Tools that can optimize your development process

Software infrastructure for Java development projects

Presentation plan

- Software Development Lifecycle Tools
  - What tools exist?
  - Where can tools help?
- Practical Applications
  - How are these tools used in the real world?
  - Work of the O2C2 group at Inland Revenue to improve the Inland Revenue Java development process

The Java Power Tools Book

Java Power Tools

Optimizing the Software Development Life Cycle

create account or login

search

Home | Book Outline | Cool Tools | Case studies | Propose a topic | Forum | Polls | The author

Cool links

Home
Get started here!

Book Outline
See what's planned for Java Power Tools!

Cool Tools
Propose cool Java tools you know!

Case studies
Write about your SDLC experiences!

Propose a topic
Write an article for Java Power Tools!

Polls
Check out who uses what!

Participating Companies
Who's been helping out

Home

This is the "official unofficial" web site for the **"Java Power Tools"** book, currently being written for publication by O'Reilly. **"Java Power Tools"** is about software tools and techniques that can contribute to improving the SDLC (Software Development Lifecyle). This includes build tools such as Maven and Ant, CI tools, code quality tools, testing tools, collaborative tools, source version control, and more! It is a collaborative book, where other authors can (and are encouraged to!) contribute material on specific subjects. It is currently scheduled for release in late 2007 to early 2008.

By the way, I'm John Ferguson Smart, the main author of the book.

PURPOSE OF THIS SITE

This site is a collaborative space where the author, contributors or potential contributors, reviewers, innocent bystanders, and any other participants can discuss book topics and help contribute to the final form and content of the book.

BOOK VISION

This book discusses key Java development problem areas and best practices, and focuses on open source tools that can help increase developer productivity in each area. The idea

The Java Power Tools Book

- **Tools to optimize the Java SDLC**
- Main author: John Smart
- Supported by Equinox
- Published by O'Reilly
- Contributions from lots of authors
- Website: *http://www.javapowertools.com*

EQUINOX
think IT · plan IT · do IT

O'REILLY®

Inland Revenue and O2C2

- Inland Revenue
  - Carries out a large number of internal and external Java-based developments

- O2C2:
  - The **Object Oriented Competency Centre**
  - Provide coaching and technical expertise in Java development
  - Optimize development practices and the SDLC infrastructure

Types of tools

- Many tools exist
- Different tools are useful for different task
  - What tools exist?
  - Where can tools help?
  - When are different tools appropriate?

Types of tools

- Let's define some categories:
  - Build Process tools
  - Version Control tools
  - Continuous Integration tools
  - Quality Metrics and code audit tools
  - Unit Testing and Test Coverage tools
  - Integration, Functional, Load and Performance Testing tools
  - Technical Documentation tools

Types of tools

- Let's define some categories:
  - **Build Process tools**
  - Version Control tools
  - Continuous Integration tools
  - Quality Metrics and code audit tools
  - Unit Testing and Test Coverage tools
  - Integration, Functional, Load and Performance Testing tools
  - Technical Documentation tools

Build Tools

- The cornerstone of the SDLC process:
  - Make your builds reproducible
  - Make your builds portable
  - Allows you to automate your build process
- Two main Java tools: **Ant** and **Maven 2**

Ant

- A procedural build scripting tool
  - Well-known and widely-used
  - Powerful and flexible
  - Lots of plugins
  - Needs lots of low-level plumbing code
  - Large scripts can become difficult to maintain

Maven 2

- A declarative build scripting framework
  - Higher level scripting
  - Strong use of standards and conventions
  - “Convention over Configuration”
  - Lots of plugins
  - Good reporting features
  - More rigid than Ant

A Standard Directory Structure

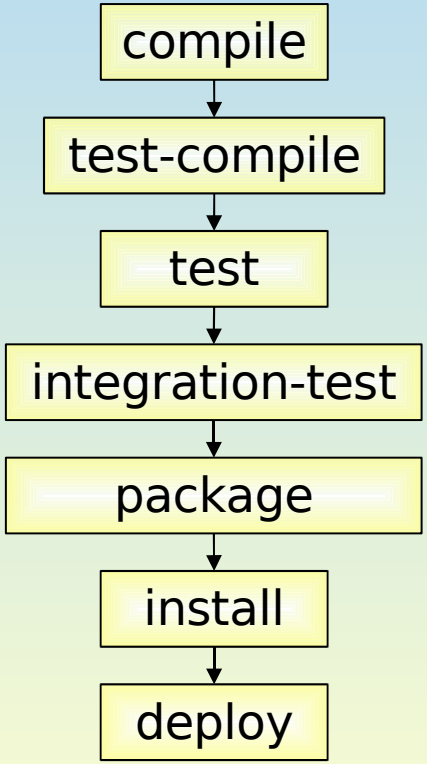
- A standard project directory structure

projects | egst-web | 50% | View as List

Name	Size	Type	Date Modified
src	3 items	folder	Fri 21 Sep 2007 11:44:33 NZST
main	4 items	folder	Fri 21 Sep 2007 11:45:15 NZST
config	7 items	folder	Thu 30 Aug 2007 16:26:18 NZST
java	7 items	folder	Thu 30 Aug 2007 16:26:18 NZST
resources	8 items	folder	Fri 07 Sep 2007 09:00:01 NZST
webapp	6 items	folder	Thu 30 Aug 2007 16:26:19 NZST
site	4 items	folder	Thu 30 Aug 2007 16:26:19 NZST
test	2 items	folder	Fri 21 Sep 2007 11:44:39 NZST
java	2 items	folder	Thu 30 Aug 2007 16:26:19 NZST
resources	5 items	folder	Thu 20 Sep 2007 13:48:45 NZST
target	18 items	folder	Fri 21 Sep 2007 10:39:52 NZST
pom.xml	18.6 KB	XML document	Fri 21 Sep 2007 09:40:35 NZST

Standard lifecycle

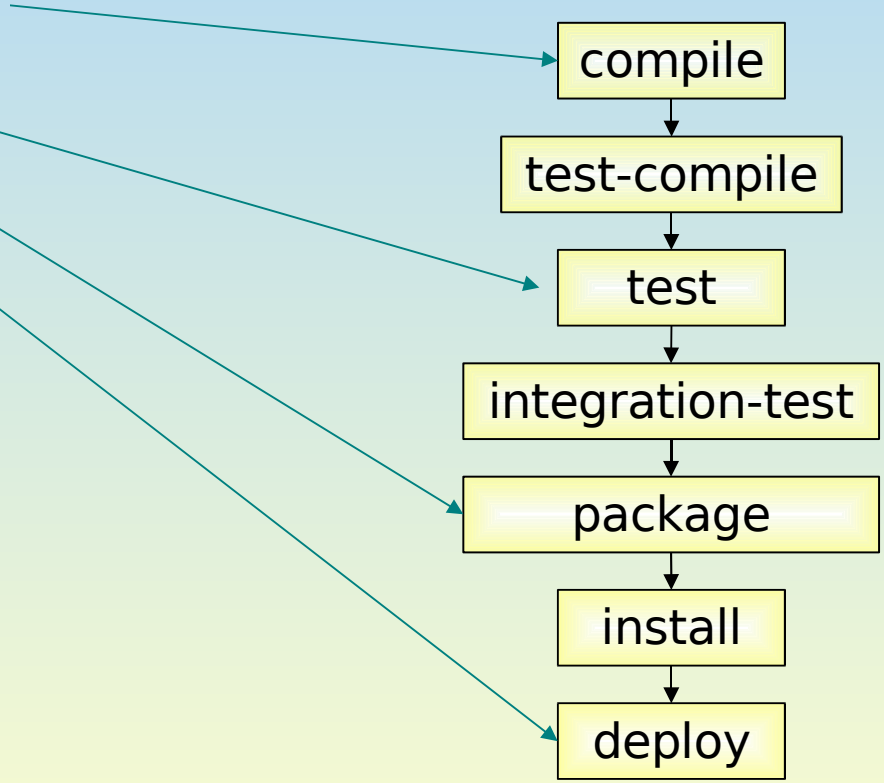
- A standard development lifecycle



Standard Build Commands

- Maven 2 - standard build commands

```
$ mvn compile
$ mvn test
$ mvn package
$ mvn deploy
...
```



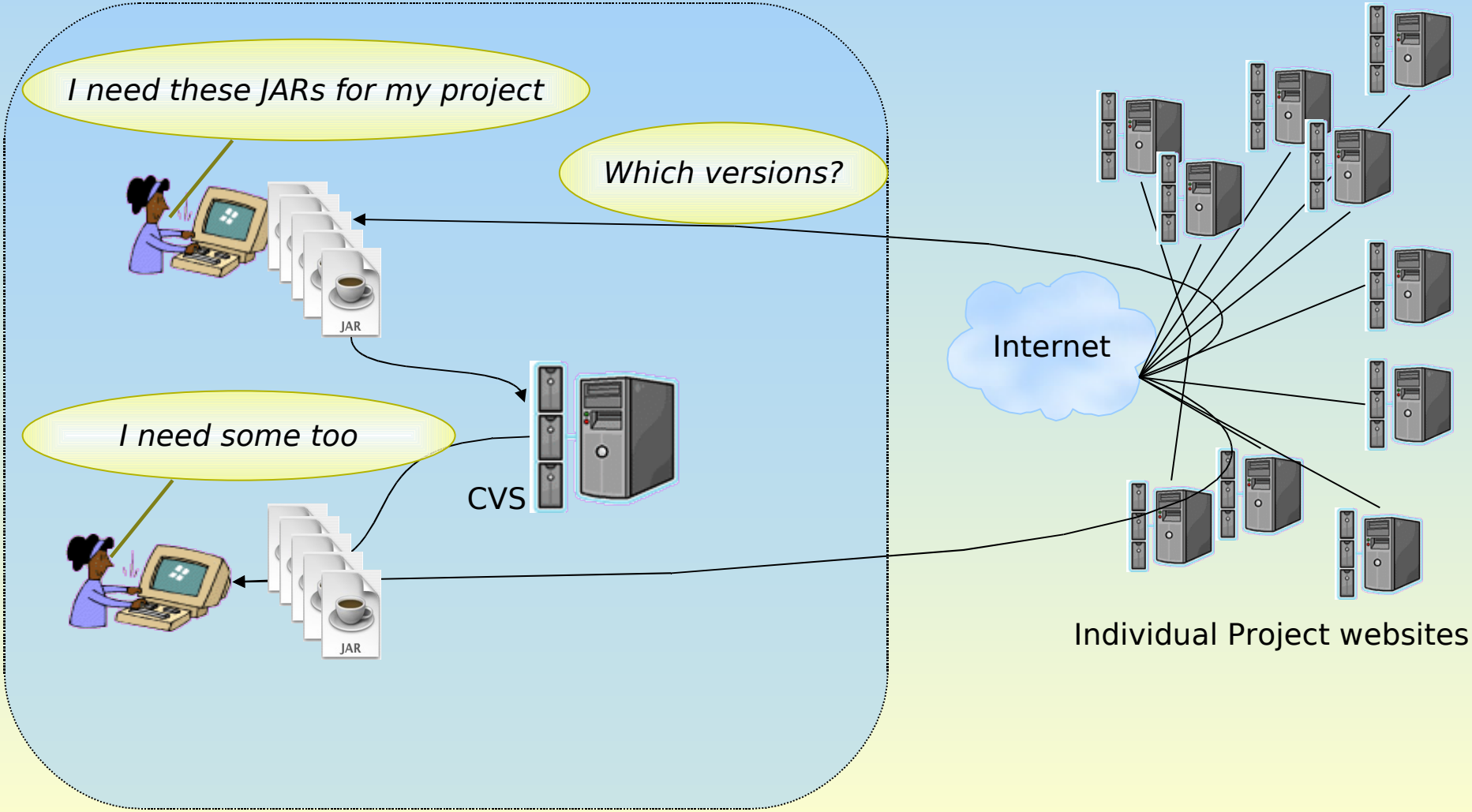
Dependency Management

- Declarative Dependency Management
  - Important feature of Maven 2
  - A major step towards a simpler build process

Traditional Dependency Management

- Traditional approach – JAR files stored in CVS
  - Each project has its own set of JAR files
  - Unnecessary duplication
  - Hard to keep track of versions
  - Errors due to incompatible JARs
  - Overloads the source code repository

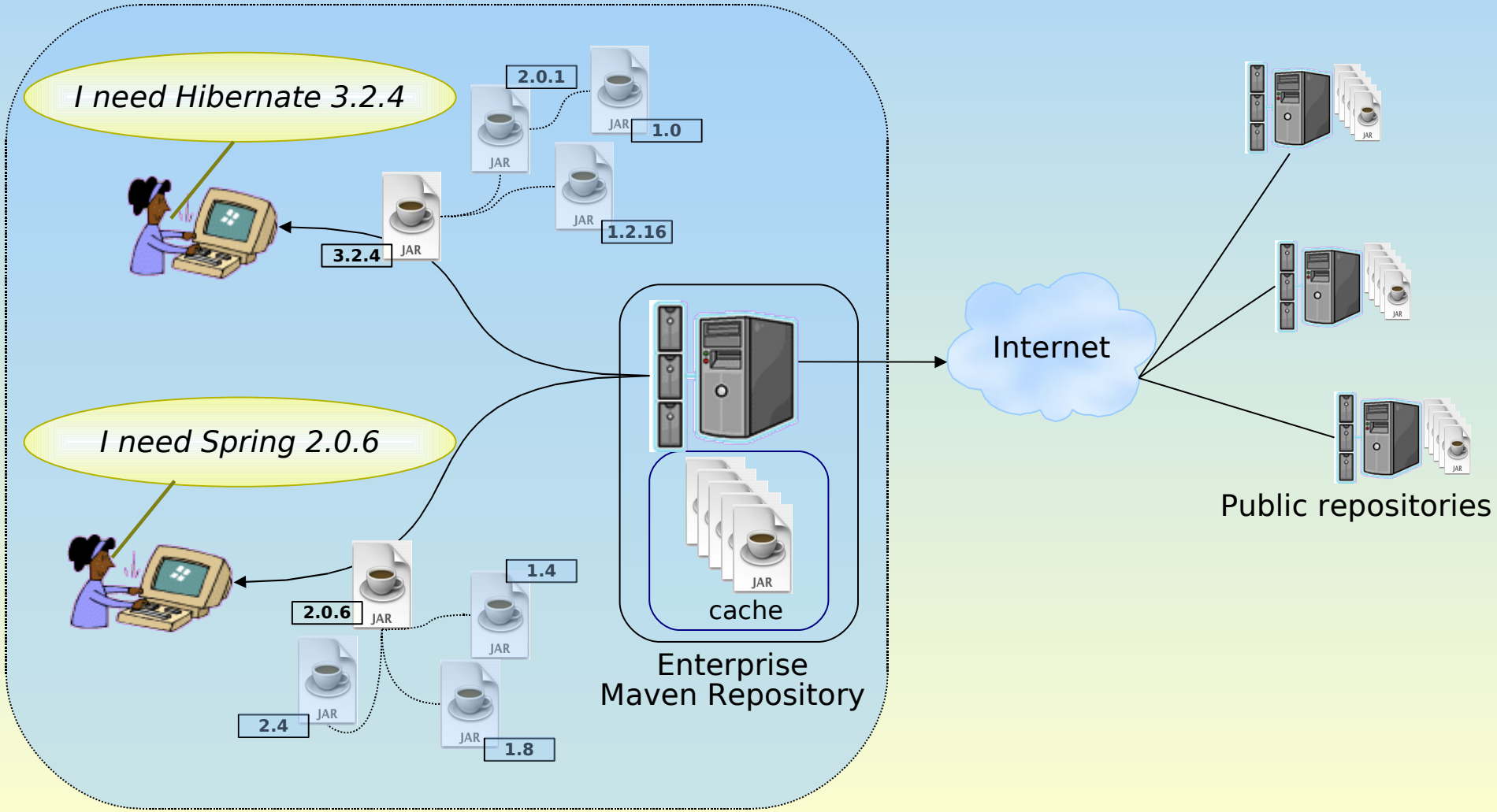
Traditional Dependency Management



Declarative Dependency Management

- Declarative dependency management
  - Versioned JARs stored on a central server
  - Each projects “declares” what libraries it needs
  - Secondary dependencies are automatically downloaded

Declarative Dependency Management

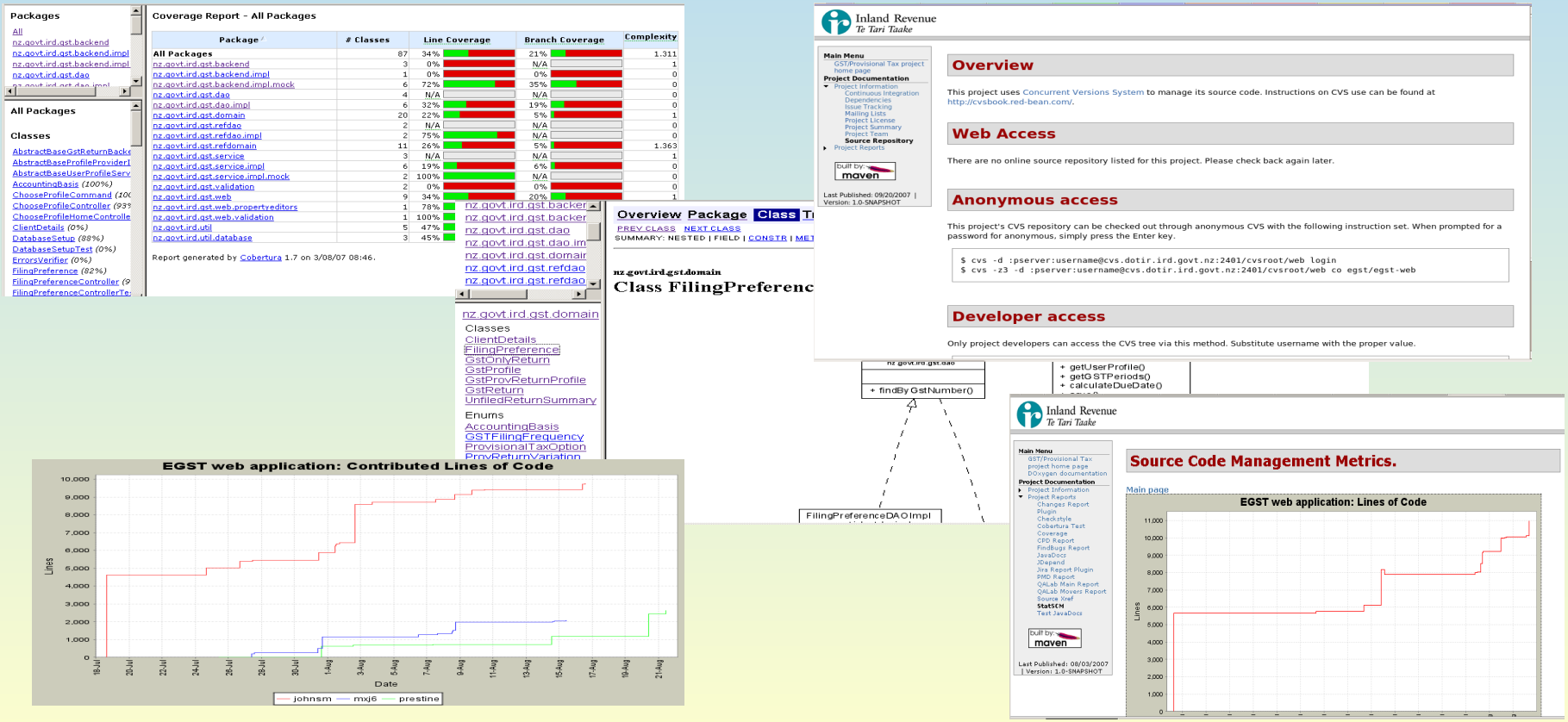


The Enterprise Maven Repository

- An Enterprise Maven Repository
  - Acts as a proxy/cache for downloaded dependencies
  - Faster and more reliable than always going to the internet
  - Store third-party and internal libraries as well

Maven 2 Reporting

- Maven 2 - reporting
  - Powerful and easy-to-use reporting features

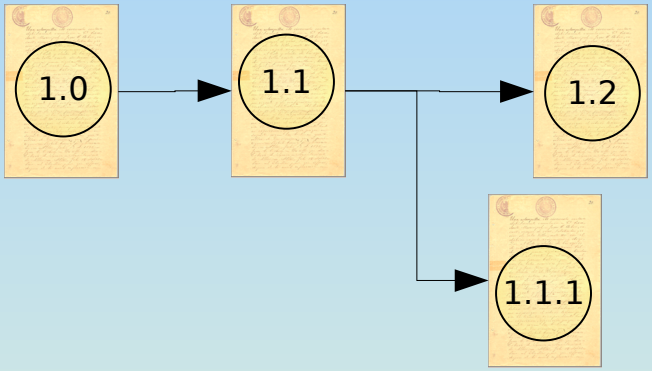


Types of tools

- Categories of tools
  - Build Process tools
  - **Version Control tools**
  - Continuous Integration tools
  - Quality Metrics and code audit tools
  - Unit Testing and Test Coverage tools
  - Integration, Functional, Load and Performance Testing tools
  - Technical Documentation tools

Version Control Tools

- Version control tools provide:
  - A central storage for source code
  - Backups and change history
  - Identify versions and releases



- Two main Open Source tools: **CVS** and **Subversion**

CVS

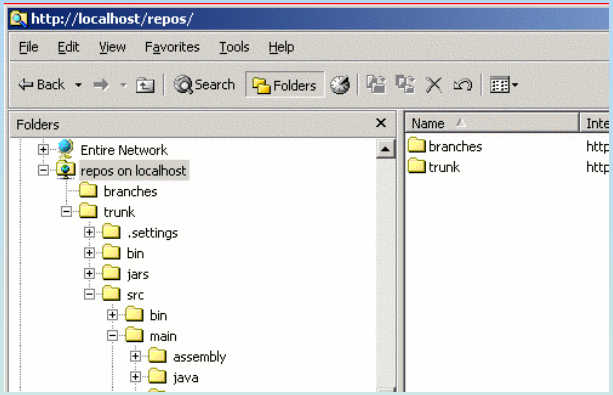
- A venerable open source version control tool
- Starting to show its age:
  - No atomic commits
  - Poor support for non-binary files
  - Difficult to rename or move directories
  - Slow tagging and logging

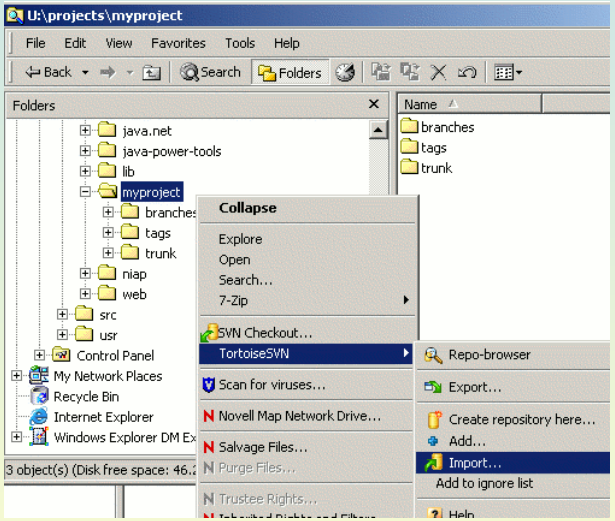
Subversion

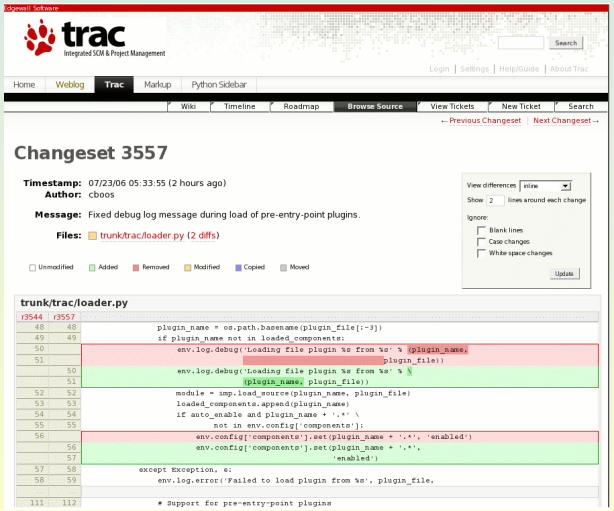
- Designed to replace CVS
- Better adapted to Java development:
  - Atomic commits
  - Good support for binary files
  - *Fast* tagging and branching

Subversion

- Has some nice advanced features:
  - HTTP / WebDAV support
  - Windows integration (TortoiseSVN)
  - Web interface (Trac)







Types of tools

- Categories of tools
  - Build Process tools
  - Version Control tools
  - **Continuous Integration tools**
  - Quality Metrics and code audit tools
  - Unit Testing and Test Coverage tools
  - Integration, Functional, Load and Performance Testing tools
  - Technical Documentation tools

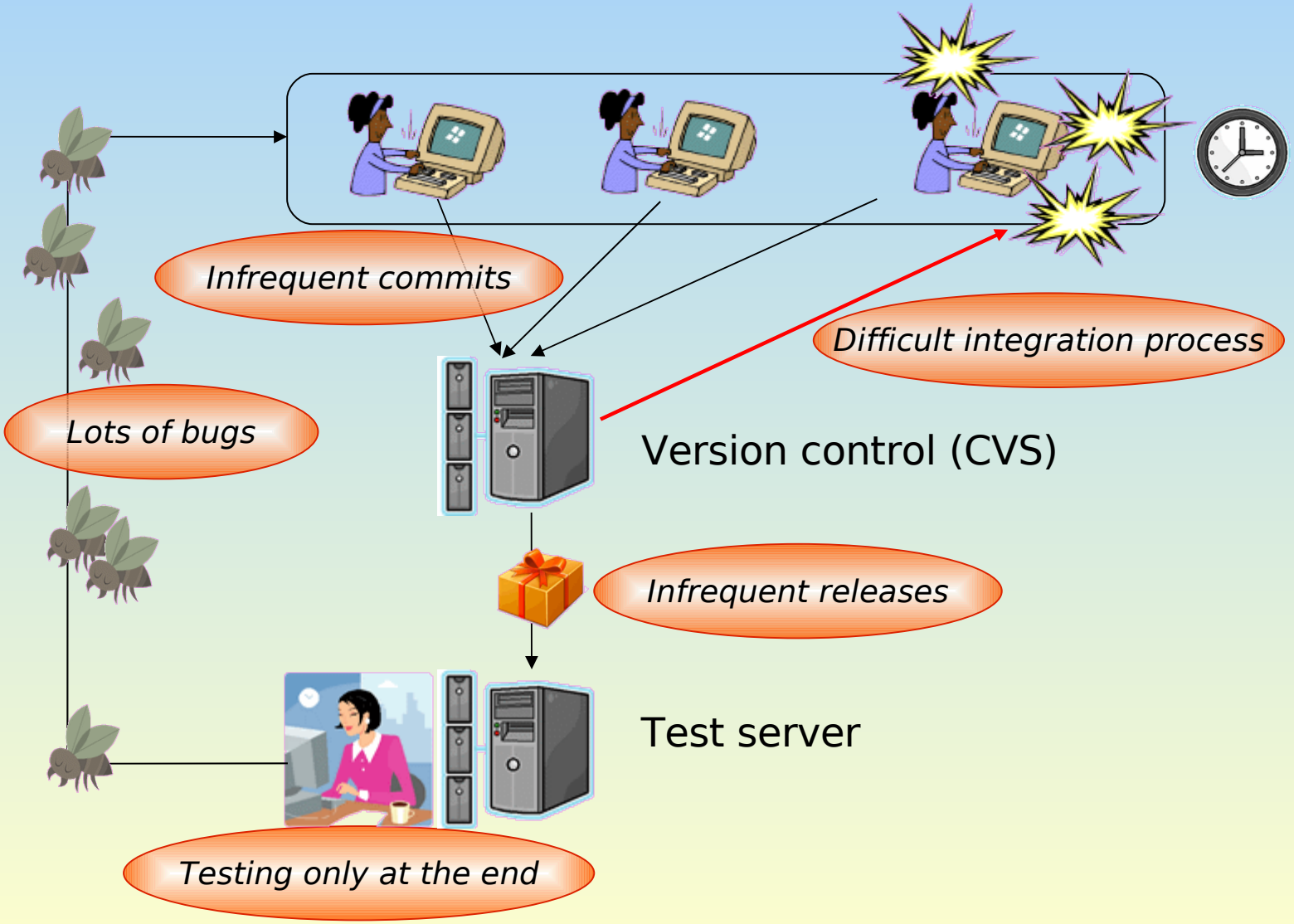
Continuous Integration

- Integrate early, integrate often:
  - Automate the build process
  - Minimise code integration issues
  - Improve visibility on the development process
  - More flexibility and agility

Continuous Integration

- The “Traditional” development process:
  - Coding
  - Maybe some unit tests
  - Integration
  - System and UAT testing
  - Production

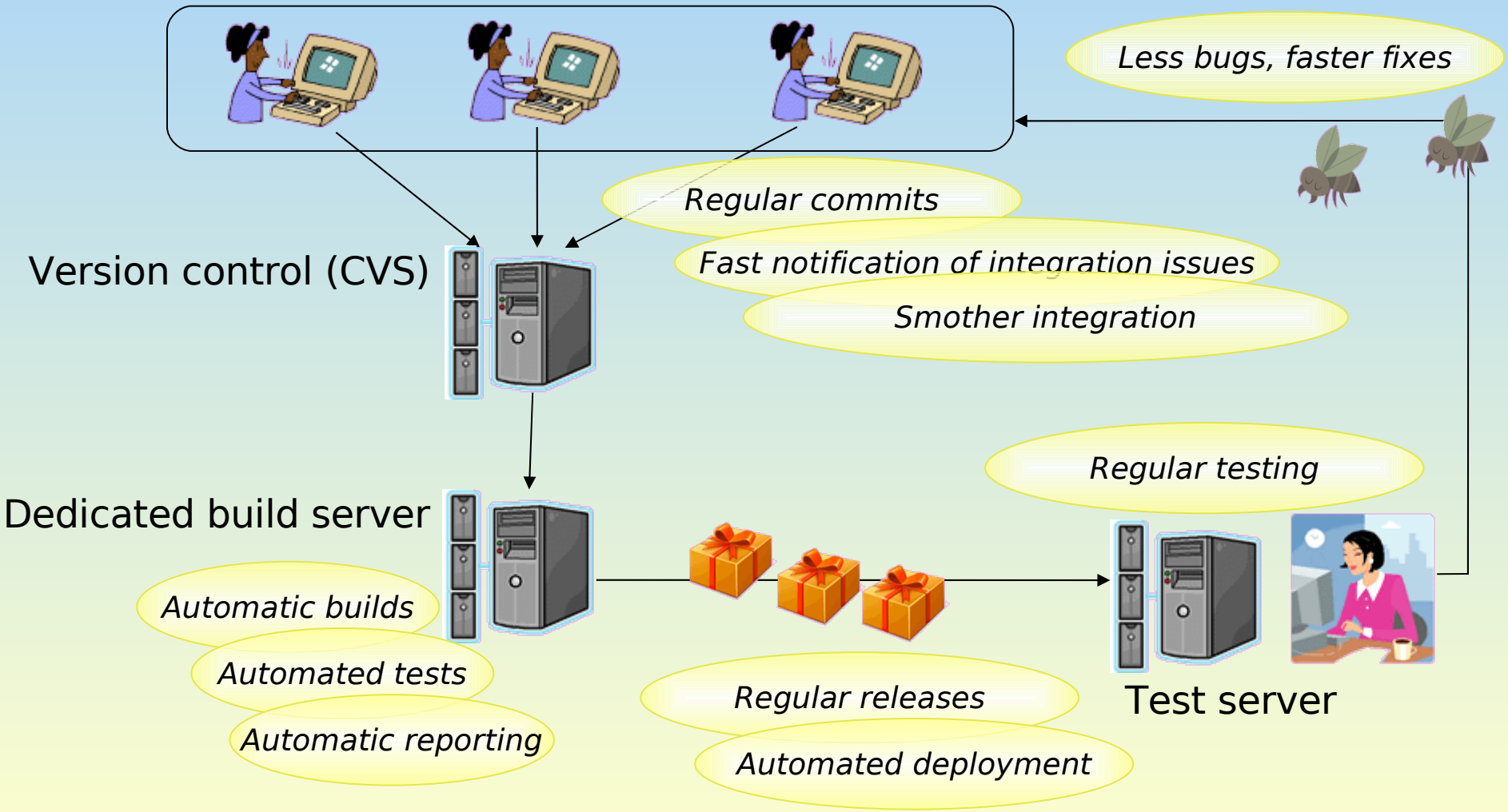
The traditional approach



The traditional approach

- 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

How Continuous Integration helps



Advantages of Continuous Integration

- An industry best practice
  - Smoother integration
  - Automatic regression testing
  - Regular working releases
  - Earlier functional testing
  - Faster and easier bug fixes
  - Better visibility

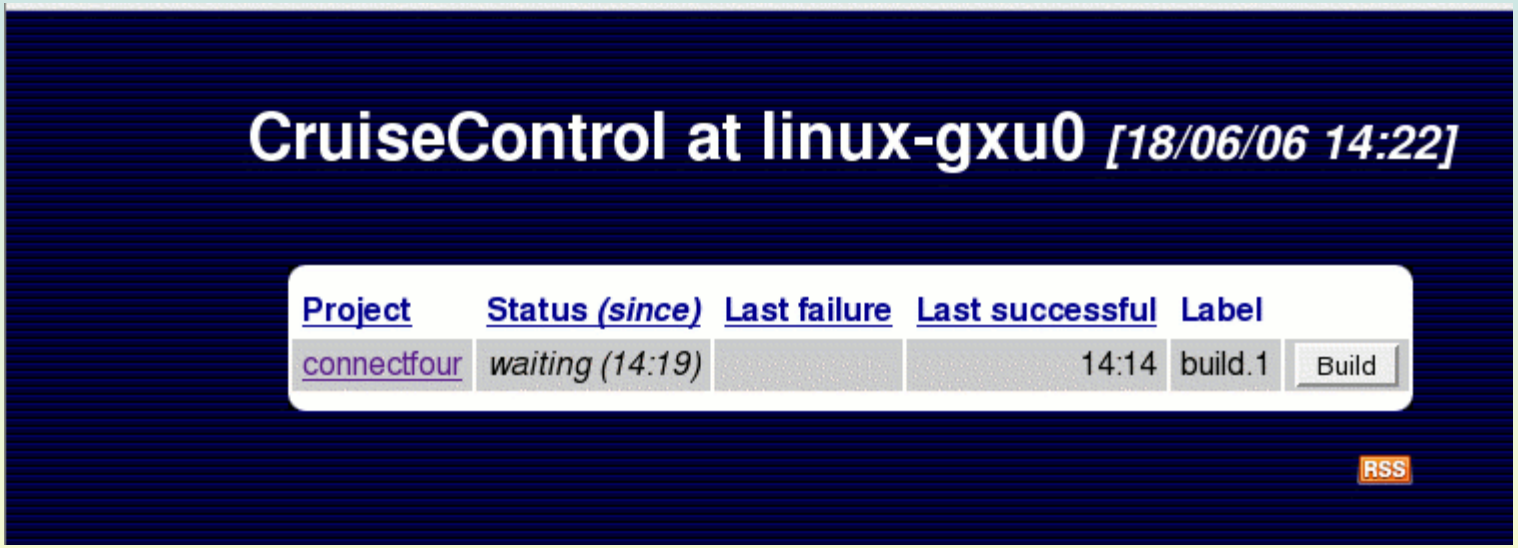
Continuous Integration tools

- Many choices
  - CruiseControl
  - LuntBuild
  - Continuum
  - Hudson
  - ...

- Some commercial tools are also available, for exampe:
  - TeamCity (JetBrains)
  - Bamboo (Atlassian)
  - Pulse (Zutubi)

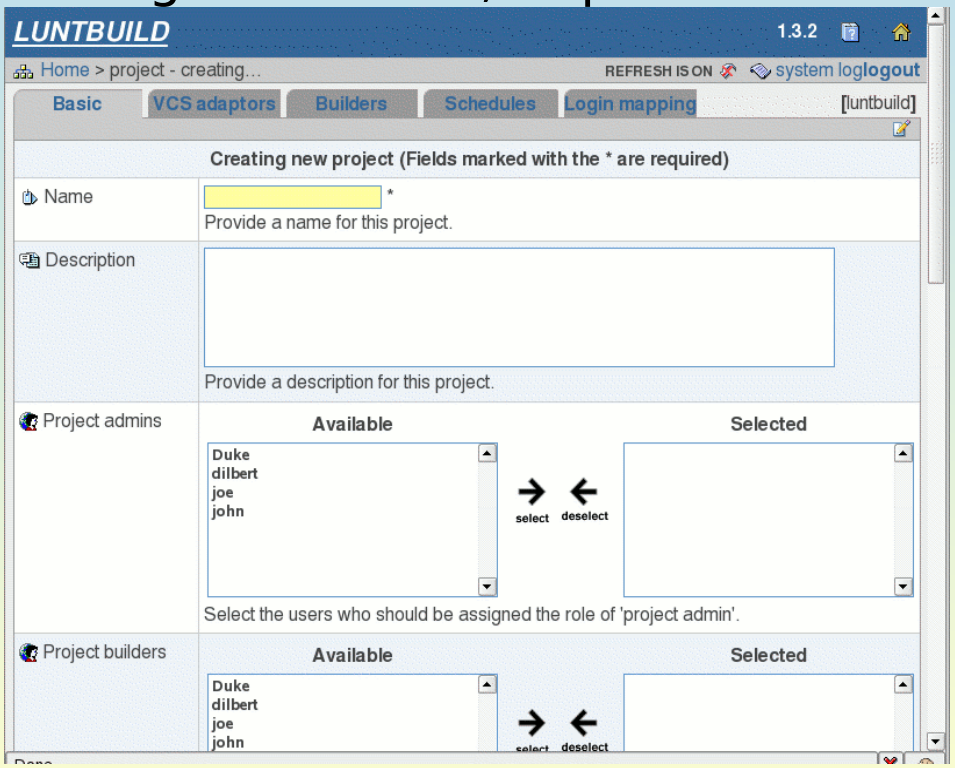
Continuous Integration tools

- CruiseControl
  - Mature tool
  - Strong user base
  - Lots of extensions
  - Hard to set up and maintain



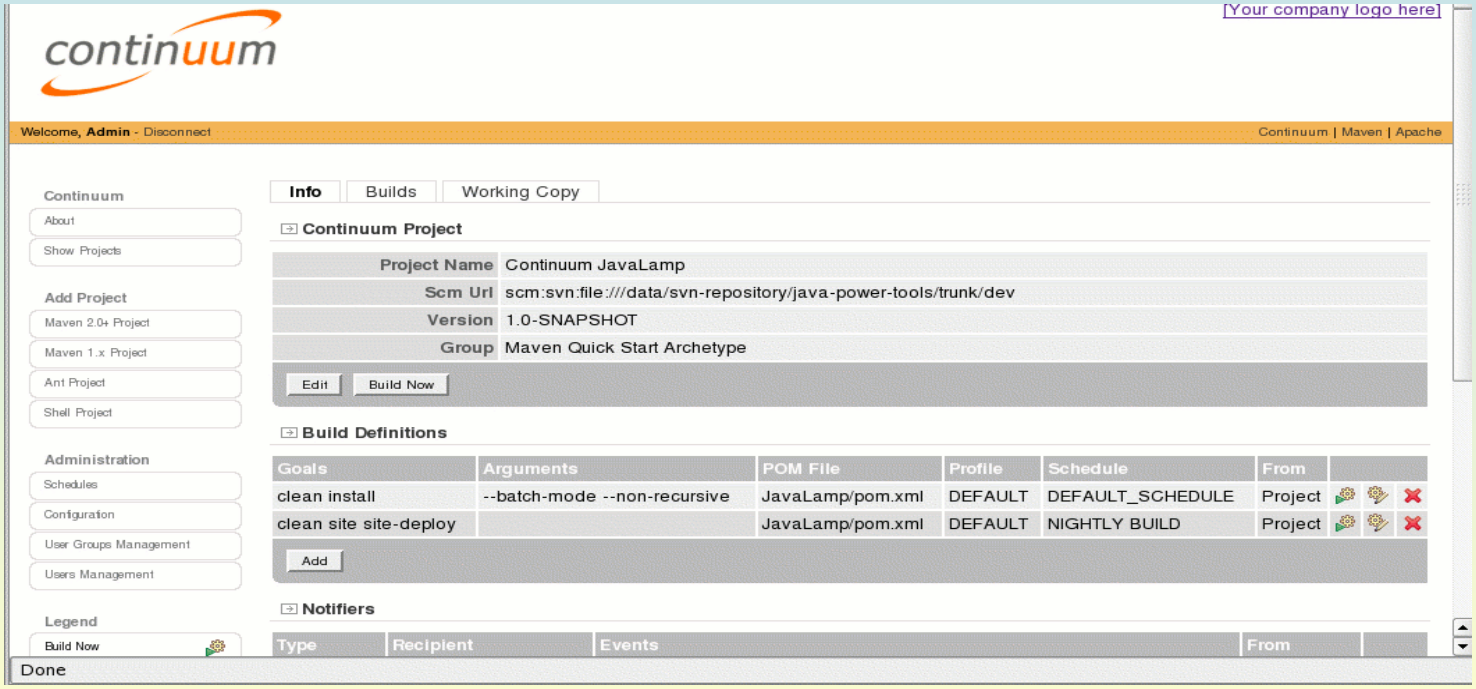
Continuous Integration tools

- LuntBuild
  - Web-based, easy to use
  - Manages artifacts, dependencies and labels



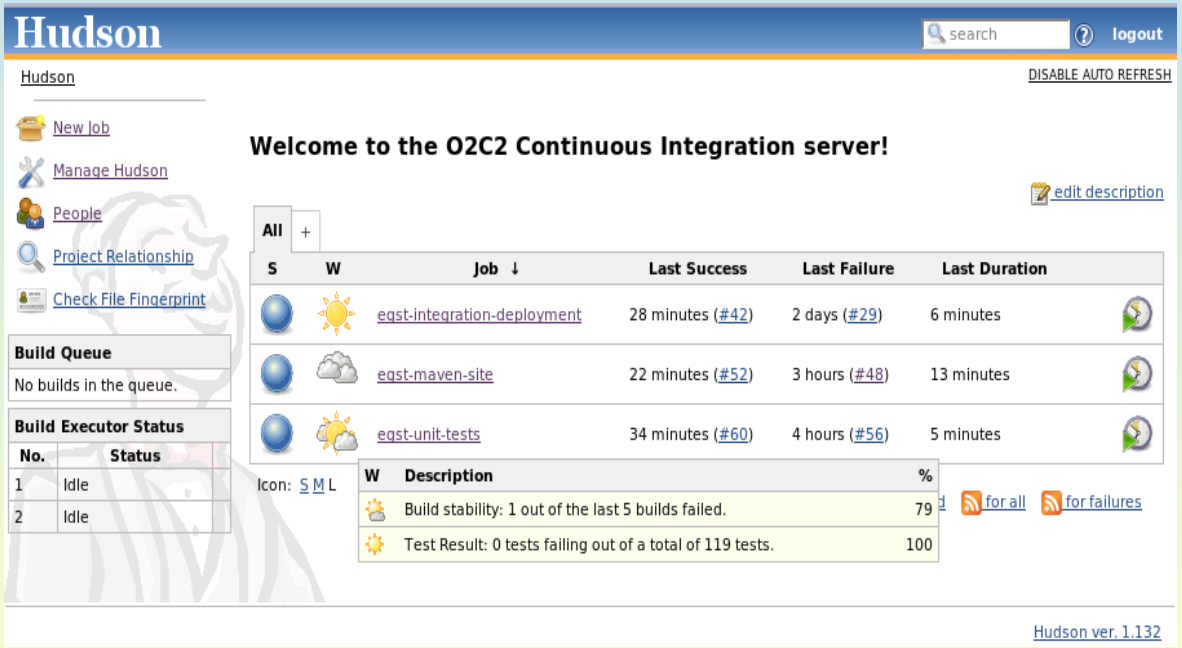
Continuous Integration tools

- Continuum
  - Web-based, easy to use
  - Not as feature-rich as the others
  - A bit clunky



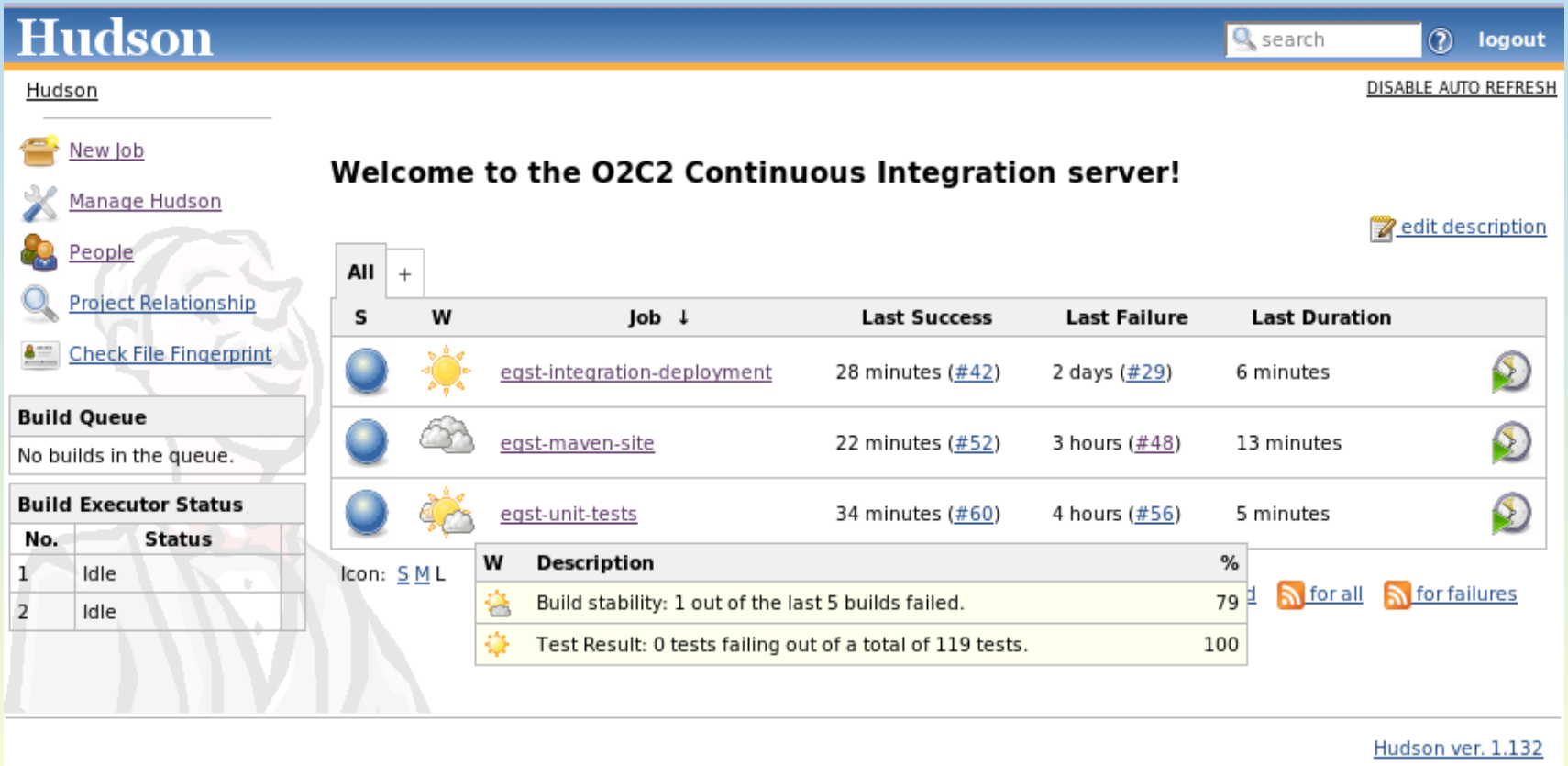
Continuous Integration tools

- Hudson
  - Web-based, easy to use
  - Lots of plugins
  - Cool user interface



The Hudson CI Server

- A project build status dashboard



The Hudson CI Server

- Automatic builds whenever code is committed

Hudson

search logout

Hudson

DISABLE AUTO REFRESH

New Job
Manage Hudson
People
Project Relationship
Check File Fingerprint

Build Queue

No builds in the queue.

Build Executor Status

No.	Status
1	Idle
2	Building egst-unit-tests #61

Welcome to the O2C2 Continuous Integration server!

edit description

All +

S	W	Job ↓	Last Success	Last Failure	Last Duration
		egst-integration-deployment	3 hours (#42)	3 days (#29)	6 minutes
		egst-maven-site	2 hours (#53)	5 hours (#48)	11 minutes
		egst-unit-tests	3 hours (#60)	6 hours (#56)	5 minutes

Icon: S M L

Legend for all for failures

The Hudson CI Server

- Notification by email
  - When a build fails
  - When it is fixed

http://gwweb.ird.govt.nz - Novell WebAccess - Mozilla Firefox

File Edit View History Bookmarks Tools Help

Mail Message

Close Previous Next Forward Reply to Sender Reply All Move Delete Read Lat

From: Hudson Build Server <hudson@ird.govt.nz>
To: "leah.xu@ird.govt.nz".INTERNET.IRDOM, "myles.johnson@ird.govt.nz".INTERNET.IRDOM, "prestine.herridge@ird.govt.nz".INTERNET.IRDOM, "john.smart@ird.govt.nz".INTERNET.IRDOM
Date: Tuesday - September 18, 2007 9:46 AM
Subject: Build failed in Hudson: egst-unit-tests #41
Mime.822 (6531 bytes) [View] [Save As]

See http://facade1.ird.govt.nz:9090/hudson/job/egst-unit-tests/41/changes

Changes:

[John Smart] Broke the build on purpose for testing purposes...

---
started
[egst-unit-tests] $ cvs -Q -z3 -d :pserver:johnsm@cvs.dotir:2401/cvsroot/web co -P -d workspace -D "Mond
2007 9:43:47 PM UTC" egst/egst-web
$ computing changelog
cvs log: Logging .
cvs log: Logging src
cvs log: Logging src/main
cvs log: Logging src/main/config
cvs log: Logging src/main/java
cvs log: Logging src/main/java/nz
cvs log: Logging src/main/java/nz/govt
cvs log: Logging src/main/java/nz/govt/ird

Done

egst-unit-tests #41 Changes [Hudson] - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://facade1.ird.govt.nz:9090/hudso

BBC GroupWise Hudson Artifactory Yahoo mail localhost:8081

Hudson search login

Hudson » egst-unit-tests » #41 DISABLE AUTO REFRESH

Back to Project
Status
Changes
Console Output
Previous Build
Next Build

Changes

Summary

1. Broke the build on purpose for testing purposes...

John Smart:

Broke the build on purpose for testing purposes...

1.2 src/main/java/nz/govt/ird/gst/web/ManageDatabaseCommand.java

Hudson ver. 1.132

Done

The Hudson CI Server

- Keeps a history of build results and artifacts

Hudson

search logout

Hudson » egst-integration-deployment

DISABLE AUTO REFRESH

Back to Dashboard
Status
Changes
Workspace
Build Now
Delete Project
Configure

Build History (trend)

#43 Sep 21, 2007 4:21:27 PM
#42 Sep 21, 2007 1:16:19 PM
#41 Sep 21, 2007 12:58:22 PM
#40 Sep 21, 2007 10:39:20 AM
#39 Sep 20, 2007 2:22:30 PM
#38 Sep 20, 2007 1:58:28 PM
#37 Sep 20, 2007 10:18:28 AM
#36 Sep 20, 2007 9:59:11 AM
#35 Sep 19, 2007 5:14:22 PM

Project egst-integration-deployment

add description

Workspace

Latest Artifacts

- egst.war

Recent Changes

Test Result Trend

count: 0, 20, 40, 60, 80, 100, 120

#1 #3 #5 #7 #9 #11 #13 #15 #17 #19 #21 #23 #25 #27 #29 #31 #33 #35 #37 #39 #41

(just show failures) enlarge

Upstream Projects

egst-unit-tests

Permalinks

- Last build (#43), 1 minute ago
- Last stable build (#42), 3 hours ago
- Last successful build (#42), 3 hours ago
- Last failed build (#29), 3 days ago

The Hudson CI Server

- View changes in a particular build
- Tag a build in CVS

Hudson

search logout

Hudson » egst-integration-deployment » #28

DISABLE AUTO REFRESH

Back to Project
Status
Changes
Console Output
Tag this build
Test Result
Previous Build
Next Build

Changes

Summary

1. Added prov tax codes.
2. Added prov tax codes.
3. Added prov tax codes.
4. Added prov tax codes.

leahxu:

Added prov tax codes.

1.15	src/test/java/nz/govt/ird/gst/dao/impl/GstTestConstants.java
1.14	src/test/java/nz/govt/ird/gst/dao/impl/GstReturnDAOImplTest.java

leahxu:

Added prov tax codes.

1.1	src/main/webapp/WEB-INF/jsp/tiles/provTaxPayment.jsp
1.1	src/main/webapp/WEB-INF/jsp/tiles/reviewProvTaxReturn.jsp
1.1	src/main/webapp/WEB-INF/tags/returnProfileClientDetailsProvCategory.tag
1.11	src/main/webapp/WEB-INF/tiles-defs.xml
1.9	src/main/webapp/WEB-INF/flows/gstreturn-flow-context.xml

Types of tools

- Categories of tools
  - Build Process tools
  - Version Control tools
  - Continuous Integration tools
  - **Quality Metrics and code audit tools**
  - Unit Testing and Test Coverage tools
  - Integration, Functional, Load and Performance Testing tools
  - Technical Documentation tools

Quality Metrics

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

Quality Metrics

- 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

Quality Metrics tools

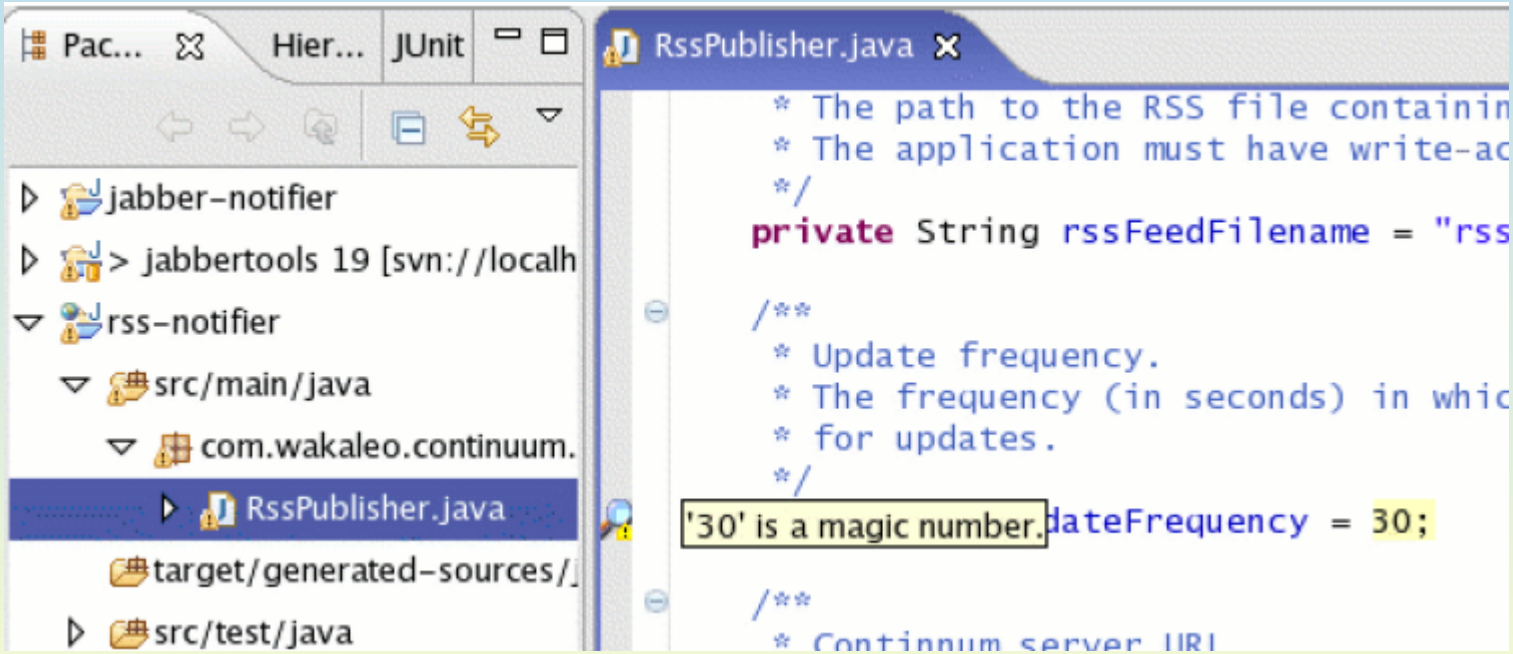
- We use three complementary tools
  - **Checkstyle** – coding standards
  - **PMD** – best practices
  - **FindBugs** – potential bugs

Quality Metrics

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

Quality Metrics

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



Quality Metrics

- **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

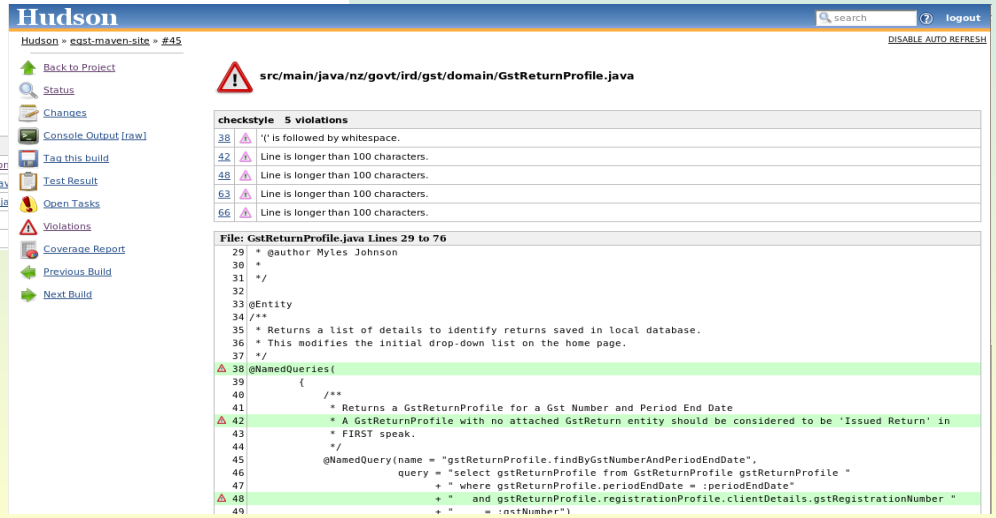
nz/govt/ird/gst/backend/UserProfileBackendService.java

Violation	Message	Line
	Line is longer than 90 characters.	29

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

Quality Metrics

- **Checkstyle** - Enforce coding standards
  - Continuous Integration server

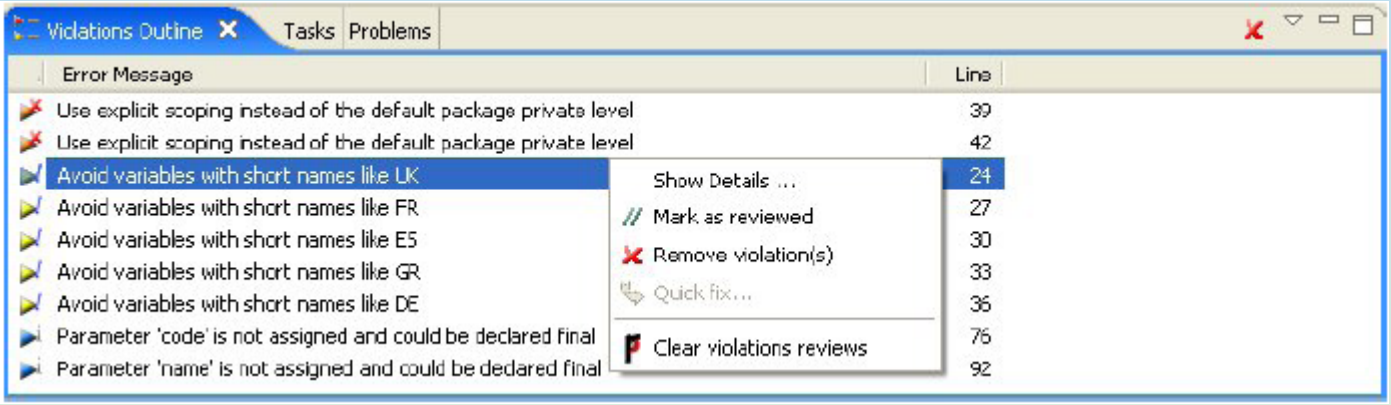


Quality Metrics

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

Quality Metrics

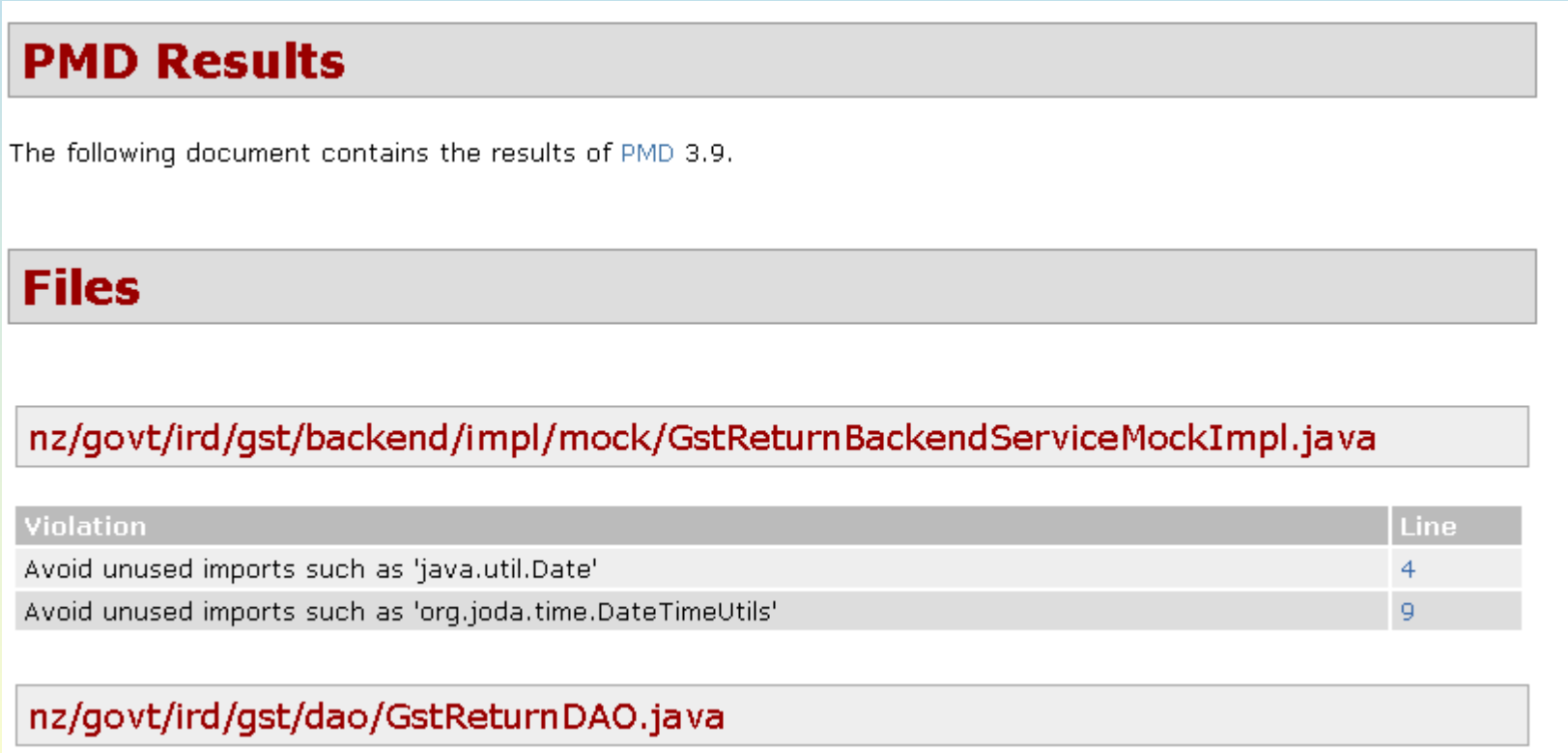
- **PMD** – Best practices
  - Eclipse plugin



DON'T SHOOT THE MESSENGER

Quality Metrics

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



PMD Results

The following document contains the results of PMD 3.9.

Files

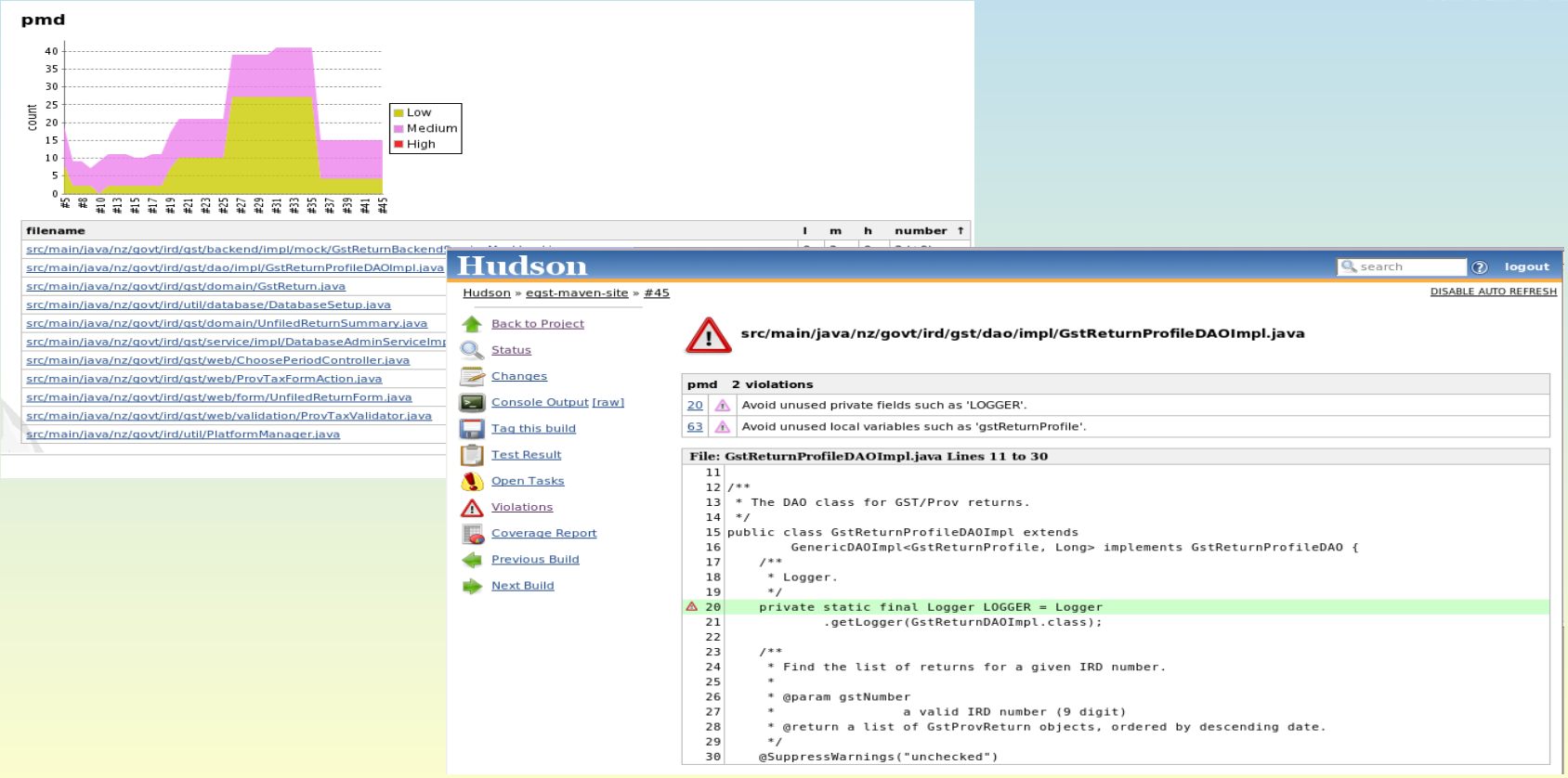
nz/govt/ird/gst/backend/impl/mock/GstReturnBackendServiceMockImpl.java

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

Quality Metrics

- **PMD** – Best practices
  - Continuous Integration server



Quality Metrics

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

Quality Metrics

- **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 | Tasks | Properties | Servers | Snippets | Console | Progress | Search | Database Explorer | JUnit

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.Libra	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.Application	ApplicationBasePag...	library-symbols-web/src/main/java/nz/govt/natlib/sym...	line 226

Quality Metrics

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

Files

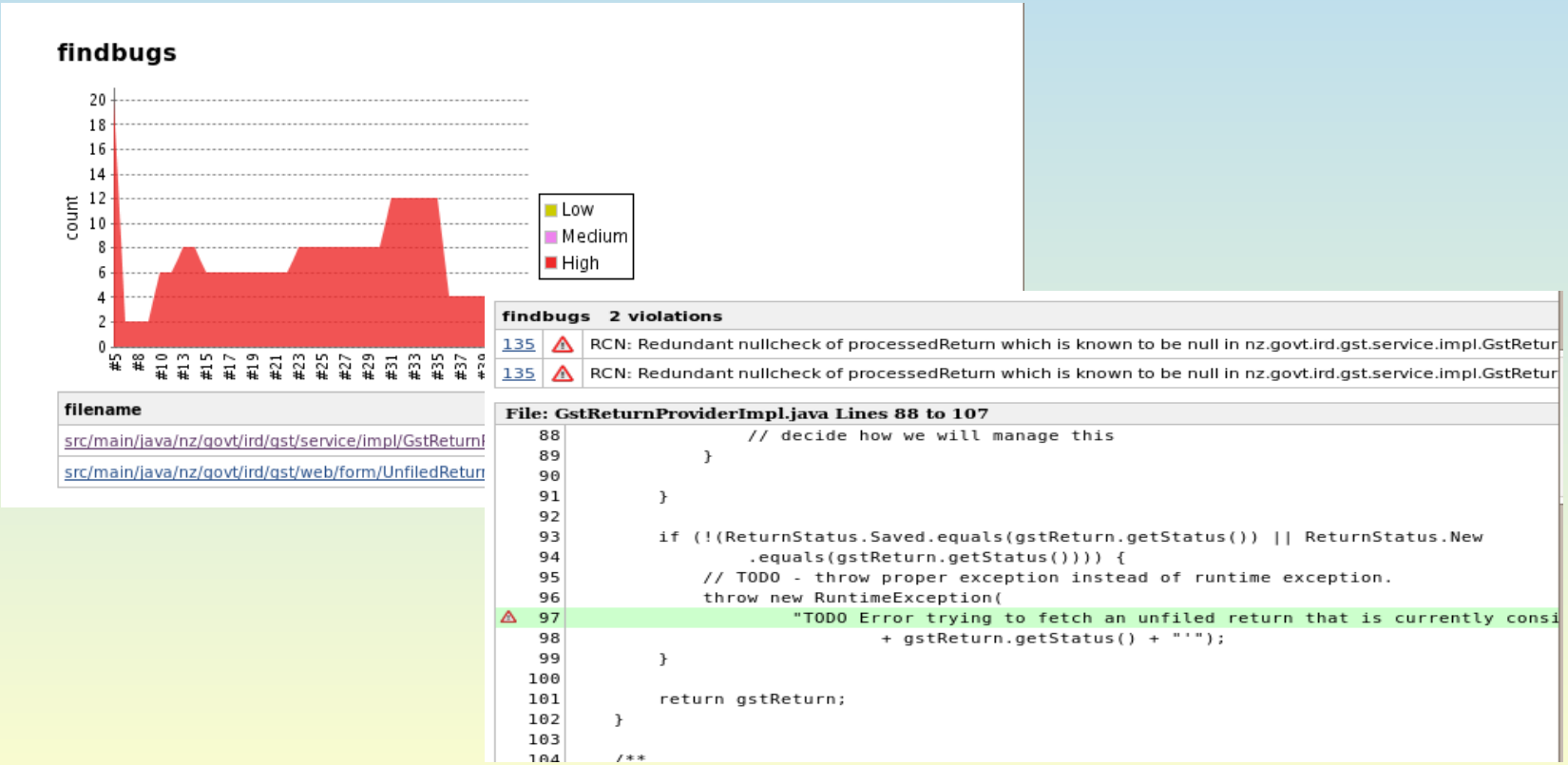
Class	Bugs
nz.govt.ird.gst.domain.HistoryDetailEntry	4
nz.govt.ird.gst.refdomain.RefGSTProfile	3
nz.govt.ird.gst.refdomain.RefGstProvProfile	3
nz.govt.ird.gst.refdomain.RefGstProvReturn	4
nz.govt.ird.gst.refdomain.RefOldGstProvReturn	4

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

Quality Metrics

- **FindBugs** – Potential defects
  - Continuous Integration server



Types of tools

- Categories of tools
  - Build Process tools
  - Version Control tools
  - Continuous Integration tools
  - Quality Metrics and code audit tools
  - **Unit Testing and Test Coverage tools**
  - Integration, Functional, Load and Performance Testing tools
  - Technical Documentation tools

Unit Testing

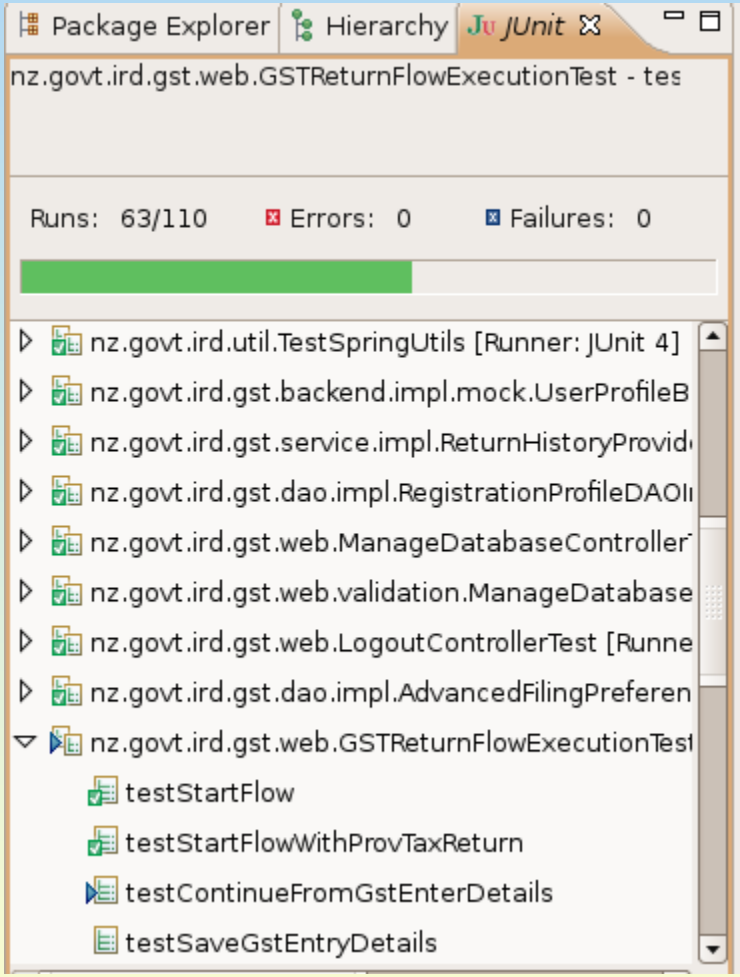
- A recommended best practice
- Unit testing results in:
  - More reliable code
  - Better designed code
  - More flexible code

Unit Testing Tools

- Main tools
  - JUnit 3.x – *the original, with lots of extensions*
  - JUnit 4 – *annotations and stuff*
  - TestNG – *the cutting edge*

Unit Testing in Eclipse

- JUnit Eclipse integration



Test reports

- Maven test reports

Summary

[Summary][Package List][Test Cases]

Tests	Errors	Failures	Skipped	Success Rate	Time
114	0	0	0	100%	17.617

Note: failures are anticipated and checked for with assertions while errors are unanticipated.

Package List

[Summary][Package List][Test Cases]

Package	Tests	Errors	Failures	Skipped	Success Rate	Time
nz.govt.ird.gst.dao.impl	29	0	0	0	100%	2.262
nz.govt.ird.util	16	0	0	0	100%	7.861
nz.govt.ird.gst.web	28	0	0	0	100%	4.463
nz.govt.ird.gst.web.propertyeditors	2	0	0	0		
nz.govt.ird.gst.web.form	3	0	0	0		
nz.govt.ird.gst.web.validation	5	0	0	0		
nz.govt.ird.gst.domain	7	0	0	0		
nz.govt.ird.util.database	2	0	0	0		
nz.govt.ird.gst.service.impl	18	0	0	0		
nz.govt.ird.gst.backend.impl.mock	4	0	0	0		

Note: package statistics are not computed recursively, they only sum up all of its test

nz.govt.ird.gst.dao.impl

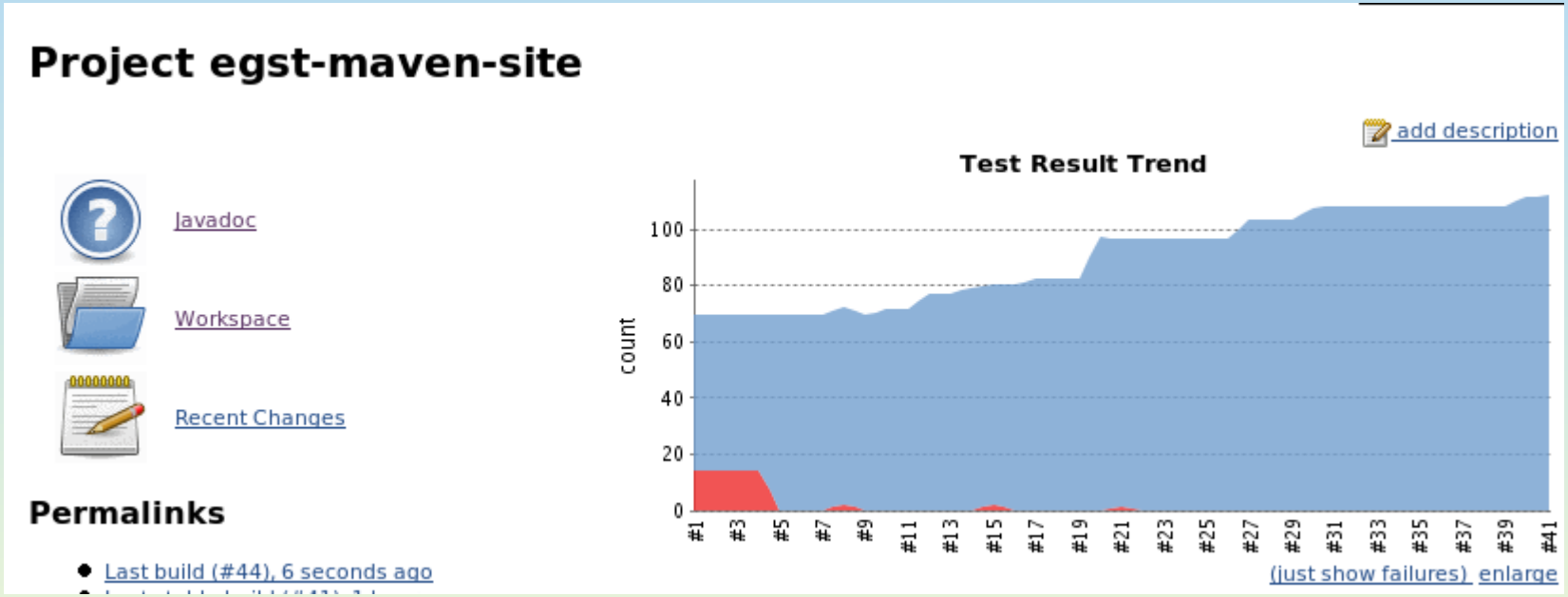
	Class	Tests	Errors	Failures	Ski
	ClientDetailsDAOImplTest	3	0	0	0
	GstReturnDAOImplTest	5	0	0	0
	FilingPreferenceDAOImplIntegrationTest	5	0	0	0
	AdvancedFilingPreferenceDAOImplTest	1	0	0	0

PlatformManagerTest

	testSetEnvironmentWithSystemProperty	0.002
	testExtendedProdValue	0.003
	testDevValue	0.004
	testDefaultDevValue	0.002
	testIntegrationValue	0.003
	testTestValue	0.003
	testUATValue	0.003
	testInvalidValue	0.004

Test result trends

- Hudson dashboard



Test Coverage

- Test Coverage:
  - Are unit tests being written for all classes?
  - How much code is really executed by the unit tests?
  - Un-executed code will contain bugs

- Tool used:
  - Cobertura

Test Coverage Reports

Maven coverage reports:

Packages

All
nz.govt.ird.gst.backend
nz.govt.ird.gst.backend.i
nz.govt.ird.gst.backend.i
nz.govt.ird.gst.dao

All Packages

Classes

AccountingBasis *(100%)*
BackToLastViewStateFlo
ChoosePeriodController
ChooseProfileCommand
ChooseProfileController
ClientDetails *(73%)*
ClientDetailsDAO *(N/A)*
ClientDetailsDAOImpl *(1*
DatabaseAdminService *(*
DatabaseAdminServiceIm
DatabaseSetup *(79%)*
DateTimeEditor *(70%)*

Coverage Report - All Packages

Package	# Classes	Line Coverage	Branch Coverage	Complexity
All Packages	76	75%	82%	1.286
nz.govt.ird.gst.backend	2	N/A	N/A	1
nz.govt.ird.gst.backend.impl	2	22%	N/A	1
nz.govt.ird.gst.backend.impl.mock	2	67%	83%	0
nz.govt.ird.gst.dao	7	N/A	N/A	0
nz.govt.ird.gst.dao.impl	7	87%	89%	0
nz.govt.ird.gst.domain	18	73%	67%	1
nz.govt.ird.gst.service	5	N/A	N/A	1
nz.govt.ird.gst.service.impl	4	80%	94%	0
nz.govt.ird.gst.service.impl.mock	2	100%	N/A	0
nz.govt.ird.gst.web	14	64%	58%	1.6
nz.govt.ird.gst.web.form	1	90%	100%	0
nz.govt.ird.gst.web.propertyeditors	2	74%	67%	0
nz.govt.ird.gst.web.validation	3	82%	93%	0
nz.govt.ird.gst.web.webflow.listeners	1	74%	92%	0
nz.govt.ird.util	3	89%	93%	1
nz.govt.ird.util.database	3	80%	100%	1.667

Report generated by Cobertura 1.7 on 20/09/07 16:38.

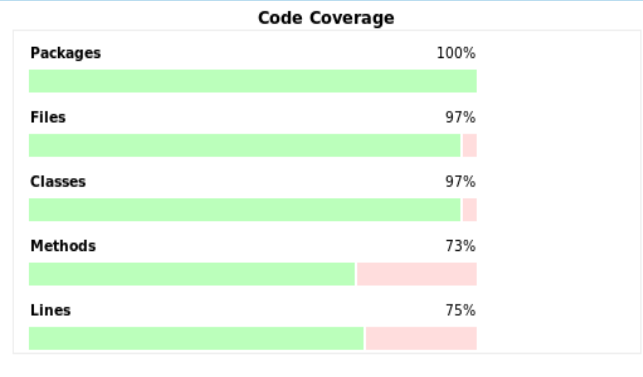
Coverage Report - nz.govt.ird.gst.backend.impl.GstReturnBackendServiceImpl

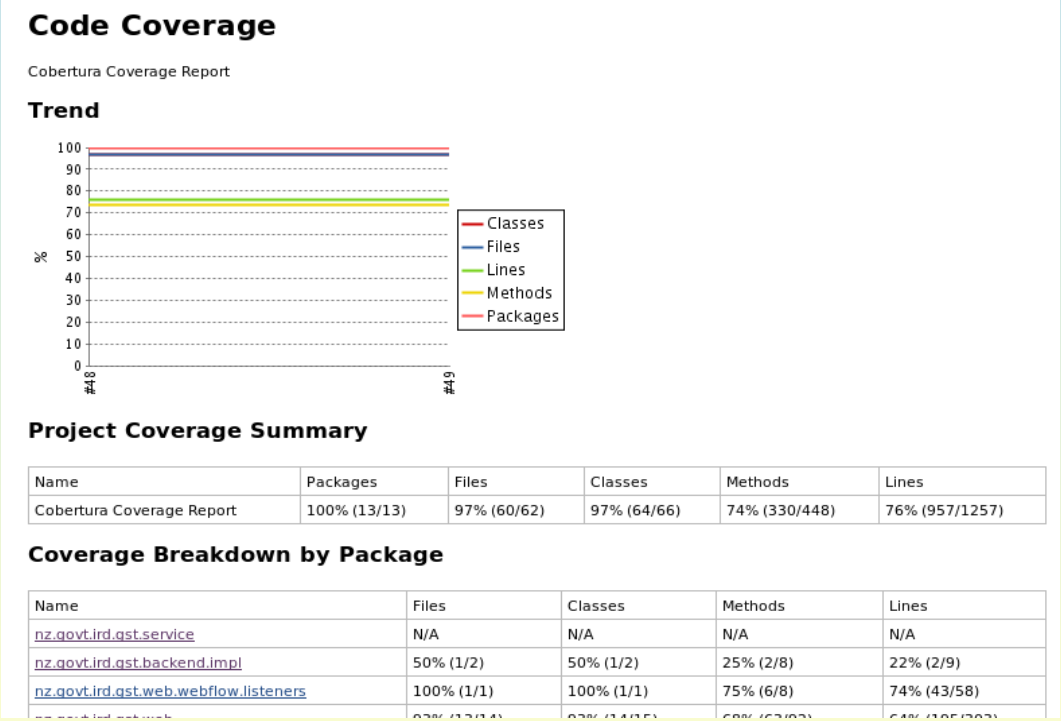
Classes in this File	Line Coverage	Branch Coverage	Complexity
GstReturnBackendServiceImpl	33%	N/A	0

```
1      package nz.govt.ird.gst.backend.impl;
2
3      import java.util.List;
4
5      import nz.govt.ird.gst.backend.GstReturnBackendService;
6      import nz.govt.ird.gst.backend.ServiceUnavailableException;
7      import nz.govt.ird.gst.backend.UserProfileBackendService;
8      import nz.govt.ird.gst.domain.GstReturnProfile;
9      import nz.govt.ird.gst.domain.UnfiledReturnSummary;
10
11     import org.joda.time.DateMidnight;
12
13     /**
14      * Provides Backend services for the GstReturn GstReturnSummary domain objects.
15      *
16      * Will access First via EAI
17      *
18      * @author MXJ6
19      *
20      */
21 1   public class GstReturnBackendServiceImpl implements GstReturnBackendService {
22
23         private UserProfileBackendService userProfileService;
24
25         public UserProfileBackendService getUserProfileService() {
26 0            return userProfileService;
27         }
28
29         public void setUserProfileService(
30                 UserProfileBackendService userProfileService) {
31 0            this.userProfileService = userProfileService;
32 0        }
33
34         /**
```

Test Coverage

- Hudson Dashboard





Code Coverage

Cobertura Coverage Report

Trend

Project Coverage Summary

Name	Packages	Files	Classes	Methods	Lines
Cobertura Coverage Report	100% (13/13)	97% (60/62)	97% (64/66)	74% (330/448)	76% (957/1257)

Coverage Breakdown by Package

Name	Files	Classes	Methods	Lines
nz.govt.ird.gst.service	N/A	N/A	N/A	N/A
nz.govt.ird.gst.backend.impl	50% (1/2)	50% (1/2)	25% (2/8)	22% (2/9)
nz.govt.ird.gst.web.webflow.listeners	100% (1/1)	100% (1/1)	75% (6/8)	74% (43/58)

Types of tools

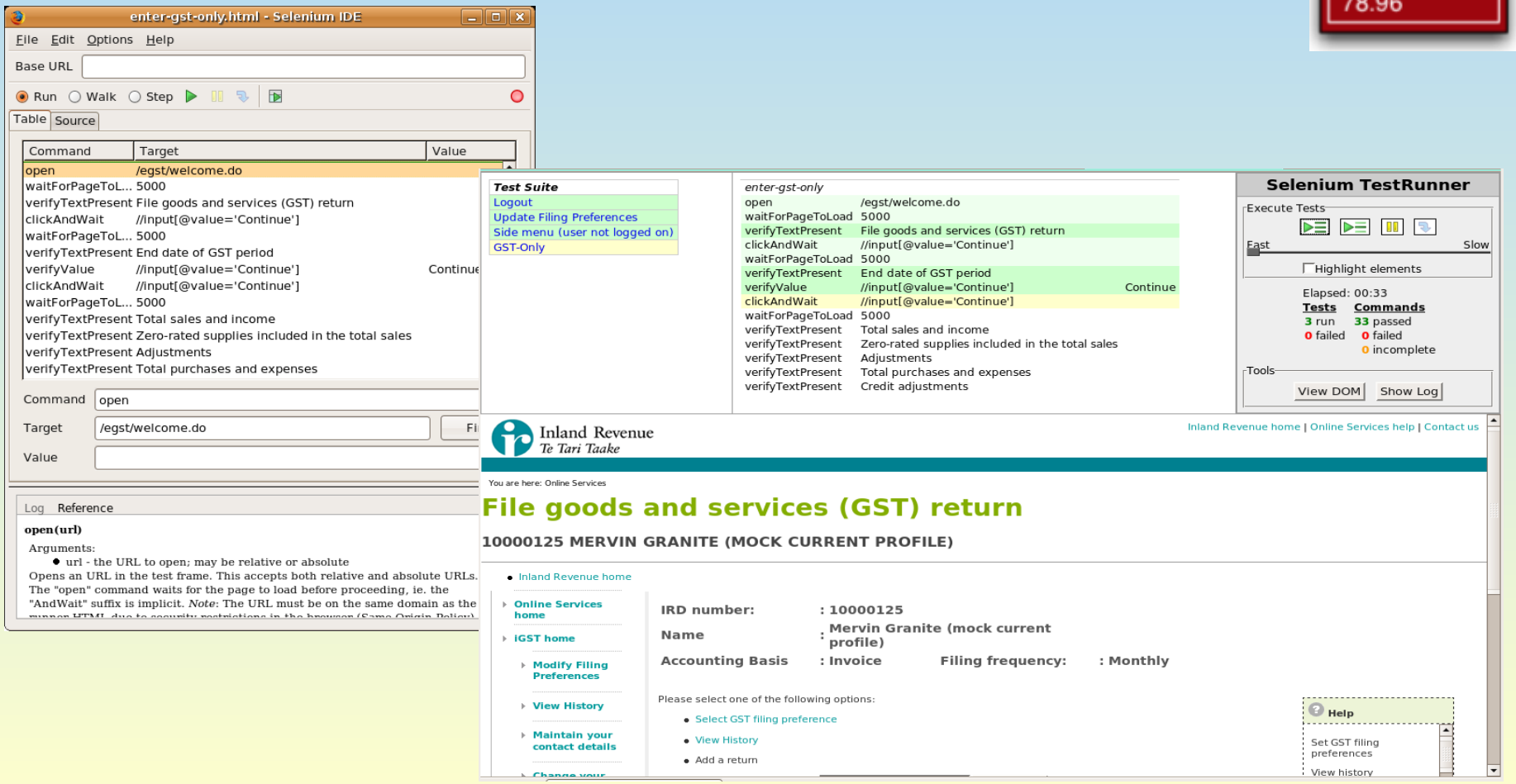
- Categories of tools
  - Build Process tools
  - Version Control tools
  - Continuous Integration tools
  - Quality Metrics and code audit tools
  - Unit Testing and Test Coverage tools
  - **Integration, Functional, Load and Performance Testing tools**
  - Technical Documentation tools

Integration, Functional, UI and Perf Tests

- Cover all your bases
  - DBUnit - *database testing*
  - Spring testing framework (Spring MVC) or StrutsTestCase
  - Eclipse Profiling (TPTP) - *profiling*
  - JMeter – *load and performance testing*
  - SoapUI – *web service testing*
  - Selenium – *web interface testing*
  - FEST – *Swing applications*
  - ...

User Interface Testing

- Automatic UI testing with Selenium:



Regression Testing

- All tests automatically run before each build:
  - Unit tests
  - Integration tests
  - User Interface tests

Types of tools

- Categories of tools
  - Build Process tools
  - Version Control tools
  - Continuous Integration tools
  - Quality Metrics and code audit tools
  - Unit Testing and Test Coverage tools
  - Integration, Functional, Load and Performance Testing tools
  - **Technical Documentation tools**

Technical Documentation

- Human-written documentation:
  - (potentially) highest possible quality
- BUT
  - Often incomplete
  - Hard to keep up-to-date

Technical Documentation

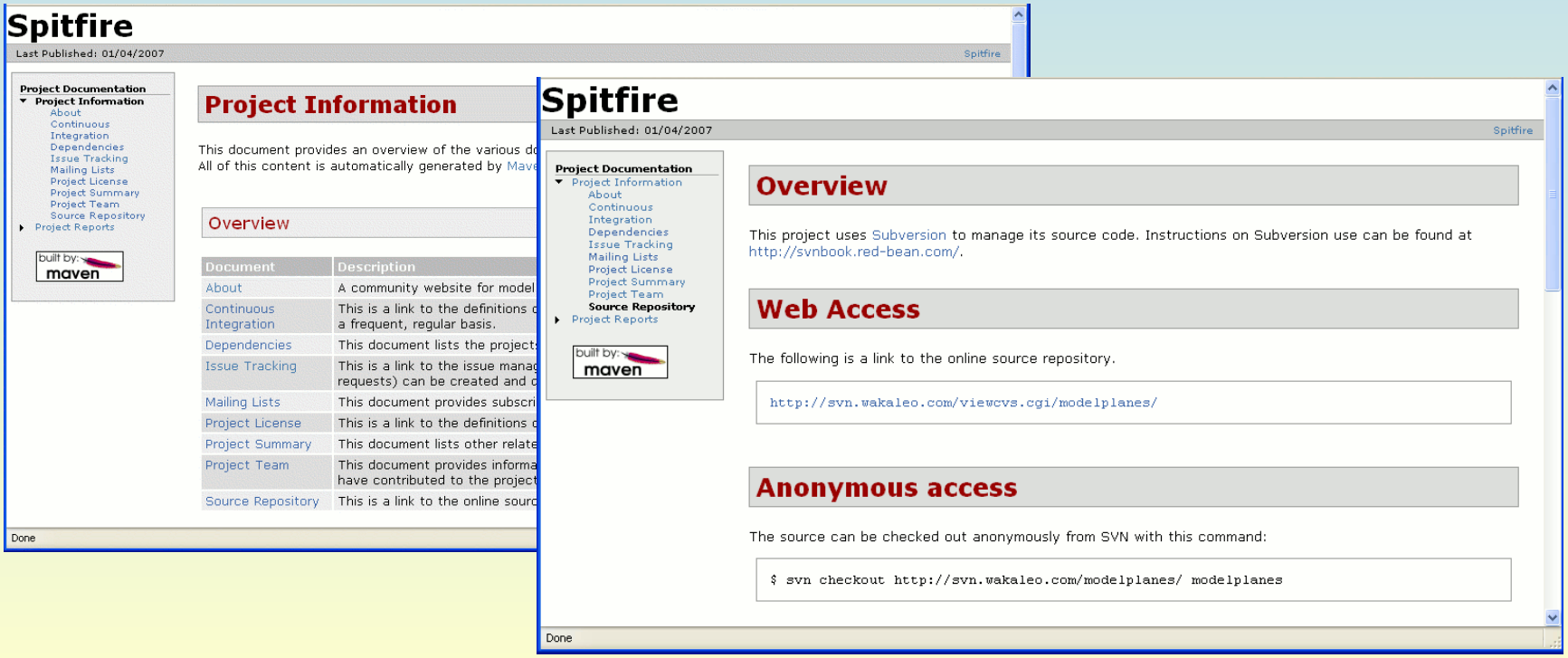
- Automatically-generated documentation:
  - Complete
  - Always up-to-date
  - Cheap to produce
- BUT
  - Lacks “higher vision”

Technical Documentation

- Many tools available:
  - Maven web site – *lots of reports and project info*
  - SchemaSpy – *database schemas*
  - Doxygen – *UML documentation*
  - UmlGraph – *UML-enabled Javadoc*

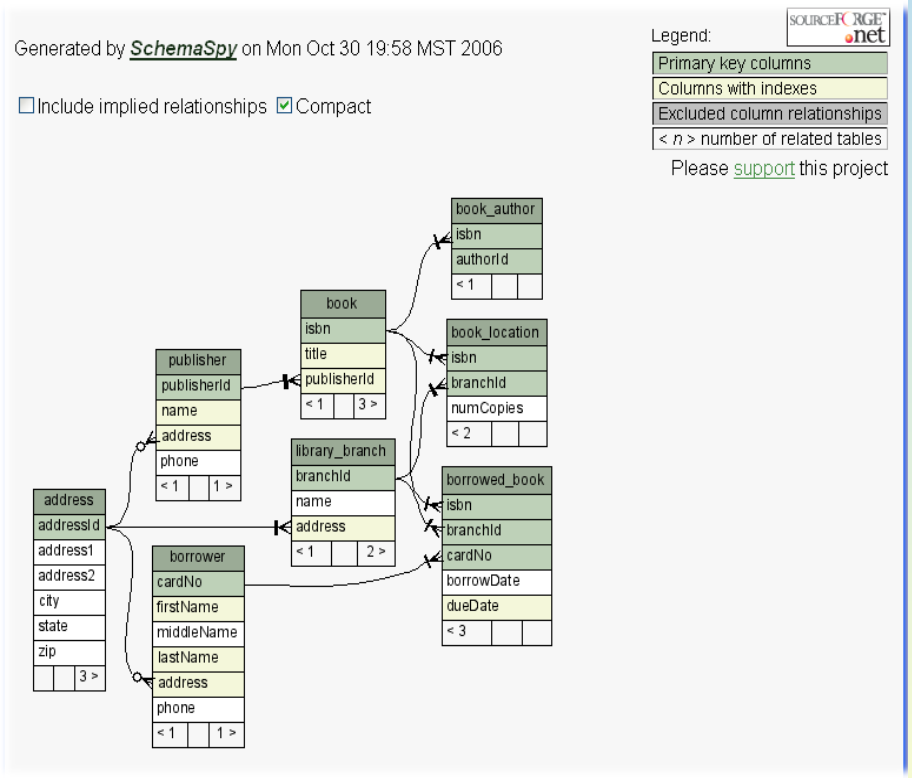
Technical Documentation

- Maven web site
  - Rich reporting capabilities
  - General project information
  - Detailed, customizable reports



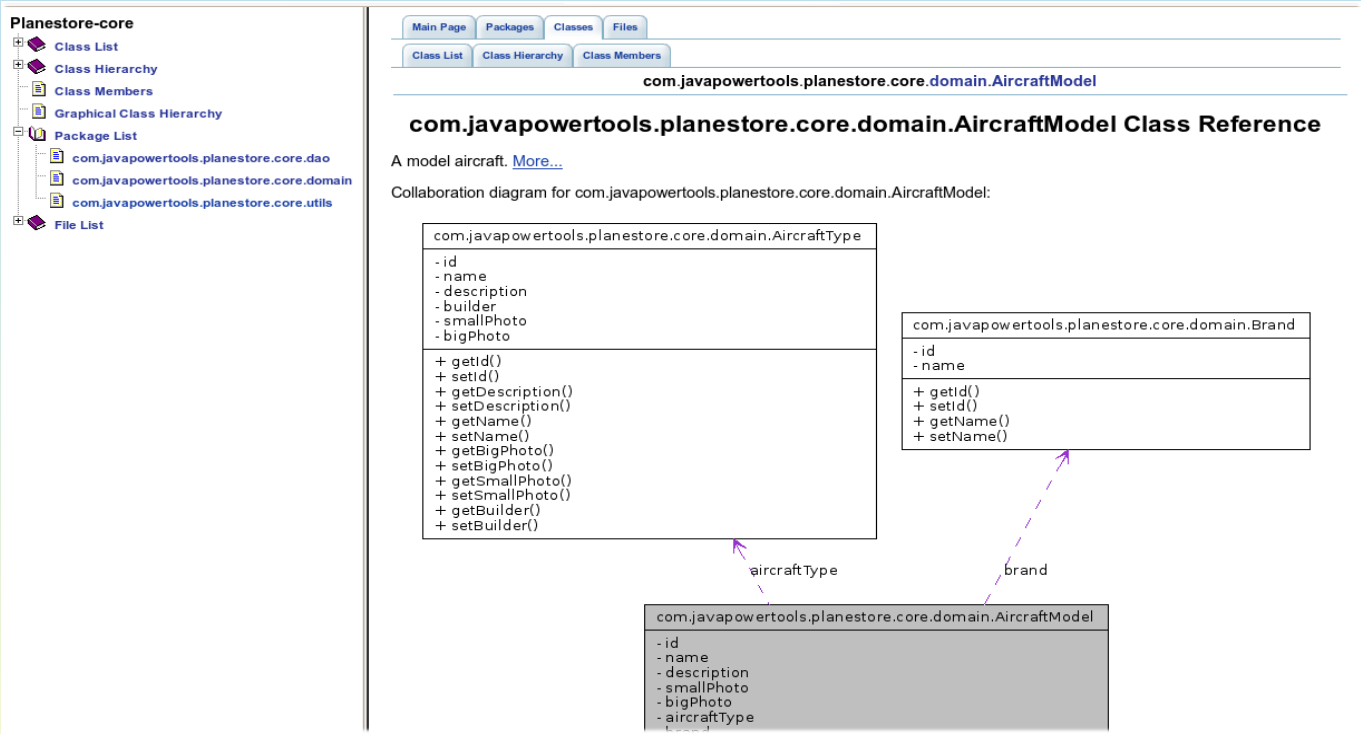
Technical Documentation

- SchemaSpy
  - Database schemas



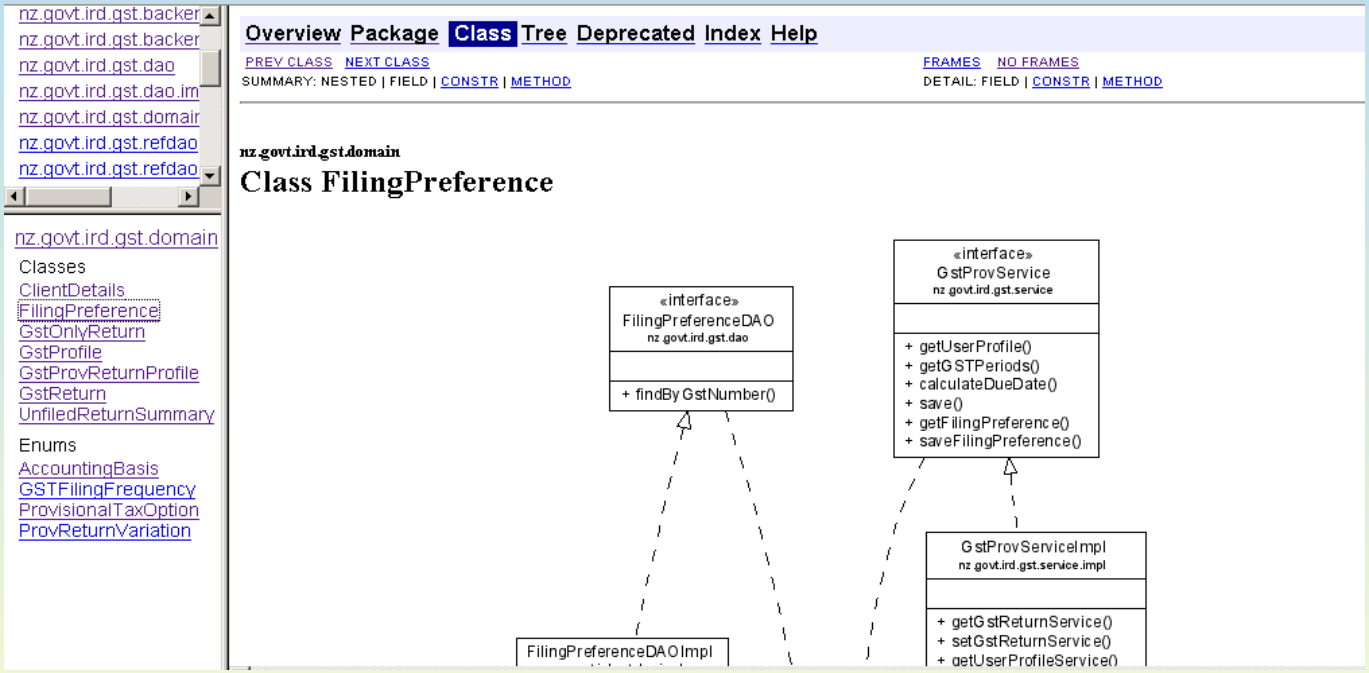
Technical Documentation

- DOxygen
  - UML-embedded documentation



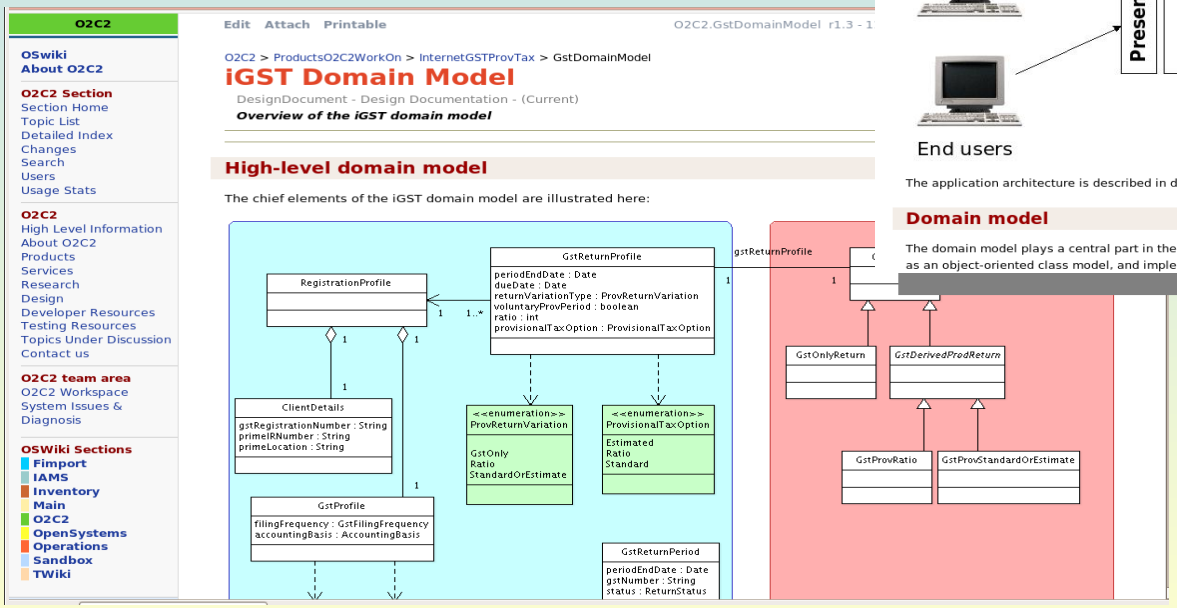
Technical Documentation

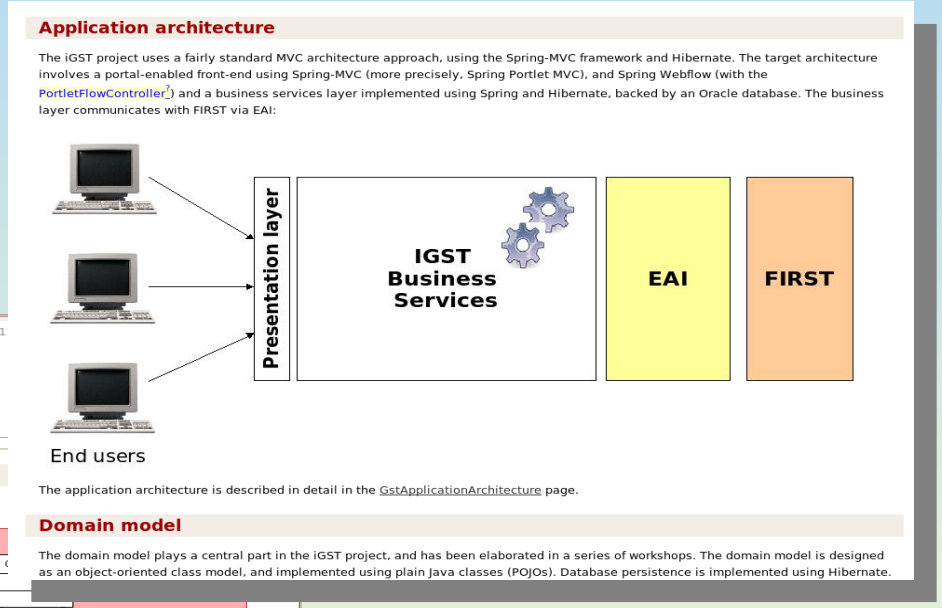
- UMLGraph
  - Javadoc with UML diagrams



Technical Documentation

- Also some human-written documentation – project Wiki
  - Architecture vision
  - High-level design
  - Collaborative





Overview

A Standard Build Process

Dependency Management

Continuous Integration

Continuous Testing

Continuous Quality

Continuous Reporting

Continuous Documentation

Questions?