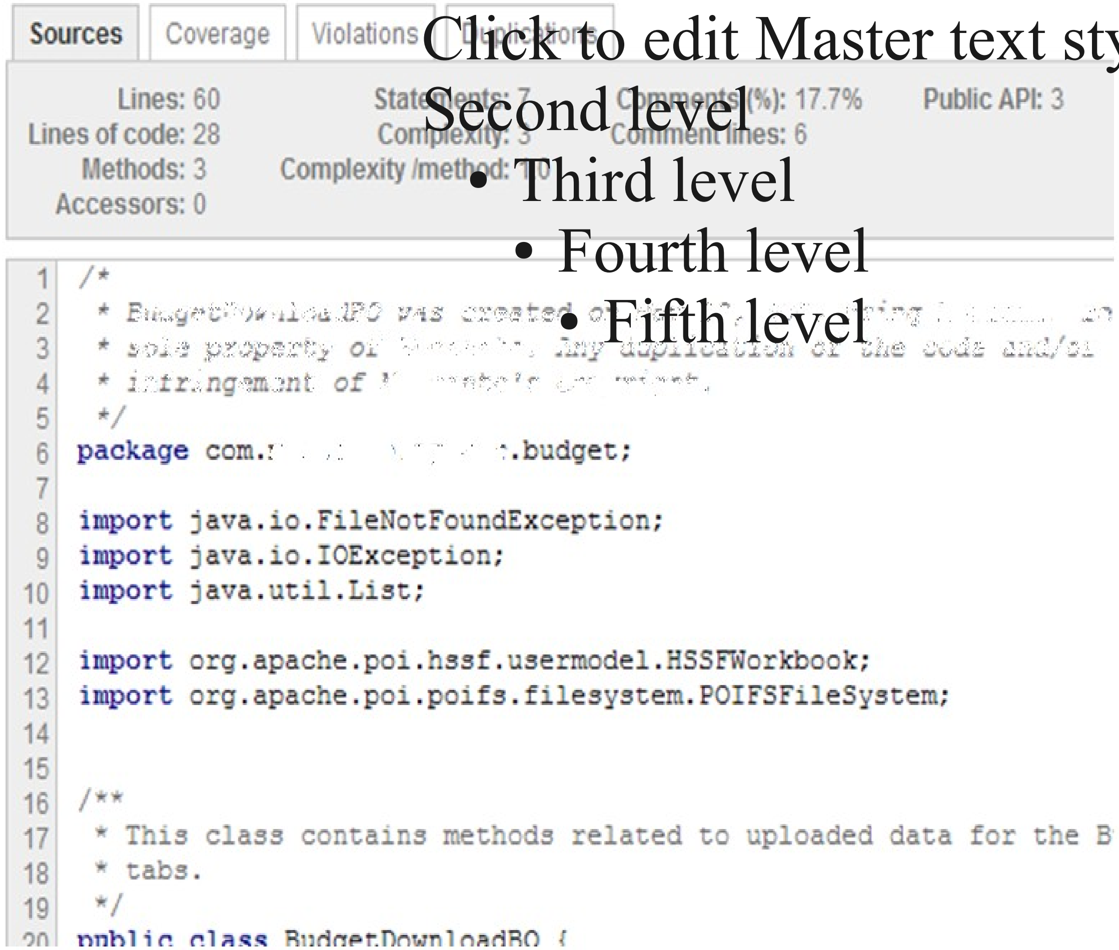
Most undocumented APIs — more

Sonar Drilldown

Click to edit Master text styles

Second level

- Third level
  - Fourth level
    - Fifth level



Sources | Coverage | Violations | Duplications

Lines: 60
Lines of code: 28
Methods: 3
Accessors: 0
Statements: 7
Complexity: 3
Complexity /method: 1.0
Comments (%): 17.7%
Comment lines: 6
Public API: 3

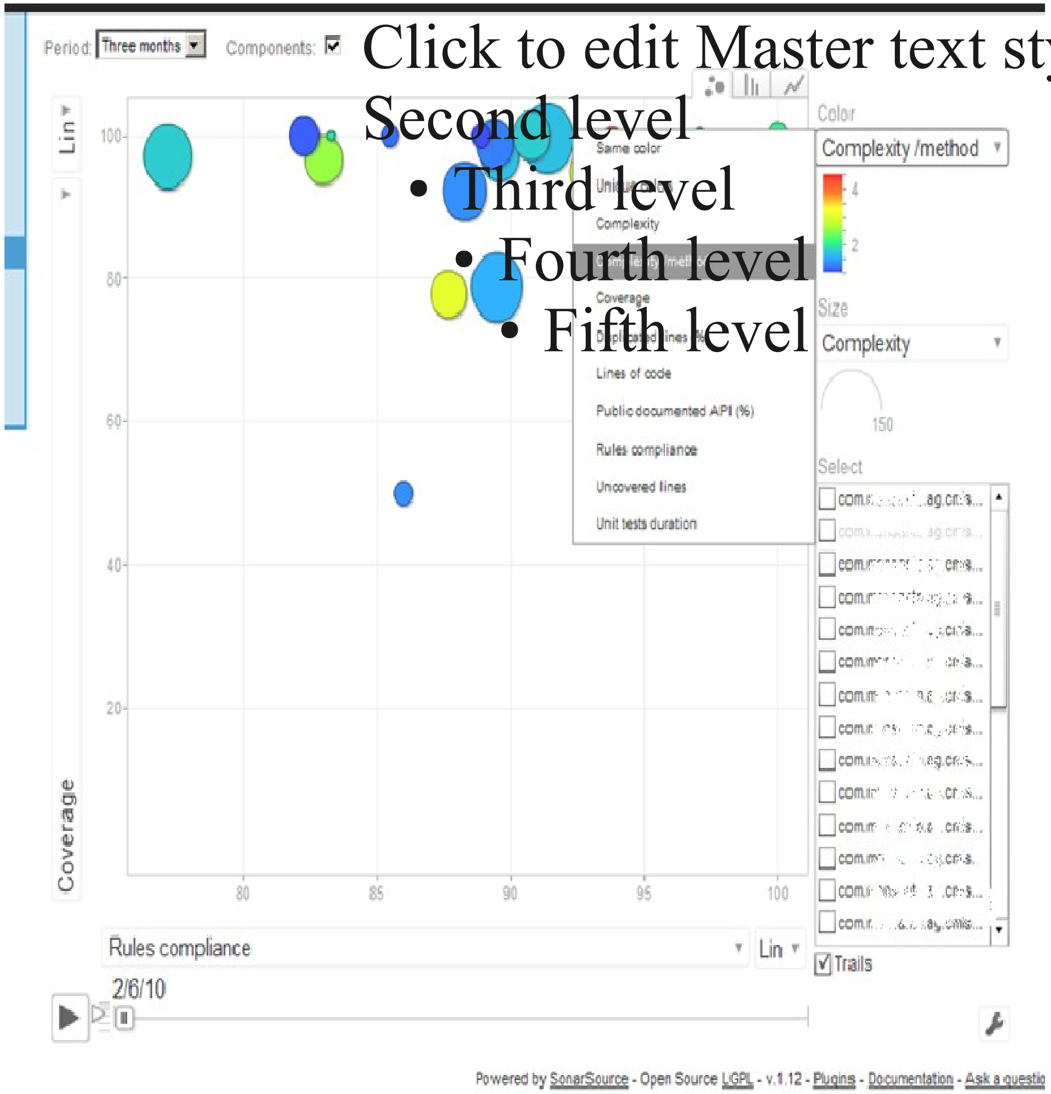
```java
/*
 * BudgetDownloadBO was created on [illegible]
 * sole property of [illegible]. Any duplication of the code and/or
 * infringement of [illegible]'s copyright.
 */
package com.[illegible].budget;

import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.List;

import org.apache.poi.hssf.usermodel.HSSFWorkbook;
import org.apache.poi.poifs.filesystem.POIFSFileSystem;


/**
 * This class contains methods related to uploaded data for the B
 * tabs.
 */
public class BudgetDownloadBO {
```

Sonar Plug-In Motion Chart

Click to edit Master text styles

Second level

- Third level
  - Fourth level
    - Fifth level



Period: Three months

Components: ✓

Color: Complexity /method

Size: Complexity

Rules compliance

Lin

2/6/10

✓ Trails

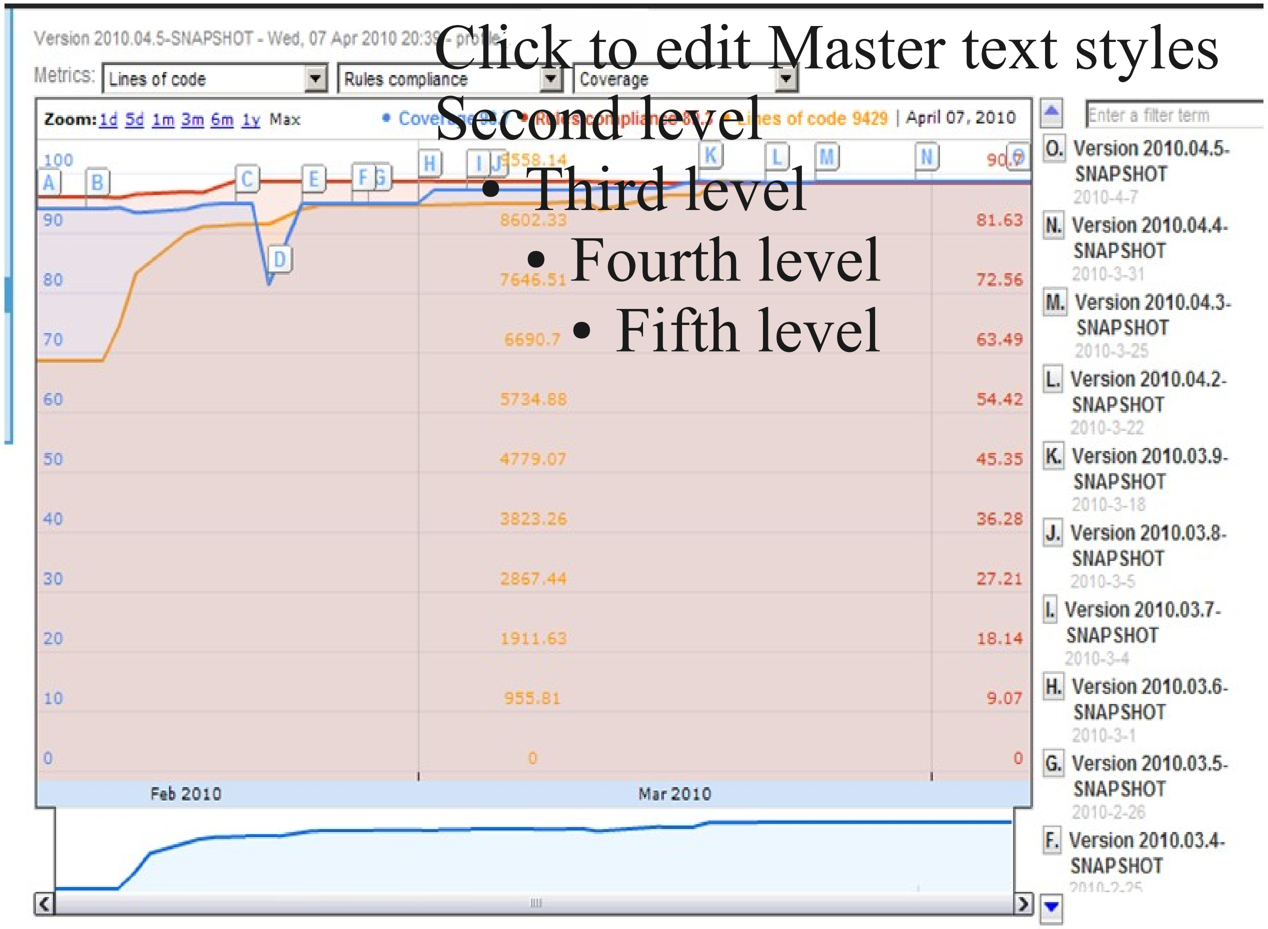
Powered by SonarSource - Open Source LGPL - v.1.12 - Plugins - Documentation - Ask a questio

Sonar Plug-In Timeline

Click to edit Master text styles

Second level

- Third level
  - Fourth level
    - Fifth level


Version 2010.04.5-SNAPSHOT - Wed, 07 Apr 2010 20:39 - profile
Metrics: Lines of code
Rules compliance
Coverage
Zoom: 1d 5d 1m 3m 6m 1y Max
• Coverage 90.7 • Rules compliance 87.2 • Lines of code 9429 | April 07, 2010
Feb 2010
Mar 2010
Enter a filter term

O. Version 2010.04.5-SNAPSHOT 2010-4-7
N. Version 2010.04.4-SNAPSHOT 2010-3-31
M. Version 2010.04.3-SNAPSHOT 2010-3-25
L. Version 2010.04.2-SNAPSHOT 2010-3-22
K. Version 2010.03.9-SNAPSHOT 2010-3-18
J. Version 2010.03.8-SNAPSHOT 2010-3-5
I. Version 2010.03.7-SNAPSHOT 2010-3-4
H. Version 2010.03.6-SNAPSHOT 2010-3-1
G. Version 2010.03.5-SNAPSHOT 2010-2-26
F. Version 2010.03.4-SNAPSHOT 2010-2-25

My Other Favorite Code Analysis Tool (IntelliJ)

Very easy to use

Comes in a free version

Easy to install

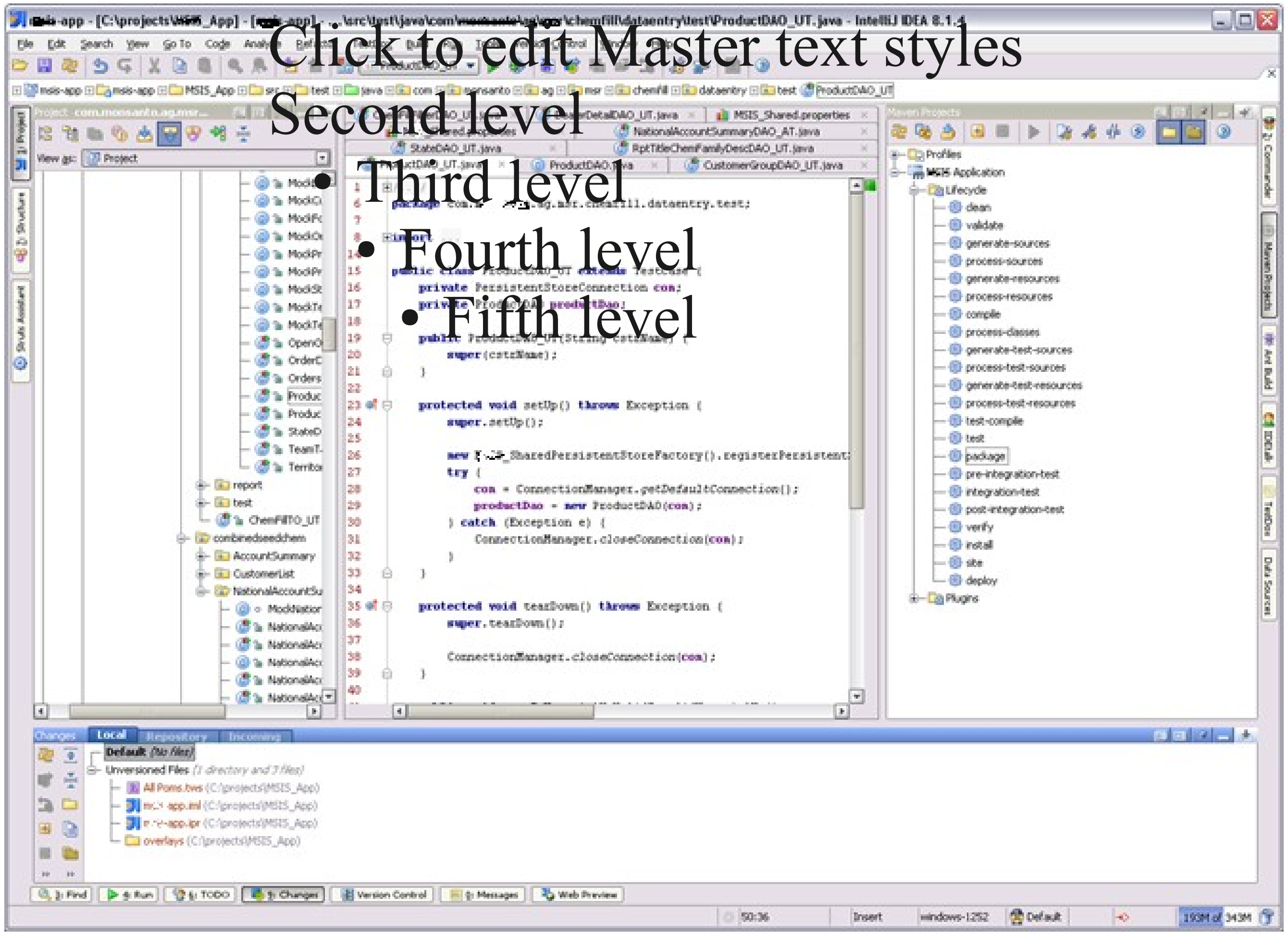
Is a Third Generation Tool

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 standards
- Conforming to specifications

IntelliJ Idea Demo



- Click to edit Master text styles
  - Second level
    - Third level
      - Fourth level
        - Fifth level

```
package com.[redacted].ag.msr.chemfill.dataentry.test;

import ...

public class ProductDAO_UT extends TestCase {
    private PersistentStoreConnection con;
    private ProductDAO productDao;

    public ProductDAO_UT(String cstrName) {
        super(cstrName);
    }

    protected void setUp() throws Exception {
        super.setUp();

        new [redacted]_SharedPersistentStoreFactory().registerPersistent
        try {
            con = ConnectionManager.getDefaultConnection();
            productDao = new ProductDAO(con);
        } catch (Exception e) {
            ConnectionManager.closeConnection(con);
        }
    }

    protected void tearDown() throws Exception {
        super.tearDown();

        ConnectionManager.closeConnection(con);
    }
```

Q&A