

Wakaleo Consulting

Optimizing your software development

<http://www.wakaleo.com>
john.smart@wakaleo.com

Java Software Quality

Tools and techniques

Introduction

- Agenda – tools to improve software quality
 - Goals of a software development team
 - Automated Testing
 - Continuous Integration
 - Code Quality Metrics
 - Technical Documentation

Introduction

- Goals of a software development team:
 - Build software to a specified set of requirements:
 - Within scope
 - Within time
 - Within budget

Yeah right.

Introduction

- Goals of a software development team:
 - Build the ***best possible application*** for the end user.
 - ***Within time and budget constraints*** defined by the project sponsor.

Introduction

- How do we achieve this?
 - Build higher quality software
 - Build more flexible software
 - Build more useful software

Introduction

- Traditional development processes
 - Room for improvement?
 - Poorly tested applications?
 - Applications that are difficult and expensive to change?
 - Lots of bugs at delivery time?
 - Inconsistent coding standards and programming habits?
 - Code that is hard to maintain and to update
 - Technical documentation out of date?

Introduction

- How can we improve?
 - Enforce good testing practices
 - Monitor developer test quality and coverage
 - Automate the build process
 - Monitor and review code quality metrics
 - Automatic technical documentation

Wakaleo Consulting

Optimizing your software development

<http://www.wakaleo.com>
john.smart@wakaleo.com

Enforce Good Testing Practices

Good Testing Practices

- Several types of developer tests
 - Unit Tests
 - Integration Tests
 - Graphical User Interface (Web) Tests
- All can be automated to varying degrees

Good Testing Practices

- Unit Tests

- A cornerstone of modern software development
- Unit tests can help you
 - Ensure that code behaves as expected
 - Make your code more flexible and easier to maintain
 - Detect regressions
 - Document your code

Introduction to Unit Testing

Traditional developer testing

- › Run against the entire application
- › Run manually in a debugger
- › Testing is ad-hoc and not reproducible
- › Testing needs human intervention

Unit testing

- › Run against classes or small components
- › Tests can be run automatically
- › Tests can be reused
- › Testing can be automated

Good Testing Practices

- Unit Tests

- The costs of writing unit tests
 - More code to write and maintain
 - More time required to write the code, *initially...*

Good Testing Practices

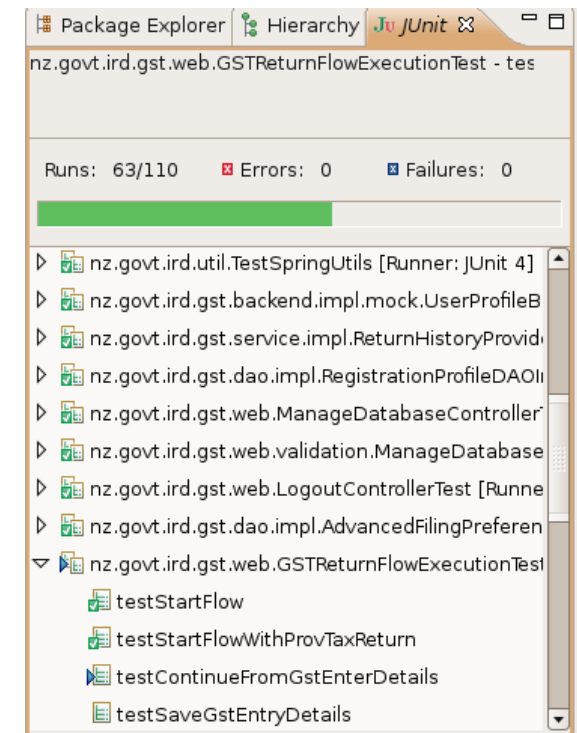
- Unit Tests
 - The benefits of writing unit tests
 - More reliable code with less bugs
 - More flexible code
 - Code that is easier to maintain
 - Automatic regression tests

Good Testing Practices

- Unit Tests

- A good unit test should:

- Test individual classes or small components
 - Test code in isolation
 - Run quickly
 - Leave the system in a predictable state
 - Be reproducible
 - Be automated

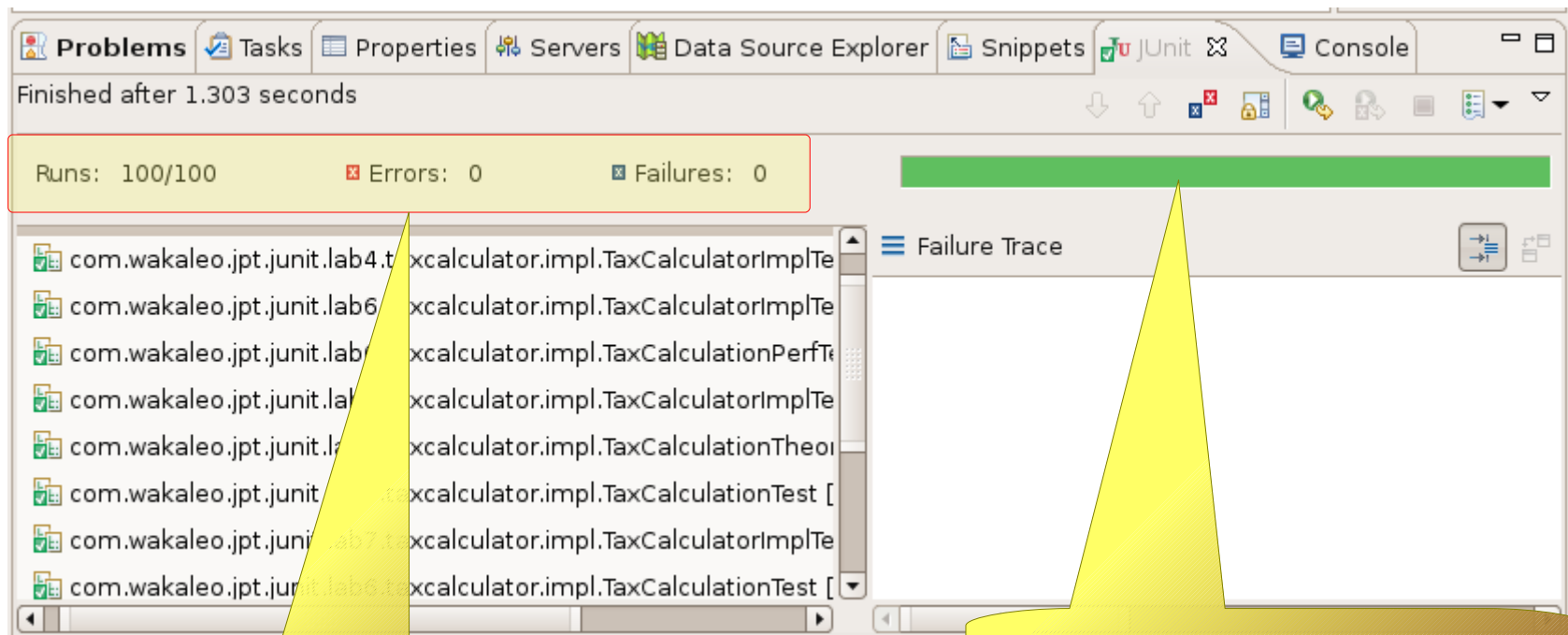


Good Testing Practices

- Unit Testing tools
 - **JUnit 3.x**
 - De facto standard in Java testing
 - Well known and well used
 - **JUnit 4**
 - A newer, more modern version of JUnit
 - **TestNG**
 - A more recent and very innovative unit testing library
 - Less widely used than JUnit

Good Testing Practices

- Running Unit Tests in Eclipse



Test result summary

The Green Bar of Success

Good Testing Practices

- Running Unit Tests in Eclipse

The screenshot shows the Eclipse IDE's 'Problems' view. At the top, it says 'Finished after 0.439 seconds'. Below that, a summary bar shows 'Runs: 100/100', 'Errors: 0', and 'Failures: 1'. A red progress bar is partially filled. The test list below shows several tests, with 'shouldUseLowestTaxRateForIncomeBelow38000' marked as failed. The 'Failure Trace' view on the right shows the error details: 'java.lang.AssertionError: Expected: is <5851.0> got: <5850.0>'.

Test result summary

There are test failures

Runs: 100/100 Errors: 0 Failures: 1

Failure Trace

```
java.lang.AssertionError:  
Expected: is <5851.0>  
got: <5850.0>  
at com.wakaleo.jpt.junit.lab7.taxcalculator.impl.TaxCalculatorImplT
```

A failed test

Test failure details

Testing web applications

- Testing web interfaces
 - Often neglected by developers and left to the testers
 - Traditionally difficult to automate
 - Involves much human judgement

Testing web applications

- Why automate web interface testing
 - Automatic smoke tests
 - Automatic regression tests

Testing web applications

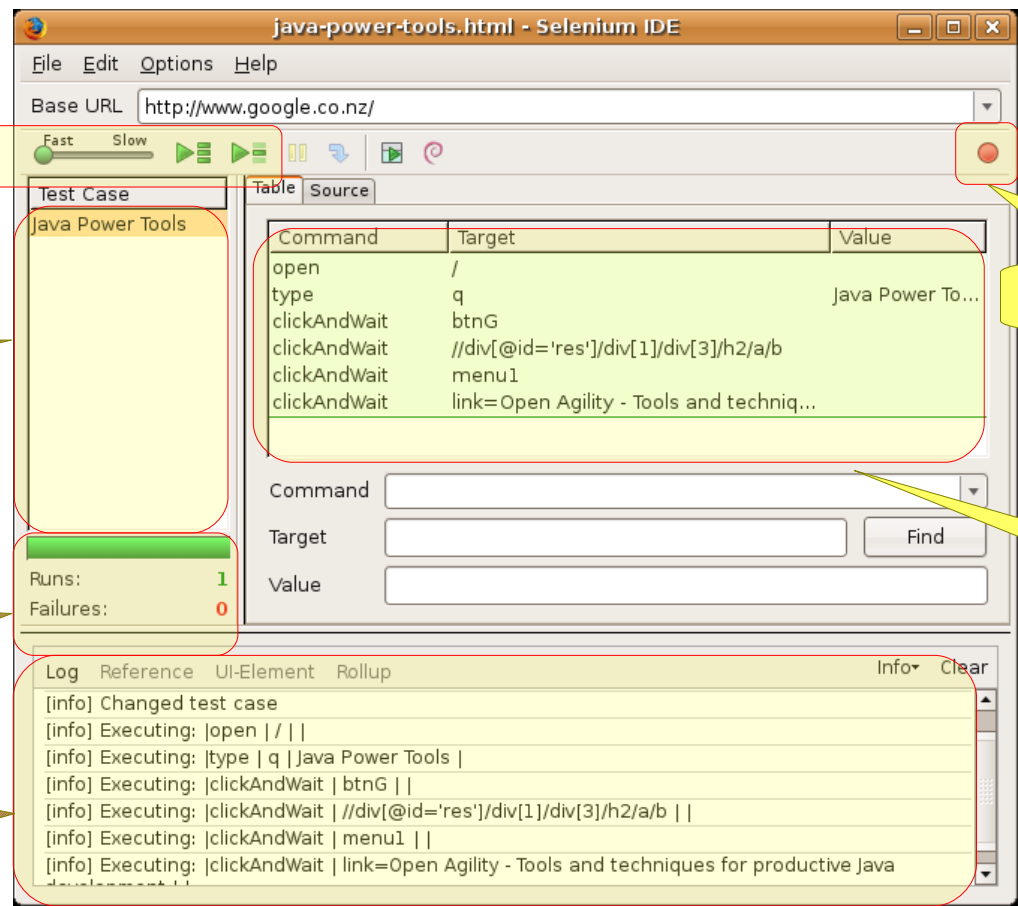
- Web Interface Testing technologies
 - Using Mock Objects
 - Spring MVC, StrutsTestCase, JSFUnit
 - Good for the controller layer
 - Doesn't test the HTML screens themselves
 - Writing tests to run against a web application
 - HTTPUnit, Cactus, Jmeter, TestMaker,...
 - Run tests from within a real web browser
 - Selenium

Testing web applications

- Testing web applications with Selenium
 - Selenium is a browser-based testing tool
 - You can
 - Record test scripts in a browser
 - Replay them manually or automatically

Testing web applications

- Testing web applications with Selenium



Replay test scripts

Manage multiple test scripts

Display latest test results

History of executed instructions

Record test scripts

Display and edit test script

Testing web applications

- Testing web applications with Selenium

The screenshot displays the Selenium Functional Test Runner v1.0-beta-1 interface within a Mozilla Firefox browser window. The browser's address bar shows the URL: `chrome://selenium-ide/content/selenium/TestRunner.html?test=/content/F`. The Selenium TestRunner panel is visible on the right side of the browser window, showing the test suite 'amazon-java-power-tools' and the test results. The test results indicate that 2 tests passed and 0 failed.

Test Suite	Test Name	Result
amazon-java-power-tools	open /	Passed
amazon-java-power-tools	type twotabsearchtextbox Java Power Tools	Passed
amazon-java-power-tools	clickAndWait navGoButtonPanel	Passed
amazon-java-power-tools	clickAndWait //img[@alt='Java Power Tools']	Passed
amazon-java-power-tools	verifyTextPresent Java Power Tools	Passed
amazon-java-power-tools	clickAndWait submit.add-to-repeatshoplist	Passed
amazon-java-power-tools	verifyTextPresent Java Power Tools	Passed
amazon-java-power-tools	clickAndWait RSL_1143067472908_delete	Passed
amazon-java-power-tools	verifyTextNotPresent Java Power Tools	Passed

The Amazon.com page is visible in the background, showing the user's shopping list. The shopping list contains one item: Java Power Tools, priced at \$59.99 (originally \$37.79). The user's name is John Smart.

Test Coverage

- Test Coverage
 - Test Coverage indicates how much application code is executed by your unit tests
 - It is especially useful for identifying code that has ***not*** been tested.

Test Coverage

- Test Coverage Tools
 - Automated Test Coverage
 - Integrated into the build process
 - Runs for every build
 - Team-wide reporting
 - Tools like Cobertura

Test Coverage

- Test Coverage - Cobertura

The screenshot displays the Cobertura Coverage Report for 'All Packages'. On the left, there are sections for 'Packages' and 'Classes'. The 'Classes' section lists various classes with their respective coverage percentages, such as 'AircraftTypePage (54%)' and 'DefaultDatabaseImpl (84%)'. The main part of the report is a table with the following columns: Package, # Classes, Line Coverage, Branch Coverage, and Complexity. The table lists several packages, with the last one, 'com.wakaleo.ipt.modelplanes.web.pages', circled in red. A yellow callout box points to this row with the text 'A package with low test coverage'.

Package	# Classes	Line Coverage	Branch Coverage	Complexity
All Packages	19	68% 171/252	73% 32/44	1316
com.wakaleo.ipt.modelplanes.core.dao	9	85% 75/88	75% 15/20	1385
com.wakaleo.ipt.modelplanes.core.domain	4	51% 38/74	N/A N/A	0
com.wakaleo.ipt.modelplanes.util	2	91% 39/43	88% 14/16	1
com.wakaleo.ipt.modelplanes.web.pages	4	40% 19/47	38% 3/8	125

Report generated by Cobertura 1.9 on 26/03/08 11:32.

A package with low test coverage

Test Coverage

- Test Coverage - Cobertura

Untested classes

Poorly tested classes

Package	# Classes	Line Coverage	Branch Coverage	Complexity
com.wakaleo.jpt.modelplanes.web.pages	4	40% 19/47	38% 3/8	1.25

Classes in this package	Line Coverage	Branch Coverage	Complexity
AircraftTypePage	54% 14/26	50% 2/4	0
CurrentUserBean	0% 0/1	N/A N/A	1
ModelPlanesPage	0% 0/1	0% 0/2	0
SpringSupport	83% 5/6	50% 1/2	15

Report generated by Cobertura 1.9 on 27/03/08 07:58.

Test Coverage

- Test Coverage - Cobertura

This code was never executed

Packages

- [All](#)
- [com.wakaleo.jpt.modelplanes.core.dao](#)
- [com.wakaleo.jpt.modelplanes.core.domain](#)
- [com.wakaleo.jpt.modelplanes.util](#)
- [com.wakaleo.jpt.modelplanes.web.pages](#)

com.wakaleo.jpt.modelplanes.web.pages

Classes

- [AircraftTypePage](#) (54%)
- [CurrentUserBean](#) (0%)
- [ModelPlanesPage](#) (0%)
- [SpringSupport](#) (83%)

```
69  */
70  public void setPlaneTypeDAO(PlaneTypeDAO planeTypeDAO) {
71  0      this.planeTypeDAO = planeTypeDAO;
72  0  }
73
74  /**
75   * Returns the list of aircraft types, which it obtains from the DAO
76   * @return
77   */
78  public List<PlaneType> getAircraftTypes() {
79  2      LOGGER.info("getAircraftTypes");
80  2      if (aircraftTypes == null) {
81  2          setAircraftTypes(getPlaneTypeDAO().findAll());
82      }
83  2      LOGGER.info("getAircraftTypes");
84  2      return aircraftTypes;
85  }
86
87  /**
88   * Assign the list of current aircraft types
89   * @param aircraftTypes
90   */
91  public void setAircraftTypes(List<PlaneType> aircraftTypes) {
92  2      LOGGER.info("setAircraftTypes");
93  2      this.aircraftTypes = aircraftTypes;
94  2  }
95
96  public String displayModels() {
97  0      LOGGER.info("displayModels()");
98  0      FacesContext facesContext = FacesContext.getCurrentInstance();
99  0      UIViewRoot root = facesContext.getViewRoot();
100  0      UIData table
101          = (UIData) root.findComponent("aircraftTypeForm")
102              .findComponent("table");
103
104  0      PlaneType planeType = (PlaneType) table.getRowData();
105
106  0      ValueBinding binding = Util.getValueBinding("#{modelPlanesPage}");
107  0      ModelPlanesPage modelPlanesPage
108          = (ModelPlanesPage) binding.getValue(facesContext);
109  0      modelPlanesPage.setPlaneType(planeType);

```

This code was executed twice

Test Coverage

- Test Coverage Tools
 - Test coverage in your IDE
 - Faster feedback for developers
 - ECLEmma - Test Coverage in Eclipse
 - Crap4j – also provides test coverage metrics

Test Coverage

- EclEmma - Test Coverage in Eclipse

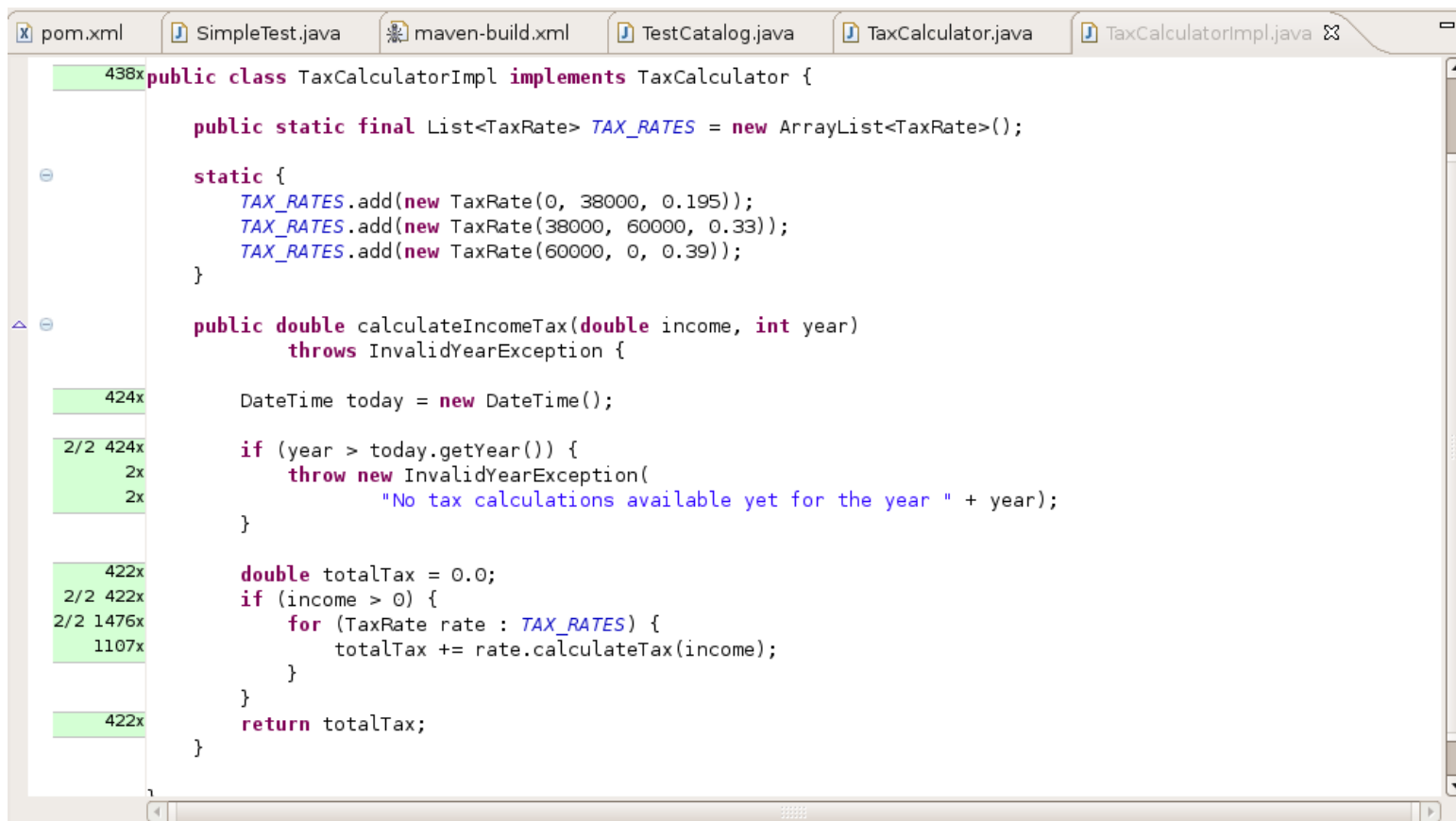
The screenshot displays the Eclipse IDE interface with the EclEmma test coverage plugin. The main editor shows the source code of `TaxRate.java` with colored highlights indicating coverage: red for uncovered code, green for fully covered code, and blue for partially covered code. The `getRate()` method is fully covered (green), while `getMaximumRevenue()` is not covered (red). The `getApplicableAmount()` method is partially covered (green and blue).

The bottom panel shows the 'Covera' view, which provides a summary of test coverage for the project and its components. The table below represents the data shown in this view:

Element	Coverage	Covered Blocks
src/main/java - tax-calculator	73.9 %	17
com.wakaleo.jpt.taxcalculator	25.0 %	1
com.wakaleo.jpt.taxcalculator.impl	84.2 %	16
TaxCalculatorImpl.java	100.0 %	9
TaxRate.java	70.0 %	7
TaxRate	70.0 %	7
TaxRate(double, double, double)	100.0 %	1
calculateTax(double)	100.0 %	1
getMaximumRevenue()	0.0 %	0

Good Testing practices

- Crap4j - Test Coverage in Eclipse



The screenshot shows the Eclipse IDE with the following tabs: pom.xml, SimpleTest.java, maven-build.xml, TestCatalog.java, TaxCalculator.java, and TaxCalculatorImpl.java. The main editor displays the code for TaxCalculatorImpl.java with test coverage annotations on the left margin:

```
438x public class TaxCalculatorImpl implements TaxCalculator {  
  
    public static final List<TaxRate> TAX_RATES = new ArrayList<TaxRate>();  
  
    static {  
        TAX_RATES.add(new TaxRate(0, 38000, 0.195));  
        TAX_RATES.add(new TaxRate(38000, 60000, 0.33));  
        TAX_RATES.add(new TaxRate(60000, 0, 0.39));  
    }  
  
    public double calculateIncomeTax(double income, int year)  
        throws InvalidYearException {  
  
        DateTime today = new DateTime();  
  
        if (year > today.getYear()) {  
            throw new InvalidYearException(  
                "No tax calculations available yet for the year " + year);  
        }  
  
        double totalTax = 0.0;  
        if (income > 0) {  
            for (TaxRate rate : TAX_RATES) {  
                totalTax += rate.calculateTax(income);  
            }  
        }  
        return totalTax;  
    }  
}
```

The test coverage annotations on the left margin are: 438x, 424x, 2/2 424x, 2x, 2x, 422x, 2/2 422x, 2/2 1476x, 1107x, and 422x.

Test Coverage

- Why use EclEmma/Crap4j and Cobertura
 - EclEmma and Crap4j allows IDE integration
 - Fast feedback for the developer
 - More convenient for the developer than using an HTML report
 - Cobertura allows project-level coverage reporting
 - Project-wide coverage statistics
 - Results can be published and reviewed
 - Can be used to enforce minimum coverage levels

Wakaleo Consulting

Optimizing your software development

<http://www.wakaleo.com>
john.smart@wakaleo.com

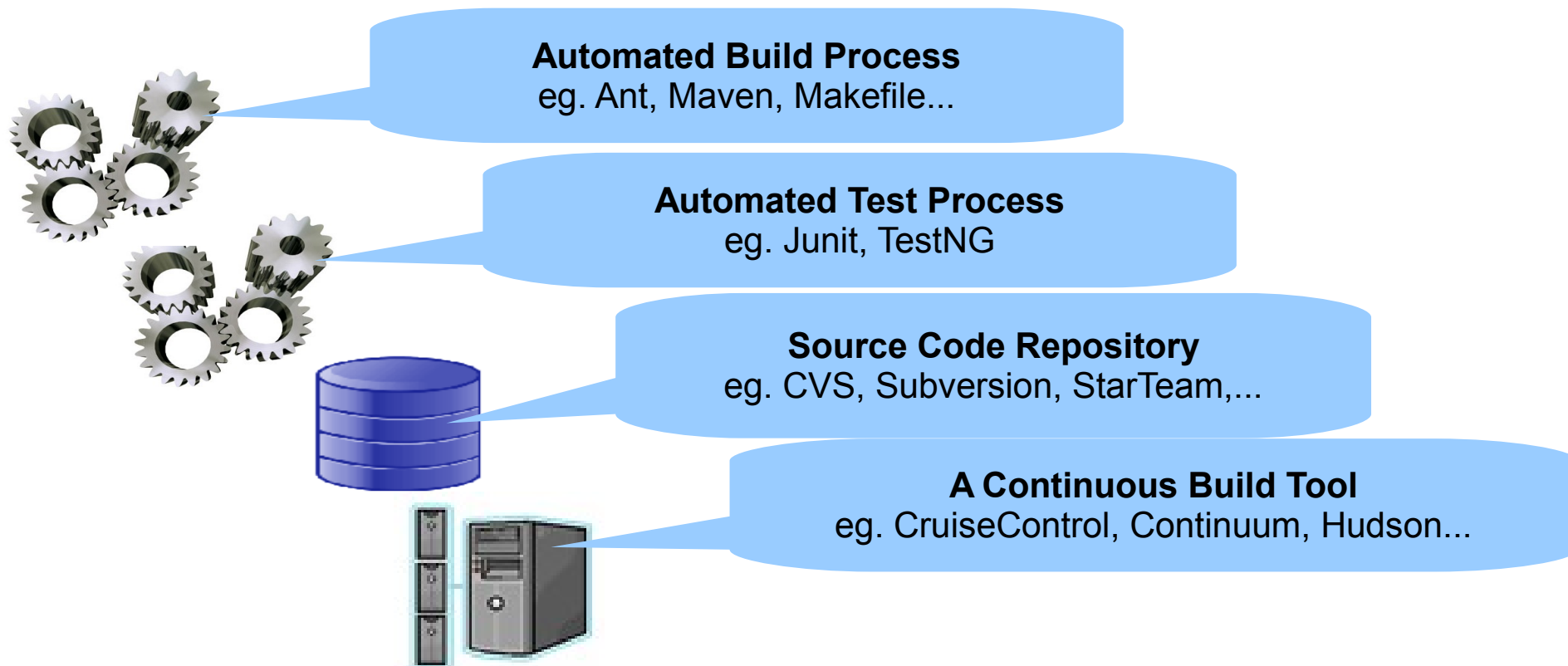
Continuous Integration

Continuous Integration

- What is Continuous Integration?
 - Automatically “integrating” and compiling source code from different developers on a central build server
 - A core best practice of modern software development

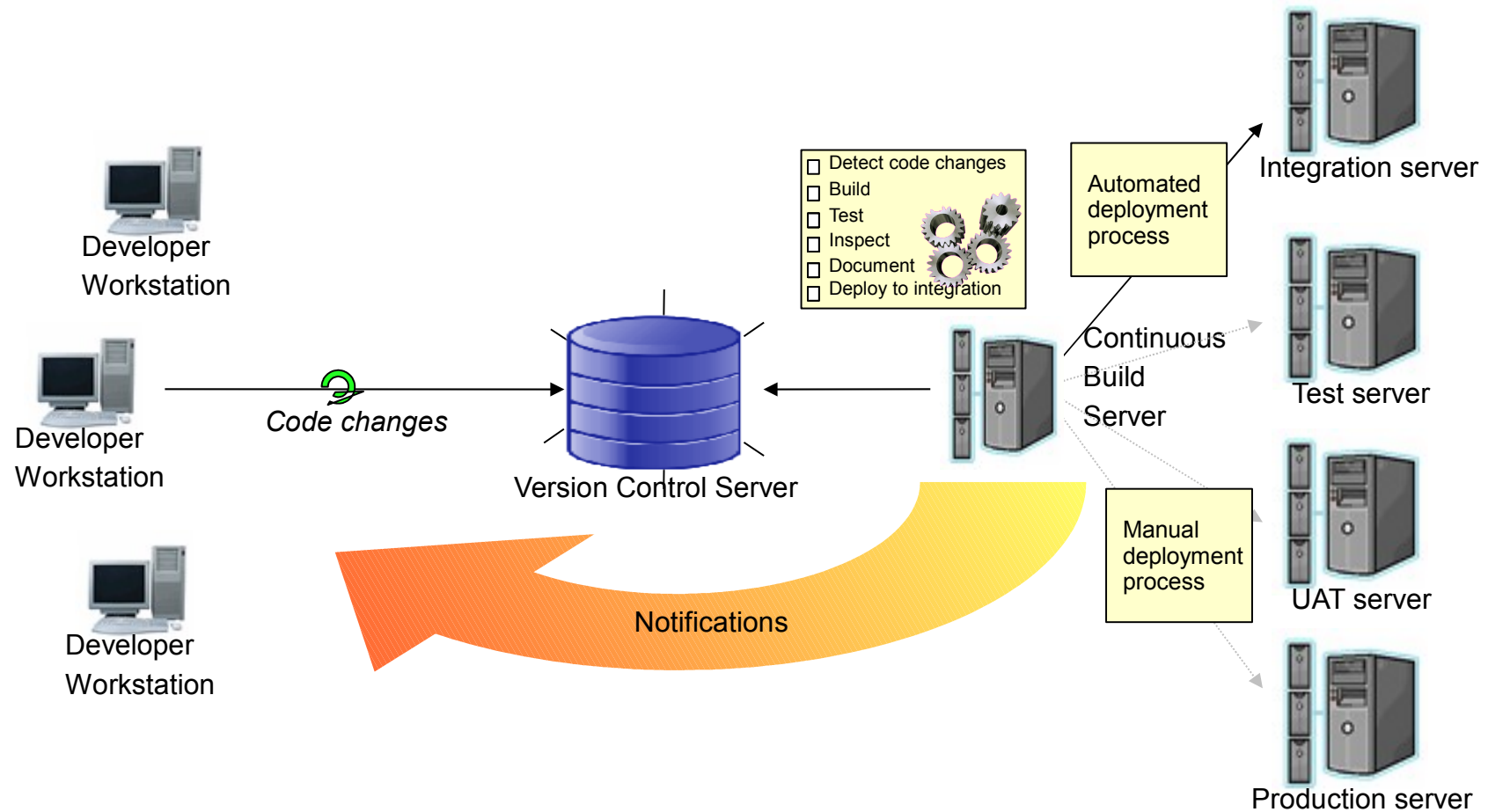
Continuous Integration

- What do you need?
 - The principal components of a Continuous Integration service:



Continuous Integration

- How does it work?

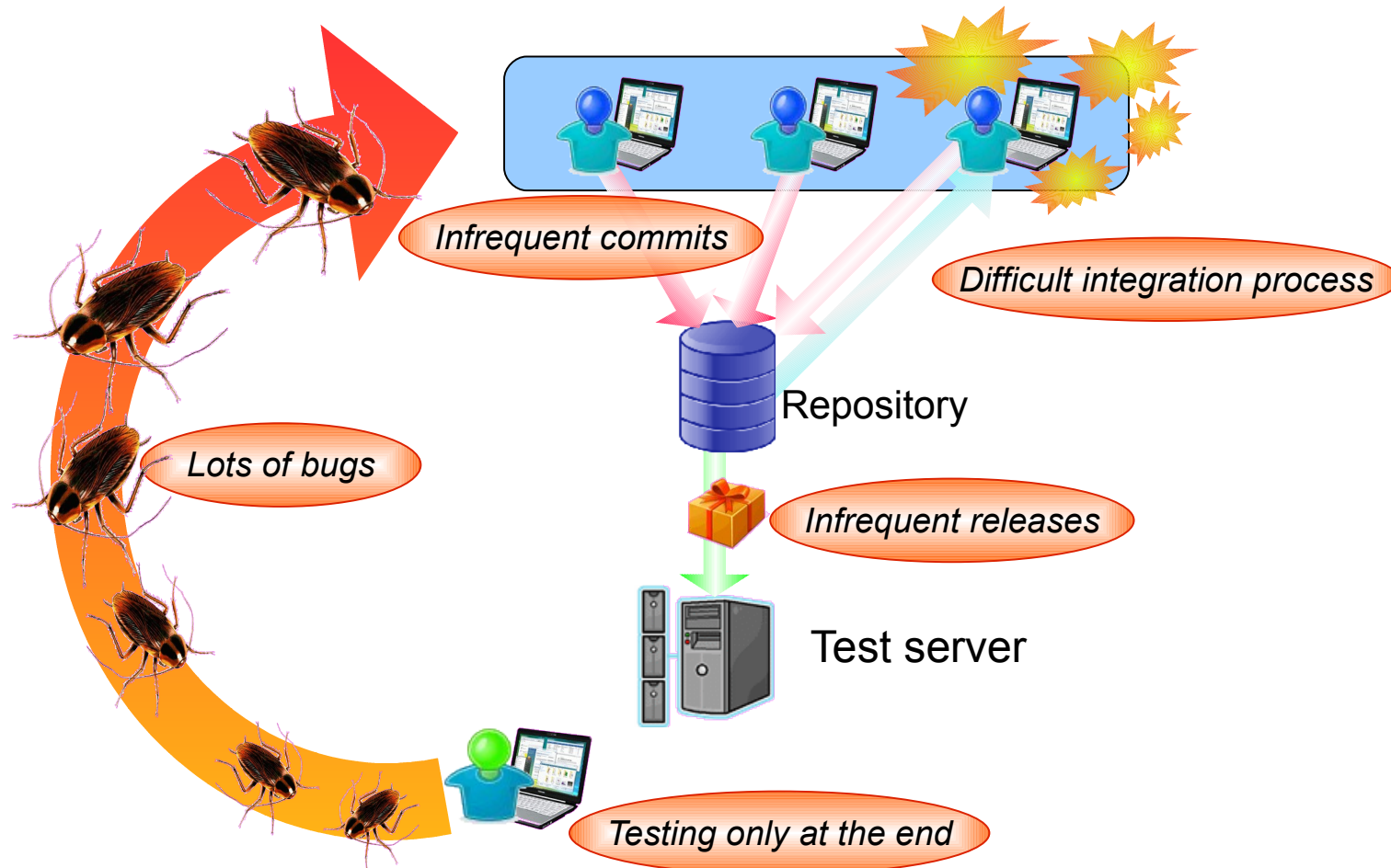


Continuous Integration

- What problem does it solve?
 - The “traditional” software process involves:
 - Coding
 - Ad-hoc testing
 - Commit changes shortly before the start of the testing phase
 - Difficult integration process

Continuous Integration

- What problem does it solve?

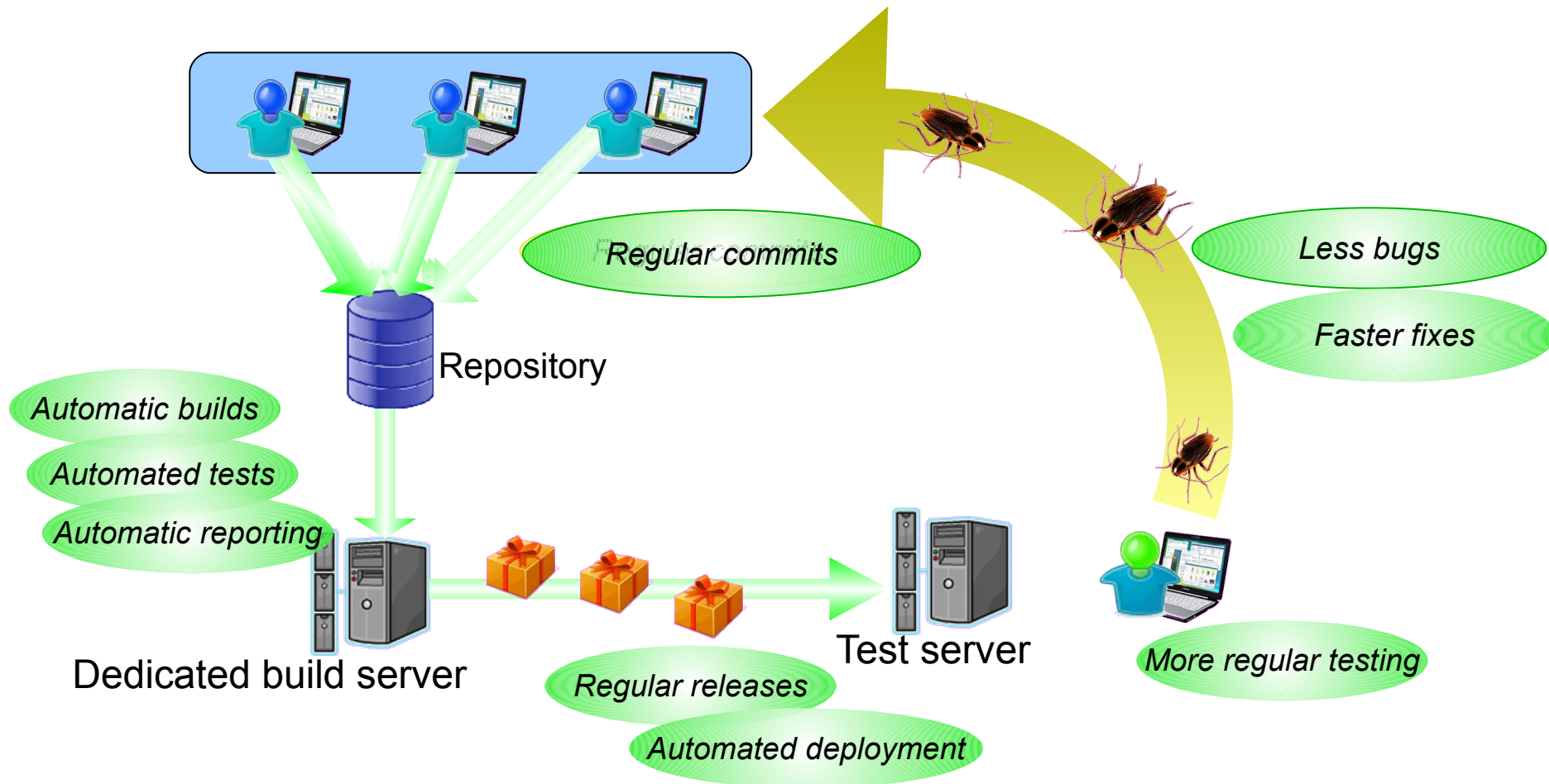


Continuous Integration

- What problem does it solve?
 - This is a flawed process
 - Testing may not be done efficiently
 - Integration is long and difficult
 - Poor visibility on development progress
 - Functional tests are done too late
 - Raised issues are harder to fix
 - The client gets a sub-optimal product

Continuous Integration

- How does it solve this problem?



Continuous Integration

- What problem does it solve?
 - Continuous Integration – an industry best practice
 - Smoother integration
 - Automatic regression testing
 - Regular working releases
 - Earlier functional testing
 - Faster and easier bug fixes
 - Better visibility

Continuous Integration

- What tools are available
 - Open Source tools
 - CruiseControl, Hudson, Continuum, LintBuild...
 - Commercial Tools
 - TeamCity, Bamboo, Pulse,...

Continuous Integration

- A Continuous Build Server - Hudson
 - An overview of all your build jobs



The screenshot shows the Hudson web interface. At the top, there's a blue header with the 'Hudson' logo, a search bar, and links for 'admin' and 'logout'. Below the header, there are navigation links for 'New Job', 'Manage Hudson', and 'People'. On the left, there are two summary boxes: 'Build Queue' (No builds in the queue) and 'Build Executor Status' (2 executors, both idle). The main content area displays a 'Welcome to the O2C2 Continuous Integration Server!' message and a table of build jobs. The table has columns for status, job name, last success, last failure, and last duration. Below the table, there are summary statistics for build stability, checkstyle violations, and test results.

S	W	Job ↓	Last Success	Last Failure	Last Duration
		debt-calculator	7 minutes 25 seconds (#124)	16 minutes (#123)	2 minutes 23 seconds
		oia-debt-calculator-core	23 minutes (#36)	28 minutes (#35)	26 seconds
		oia-debt-calculator-core-site	7 minutes 20 seconds (#31)	16 minutes (#30)	2 minutes 49 seconds
		oia-debt-calculator-portlet	1 hour 9 minutes (#81)	1 hour 13 minutes (#80)	1 minute 32 seconds
		oia-debt-calculator-portlet-integration-tests	1 hour 6 minutes (#37)	N/A	1 minute 25 seconds
		oia-debt-calculator-portlet-site	19 minutes (#80)	38 minutes (#79)	1 minute 49 seconds

W	Description	%
	Build stability: 4 out of the last 5 builds failed.	19
	Number of checkstyle violations is 7	87
	Test Result: 0 tests failing out of a total of 109 tests.	100

Legend for all for failures

Hudson ver. 1.180

Continuous Integration

- The Hudson Dashboard
 - An overview of all your build jobs

Hudson search admin | logout

Hudson

[New Job](#)
[Manage Hudson](#)
[People](#)

Build Queue
No builds in the queue.

Build Executor Status

No.	Status
1	Idle
2	Idle

Welcome to the O2C2 Continuous In

All Debt Calculator Core Debt Calculator Portlet +

S	W	Job ↓				
		debt-calculator				
		oia-debt-calculator-core				
		oia-debt-calculator-core-site				
		oia-debt-calculator-portlet				
		oia-debt-calculator-portlet-integration-tests	1 hour 6 minutes (#37)	N/A	1 minute 25 seconds	
		oia-debt-calculator-portlet-site	19 minutes (#80)	38 minutes (#79)	1 minute 49 seconds	

Build Status Legend:

- Build successful
- Build unstable
- Build failed
- Build in progress

Quick status for each project

W	Description	%
	Build stability: 4 out of the last 5 builds failed.	19
	Number of checkstyle violations is 7	87
	Test Result: 0 tests failing out of a total of 109 tests.	100

Legend for all for failures

Hudson ver. 1.180

Continuous Integration

- The Hudson Dashboard
 - An overview of all your build jobs

The screenshot shows the Hudson dashboard for a project named 'O2C2 Continuous Integr...'. The top navigation bar includes 'Hudson', a search box, and user links for 'admin' and 'logout'. On the left, there are links for 'New Job', 'Manage Hudson', and 'People', along with sections for 'Build Queue' (empty) and 'Build Executor Status' (two idle executors).

The main area displays a list of build jobs with status icons. A red circle highlights the status icons for the first few jobs. A callout box with a green-to-red gradient arrow points to these icons, with 'Build stable' at the top (sun icon) and 'Build unstable' at the bottom (cloud with rain icon).

No.	Status
1	Idle
2	Idle

Job	Status	Last Duration
debt-calculator	Build unstable	minutes 23 seconds
ia-debt-calculator-core	Build stable	seconds
ia-debt-calculator-core-site	Build unstable	minutes 49 seconds
ia-debt-calculator-portlet	Build stable	minute 32 seconds
ia-debt-calculator-portlet-integration-tests	Build stable	minute 25 seconds
ia-debt-calculator-portlet-site	Build unstable	1 minute 49 seconds

W	Description	%
☁	Build stability: 4 out of the last 5 builds failed.	19
☀	Number of checkstyle violations is 7	87
☀	Test Result: 0 tests failing out of a total of 109 tests.	100

Legend | RSS for all | RSS for failures

Hudson ver. 1.180

Quick status for each project

Continuous Integration

- The Hudson Dashboard
 - An overview of all your build jobs

The screenshot shows the Hudson CI dashboard. On the left, there are navigation links for 'New Job', 'Manage Hudson', and 'People'. Below these are sections for 'Build Queue' (No builds in the queue) and 'Build Executor Status' (2 executors, both idle). The main area displays a 'Welcome to the O2C2 Continuous Integration Server!' message and a table of build jobs. The table has columns for 'Job', 'Last Success', 'Last Failure', and 'Last Duration'. A red circle highlights the detailed view of a failed build for the job 'oja-debt-calculator-portlet-site'. This view shows a 'Build stability' of 19% (4 out of 5 builds failed), 87 checkstyle violations, and a test result of 0 tests failing out of 109. A yellow callout box with the text 'Display detailed build results' points to this section.

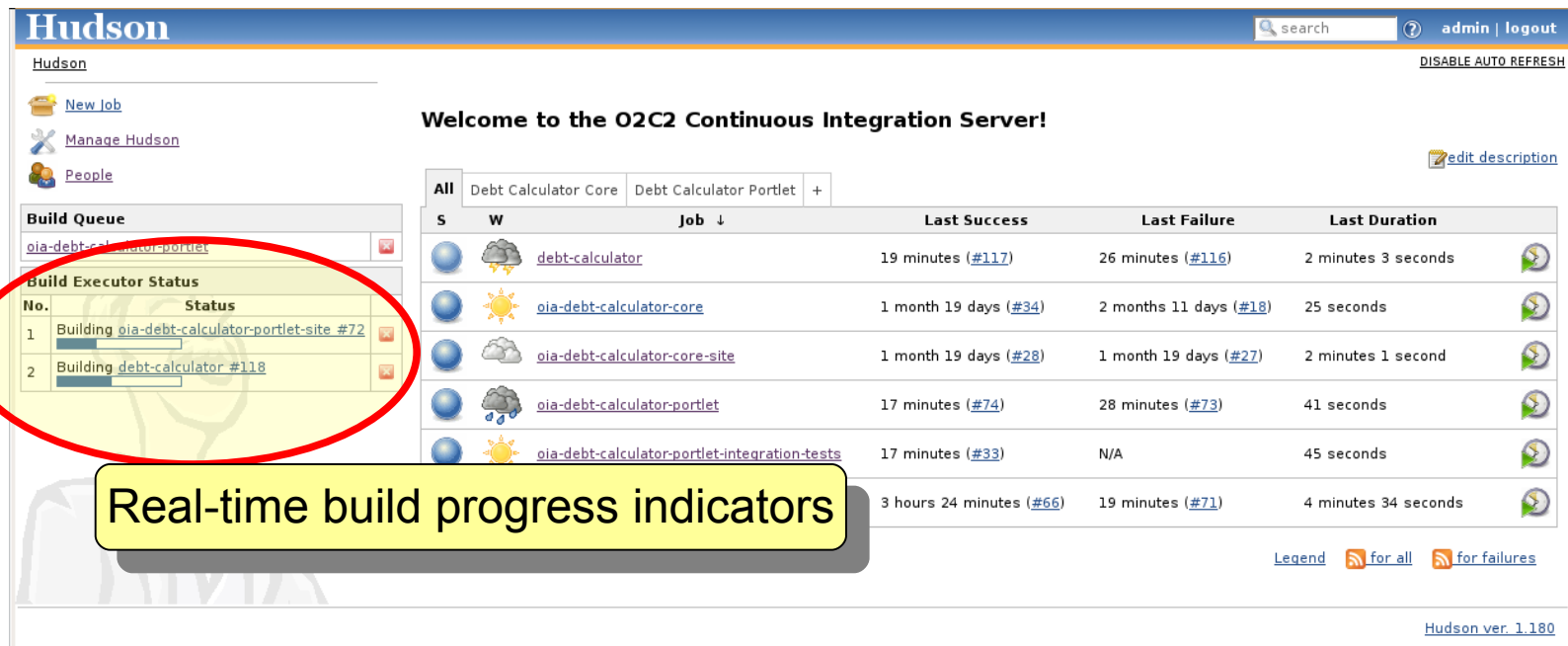
S	W	Job ↓	Last Success	Last Failure	Last Duration
●	☁	debt-calculator	7 minutes 25 seconds (#124)	16 minutes (#123)	2 minutes 23 seconds
●	☀	oia-debt-calculator-core	23 minutes (#36)	28 minutes (#35)	26 seconds
●	☁	oia-debt-calculator-core-site	7 minutes 20 seconds (#31)	16 minutes (#30)	2 minutes 49 seconds
●	☁	oia-debt-calculator-portlet	1 hour 9 minutes (#81)	1 hour 13 minutes (#80)	1 minute 32 seconds
●	☀	oia-debt-calculator-portlet-integration-tests	1 hour 6 minutes (#37)	N/A	1 minute 25 seconds
●	☁	oja-debt-calculator-portlet-site	19 minutes (#80)	38 minutes (#79)	1 minute 49 seconds

W	Description	%
☁	Build stability: 4 out of the last 5 builds failed.	19
☀	Number of checkstyle violations is 7	87
☀	Test Result: 0 tests failing out of a total of 109 tests.	100

Display detailed build results

Continuous Integration

- The Hudson Dashboard
 - An overview of all your build jobs



The screenshot shows the Hudson CI dashboard for the O2C2 Continuous Integration Server. The main area displays a table of build jobs with columns for status, job name, last success, last failure, and last duration. A red circle highlights the 'Build Executor Status' section, which shows two active builds: 'Building oia-debt-calculator-portlet-site #72' and 'Building debt-calculator #118'. A yellow callout box with the text 'Real-time build progress indicators' points to this section.

All	Debt Calculator Core	Debt Calculator Portlet	+		
S	W	Job ↓	Last Success	Last Failure	Last Duration
🌧️	☁️	debt-calculator	19 minutes (#117)	26 minutes (#116)	2 minutes 3 seconds
🌞	☀️	oia-debt-calculator-core	1 month 19 days (#34)	2 months 11 days (#18)	25 seconds
🌧️	☁️	oia-debt-calculator-core-site	1 month 19 days (#28)	1 month 19 days (#27)	2 minutes 1 second
🌧️	☁️	oia-debt-calculator-portlet	17 minutes (#74)	28 minutes (#73)	41 seconds
🌞	☀️	oia-debt-calculator-portlet-integration-tests	17 minutes (#33)	N/A	45 seconds
🌧️	☁️		3 hours 24 minutes (#66)	19 minutes (#71)	4 minutes 34 seconds

Real-time build progress indicators

Continuous Integration

- Displaying Build Results
 - View the details of any particular build job

The screenshot displays the Hudson web interface for the project 'oia-debt-calculator-portlet'. The interface includes a navigation menu on the left with links for 'Back to Dashboard', 'Status', 'Changes', 'Workspace', 'Build Now', 'Delete Project', 'Configure', and 'Subversion Polling Log'. The main content area shows the project name, workspace information, and a 'Test Result Trend' graph. The graph plots the number of test failures (count) against build numbers (#1 to #79). The count starts at 0, remains low until build #30, then increases significantly, reaching a peak of approximately 100 failures by build #79. A red circle highlights the graph area, and a yellow box with the text 'Test Results' is overlaid on the bottom right of the graph. Below the graph, there are sections for 'Downstream Projects' and 'Permalinks'.

Build History (trend)

Build #	Timestamp
#81	Feb 8, 2008 3:55:49 AM
#80	Feb 8, 2008 3:51:49 AM
#79	Feb 8, 2008 3:48:01 AM
#78	Feb 8, 2008 3:46:43 AM
#77	Feb 8, 2008 3:34:11 AM
#76	Feb 8, 2008 3:10:11 AM
#75	Feb 8, 2008 3:02:55 AM
#74	Feb 8, 2008 2:44:54 AM
#73	Feb 8, 2008 2:34:42 AM
#72	Feb 8, 2008 2:20:47 AM

Test Result Trend

Count vs. Build #

Test Results

Continuous Integration

- Displaying Build Results
 - View the details of any particular build

The screenshot shows the Hudson web interface for the project 'oia-debt-calculator-portlet'. The page includes a navigation sidebar on the left with links like 'Back to Dashboard', 'Status', 'Changes', 'Workspace', 'Build Now', 'Delete Project', 'Configure', and 'Subversion Polling Log'. The main content area features a 'Test Result Trend' chart showing a blue area graph of test counts over build numbers #7 to #79. A red circle highlights the 'Recent Changes' link, which is annotated with a yellow box containing the text 'Change history'. Below the chart are sections for 'Downstream Projects' (listing 'oia-debt-calculator-portlet-integration-tests') and 'Permalinks' (listing links for the last, last stable, last successful, and last failed build).

Hudson search admin | logout DISABLE AUTO REFRESH

Hudson » [oia-debt-calculator-portlet](#)

[Back to Dashboard](#)

[Status](#)

[Changes](#)

[Workspace](#)

[Build Now](#)

[Delete Project](#)

[Configure](#)

[Subversion Polling Log](#)

Project oia-debt-calculator-portlet

[add description](#)

[Workspace](#)

Last Successful Artifacts

- debtcalculator-portlet.war

[Recent Changes](#)

Test Result Trend

count

#7 #9 #11 #13 #15 #17 #19 #21 #24 #26 #28 #30 #32 #34 #36 #38 #40 #42 #44 #51 #53 #55 #57 #59 #61 #63 #67 #74 #77 #79

[\(just show failures\)](#) [enlarge](#)

Downstream Projects

- [oia-debt-calculator-portlet-integration-tests](#)

Permalinks

- [Last build \(#81\), 31 minutes ago](#)
- [Last stable build \(#81\), 31 minutes ago](#)
- [Last successful build \(#81\), 31 minutes ago](#)
- [Last failed build \(#80\), 35 minutes ago](#)

Build History (trend)

Build #	Timestamp
#81	Feb 8, 2008 3:55:49 AM
#80	Feb 8, 2008 3:51:49 AM
#79	Feb 8, 2008 3:48:01 AM
#78	Feb 8, 2008 3:46:43 AM
#77	Feb 8, 2008 3:34:11 AM
#76	Feb 8, 2008 3:10:11 AM
#75	Feb 8, 2008 3:02:55 AM
#74	Feb 8, 2008 2:44:54 AM
#73	Feb 8, 2008 2:34:42 AM
#72	Feb 8, 2008 2:20:47 AM

Continuous Integration

- Displaying Build Results
 - View the details of any particular build job

The screenshot displays the Hudson web interface. The top navigation bar includes the 'Hudson' logo, a search box, and user information 'admin | logout'. The main content area is titled 'Project oia-debt-calculator-portlet'. On the left sidebar, the 'Workspace' link is circled in red, with a yellow callout box containing the text 'View workspace'. Below the sidebar is a 'Build History' table with columns for build number, date, and time. The main content area shows a file browser for the 'debtcalculator-core / src /' directory, listing folders like '.svn', 'argouml', 'main', 'site', and 'test'. Below the file browser is another 'Build History' table and a 'Downstream Projects' section listing 'oia-debt-calculator-portlet-integration-tests'. At the bottom, there is a 'Permalinks' section with links to the last build, last stable build, last successful build, and last failed build.

Build History	(trend)
#81	Feb 8, 2008 3:55:49 AM
#80	Feb 8, 2008 3:51:49 AM
#79	Feb 8, 2008 3:48:01 AM
#78	Feb 8, 2008 3:46:43 AM
#77	Feb 8, 2008 3:34:11 AM
#76	Feb 8, 2008 3:10:11 AM
#75	Feb 8, 2008 3:02:55 AM
#74	Feb 8, 2008 2:44:54 AM
#73	Feb 8, 2008 2:34:42 AM
#72	Feb 8, 2008 2:20:47 AM

Build History	(trend)
(all files in zip)	

Downstream Projects

- oia-debt-calculator-portlet-integration-tests

Permalinks

- Last build (#81), 31 minutes ago
- Last stable build (#81), 31 minutes ago
- Last successful build (#81), 31 minutes ago
- Last failed build (#80), 35 minutes ago

Continuous Integration

- Displaying Build Results
 - View the details of any particular build job

The screenshot shows the Hudson web interface for the project 'oia-debt-calculator-portlet'. The page includes a navigation sidebar on the left with links for 'Back to Dashboard', 'Status', 'Changes', 'Workspace', 'Build Now', 'Delete Project', 'Configure', and 'Subversion Rolling Log'. The main content area is titled 'Project oia-debt-calculator-portlet' and contains several sections: 'Workspace' with a folder icon and 'Last Successful Artifacts' listing 'debtcalculator-portlet.war'; 'Recent Changes' with a document icon and a link to 'Latest Test Result (no failures)'; 'Downstream Projects' listing 'oia-debt-calculator-portlet-integration-tests'; and 'Permalinks' with links for 'Last build (#81), 31 minutes ago', 'Last stable build (#81), 31 minutes ago', 'Last successful build (#81), 31 minutes ago', and 'Last failed build (#80), 35 minutes ago'. A 'Test Result Trend' chart is displayed on the right, showing a blue area chart of test counts over 79 builds, with a red line indicating failures. A 'Build History' table is circled in red, showing a list of builds with their IDs, dates, and times. A yellow callout box with the text 'Build History' points to this table.

Build ID	Date	Time
#81	Feb 8, 2008	3:55:49 AM
#80	Feb 8, 2008	3:51:49 AM
#79	Feb 8, 2008	3:48:01 AM
#78	Feb 8, 2008	3:46:43 AM
#77	Feb 8, 2008	3:34:11 AM
#76	Feb 8, 2008	3:10:11 AM
#75	Feb 8, 2008	2:44:54 AM
#74	Feb 8, 2008	2:34:42 AM
#73	Feb 8, 2008	2:28:47 AM

Continuous Integration

- Displaying Build Results
 - View the details of a particular build

The screenshot displays the Hudson web interface for a specific build. The page title is "Hudson" and the breadcrumb is "Hudson > oia-debt-calculator-core > #49". The main content area shows "Build #49 (Mar 5, 2008 4:23:55 PM)" with a status icon. To the right, it indicates "Started 1 day 0 hours ago" and "Took 1 minute 3 seconds". Below this, there are sections for "Build Artifacts" (listing "debtcalculator-core-1.0-SNAPSHOT.jar"), "Revision: 172", "Changes" (listing "1. test (detail/Trac)"), and "Test Result (no fail)". A "Permalinks" section at the bottom lists "Build number". The interface includes a search bar, a login link, and a "Back to Project" link. The version "Hudson ver. 1.180" is shown at the bottom right.

Build History

Build #49 (Mar 5, 2008 4:23:55 PM)

Started 1 day 0 hours ago
Took 1 minute 3 seconds

The artifact for this build

Build Artifacts

- debtcalculator-core-1.0-SNAPSHOT.jar

Revision: 172

The changes that caused this build

Changes

- test (detail/Trac)

Test Result (no fail)

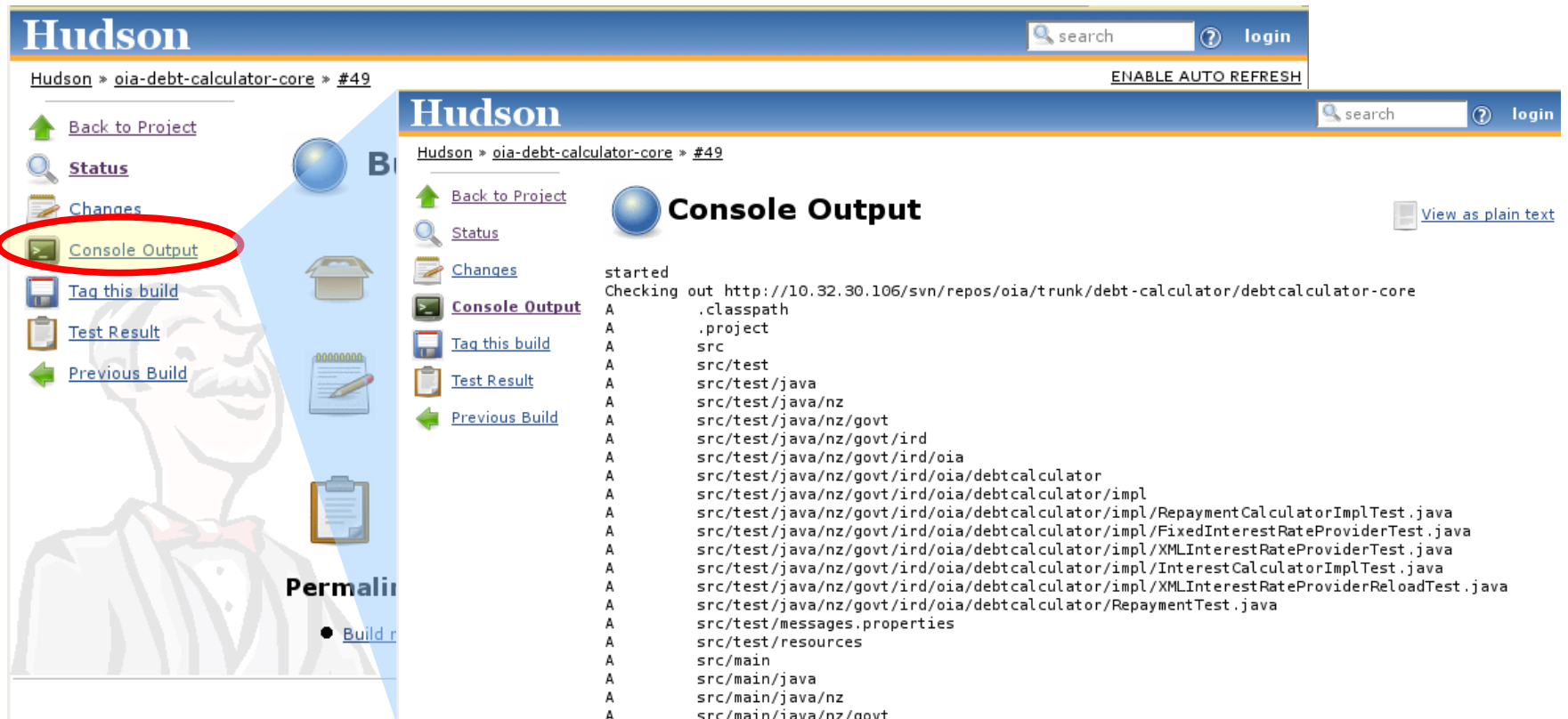
Permalinks

- Build number

Hudson ver. 1.180

Continuous Integration

- Displaying Build Results
 - View the details of a particular build – in real time!



The screenshot displays the Hudson web interface for a specific build. The top navigation bar includes the 'Hudson' logo, a search field, and a 'login' button. Below the navigation bar, the breadcrumb path is 'Hudson > oia-debt-calculator-core > #49'. On the left sidebar, several navigation links are visible: 'Back to Project', 'Status', 'Changes', 'Console Output' (highlighted with a red circle), 'Tag this build', 'Test Result', and 'Previous Build'. The main content area shows the 'Console Output' for build #49, with a 'View as plain text' link on the right. The console output text is as follows:

```
started
Checking out http://10.32.30.106/svn/repos/oia/trunk/debt-calculator/debtcalculator-core
A      .classpath
A      .project
A      src
A      src/test
A      src/test/java
A      src/test/java/nz
A      src/test/java/nz/govt
A      src/test/java/nz/govt/ird
A      src/test/java/nz/govt/ird/oia
A      src/test/java/nz/govt/ird/oia/debtcalculator
A      src/test/java/nz/govt/ird/oia/debtcalculator/impl
A      src/test/java/nz/govt/ird/oia/debtcalculator/impl/RepaymentCalculatorImplTest.java
A      src/test/java/nz/govt/ird/oia/debtcalculator/impl/FixedInterestRateProviderTest.java
A      src/test/java/nz/govt/ird/oia/debtcalculator/impl/XMLInterestRateProviderTest.java
A      src/test/java/nz/govt/ird/oia/debtcalculator/impl/InterestCalculatorImplTest.java
A      src/test/java/nz/govt/ird/oia/debtcalculator/impl/XMLInterestRateProviderReloadTest.java
A      src/test/java/nz/govt/ird/oia/debtcalculator/RepaymentTest.java
A      src/test/messages.properties
A      src/test/resources
A      src/main
A      src/main/java
A      src/main/java/nz
A      src/main/java/nz/govt
```

Wakaleo Consulting

Optimizing your software development

<http://www.wakaleo.com>
john.smart@wakaleo.com

Code Quality

Code Quality

- Why enforce coding standards?
 - Better quality code
 - Code is easier to maintain
 - Detect potential bugs
 - Train staff

Code Quality

- Manual code reviews are good, but...
 - Slow and time-consuming
 - Tend not to be done systematically
- Automatic code audits
 - Automatically enforce organisation coding standards
 - Detect bad coding practices and potential bugs
 - Facilitate developer training

Code Quality

- Quality Metrics Tools
 - Several complementary tools
 - **Checkstyle** – coding standards
 - **PMD** – best practices
 - **FindBugs** – potential bugs
 - **Crap4j** – overly complex and poorly tested classes

Code Quality

- **Checkstyle** - Enforce coding standards
 - Formatting and indentation
 - Naming conventions
 - Javadocs
 - etc...

Code Quality

- **Checkstyle** - Enforce coding standards
 - Eclipse plugin



```
    * The path to the RSS file containin
    * The application must have write-ac
    */
    private String rssFeedFilename = "rss

    /**
     * Update frequency.
     * The frequency (in seconds) in whic
     * for updates.
     */
    '30' is a magic number. dateFrequency = 30;

    /**
     * Continuum server IRI
```

Code Quality

- **Checkstyle** - Enforce coding standards
 - Hudson reports



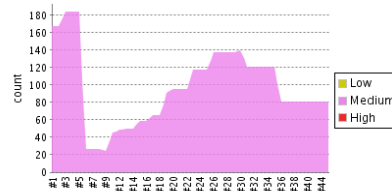
Hudson search ? logout
Hudson » egst-maven-site » #45 DISABLE AUTO REFRESH

- [Back to Project](#)
- [Status](#)
- [Changes](#)
- [Console Output \[raw\]](#)
- [Tag this build](#)
- [Test Result](#)
- [Open Tasks](#)
- [Violations](#)**
- [Coverage Report](#)
- [Previous Build](#)
- [Next Build](#)

Violations Report /hudson/job/egst-maven-site/45 for build 45

Type	Violations	Files in violation
checkstyle	80 (±0)	22 (±0)
cpd	2 (±0)	2 (±0)
findbugs	4 (±0)	2 (±0)
pmd	15 (±0)	11 (±0)

checkstyle



filename
src/main/java/nz/govt/ird/gst/web/webflow/listeners/BackToLastViewStateFlowExecution
src/main/java/nz/govt/ird/gst/backend/impl/mock/GstReturnBackendServiceMockImpl.java
src/main/java/nz/govt/ird/gst/backend/impl/mock/UserProfileBackendServiceMockImpl.java
src/main/java/nz/govt/ird/gst/domain/GstReturn.java
src/main/java/nz/govt/ird/gst/domain/GstReturnProfile.java

Hudson search ? logout
Hudson » egst-maven-site » #45 DISABLE AUTO REFRESH

- [Back to Project](#)
- [Status](#)
- [Changes](#)
- [Console Output \[raw\]](#)
- [Tag this build](#)
- [Test Result](#)
- [Open Tasks](#)
- [Violations](#)**
- [Coverage Report](#)
- [Previous Build](#)
- [Next Build](#)

src/main/java/nz/govt/ird/gst/domain/GstReturnProfile.java

checkstyle	5 violations
38	' ' is followed by whitespace.
42	Line is longer than 100 characters.
48	Line is longer than 100 characters.
63	Line is longer than 100 characters.
66	Line is longer than 100 characters.

```
File: GstReturnProfile.java Lines 29 to 76
29 * @author Myles Johnson
30 *
31 */
32
33 @Entity
34 /**
35 * Returns a list of details to identify returns saved in local database.
36 * This modifies the initial drop-down list on the home page.
37 */
38 @NamedQueries({
39     {
40         /**
41          * Returns a GstReturnProfile for a Gst Number and Period End Date
42          * A GstReturnProfile with no attached GstReturn entity should be considered to be 'Issued Return' in
43          * FIRST speak.
44          */
45         @NamedQuery(name = "gstReturnProfile.findByGstNumberAndPeriodEndDate",
46             query = "select gstReturnProfile from GstReturnProfile gstReturnProfile "
47                 + "where gstReturnProfile.periodEndDate = :periodEndDate "
48                 + "and gstReturnProfile.registrationProfile.clientDetails.gstRegistrationNumber = "
49                 + ":gstNumber")
49     }
50 })
```


Code Quality

- **Checkstyle** - Enforce coding standards
 - Maven reports




	Line is longer than 90 characters.	62
	Line is longer than 90 characters.	123

`nz/govt/ird/gst/dao/impl/RegistrationProfileDAOImpl.java`

Violation	Message	Line
	Unused import - nz.govt.ird.gst.domain.UnfiledReturnSummary.	9

`nz/govt/ird/gst/backend/UserProfileBackendService.java`

Violation	Message	Line
	Line is longer than 90 characters.	29

`nz/govt/ird/gst/web/GstFormAction.java`

Code Quality

- **PMD** – Best practices
 - Empty try/catch/finally blocks
 - Incorrect null pointer checks
 - Excessive method length or complexity
 - etc...
 - Some overlap with Checkstyle



Code Quality

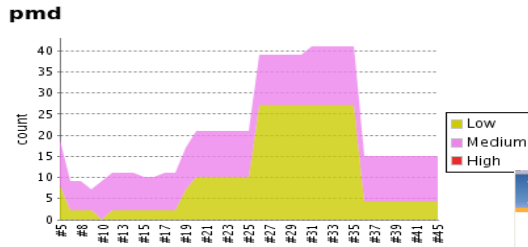
- **PMD** – Best practices
 - Eclipse plugin

A screenshot of the Eclipse IDE's 'Violations Outline' window. The window has a title bar with 'Violations Outline' and standard window controls. Below the title bar are tabs for 'Tasks' and 'Problems'. The main area is a table with two columns: 'Error Message' and 'Line'. The table contains several rows of violations, with the first two rows having a red 'X' icon and the last two rows having a blue 'P' icon. A context menu is open over the row with the message 'Avoid variables with short names like UK', showing options like 'Show Details ...', 'Mark as reviewed', 'Remove violation(s)', 'Quick fix ...', and 'Clear violations reviews'.

Error Message	Line
Use explicit scoping instead of the default package private level	39
Use explicit scoping instead of the default package private level	42
Avoid variables with short names like UK	24
Avoid variables with short names like FR	27
Avoid variables with short names like E5	30
Avoid variables with short names like GR	33
Avoid variables with short names like DE	36
Parameter 'code' is not assigned and could be declared final	76
Parameter 'name' is not assigned and could be declared final	92

Code Quality

- **PMD – Best practices**
 - Hudson reports



filename
src/main/java/nz/govt/ird/gst/backend/impl/mock/GstReturnBackendImpl.java
src/main/java/nz/govt/ird/gst/dao/impl/GstReturnProfileDAOImpl.java
src/main/java/nz/govt/ird/gst/domain/GstReturn.java
src/main/java/nz/govt/ird/util/database/DatabaseSetup.java
src/main/java/nz/govt/ird/gst/domain/UnfiledReturnSummary.java
src/main/java/nz/govt/ird/gst/service/impl/DatabaseAdminServiceImpl.java
src/main/java/nz/govt/ird/gst/web/ChoosePeriodController.java
src/main/java/nz/govt/ird/gst/web/ProvTaxFormAction.java
src/main/java/nz/govt/ird/gst/web/form/UnfiledReturnForm.java
src/main/java/nz/govt/ird/gst/web/validation/ProvTaxValidator.java
src/main/java/nz/govt/ird/util/PlatformManager.java

Hudson search ? logout

Hudson > egst-maven-site > #45 DISABLE AUTO REFRESH

- [Back to Project](#)
- [Status](#)
- [Changes](#)
- [Console Output \[raw\]](#)
- [Tag this build](#)
- [Test Result](#)
- [Open Tasks](#)
- [Violations](#)
- [Coverage Report](#)
- [Previous Build](#)
- [Next Build](#)

src/main/java/nz/govt/ird/gst/dao/impl/GstReturnProfileDAOImpl.java

pmd 2 violations

20	⚠	Avoid unused private fields such as 'LOGGER'.
63	⚠	Avoid unused local variables such as 'gstReturnProfile'.

File: GstReturnProfileDAOImpl.java Lines 11 to 30

```
11
12 /**
13  * The DAO class for GST/Prov returns.
14  */
15 public class GstReturnProfileDAOImpl extends
16     GenericDAOImpl<GstReturnProfile, Long> implements GstReturnProfileDAO {
17     /**
18      * Logger.
19      */
20     private static final Logger LOGGER = Logger
21         .getLogger(GstReturnDAOImpl.class);
22
23     /**
24      * Find the list of returns for a given IRD number.
25      *
26      * @param gstNumber
27      *     a valid IRD number (9 digit)
28      * @return a list of GstProvReturn objects, ordered by descending date.
29      */
30     @SuppressWarnings("unchecked")
```

Code Quality

- **PMD** – Best practices
 - Maven reports



PMD Results

The following document contains the results of [PMD 3.9](#).

Files

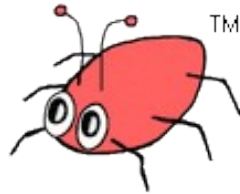
[nz/govt/ird/gst/backend/impl/mock/GstReturnBackendServiceMockImpl.java](#)

Violation	Line
Avoid unused imports such as 'java.util.Date'	4
Avoid unused imports such as 'org.joda.time.DateTimeUtils'	9

[nz/govt/ird/gst/dao/GstReturnDAO.java](#)

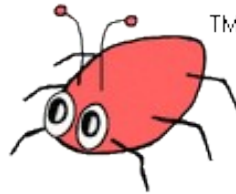
Code Quality

- **FindBugs** – Potential defects
 - Potential NullPointerExceptions
 - Infinite loops
 - etc...



Code Quality

- **FindBugs** – Potential defects
 - Eclipse plugin



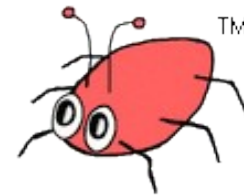
```
if (ref == null && ( updated != null && updated instanceof Collection) ) {
NP: Load of known null value in nz.govt.natlib.symbols.core.updates.ChangedLibraries.isCollection(java.lang.reflect.Method)1);
    getObjectChanges() .add(changedCollection);
    return true;
}

if (( ref != null && ref instanceof Collection) && updated == null) {
    ChangedCollection changedCollection = new ChangedCollection(ref,updated);
    getObjectChanges() .add(changedCollection);
}
```

Problems 18 errors, 288 warnings, 0 infos

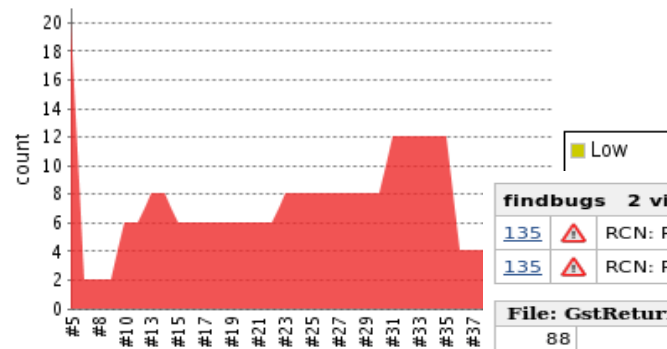
Description	Resource	Path	Location
Errors (18 items)			
Warnings (100 of 288 items)			
⚠️ DLS: Dead store to baseDir in method nz.govt.natlib.symbols.core.searchengine.Di	DirectoryProviderI...	library-symbols-web/src/test/java/nz/govt/natlib/symb...	line 80
⚠️ DLS: Dead store to c in method nz.govt.natlib.symbols.core.updates.ChangedLibra	ChangedLibraries.j...	library-symbols-web/src/main/java/nz/govt/natlib/sym...	line 36
⚠️ DLS: Dead store to c in method nz.govt.natlib.symbols.core.updates.ChangedLibra	ChangedLibrary.java	library-symbols-web/src/main/java/nz/govt/natlib/sym...	line 60
⚠️ DLS: Dead store to categories in method nz.govt.natlib.symbols.core.updates.Lib	LibrarySymbolsUpd...	library-symbols-web/src/main/java/nz/govt/natlib/sym...	line 156
⚠️ DLS: Dead store to cycle0 in method nz.govt.natlib.symbols.web.pages.Applicatio	ApplicationBasePan	library-symbols-web/src/main/java/nz/govt/natlib/sym	line 226

Code Quality



- **FindBugs** – Potential defects
 - Hudson reports

findbugs

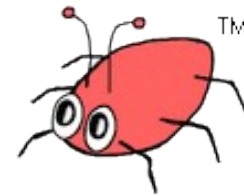


filename
src/main/java/nz/govt/ird/gst/service/impl/GstReturnProviderImpl.java
src/main/java/nz/govt/ird/gst/web/form/UnfiledReturnForm.java

```
findbugs 2 violations
135 [A] RCN: Redundant nullcheck of processedReturn which is known to be null in nz.govt.ird.gst.service.impl.GstReturnProviderImpl.java
135 [A] RCN: Redundant nullcheck of processedReturn which is known to be null in nz.govt.ird.gst.service.impl.GstReturnProviderImpl.java

File: GstReturnProviderImpl.java Lines 88 to 107
88 // decide how we will manage this
89 }
90 }
91 }
92 }
93 if (!(ReturnStatus.Saved.equals(gstReturn.getStatus()) || ReturnStatus.New
94 .equals(gstReturn.getStatus())) {
95 // TODO - throw proper exception instead of runtime exception.
96 throw new RuntimeException(
97 "TODO Error trying to fetch an unfiled return that is currently consid
98 + gstReturn.getStatus() + """);
99 }
100 }
101 return gstReturn;
102 }
103 }
104 /**
```

Code Quality



- **FindBugs** – Potential defects
 - Maven reports

Files

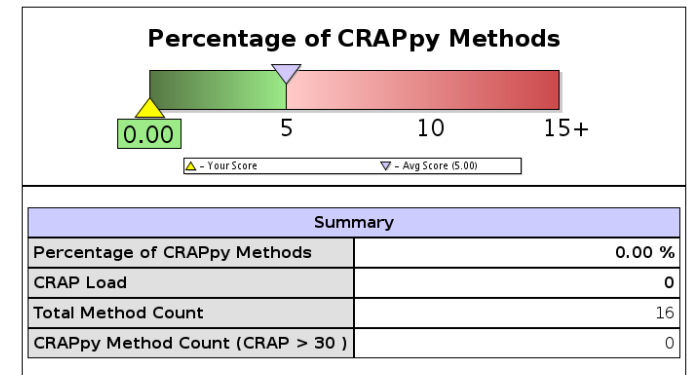
Class	Bugs
nz.govt.ird.gst.domain.HistoryDetailEntry	4
nz.govt.ird.gst.refdomain.RefGSTProfile	3
nz.govt.ird.gst.refdomain.RefGstProvProfile	3
nz.govt.ird.gst.refdomain.RefGstProvReturn	4
nz.govt.ird.gst.refdomain.RefOldGstProvReturn	4

[nz.govt.ird.gst.domain.HistoryDetailEntry](#)

Bug	Category	Details	Line
nz.govt.ird.gst.domain.HistoryDetailEntry.getRtnPeriodEnd() may expose internal representation by returning nz.govt.ird.gst.domain.HistoryDetailEntry.rtnPeriodEnd	MALICIOUS_CODE	EI_EXPOSE_REP	27
nz.govt.ird.gst.domain.HistoryDetailEntry.getStatusDate() may expose internal representation by returning _____	MALICIOUS_CODE	EI_EXPOSE_REP	43

Code Quality

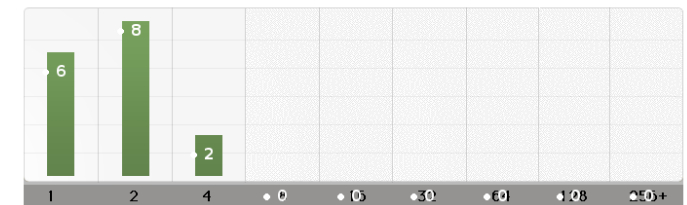
- **Crap4j** – Complex and untested code
 - Uses code coverage and code complexity, ~~metrics~~
 - Eclipse plugin



[Share and Compare](#)

Share your results, (anonymously or publicly) & compare your project to others.

Method CRAP Distribution



Wakaleo Consulting

Optimizing your software development

<http://www.wakaleo.com>
john.smart@wakaleo.com

Automated Documentation

Automated Documentation

- Automatically-generated documentation:
 - Complete
 - Always up-to-date
 - Cheap to produce
- **BUT**
 - Lacks “higher vision”

Automated Documentation

- Human-written documentation – use a project Wiki
 - Architecture vision
 - High-level design
 - Collaborative

Application architecture

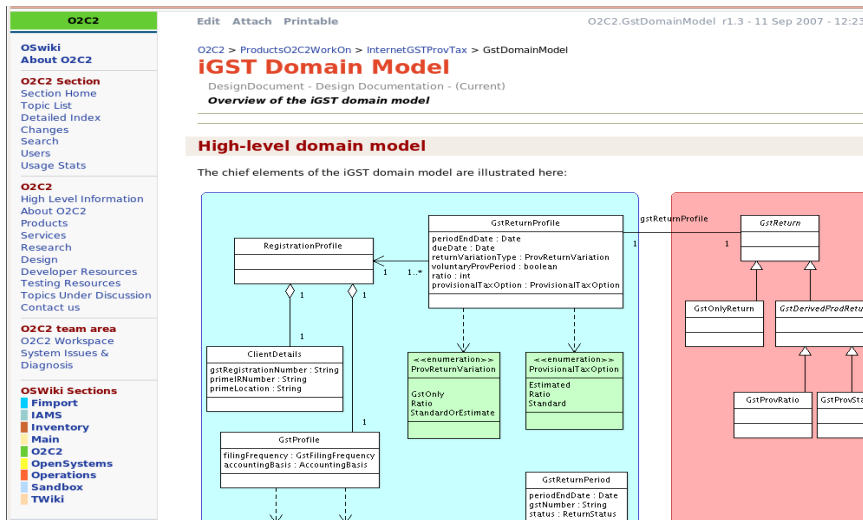
The iGST project uses a fairly standard MVC architecture approach, using the Spring-MVC framework and Hibernate. The target architecture involves a portal-enabled front-end using Spring-MVC (more precisely, Spring Portlet MVC), and Spring Webflow (with the [PortletFlowController](#)) and a business services layer implemented using Spring and Hibernate, backed by an Oracle database. The business layer communicates with FIRST via EAI:



The application architecture is described in detail in the [GstApplicationArchitecture](#) page.

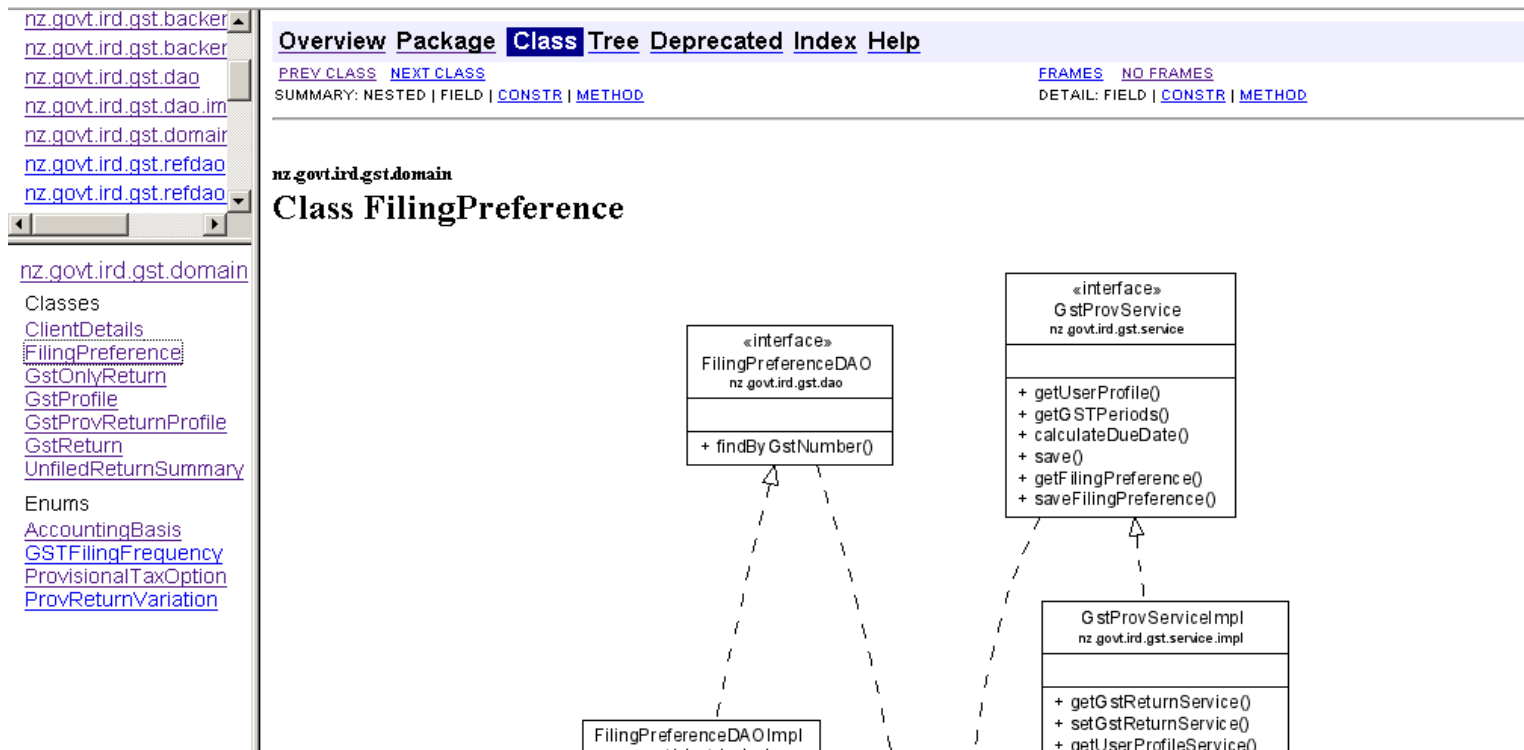
Domain model

The domain model plays a central part in the iGST project, and has been elaborated in a series of workshops. The domain model is designed as an object-oriented class model, and implemented using plain Java classes (POJOs). Database persistence is implemented using Hibernate.



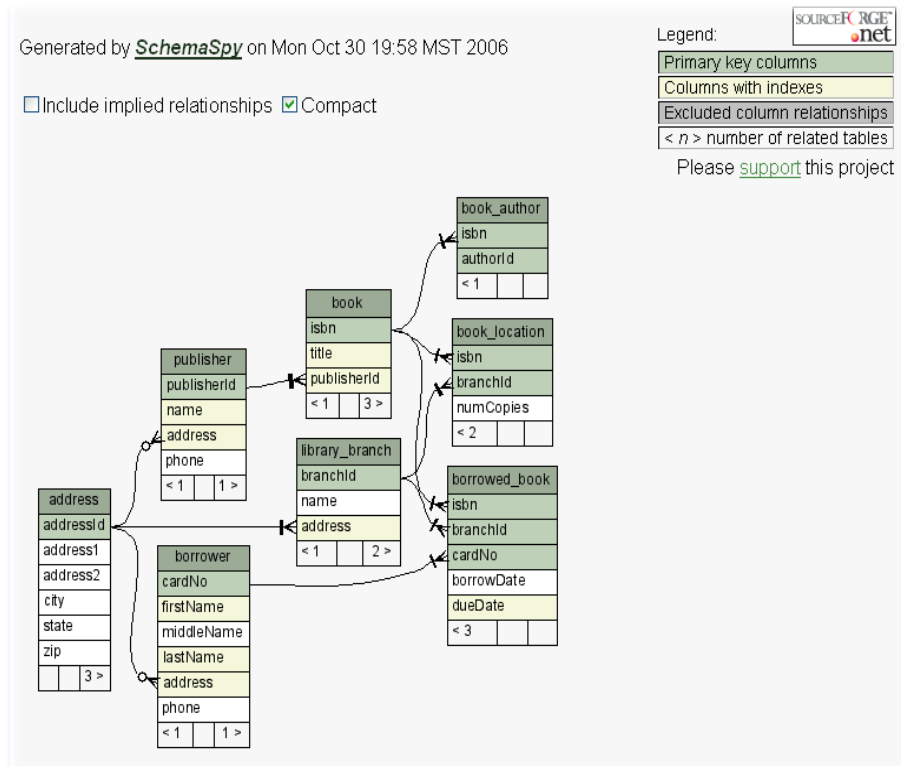
Automated Documentation

- Automatically-generated UML/Javadoc
 - UmlGraph



Automated Documentation

- Automatically-generated database models
 - SchemaSpy

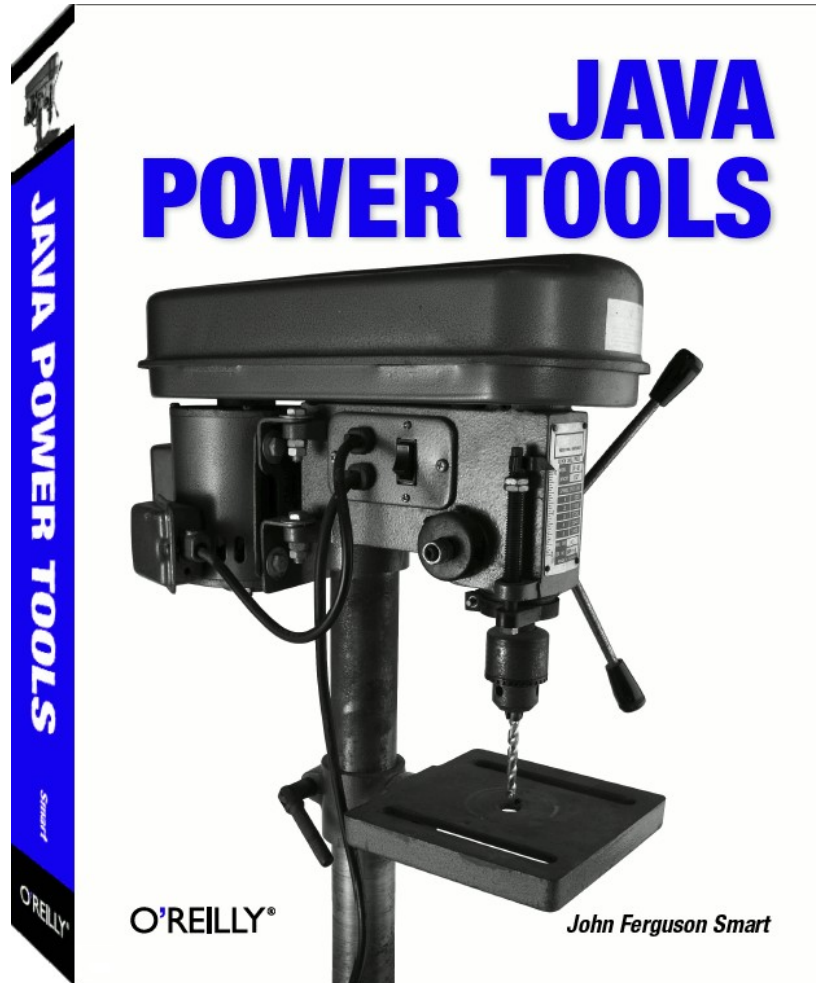


Conclusion

- Optimal development practices
 - So how does your team measure up?
 - Unit testing practices?
 - Automated web testing
 - Code Coverage Metrics
 - Continuous Integration?
 - Continuous Quality?
 - Automated Technical Documentation?



To learn more...



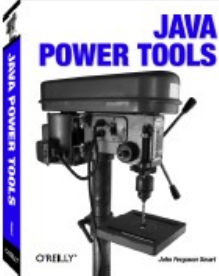
Wakaleo Consulting
Optimizing your software development process

<http://www.wakaleo.com>

The Java Power Tools Bootcamp

Code better - Code faster - Code smarter

The Java Power Tools Bootcamp is a comprehensive, innovative and hands-on workshop covering best-of-breed open source tools and techniques for Agile Development in Java. Learn how to optimize your development process, hone your programming skills and know-how, and ultimately produce better software. And have fun while you're doing it!



Course Objectives

Students will come away from this workshop with a solid understanding of how they can improve their development practices back in the real world, as well as an abundance of practical tips and tricks that they can use in their day-to-day work. Notably, after the course, students will:

- ✓ Have a practical understanding and experience of Maven 2, and be able to determine for themselves if it is suitable for their project or organisation.
- ✓ Understand the issues around dependency management in Java development, and be able to implement declarative dependency management in a corporate environment using both Maven and Ant.
- ✓ Know how to write effective unit tests, and understand how to use unit testing practices to write more reliable code faster.
- ✓ Be able to write automated database and web interface tests.
- ✓ Understand how to use code quality and test coverage metrics to improve your code, and understand what the various metrics can tell you, and also what they can't.
- ✓ Have a solid working knowledge of Subversion in the real world.
- ✓ Know how to set up a working Continuous Integration server, complete with automated builds, tests, code quality audits and reports, and automatic deployment to an integration server

→ Wellington

28 April-1 May, 2008

→ Auckland

11-14 August, 2008

Thank you

- Questions?