



# ***E-vote 2011***

---

**SSA-U Appendix 5**

**Testing and Approval**

**Project: E-vote 2011**

---



## CONTENT

<b>1. TEST GUIDELINES</b>	<b>3</b>
1.1. Introduction	3
1.2. Scope of each test	3
1.3. Requirements and criteria for implementation	6
1.3.1. Start criteria	6
1.3.2. Requirements for ending the test	6
1.3.3. Test execution	6
1.3.4. Classification of findings	6
1.3.5. Criteria for Acceptance or Rejection	7
1.3.6. Abandonment	7
1.3.7. Recurrence	7
1.4. Test documentation	8
1.5. Test Plan	8
1.6. Test log	9
1.7. Test environment	10
<b>2. ACCEPTANCE TEST AND APPROVAL PERIOD</b>	<b>10</b>
2.1. Introduction	10
2.2. Requirements for execution of acceptance test	10
2.3. Precondition before installation for acceptance test	11
2.4. Strategy	11
2.5. Stop and restart of Acceptance test	11
2.6. Acceptance of the delivery	11
2.7. Approval period	12
2.7.1. Introduction	12
2.7.2. Final acceptance of the delivery	12
2.8. Error reporting and management	12
2.9. Registration of errors	13
<b>3. TESTING TOOLS AND OTHER SUPPORTING TOOLS</b>	<b>14</b>
<b>4. QUALITY ASSURANCE, ERROR HANDLING AND CONFIGURATION CONTROL</b>	<b>15</b>
4.1. Quality Management	15
4.2. QA Assurance and Control	16
4.2.1. Implementation	16
4.2.2. Regression tests	17
4.2.3. Usability and accessibility testing:	18

---



4.2.4.	Security testing	18
4.2.5.	Volume and performance testing	18
4.3.	Releasing	19
4.4.	Configuration Management	21
4.5.	Change Management	21
4.6.	Tools Used	21
4.6.1.	Bug Tracking – Bugzilla	22
4.6.2.	Test Management – Hudson Continuous Integration Server	22
4.6.3.	Build Management – Maven	23
4.6.4.	Configuration/Version Management - Subversion	25
4.6.5.	Load testing – JMeter	26
4.6.6.	Rational Purify	27
4.6.7.	FindBugs	27
4.6.8.	Cobertura	28
4.6.9.	Checkstyle	29
4.6.10.	Jemmy	30
4.6.11.	WAVE	30
4.6.12.	TAW	30
4.6.13.	W3C Markup Validation Service	31
4.6.14.	JUnit	31
4.6.15.	JAWS	32
4.6.16.	Paros	33
4.6.17.	WebScarab	33



## 1. Test Guidelines

### 1.1. Introduction

This section describes the guidelines for tests of the system delivered.

Detailed test plans for each test cycle will be developed during the project. Each detailed test plan will cover the following topics:

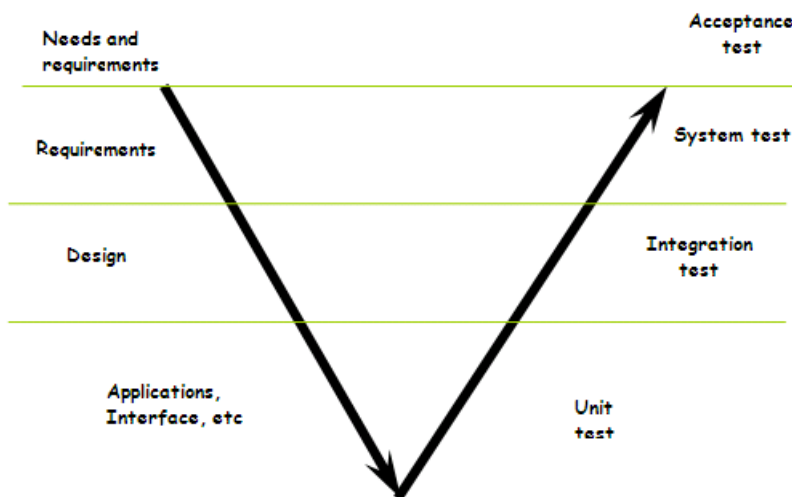
- The overall intention for the test.
- A description of all tests that should be performed.
- Planning of the tests.
- How each test should be performed as well as the expected results.
- Who will participate in the test.
- How the test results will be logged and how any deviations from the expected result should be handled.

### 1.2. Scope of each test

#### General

The purpose of all test activities in the project is to identify as many errors as possible in an efficient manner, as early as possible, and within the time that is available. At each level the goal is to correct errors before the next type of test is performed. Through these test iterations, an accepted and approved high quality solution will be obtained.

The figure below shows the principle for how testing is conducted in the project with the types of tests to be conducted in the project.





The types of tests that will be implemented in the project are listed in the table below. The table shows which tests are performed in which phases and who will be responsible for test planning and execution.

Test type	Abbreviation	Responsible	Specification phase	Development phase	Acceptance and Approval period
Module- / Unit test	MT	Vendor		X	
Integration test	IT	Vendor		X	
System test	ST	Vendor		X	
Performance test	PT	Vendor		X	
Deployment and Installation test	DT	Vendor / Hosting provider		X	X
Acceptance test	AT	Customer	X	X	X
Continuous test	CT	Customer / Hosting provider			X



The table below shows the different areas that are covered in each test:

Characteristic	MT	IT	ST	PT	DT	AT	CT
<b>Functionality</b>							
Functionality described in the requirement specification			X			X	
Manual routines			X			X	X
Correctness	X	X	X			X	X
Auditability			X			X	
Security			X			X	X
Interaction with other systems		X	X		X	X	X
Accessibility			X			X	
Compatibility			X		X	X	
<b>Reliability</b>							
Maturity for operations			X	X		X	X
Robustness	X	X	X			X	X
Restore after system stop			X			X	X
Stress tolerance			X	X		X	
Simultaneity			X	X		X	
Availability			X	X		X	X
<b>Performance</b>							
Response time for end user functions			X	X		X	X
Processing time for heavy batches				X		X	
The usage of machine resources				X		X	
Usage if other type of resources				X		X	
<b>Maintainability</b>							
Analyzable	X	X	X			X	X
Predictable			X	X		X	X
Ease of alteration			X			X	
Testability			X			X	
Completeness of documentation			X			X	
<b>Transferable</b>							
Adaptability			X			X	
Ease of installation			X		X	X	
Standardizing			X			X	X



### 1.3. Requirements and criteria for implementation

#### 1.3.1. Start criteria

Start criteria for all tests are:

- The test must be properly planned and the test plan must be approved.
- The previous test level must have been completed.
- Relevant test data must be available.
- The test environment must have been established including all interfaces.
- The resources needed to complete the test must have been allocated.

#### 1.3.2. Requirements for ending the test

The following general requirements must be met in order to end a test:

- All test cases and test objects must have been tested and documented, including test results. The term test object covers among others requirement, module, interface, function and subsystem depending on the level of the test.
- The predefined criteria for successful test have been met.

#### 1.3.3. Test execution

The following general requirements apply to the execution of each test cycle:

- The test must follow the test predefined test descriptions.
- All test cases must be covered and all findings and results must be properly recorded.
- All deviations must be located, evaluated and classified.

#### 1.3.4. Classification of findings

The following categories are used for classification of findings:

Level	Category	Description
A	Critical error	<ul style="list-style-type: none"> <li>- Error that results in the stoppage of the system, the loss of data, or in other functions that are of critical importance to the Customer not being delivered or not working as agreed.</li> <li>- The documentation being incomplete or misleading, and this resulting in the Customer being unable to use the system or material parts thereof.</li> </ul>



B	Serious error	<ul style="list-style-type: none"> <li>- Error that results in functions of importance to the Customer not working as described in the Agreement, and which it is time-consuming or costly to avoid.</li> <li>- The documentation being incomplete or misleading, and this resulting in the Customer being unable to use functions that are of importance to the Customer.</li> </ul>
C	Less serious error	<ul style="list-style-type: none"> <li>- Error that results in individual functions not working as intended, but which can be avoided with relative ease by the Customer.</li> <li>- The documentation being incomplete, imprecise or easily misunderstood.</li> </ul>

### 1.3.5. *Criteria for Acceptance or Rejection*

Each test plan shall include precise criteria for acceptance. The criteria should be like:

- All test cases / procedures shall be executed.
- The number of errors found shall be less than the defined maximum.
- The last test cycle shall contain no errors in category Critical or Serious.

### 1.3.6. *Abandonment*

A test can be stopped completely:

- If errors make it impossible to finish the execution of a test (e.g. application abortions, data errors).
- If the number of Critical or Serious errors are so high that there is no point carrying on.

The test can be paused:

- If there is a heavy backlog in the error corrections that prevent the test to be performed on the latest version of the function/subsystem.
- If serious deficiencies or errors in the test basis or test data is discovered.
- If an error is found that is critical and prevent other features to be tested.

### 1.3.7. *Recurrence*

The test manager will, in cooperation with project managers decide whether a test must be repeated or not. Different situations in which this must be considered are:

- Test data is not sufficient.
- Reported errors are not possible to reproduce.
- The number of findings is assumed too high.





It may also be required to perform a regression test if the System test has found errors in one set of modules and the bug-fixes is assumed to influence other modules. Regression testing is done automatically during the build process, as part of the standard Continuous Integration environment.

#### 1.4. Test documentation

The purpose of the test documentation is:

- It shall ensure that the test may be carried out in an effective way.
- It shall document the results from the tests and disseminate the knowledge achieved during the test.
- It shall document all test that have been performed and explicit explore the test that have not been performed, if any.

The contractor is responsible for all test documentation for the Module test, the Integration test, the System test, the Performance test as well as the Deployment and Installation test prior to customer acceptance test.

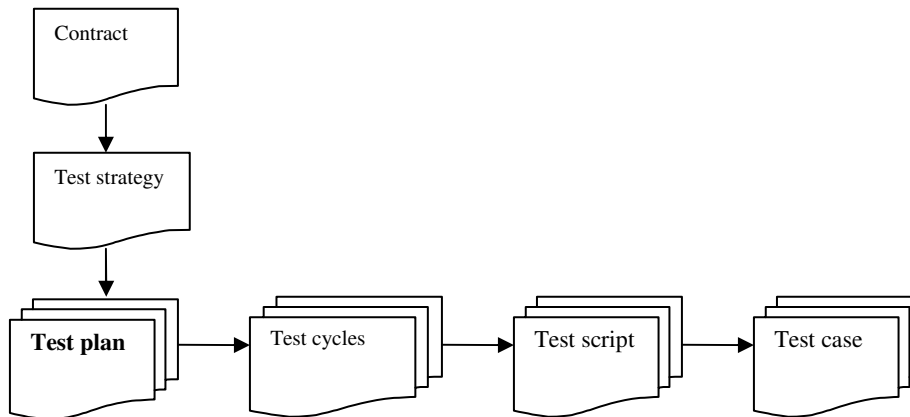
#### 1.5. Test Plan

The test plan shall support the planning and execution of each test. The main goal is that the test may be performed in an efficient way and that the needed resources (human resources, infrastructure, surrounding systems and test data) are allocated as expected. Below is a brief description of the elements in the test plan that should be developed for each test:

- Introduction.
- Deviation from overall test strategy (if any) and the reason for this.
- Which objects are to be tested.
- Which characteristics that are to be tested (refer the table in section 1.2).
- Any preconditions or assumptions.
- Acceptance criteria.
- Abandonment criteria.
- The type of test documentation to be produced.
- All activities that need to be carried out to perform the test.
- Any requirements for test infrastructure and basis software.
- Responsibility and roles.
- Manning and need for training.
- A risk analysis.
- A progress plan with milestones.



- An overall description of test procedures.



## 1.6. Test log

The test log is recorded during the execution of the test. It is a consecutive report of all activities and findings. The test log contains:

- The unique identifier for the test.
- The person(s) that performed the test and the date and time.
- Which iteration this was of the test.
- Any references to detailed test steps.
- All test observations.
- Result (including first categorization of findings).
- The status (accepted or not accepted).Test report

After each test has been completed a test report shall be produced:

- Summary
- Any adaptations or deviations to the test plan that has been needed.
- The actual tests contribution rate to fulfill the intention for the test
- Evaluation of the test results (according to acceptance, defects, consequences and future risk)
- A summary of the test activities



## 1.7. Test environment

Module test:	Contractor development environment
Integration test:	Contractor development environment
System test:	Contractor system test environment
Performance test:	Contractor system test environment (not full performance test)
Deployment and Installation test:	Contractor system test environment
Acceptance test:	Customer acceptance test environment
Approval period:	Customer hosting environment

## 2. Acceptance test and approval period

### 2.1. Introduction

This chapter details:

- Execution of test and acceptance of the delivery.
- The approval period.
- Criteria for acceptance.
- Procedure for error reporting and correction.

The details of Acceptance Test will be defined in the Acceptance Test Plan.

### 2.2. Requirements for execution of acceptance test

The Customer is responsible for the execution of the acceptance test. The contractor shall support the customer during the acceptance test.

If the Customer finds errors or defects, these findings shall be evaluated and categorized according to section 1.3.4, and be reported without unfounded delay to the Contractor.

All findings shall be described in detail by the Customer and be assigned a unique reference number. The description shall reference a unique test that is identified in the agreed test specification developed for the acceptance test.

The Contractor shall record all findings reported, and must continuously be able to report the status on the analysis and correction process.

The Contractor is obliged to start working on findings categorized as Critical or Serious without unfounded delay (within normal business hours).



### 2.3. Precondition before installation for acceptance test

The following preconditions must exist before the Acceptance test can be started:

- The number of errors from the Contractors System test shall be within the agreed limits (refer section 1.3.5).
- The System test results shall be accepted by the responsible authority at the Contractor and the Contractor must have provided the Customer with the System test report.
- The Deployment and Installation test report from the Contractor shall be provided to the customer.

### 2.4. Strategy

During the Acceptance test the Customer shall ensure that the delivery fulfills the requirement specification. The Acceptance test shall also form the basis for the decision whether the delivery can be put into production or not.

The Customer decides which parts that should be tested and the detail of each test. Test reports and test procedures from the Contractors previous tests may be reused or modified.

### 2.5. Stop and restart of Acceptance test

The Acceptance test will be paused if one of the following occurs:

- The error found prevents further testing of one or several sub-systems.
- Larger changes are needed to fix the error and it is considered that further test will have to be repeated after the changes have been implemented.
- The error will have large implications on the results of further test in a way that make it hard to decide if deviations from expected result are a consequence of the original error or a new finding.

The Acceptance test will be resumed when the error(s) that caused the test to be paused has been fixed and a new version of the module(s) is installed. It will be an individual evaluation at which stage in the test the test will carry on or if the test should start from the beginning.

The acceptance test will be stopped if:

- Critical errors are found.
- The total number of errors found makes it obvious that the delivery will not be accepted.

The Acceptance test will be restarted when the error(s) that caused the test to be paused has been fixed and a new version of the module(s) is installed. In this case the Acceptance test will always restart at the beginning.

### 2.6. Acceptance of the delivery

The delivery shall be accepted when the Acceptance test has been conducted and the following conditions are fulfilled:

---



- There shall be no open errors categorized as Critical or Serious.
- According to the customer judgment the number of open errors categorized as Less serious does not prevent the system from being put into production.
- A plan for correction of remaining errors has been accepted by both parties.

## **2.7. Approval period**

### **2.7.1. Introduction**

During the approval period the Customer is responsible for the collection and registration of any findings and to report them to the Contractor.

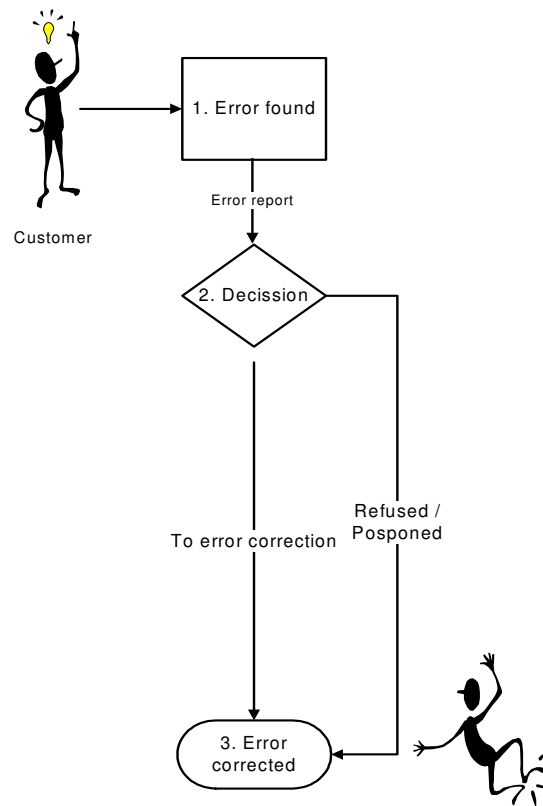
### **2.7.2. Final acceptance of the delivery**

The delivery shall be accepted when the expiration of the approval period and the following conditions are fulfilled:

- There shall be no open errors categorized as Critical or Serious.
- According to the customer judgment the number of open errors categorized as Less serious are acceptable.
- An agreement on which of the remaining errors should be fixed, based on their priority and a plan for their correction errors has been accepted by both parties.

## **2.8. Error reporting and management**

If errors are found the following procedure shall be followed:



### 1. Error found

All findings shall be registered in a bug tracking system.

### 2. Decision

The Contractor evaluate the reported error and propose one of the following decisions:

- The error reported is refused. The Contractor evaluates the reported incident to be caused by wrong usage or wrong data input.
- The error reported is assumed minor and the error correction should be postponed to next maintenance release.
- The error will be corrected as soon as possible.

Any other action that is proposed than error correction must be accepted by the Customer.

### 3. Error corrected

The error is corrected, tested and correct operation has been verified. The end-user(s) that reported the error has been informed.

## 2.9. Registration of errors

When an error is found, the following information must be registered by the Customer and the relevant information objects should be updated by the Contractor to reflect progress in the error correction procedure.

- The name of the person that has registered the error.
- The department or organization of that person.



- Date and time.
- Title: Short description of the error.
- Error reference/number (will usually be automatically generated).
- Current status: Registered, Assigned, In progress, Error corrected, Correction verified).
- The module, application or function where the error was discovered.
- Classification: Critical, Serious, Less serious.
- Detailed description of the error including the data or external objects involved.
- Where possible steps to reproduce the error.
- Any suggested solutions.
- Consequence if the error is not corrected.

### 3. Testing tools and other supporting tools

The following list of test-and supporting tools that are used is taken from Appendix 3, Customer technical platform:

Tool	Product	Version	Vendor	License
Bug tracking	Bugzilla	3.0.5	Mozilla	Mozilla Public License
Test Management	Hudson Continuous Integration Server	1.329	Sun Microsystems	MIT license
Build Management	Maven	2.2.1	Apache Software Foundation	The Apache Software License, Version 2.0
ConfigurationManagement	Subversion		Tigris.org	Subversion License
Performance test	Jmeter	2.3.4	Apache Software Foundation	The Apache Software License, Version 2.0
Profiling	Rational PurifyPlus	7	IBM	Closed source
Code analysis	FindBugs	1.3.9	Sourceforge	GNU LGPL
Test coverage	Cobertura	1.9.3	Sourceforge	Apache Software License, Version 1.1 , GNU General Public License, Version 2.0
Standards compliance	Checkstyle	5.0	Sourceforge	GNU LGPL
Functional tests	Jemmy	V2	Java.net	<a href="#">OSI</a> – CDDL
Security testing	WebScarab	20080814	OWASP	CreativeCommons 3
Securit testing	Paros Proxy	3.2	MileSCAN Technologies	Freeware



## 4. Quality assurance, error handling and configuration control

### 4.1. Quality Management

To ensure that the project is evolving within the expected, Quality Management measures will be used during the project life cycle. The project quality objectives are as follows:

- a) To deliver the agreed project outcomes on schedule and within budget.
- b) To achieve the business goals of the project.
- c) To manage the project using a defined and documented methodology.
- d) To track the project performance metrics to determine if it is performing above the defined thresholds.

The Quality Management measures will be defined in the Project Quality Plan, and executed by the Quality Manager. The Quality Management Plan will define:

- a) The quality assurance activities to be followed to the full extent of work, such as progress reviews, program scheduling & control.
- b) The organization and responsibilities for all project quality activities within the project.
- c) Definition of tools and methodologies used in the project.
- d) Reference of all activities to ensure final acceptance of the product.
- e) Definition of the corrective action process within the project.
- f) Definition of the project performance metrics and their threshold values.

This plan will require regular review and updates as the project progresses to ensure it is effective throughout the project lifecycle.

There are three major processes in quality management:

- a) **Quality Planning:** Quality planning identifies quality standards that are relevant to the project, defines new quality standards required by the project and plans activities needed to satisfy these requirements. It is a process that supports overall project planning and shall be performed regularly throughout the project lifecycle. Quality planning is performed in parallel with other planning processes. The impact to other processes to meet quality standards might include changes to functional specifications of the product, adjustments to the cost or schedule, and/or additional risks for the project. Planning activities will include articulating or at a minimum, referencing the deliverables, their acceptance criteria, and the process to be used for review and acceptance to ensure the delivery of a quality solution. Such activities will be approved by the customer before proceeding with project execution activities.
- b) **Quality Assurance (“QA”):** QA is the planned activities of a project that monitor all other quality management processes to ensure that the project will meet the quality standards defined in the quality plan. QA monitors the levels of quality being achieved and shall be performed throughout the life of the project. Specific objectives of the QA process include: represent quality concerns in walk-through reviews, provide an assessment of the completeness and consistency of deliverables, ensure that project





deliverables meet the quality requirements, and conduct project reviews to determine the health of the project.

- c) **Quality Control (“QC”)**: QC is the monitoring and analysis of certain project results and data to determine if they comply with the relevant quality standards and defined project performance metrics. Analysis is performed to determine ways to eliminate causes of unsatisfactory results. QC should be performed throughout the length of the project execution.

#### Deliverables:

- a) **Project Quality Plans**: Besides the company level Quality Manual, specific project quality needs will be included in the Project Quality plan, which is part of the Project Management Plan. This plan is based on a standard Project Management Plan, which is in turn, tailored to suit the needs of every specific project.

## 4.2. QA Assurance and Control

### 4.2.1. Implementation

A continuous integration process has been put in place that helps accelerate the speed at which errors are detected and notified to developers and QA managers. Several tools are also used to automate as much as possible the detection of bugs.

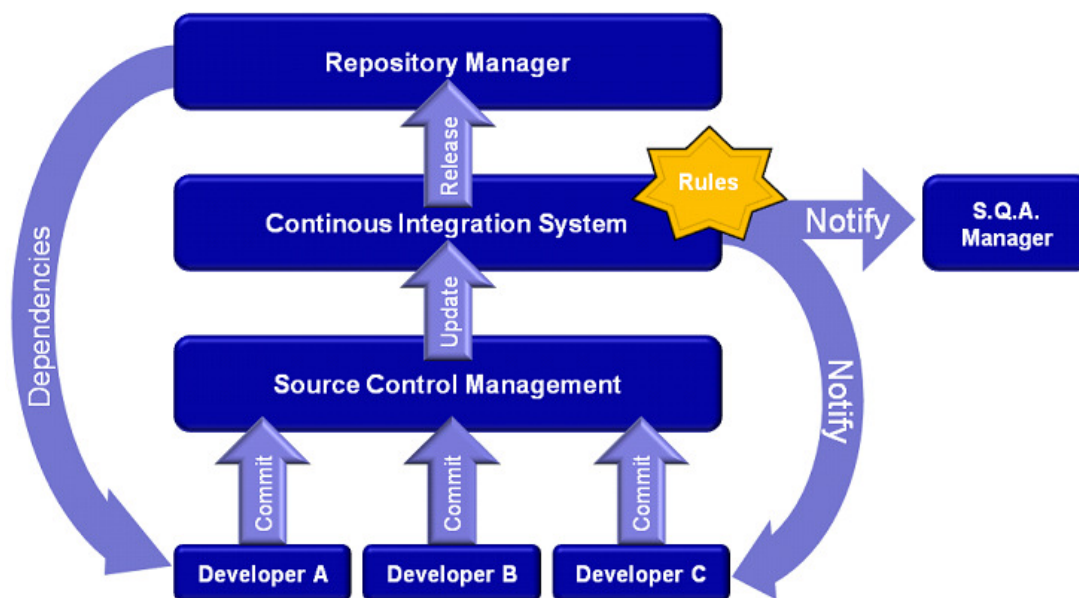


Figure 2 – Continuous Integration Process

**Process description:**

- Software Development Manager creates entries in the Bug Tracking System and assigns work
- Developer implements code functionality and Unit Tests according to the specified requirements. Functionality is developed using the Eclipse IDE with Checkstyle, Findbugs and several other plug-ins that accelerate the development speed and allow doing quality checks as code is being written. That means that any problems are detected earlier in the development cycle.
- Developer uses Maven to build code and run Unit Tests and Manual tests.
- Periodic Peer Reviews are done on the code.
- When Unit Tests and Manual tests are successful, the code is committed to Source Control, associated with the Bug Tracking System entry ID.
- The Test Management – Hudson Continuous Integration Server gets the later code updates and runs several verifications:
  - Runs Checkstyle to verify coding standards compliance.
  - Runs FindBug to identify known bug patterns in the code.
  - Runs automated (scripted) functional tests / smoke test.
  - Compiles the code using Build Management – Maven.
  - Runs the Unit Tests using Junit and Cobertura to determine the level of code coverage that the Unit Tests provide.
  - Runs tests using Load testing – JMeter and Jemmy for GUI tools.
- If there is any error (such as a compilation error) or any of the established rules regarding, FindBug warnings, Checkstyle warnings or code coverage are not met, the developer is immediately notified using internal Instant Messaging. An e-mail is also sent to the QA Manager and the developer.

Peer Review is periodically arranged on each developer's code, that is, a second member of the team reviews the written code and verifies it is correct and fulfills Software Quality Standards followed by the Contractor. This systematic examination of computer source code allows finding and fixing mistakes overlooked in the initial development phase, improving both the overall quality of software and the developers' skills.

Periodically, intermediate releases are created and, in addition to functional and integration tests, they are run through the tests described in the sections below.

#### **4.2.2. Regression tests**

Regression tests validate that old functionality has not been broken by newer functionality. Regression tests are done in two ways: By automated tests scripts (using Jemmy or custom scripts) that use the entire application and by repeating manual test cases, even those that belong to functionality that has not changed.

- Procedure description:
  - Release is created.
  - Every manual test case for the application is executed.
  - Every automated test case is executed.



- If any test fails to meet the success criteria, it is added to the issues reporting database, to be prioritized and fixed.

#### ***4.2.3. Usability and accessibility testing:***

Please refer Appendix 6 section 4 “Accessibility and Usability” for our user centric development method.

#### ***4.2.4. Security testing***

In addition to the security testing procedures, Scytl Engineers receive specific secure software development techniques training, that includes information on how to program defensively to prevent code patterns that might lead to potential risks such as buffer overflows, SQL injection, cross site scripting, etc. That ensures that security is programmed into the application from the beginning.

- Process description:
  - When compilation is done, Findbugs is used to detect common error patterns.
  - Mandatory code reviews are done on security related code by other team members and by R&D department members.
  - Web applications are tested using tools such as aParos and WebScarab.
  - Penetration tests are performed on production releases by an external company, which follows OpenSource OSSTMM e ISSAF techniques for penetration testing. This guarantees the objectivity of the audit being done, since the audit team is completely unrelated to the development team.
  - Any issue resulting from the audit and tests are applied added to the issue reporting database to be prioritized and fixed.

#### ***4.2.5. Volume and performance testing***

Each release is tested under high loads to verify that it behaves correctly.

- Process description:
  - Release is created.
  - High load test cases are defined for each release.
  - These test are automated using Jmeter and include:
    - Tests with low amount of data.
    - Tests with high amounts of data, within the expected application operation range (Volume testing).
    - Tests with higher data than the system is expected to operate with (Stress testing).



- Analysis of memory usage and performance bottlenecks using Rational Purify.
- A success/failure criteria is defined for every test case.
- These test cases are executed and the system performance is measured and compared to the success criteria.
- Any value outside the expected range is reported as an issue to be prioritize and fixed.

After the tests, all resulting issues are prioritized by the project management and an agreement on which ones will be fixed is done. After those issues are fixed, the tests are run again until the project management determines that the implemented code/feature/module fulfills the defined requirements.

This phase is complete when the following have been reviewed and, in those instances identified by an \*, brought under configuration control:

- Baselined software product\*
- Unit test, unit test results\*
- Function integration tests, function integration test results\*
- Administration documentation
- System-level test case description\* - signature VP engineering, product manager, quality assurance manager

The following products are the outputs of this phase:

- Baselined software product
- Unit tests
- Unit test results
- Function integration tests
- Function integration test results
- System-level test cases
- Code Review Report

### **4.3. Releasing**

The objective of this procedure is to prepare and deploy a release of the project. The release can be an internal or an external release. Internal releases are the releases done inside the Software Engineering department as milestones of the development process and never reach the customer. External releases are all the releases that go to the customer for acceptance.

The contents of every specific release will be detailed in the acceptance test plan. In general terms, a release will include one or many of the following (when applicable)

- Source code
- Unit tests
- Compilation script
- Technical and functional documentation
- Test cases and scripts
- Results of internal testing



The release procedure is always applied after the implementation of a version of the software module/application. The release procedure does not include the on-site installation to the customer, neither the maintenance procedure.

When the release is external, then the system must be installed in the testing or pre-production environment so the customer can validate, with the Contractor's support, the correct behavior performance of the developed system, basing such testing on the defined User test cases.



## QP207 Release

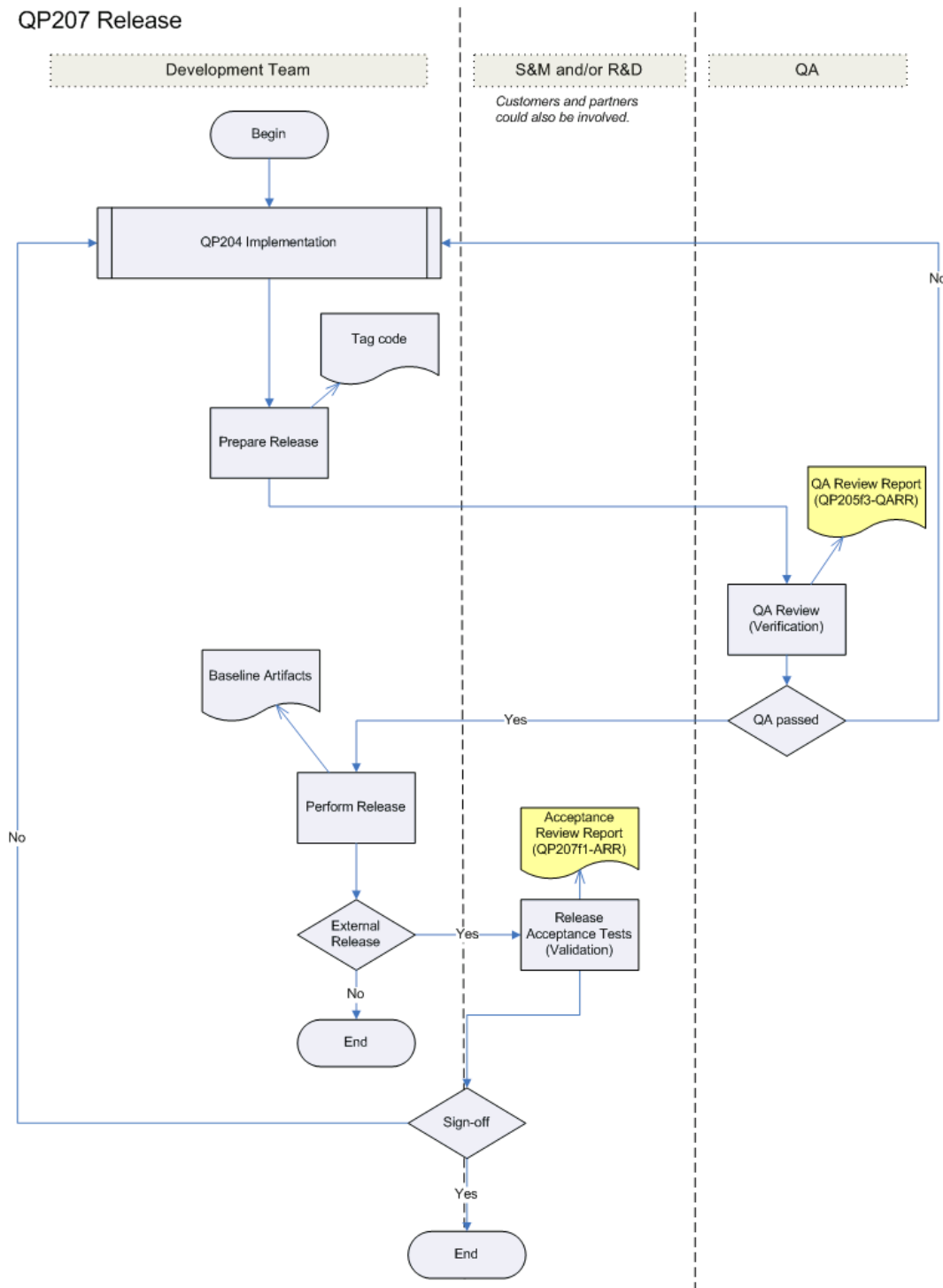


Figure 3 – Releasing process



#### 4.4. Configuration Management

Software configuration management (or SCM) is the task of tracking and controlling changes in the software. It includes revision control and baselines.

SCM procedures are supported by version control tools. Every configuration item (document, source code file, unit test, test cases, scripts, configuration files, etc) is stored in the centralized repository of the Configuration/Version Management - Subversion version control server. This tool keeps track of every change made. At specific moments, such as when a delivery is accepted, the current status of the items under version control will be baselined and a tag will be created in the version control server.

Configuration Management procedures will be described as part of a Configuration Management Plan, that will be developed during the project.

#### 4.5. Change Management

The Project Manager is responsible for executing the change management process.

Changes to scope during the delivery of any project introduce risks that can, if not managed effectively, cause projects to fail. These failures may be due to reasons of time constraint, budget overrun, and inadequately assessed functional or technical impact of the change. Uncontrolled changes can disrupt development and implementation and impact the balance between schedule and quality. Anyone associated with the project may initiate a request for a change. However, a baseline must be maintained against which the impact of approved changes is assessed.

Changes to the project that impact schedule and cost will be reviewed in detail. Any possible financial impact of the change will be determined, and if appropriate, further charges will be documented by the Contractor and passed on to the customer. The change will be documented and signed off by both the Contractor and the customer prior to work commencing.

Deliverables (if required):

- a) **Change Request:** this document reflects the changes agreed on the project, including their impact on schedule/pricing/functionalities.
- b) **Delay Notification:** this document reflects project delays agreed on the project schedule.

#### 4.6. Tools Used

Below there is a list of the main tools employed by the Contractor to manage its projects and develop software. The Contractor is continuously monitoring the market to adopt the best available practices and tools. With the sole exception of Rational Purify, all the testing and configuration control tools used are Open Source. That means that it would be possible for the customer to set up an equivalent test environment without having to buy additional licenses.



#### 4.6.1. Bug Tracking – Bugzilla

Bugzilla is a Web-based general-purpose bugtracker tool originally developed and used by the Mozilla project, and licensed under the Mozilla Public License. The Contractor adopted it long ago for use as a defect tracker for it solutions.

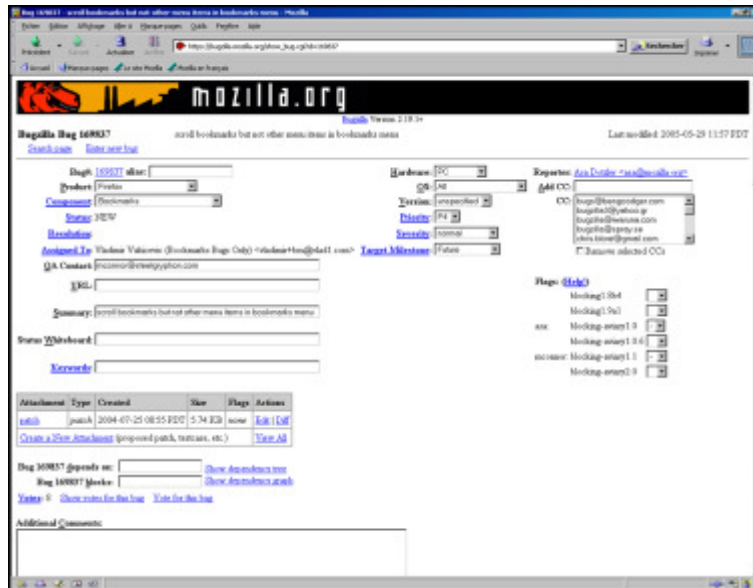


Figure 4 – Bugzilla screenshot

While the potential exists in the code to turn Bugzilla into a technical support ticket system, task management tool, or project management tool, Bugzilla's developers have chosen to focus on the task of designing a system to track software defects. Mandated design requirements include:

- The ability to run on freely available, open source tools. While Bugzilla development includes work to support commercial databases, tools, and operating systems, this is not intended to come at the expense of open source ones.
- The maintenance of speed and efficiency at all costs. One of Bugzilla's major attractions to developers is its lightweight implementation and speed, so calls into the database are minimized whenever possible, data fetching is kept as light as possible, and generation of heavy HTML is avoided.
- Tickets. For instance, Mozilla.org and the MediaWiki project use it to track feature requests as well. Bugs can be submitted by anybody, and will be assigned to a particular developer. Various status updates for each bug are registered and maintained, together with user notes and bug examples. In practice, most Bugzilla projects allowing the public to file bugs — such as the Bugzilla bug database for Bugzilla itself — assign all bugs to a gatekeeper, whose job it is to assign responsibility and priority level.

#### 4.6.2. Test Management – Hudson Continuous Integration Server

Hudson is a continuous integration tool written in Java, and runs in a servlet container, such as Apache Tomcat or the Glassfish application server. It supports SCM tools including CVS and Subversion, and can execute Apache Ant and Apache Maven based projects, as well as arbitrary shell scripts and Windows batch commands. The Contractor uses it to configure automated tests on all its developed applications.



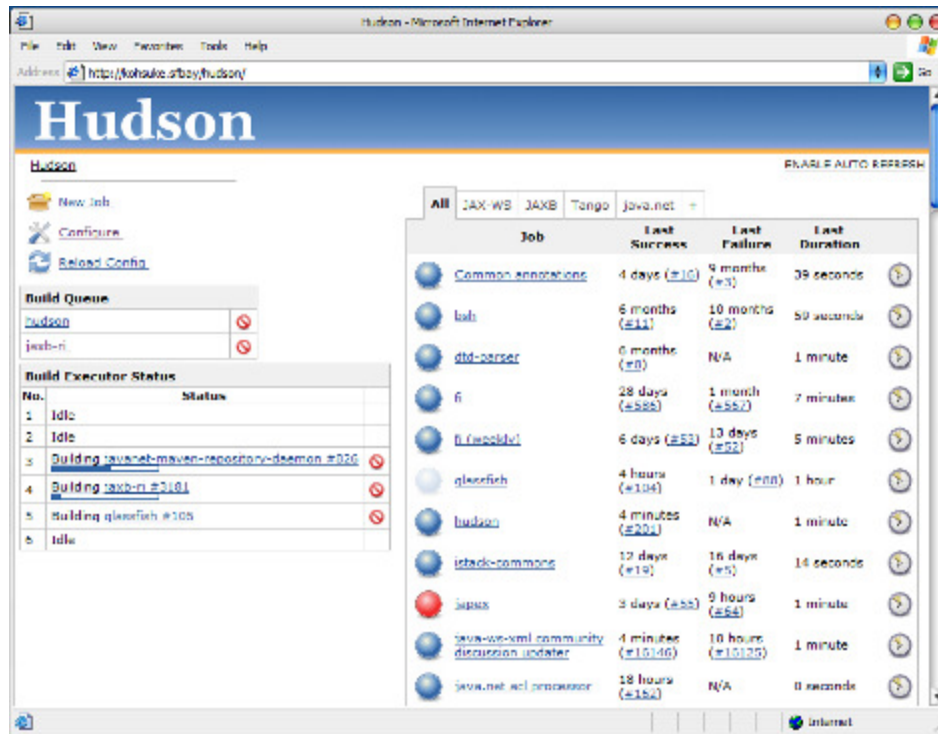


Figure 5 – Hudson screenshot

Builds can be started by various means, including scheduling via a cron-like mechanism, building when other builds have completed, and by requesting a specific build URL.

During recent years, Hudson has become a popular alternative to CruiseControl and other open-source build servers. At the JavaOne conference in May 2008, it was the winner of Duke's Choice Award in Developer Solutions category.

#### 4.6.3. Build Management – Maven

Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information. This is its primary use by The Contractor.

Maven 2.0 is based around the central concept of a build lifecycle. What this means is that the process for building and distributing a particular artifact (project) is clearly defined.

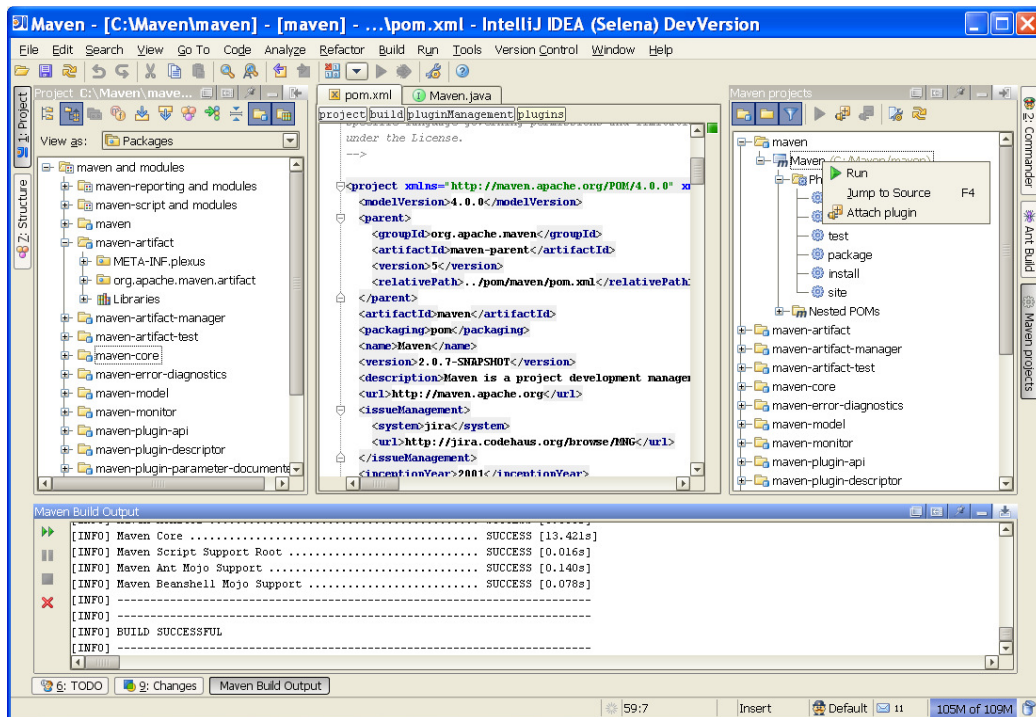


Figure 6 – Maven screenshot

For the person building a project, this means that it is only necessary to learn a small set of commands to build any Maven project, and the POM will ensure they get the results they desired.

There are three built-in build lifecycles: default, clean and site. The default lifecycle handles project deployment; the clean lifecycle handles project cleaning, while the site lifecycle handles the creation of project's site documentation.

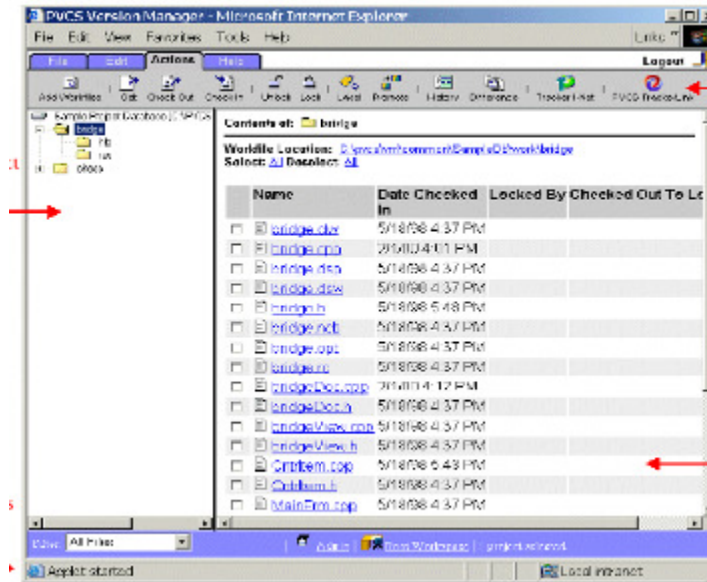
For example, the default lifecycle has the following build phases:

- validate - validate the project is correct and all necessary information is available.
- compile - compile the source code of the project.
- test - test the compiled source code using a suitable unit testing framework. These tests should not require the code be packaged or deployed.
- package - take the compiled code and package it in its distributable format, such as a JAR.
- integration-test - process and deploy the package if necessary into an environment where integration tests can be run.
- verify - run any checks to verify the package is valid and meets quality criteria.
- install - install the package into the local repository, for use as a dependency in other projects locally.
- deploy - done in an integration or release environment, copies the final package to the remote repository for sharing with other developers and projects.



#### 4.6.4. Configuration/Version Management - Subversion

The version control tool provides assistance for the control of the production process and modification of an application. Subversion is the main tool used by The Contractor for such tasks.



### Figure 7 – Subversion screenshot

It lowers the risk of error and ensures consistency of developments within teams. This tight control is reached by concurrent access to a component and the lock components extracted for updates.

It allows control of developments faster and better through the management of releases, and concurrent and parallel developments.

Teams keep the change history of all components, which allows you to find and restore any previous version of an application, maintaining it reliable, consistent and reproducible.

It provides the configuration manager(as well as the developers) utilities as effective tools for management of application versions, construction of the deliverables, querying the repository for a list of locked components, report changes in a component, and the list of components.



#### 4.6.5. Load testing – JMeter

JMeter is an Apache Jakarta project that can be used as a load testing tool for analyzing and measuring the performance of a variety of services, with a focus on web applications.

JMeter can be used as a unit test for JDBC database connections, FTP, LDAP, Webservices, JMS, HTTP and generic TCP connections. JMeter can also be configured as a monitor, although this is typically considered an ad-hoc solution in lieu of advanced monitoring solutions.

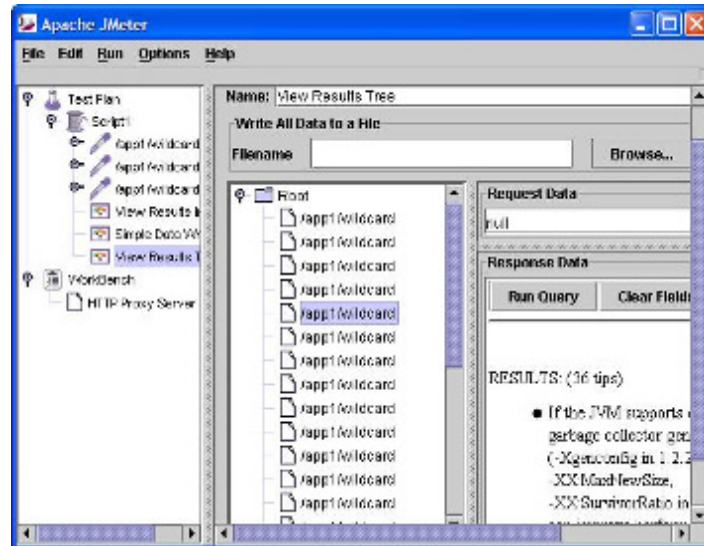


Figure 1 – JMeter screenshot

While JMeter is classified as a "load generation" tool, this is not a complete description of the tool. JMeter supports assertions to ensure the data received is correct, per thread cookies, configuration variables and a variety of reports.



#### 4.6.6. Rational Purify

IBM Rational Purify is a runtime analysis solution designed to help developers write more reliable code. Reliability is ensured via two crucial functions: memory corruption detection and memory leak detection. Rational Purify packages support for these two runtime analysis capabilities in a single product.

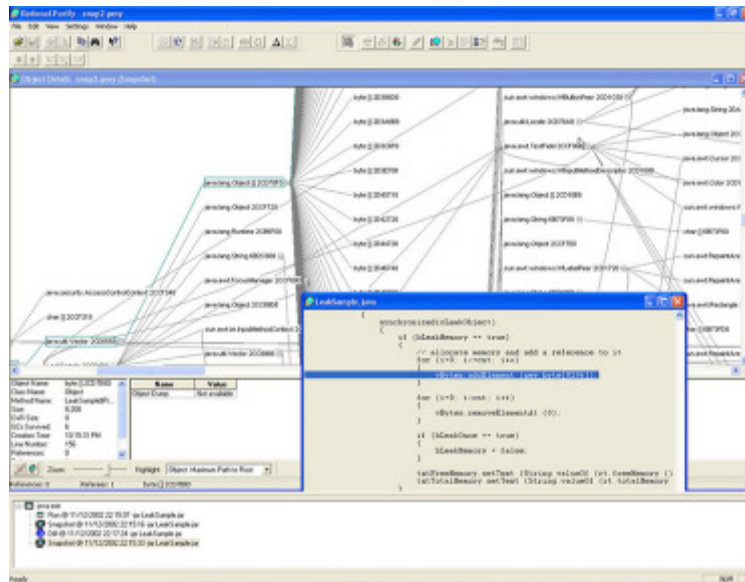


Figure 9 – Rational Purify screenshot

#### 4.6.7. FindBugs

FindBugs looks for bugs in Java programs. It is based on the concept of bug patterns. A bug pattern is a code idiom that is often an error. Bug patterns arise for a variety of reasons:

- 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 inspect Java bytecode for occurrences of bug patterns. Static analysis means that FindBugs can find bugs by simply inspecting a program's code: executing the program is not necessary. This makes FindBugs very easy to use: in general, you should be able to use it to look for bugs in your code within a few minutes of downloading it. FindBugs works by analyzing Java bytecode (compiled class files), so you don't even need the program's source code to use it.

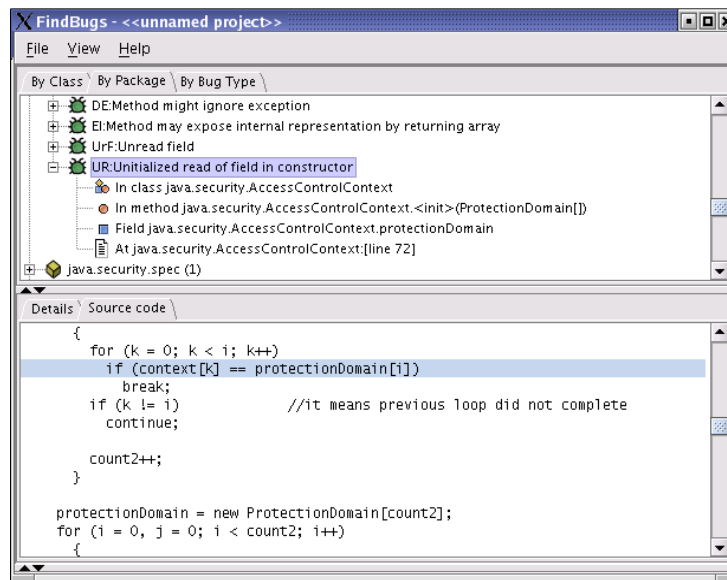


Figure 9 – FindBugs screenshot

FindBugs is free software, available under the terms of the Lesser GNU Public License. It is written in Java, and can be run with any virtual machine compatible with Sun's JDK 1.5. It can analyze programs written for any version of Java. FindBugs uses BCEL to analyze Java bytecode. As of version 1.1, FindBugs also supports bug detectors written using the ASM bytecode framework. FindBugs uses dom4j for XML manipulation.

#### 4.6.8. Cobertura

Cobertura is a free Java tool that calculates the percentage of code accessed by tests. It can be used to identify which parts of your Java program are lacking test coverage. It is based on jcoverage. Features:

- Can be executed from ant or from the command line.
- Instruments Java bytecode after it has been compiled.
- Can generate reports in HTML or XML.
- Shows the percentage of lines and branches covered for each class, each package, and for the overall project.
- Shows the McCabe cyclomatic code complexity of each class, and the average cyclomatic code complexity for each package, and for the overall product.
- Can sort HTML results by class name, percent of lines covered, percent of branches covered, etc. And can sort in ascending or descending order.



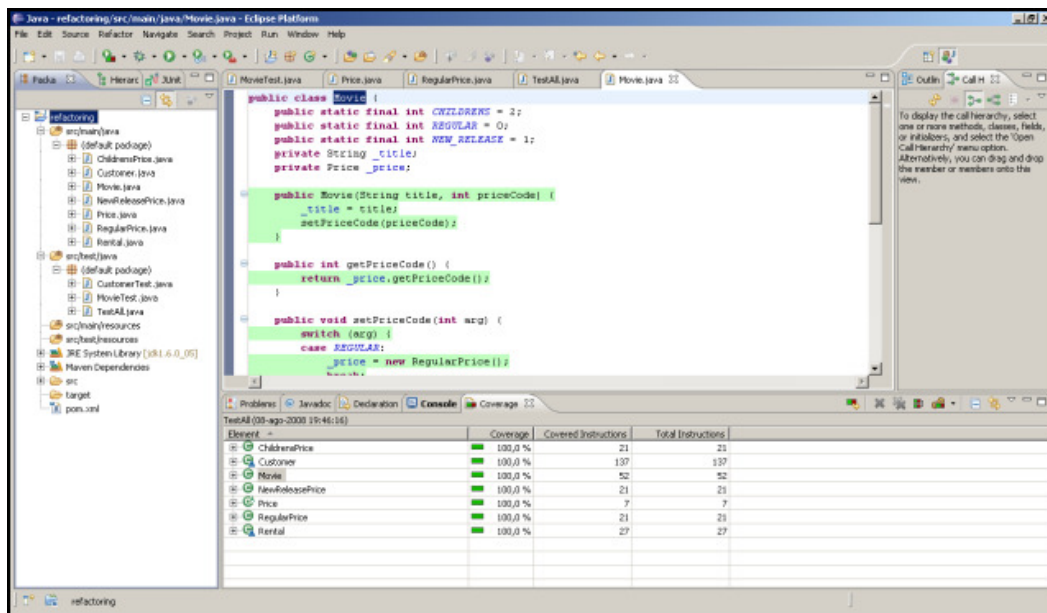


Figure 10 – Cobertura screenshot

#### 4.6.9. Checkstyle

Checkstyle is a 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. This makes it ideal for projects that want to enforce a coding standard.

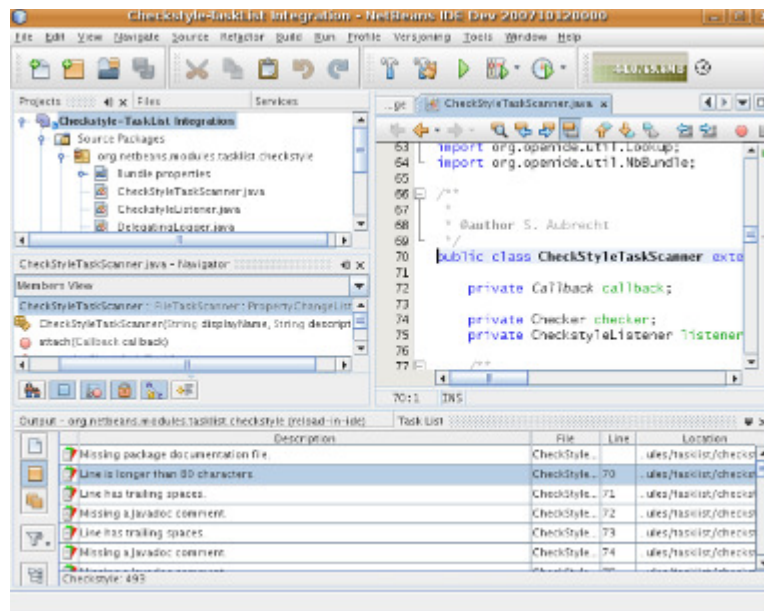


Figure 11 – Checkstyle screenshot



Checkstyle is highly configurable and can be made to support almost any coding standard. An example configuration file is supplied supporting the Sun Code Conventions. As well, other sample configuration files are supplied for other well-known conventions.

#### 4.6.10. Jemmy

Jemmy is a tool that allows the creation of automated (scripted) tests for Java GUI applications.

Jemmy represents the most natural way to automate the testing of Java GUIs - the automated tests are themselves written in Java. Having the tests written in Java allows all the flexibility of a high level language, without having to learn yet *another* programming language. Jemmy is a Java library allowing you to simulate user actions to test Java GUI logic and then automatically compare the GUI's actual behaviour with expected behaviour. Jemmy is obviously perfect for automated Test Driven Development and/or Concept Oriented Development.

#### 4.6.11. WAVE

WAVE is a free web accessibility evaluation tool provided by WebAIM. It is used to aid humans in the web accessibility evaluation process. Rather than providing a complex technical report, WAVE shows the original web page with embedded icons and indicators that reveal the accessibility of that page.

skip navigation | about | contact/feedback | english | español

**WAVE**  
web accessibility evaluation tool

Web page address...

or

icons key toolbar help blog

### Welcome to WAVE

WAVE is a free web accessibility evaluation tool provided by [WebAIM](#). It is used to aid humans in the web accessibility evaluation process. Rather than providing a complex technical report, WAVE shows the original web page with embedded icons and indicators that reveal the accessibility of that page.

#### Enter a web site address

Enter the URL of the web site you want to evaluate:

#### Upload a file

If you have files that are not publicly available on the internet, you can upload the files for WAVE evaluation. Simply browse to the file using the form below.

#### Check HTML code

### WAVE Blog

- [WAVE User Survey](#)  
July 31, 2009
- [WAVE Update](#)  
July 28, 2009
- [New Rules and Bug Fixes](#)  
May 28, 2009
- [WAVE in Spanish](#)  
May 28, 2009
- [Updated Toolbar Released](#)  
April 24, 2009

[Read more of the blog...](#)

#### 4.6.12. TAW

TAW is an on-line web accessibility analyzer





#### 4.6.13. W3C Markup Validation Service

This is an online web offered by the W3C that allows to validate a web page for conformance to the HTML standard.

This validator checks the [markup validity](#) of Web documents in HTML, XHTML, SMIL, MathML, etc. If you wish to validate specific content such as [RSS/Atom feeds](#) or [CSS stylesheets](#), [MobileOK content](#), or to [find broken links](#), there are [other validators and tools](#) available.

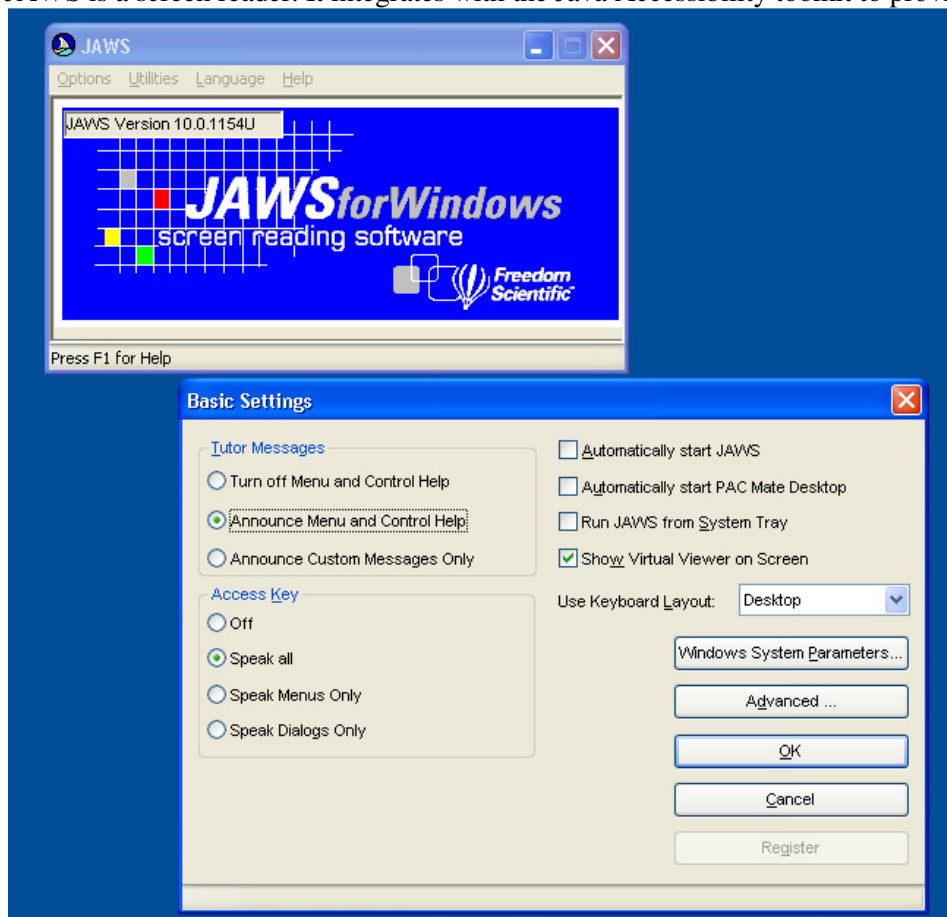
#### 4.6.14. Junit

Junit is an Open Source Java Unit Test framework.



#### 4.6.15. JAWS

JAWS is a screen reader. It integrates with the Java Accessibility toolkit to provide accessible Java applications.





#### 4.6.16. Paros

"Paros" is a software for people who need to evaluate the security of their web applications. It is free of charge and completely written in Java. Through Paros's proxy nature, all HTTP and HTTPS data between server and client, including cookies and form fields, can be intercepted and modified.

**PAROS** MITM Proxy + Spider + Scanner + Your Imagination  
The Magic for Web Security


**Home**  
**Paros Proxy**  
Download  
Features  
Installation  
Mini FAQ  
Donation  
**Contact**

**Paros - for web application security assessment**

We wrote a program called "**Paros**" for people who need to evaluate the security of their web applications. It is free of charge and completely written in Java. Through Paros's proxy nature, all HTTP and HTTPS data between server and client, including cookies and form fields, can be intercepted and modified.

We hope you can benefit from our work and products.

If you want to support our project or obtain formal support from us, please check out the product [Milescan Web Security Auditor](#), which is further developed by our core team member and supported by us as well.

 **MileSCAN**  
*Web Security Auditor*

**About Us**

We are a group of IT security professionals. We have years of experience in security design and risk assessment of web application, especially for those of financial institutions. The aim of this website is to:

1. Provide a standardized methodology for doing web application security assessment
2. Provide guidelines to web developers for secure web application design
3. Express our view on issues about web application security

Copyright © 2004, Chinotec Technologies Company. All Rights Reserved.

#### 4.6.17. WebScarab

WebScarab is a framework for analysing applications that communicate using the HTTP and HTTPS protocols. It is written in Java, and is thus portable to many platforms. WebScarab has several modes of operation, implemented by a number of plugins. In its most common usage, WebScarab operates as an intercepting proxy, allowing the operator to review and modify requests created by the browser before they are sent to the server, and to review and modify responses returned from the server before they are received by the browser. WebScarab is able to intercept both HTTP and HTTPS communication. The operator can also review the conversations (requests and responses) that have passed through WebScarab.

