Trustworthy Factory

Release

January 14, 2016

Contents

CHAPTER 1

Introduction

This section describes how to use the different factories used by the software developers and by the software designers. These factories are used to develop applications by writing source code and to certify application. For each one, a global scenario of development is given and the detailed activities. For java (and mobile) development factory, the pre-requisite is that the developers have enough skills to write java code. For the certifier, the goal of certification tool is to help the evaluator to produce evidences and, with that and with some other information, to deliver a Digital Trustworthiness Certificate (DTWC).

CHAPTER 2

User Guide

2.1 Global scenario description

Two main scenarios, one for each profile. The development scenario

The basic scenario for the trustworthy java application development factory is the following:

- The developer starts his trustworthy java application development factory. During this step, he enters his credentials in order to access the environment;
- The developer configures the environment by using the preference pages. During this phase, he configures the access to the SVN repository;
- The developer creates a new Optet project. He uses the Optet wizard to configure this new project.
- The developer writes the application source code following the guidelines and performing the tasks he has to follow for a trustworthiness development.
- During the project, the developer could run static and runtime analysis in order to check the compliance with the TWAttributes requirements;
- When the development of the application is finished, the developer archives the source code in the secure repository.

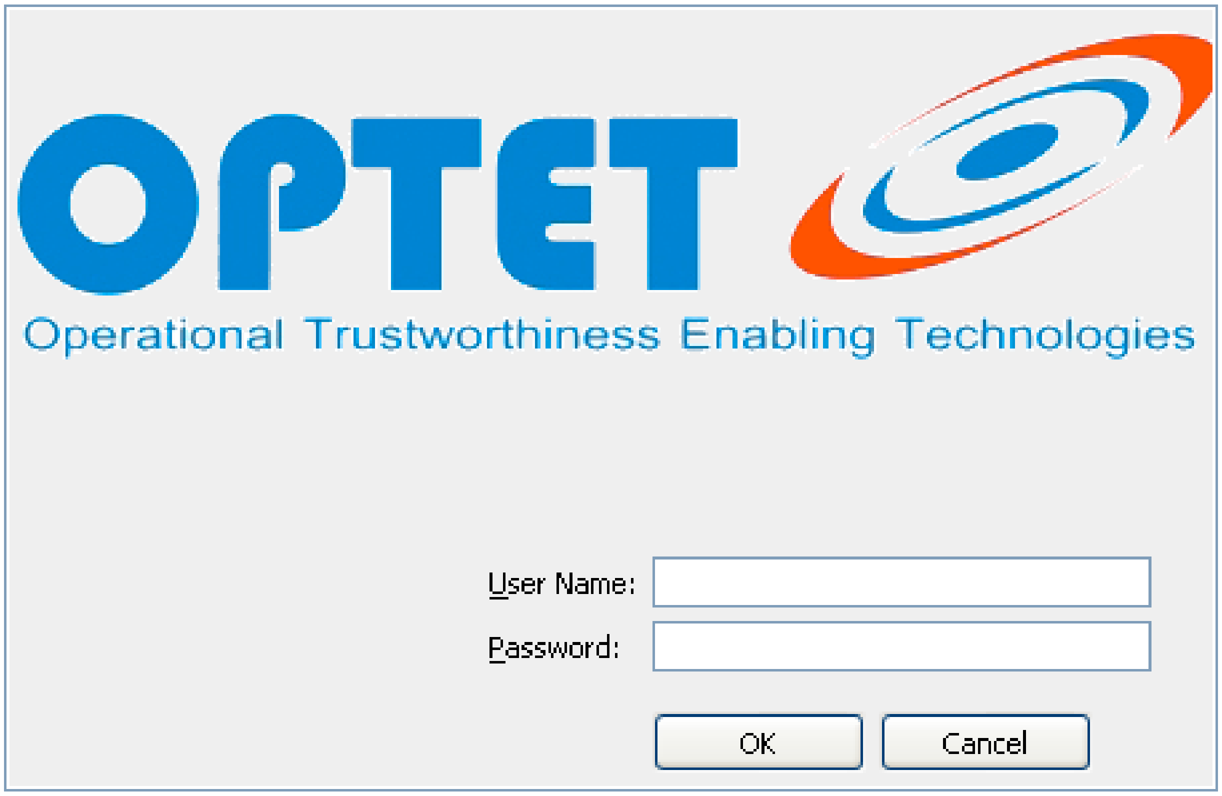
The certification scenario The basic scenario for the certification tool is the following:

- The certifier starts his certification tools. During this step, the user enters his credentials in order to be admitted into the environment;
- The certifier configures the environment using the preference pages. During this phase, he configures the access to the SVN repository and the certificate used to signed the DTWC;
- The certifier creates a new Optet project. He uses the Optet wizard to configure this new project. At this stage, the certifier must select a project with the sources (coming from the development phase);
- **On the project, the certifier opens the workflow view of the certification process:**
  - The first step enables to define the software description regarding the input of the SVN file (TWProfile and CSM);
  - The second phase enables to define the trustworthiness problem definition;
  - The third phase enables to define the trustworthiness property specification;
  - The fourth phase enables to run the analysis using the analysis plug-in (using the automatic functionality);
  - The last phase enables to generate the DTWC[2] and the component.

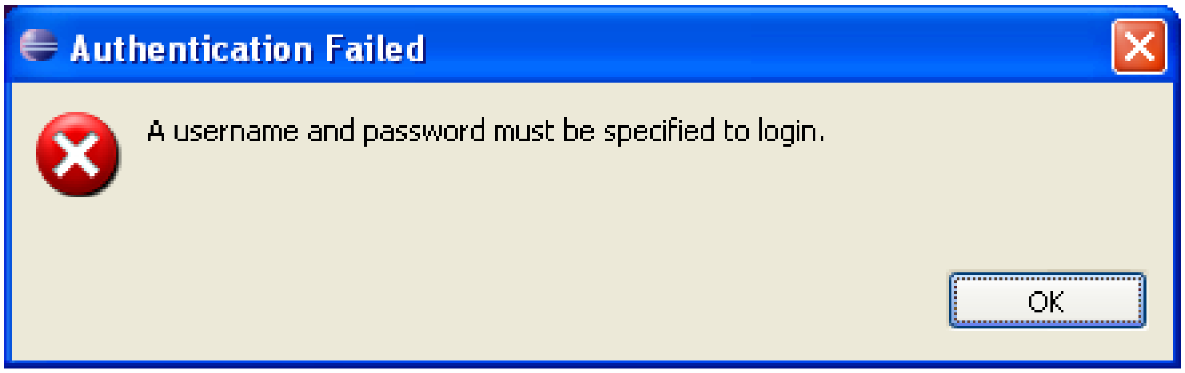
- When the certificate and the component are created, the certifier saves them in the secure repository.

2.2 Authentication phase

When the user wants to develop a new component by using the Trustworthy Java applications development factory, he starts the factory and enters his credentials in order to be authenticated. During the start-up of the factory based on eclipse, the following view appears in order to enter credentials:



In this view, the login and the password must be entered. Then press ?OK? to validate. If the credentials are valid, the user reaches the workspace development view. If not, an error message appears:

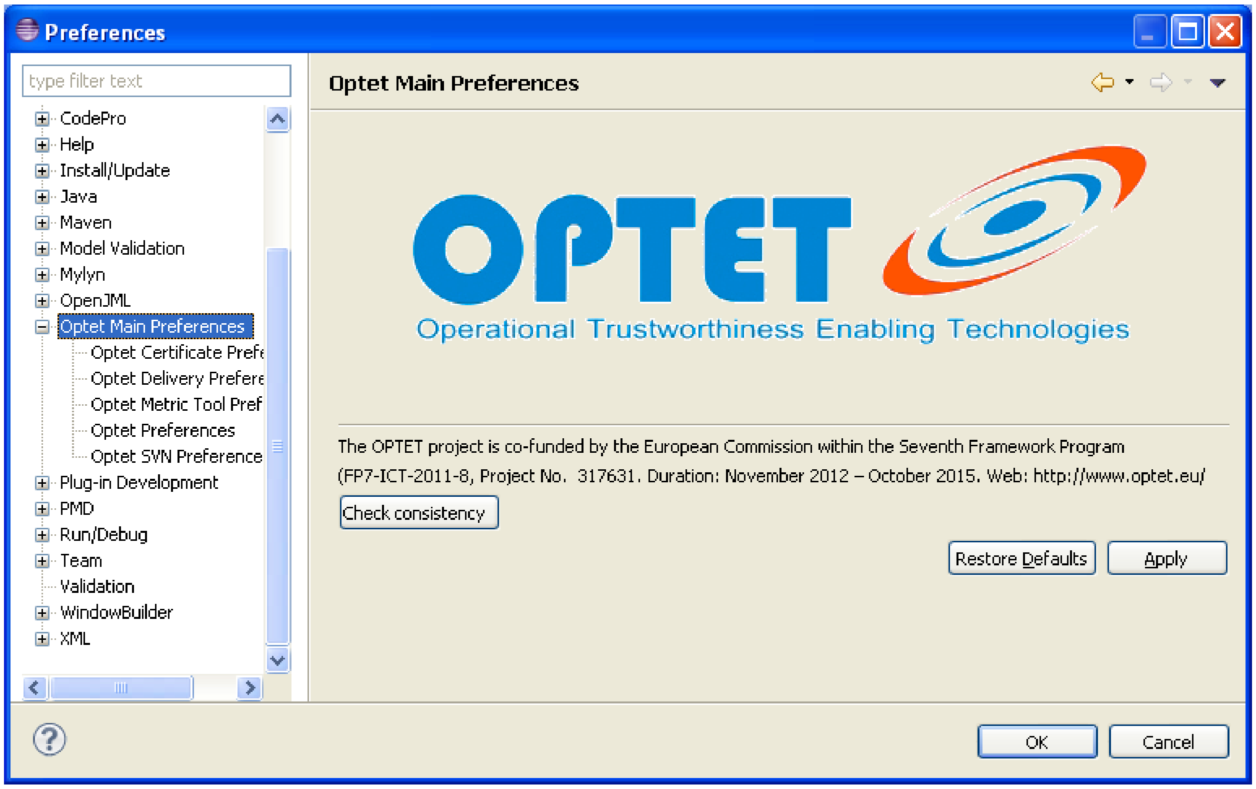


A username and password must be specified to login.

In this version, the login password for the development factory is dev/dev and for the certification tool is cert/cert.

2.3 Configuration of the environment

The developer has to set the Optet. The configuration is done under a unique entry point into the Preferences pages of Eclipse. To access the Optet Preferences, click on Windows>Preferences>Optet Main Preferences The following page appears:



The OPTET project is co-funded by the European Commission within the Seventh Framework Program (FP7-ICT-2011-8, Project No. 317631. Duration: November 2012 – October 2015. Web: http://www.optet.eu/

This page is the mother page presenting the Optet project. Under this page, all the Optet configuration pages are located, i.e.:

- Optet Certificate Preferences
- Optet SVN Preferences
- ...

For the developer, only the Optet SVN preferences need to be used.

**1. Optet SVN Preferences

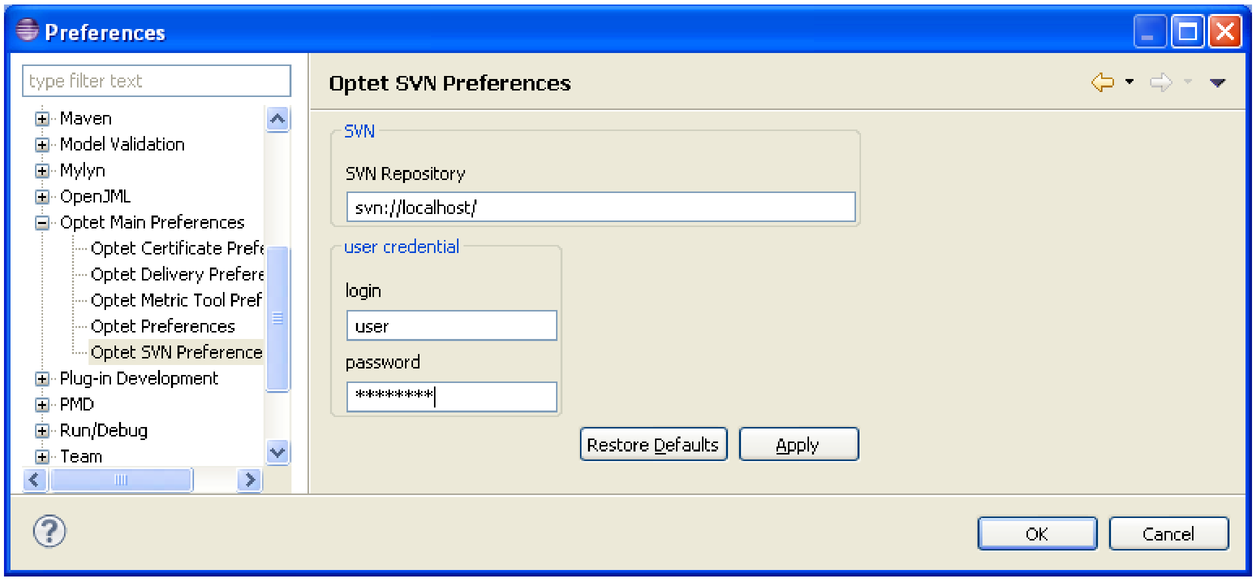
SVN tool plays the role of secure repository. This secure repository is used to store the Trustworthiness project coming from the design phase.

The required data is: * The SVN Repository: the URL of the repository used shall be entered * The user credentials: the developer enters the SVN credentials used to interact with the SVN repository. (The credentials can be different from the factory credentials). The deployment of a SVN Repository is a pre-requisite for Java Trustworthiness factory and is not in his scope.

**2. Optet Certification Preferences

The Certificate Preferences allows the certifier to specify some required data about the generation of the certificate.

At the end of the certification process, the certificate generated by the automatic process is signed using the certificate of the certifier. For that, the certifier must enter the element require in order to use his certificate ((see the keytool part).



Optet SVN Preferences

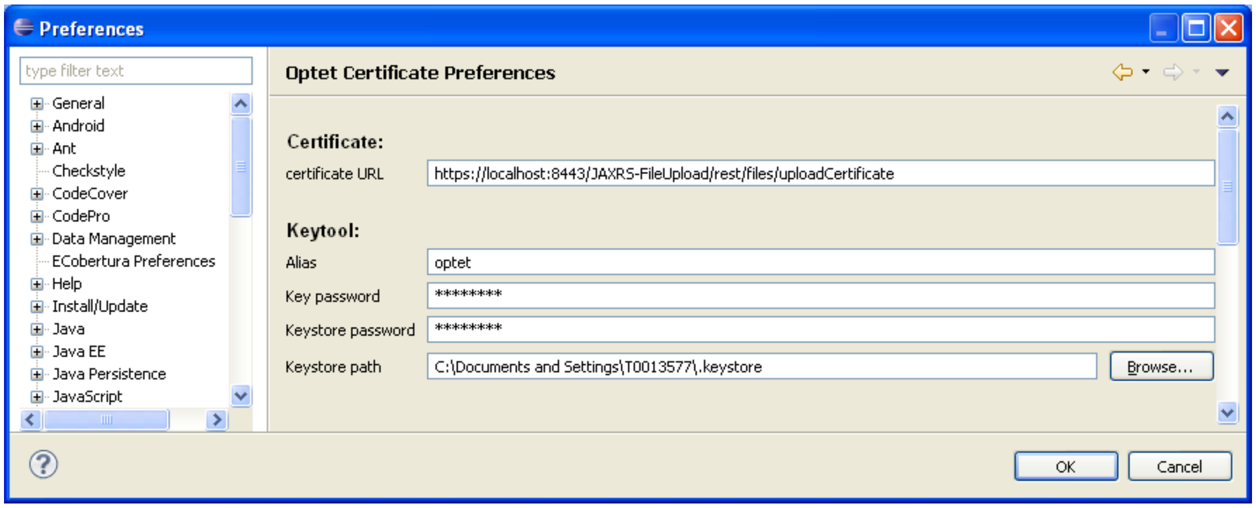
SVN

SVN Repository: svn://localhost/

user credential

login: user

password: *********



Optet Certificate Preferences

Certificate:

certificate URL: https://localhost:8443/JAXRS-FileUpload/rest/files/uploadCertificate

Keytool:

Alias: optet

Key password: ********

Keystore password: ********

Keystore path: C:\Documents and Settings\T0013577\.keystore

This certificate is stored in a key store and the access to this key store is specified (the path, the alias of the certificate, the password of the certificate, the password of the key store).

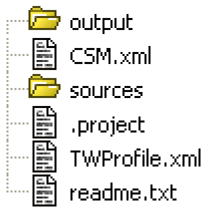
When the certificate is generated, it?s stored locally and also pushed to external service (web service) in charge of storing all the certificates generated by the certification tool.

2.4 Development of a Java application

The development of a trustworthy Java application is based on the same mechanisms than for a standard Java application. The main steps are: * To link this java development with trustworthy requirement coming from the secure repository. * To write java source code for the application.

**1. Trustworthy requirements

The trustworthy requirements for the Java development are present into the secure repository configured previously. This secure repository is composed of a set of projects. Each project is linked to a dedicated application which must be developed with the factory. The structure of each project is the following and it integrates data coming from the design phase (The TW profile, the CSM (Concrete System Model), etc...). All these data need to be set into the Secure repository during the design phase under a specific tree:



- An output directory which contains, at the end, the delivery and the TW certificate;
- The sources directory which contains the sources produced during the development phase;
- The TWProfile.xml file which is the file containing the evidence required by the assets;
- The CSM.xml file which contains design data of the component to be developed;
- The readme.txt file which contains information about the project required by the developer and /or the certifier.

All these files are mandatory in order to start the development.

**2. Java Project Creation

The developer has fourpossibilities for a development using the Optet environment:

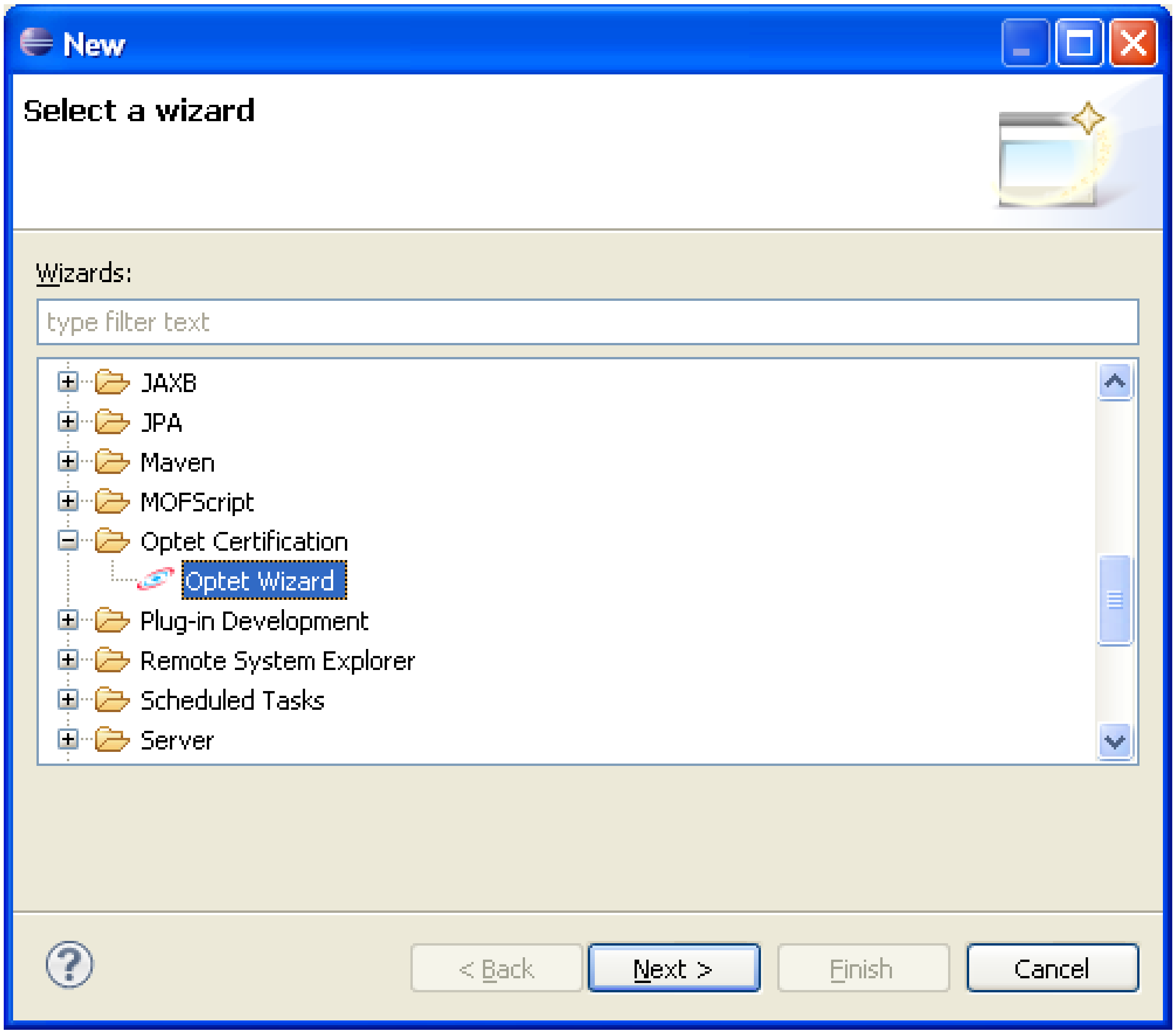
- Start a new development from scratch;
- Recover a previous development already developed using Optet
- Convert a standard java project into an Optet development.
- Convert manually a standard java project into an Optet development.

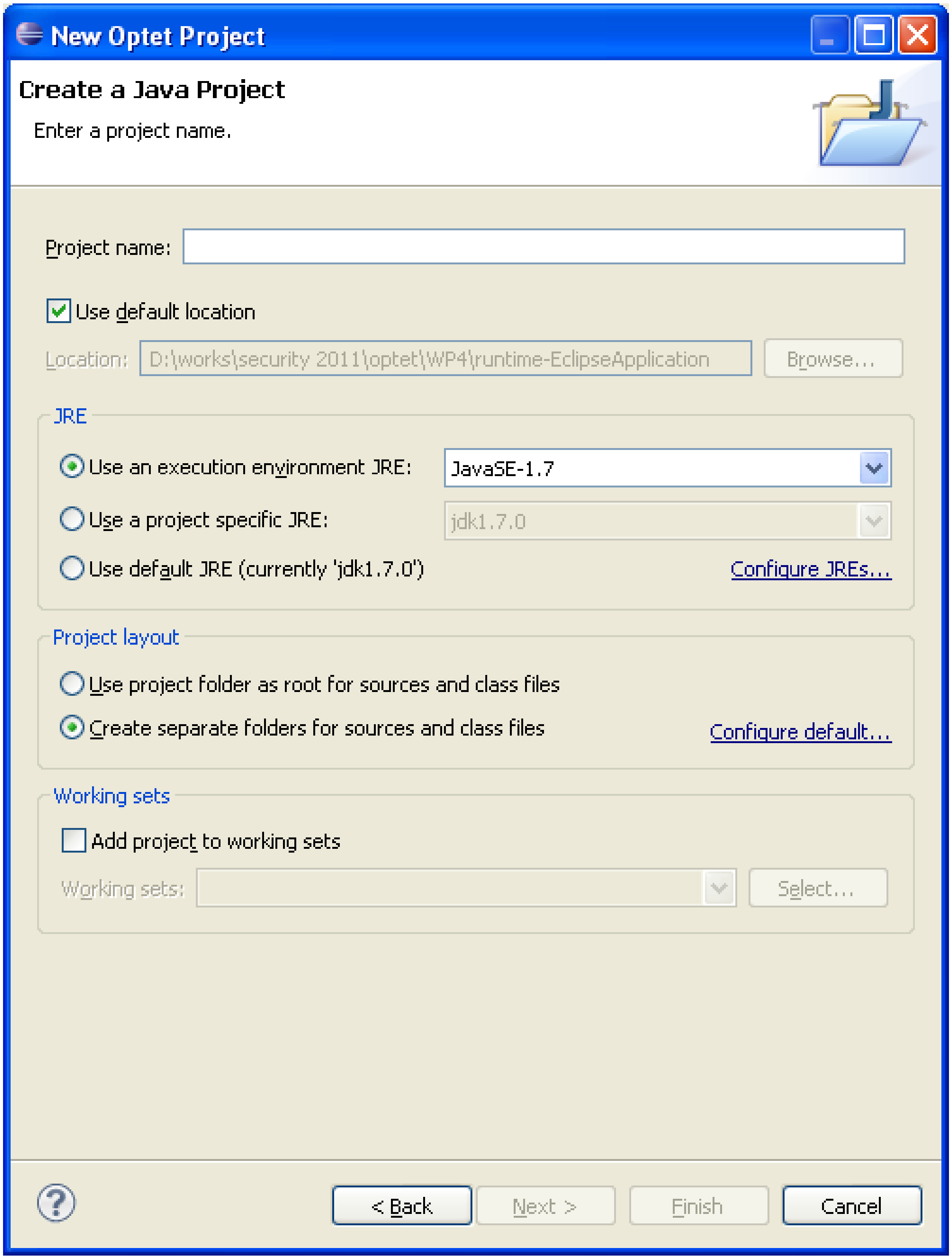
**2.1 Start an empty SVN project

In order to create a new Optet project, the developer must use the Optet Wizard. For this, click on File>New>Other>Optet certification>Optet Wizard

At this step, the two next views will be the same as for a standard Java project.

The developer must enter the name of the project, the location, the JRE used, etc... The second view is used to configure the Java settings


New
Select a wizard
Wizards:
type filter text
JAXB
JPA
Maven
MOFScript
Optet Certification
Optet Wizard
Plug-in Development
Remote System Explorer
Scheduled Tasks
Server
< Back
Next >
Finish
Cancel



Create a Java Project

Enter a project name.

Project name:

[✓] Use default location

Location: D:\works\security 2011\optet\WP4\runtime-EclipseApplication

JRE

[● Use an execution environment JRE:] JavaSE-1.7

Use a project specific JRE: jdk1.7.0

Use default JRE (currently 'jdk1.7.0')

Project layout

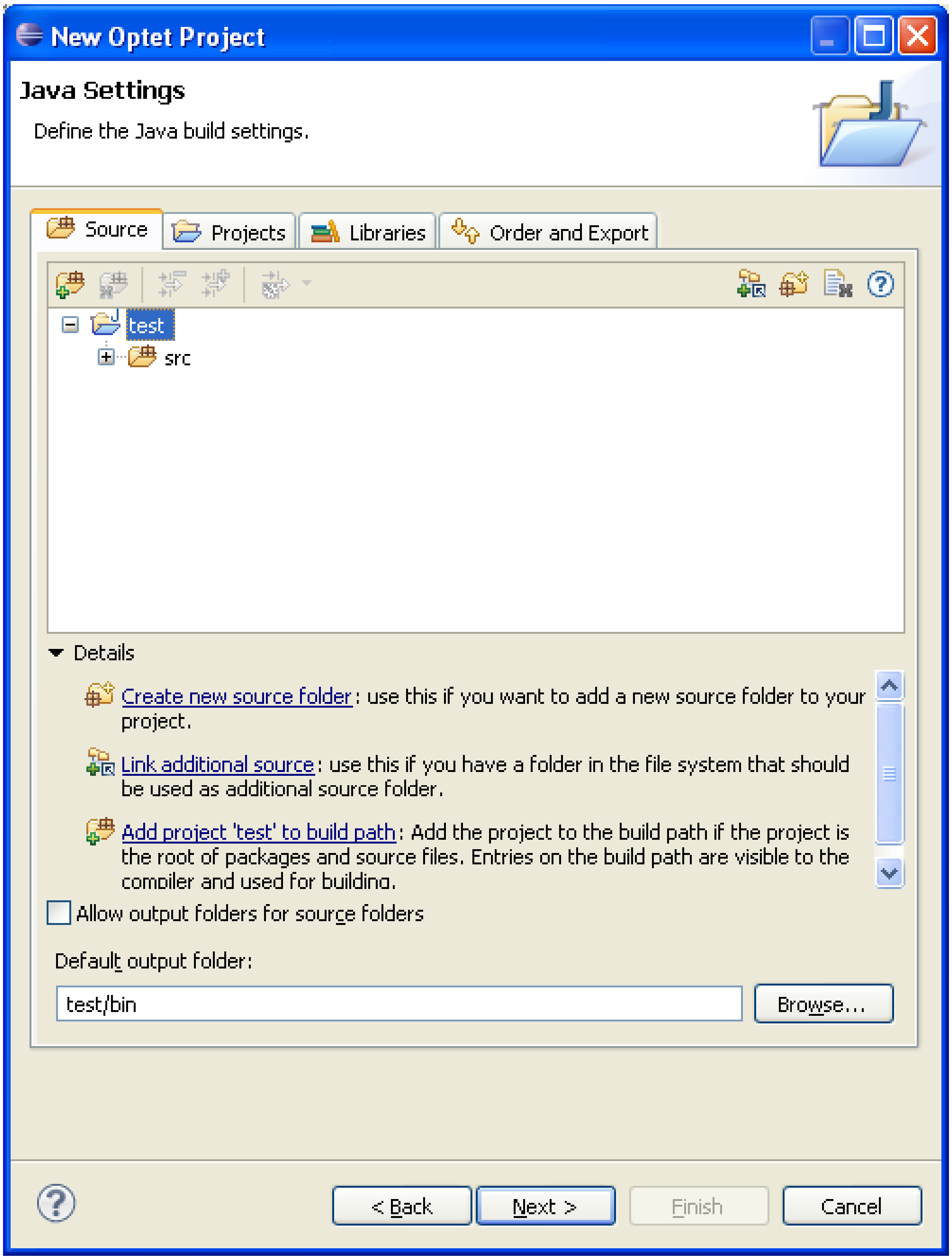
Use project folder as root for sources and class files

[● Create separate folders for sources and class files]

Working sets

[ ] Add project to working sets

Working sets:



Create new source folder: use this if you want to add a new source folder to your project.

Link additional source: use this if you have a folder in the file system that should be used as additional source folder.

Add project 'test' to build path: Add the project to the build path if the project is the root of packages and source files. Entries on the build path are visible to the compiler and used for building.

Allow output folders for source folders

Default output folder: test/bin

The last view links the future development with the SVN Optet project:

With this view, the developer is connected to the secure SVN repository containing the entire trustworthiness project. The developer selects the project he wants to develop. Using the list on the left, the developer browses through the entire created project and sees the description and the content of the SVN repository. In this example, just the CSM.xml, the TWProfile.xml and the readme.txt file are present. In this example, the checkout option shall not be set because the developer starts with an empty source folder. At the end, the project is created and is ready to be updated by the developer.

**2.2. From an existing SVN project

In this example, the developer must follow the development of an application already stored in the secure SVN repository. In this case, the principle is the same as previously except the phase of checkout. When the developer selects the project to extract, he selects the checkout option. At the end of the project, the sources of the development will be checked-out into the project workspace.

**2.3. From an existing development

The last example is when a development has already been done by the developer but was not attached to an Optet project. In this case, the developer must click on the Eclipse Toolbar optetMenu>Select project. The following view appears in order to give the opportunity to the developer to select the Optet project:

When the project is selected, the developer validates his choice and selects the Java project he wants to associate. For this, the next view appears:

The developer selects the project and validates the choice. After this, the current Java project is associated with the SVN Optet project defined in the secure SVN repository.

**2.4. Manually from an existing development

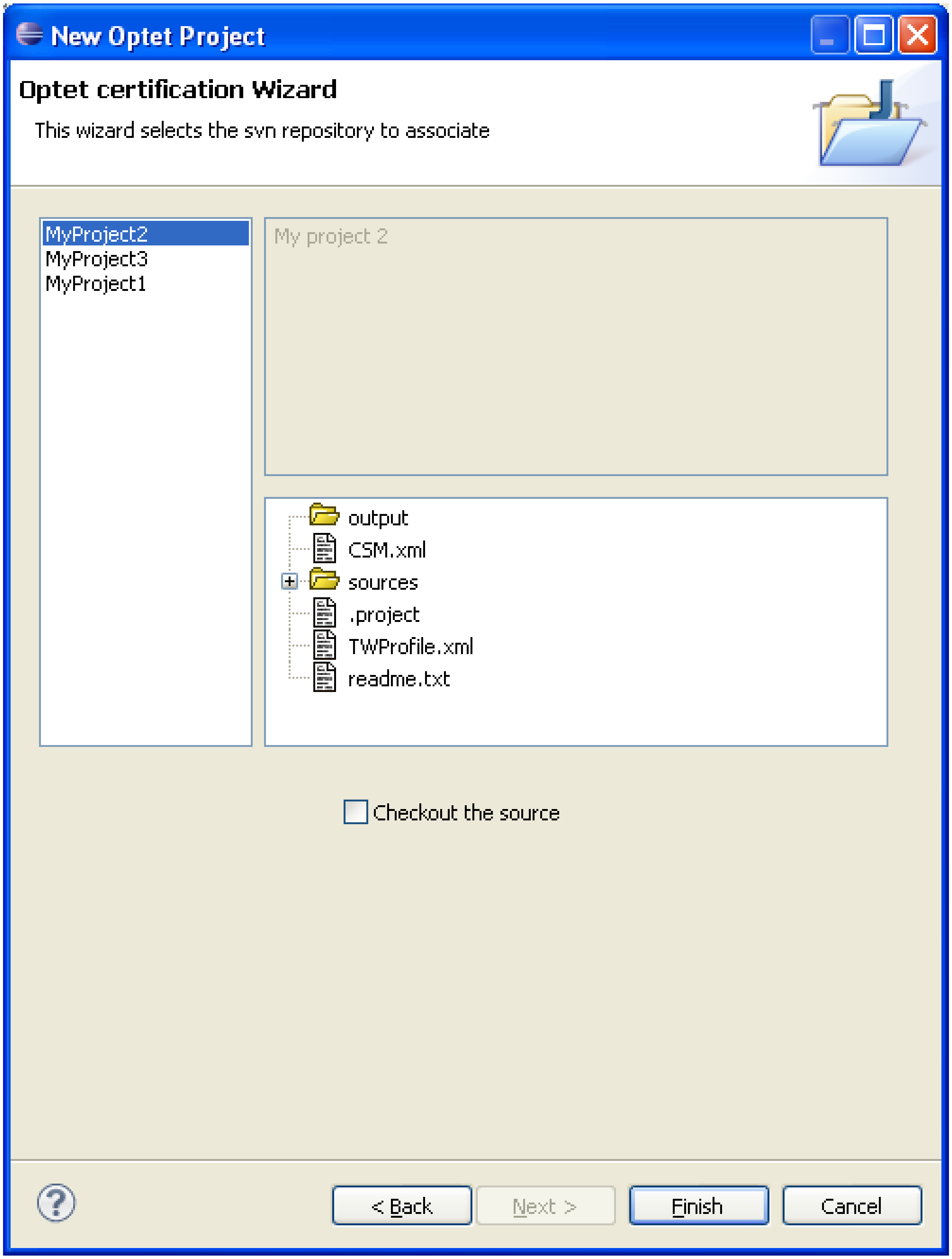
The TWProfile.xml and Optet.properties files must be created manually into an Optet directory on the project

The TWProfile.xml must contains de TWAttributes expected with the metric and excepted values. The format must be like this

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<TWProfile>
        <TWAttribute name="Maintainability" attributeID="44">
                <Metric name="documented elements" metricID="253" expectedValue="80"></Metric>
        </TWAttribute>
        <TWAttribute name="Flexibility/Robustness" attributeID="40">
                <Metric name="interceptet errors" metricID="235" expectedValue="100"></Metric>
        </TWAttribute>
        <TWAttribute name="Change Cycle/Stability" attributeID="25">
                <Metric name="Compliance with best programing practices" metricID="1154" expectedValu
                <Metric name="unit test coverage for stability" metricID="1153" expectedValue="60"></
                        </TWAttribute>
        <TWAttribute name="Reliability" attributeID="41">
                <Metric name="Reliability of Software" metricID="243" expectedValue="90"></Metric>
        </TWAttribute>
        <TWAttribute name="Software Complexity" attributeID="41">
                <Metric name="structure of the software" metricID="243" expectedValue="90"></Metric>
        </TWAttribute>
</TWProfile>
```

The Optet.properties file must be like this

```
#Optet properties
#Sat Sep 06 16:33:16 CEST 2014
project.type=java
svn.project.selected=
```



Optet certification Wizard

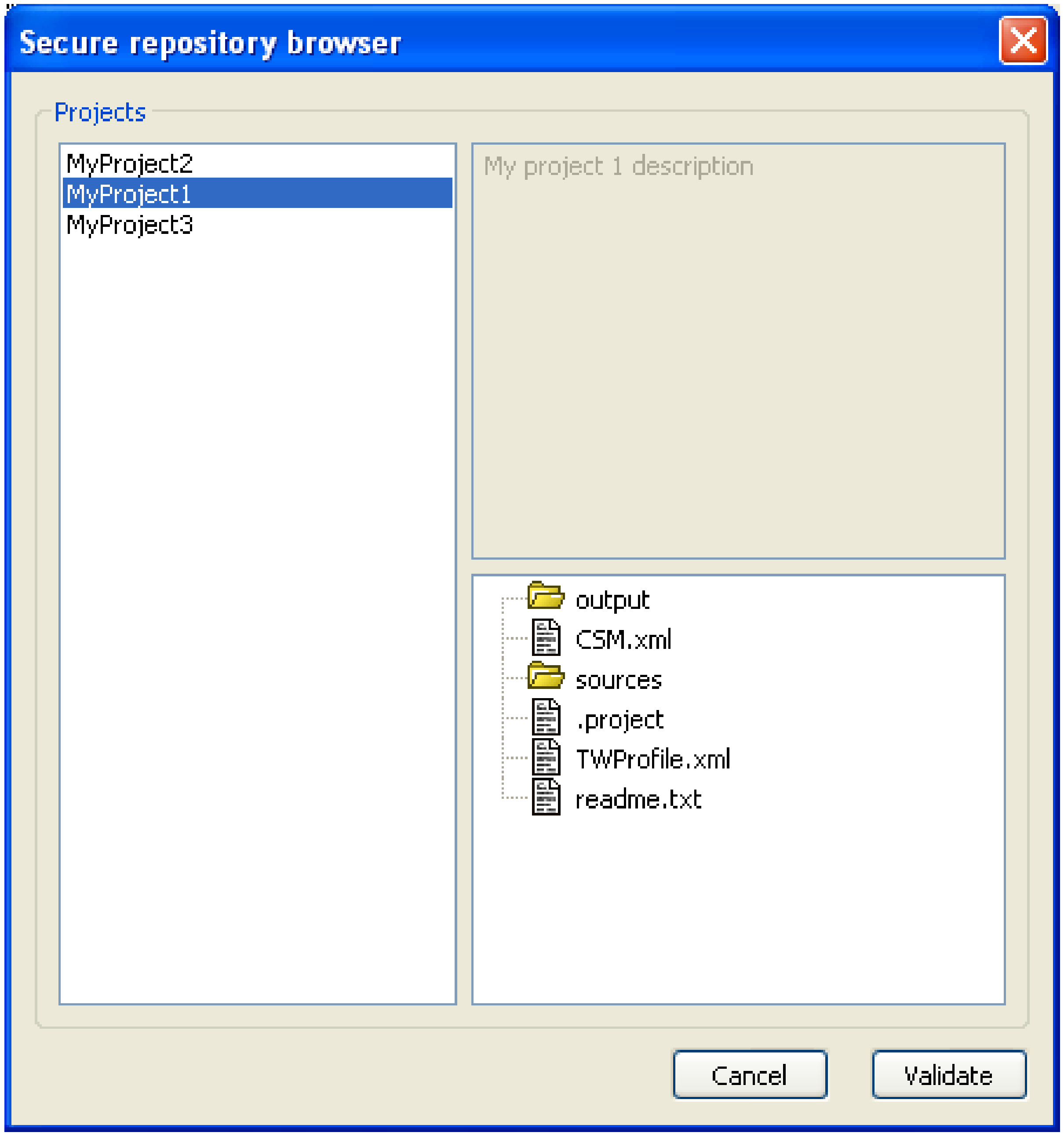
This wizard selects the svn repository to associate

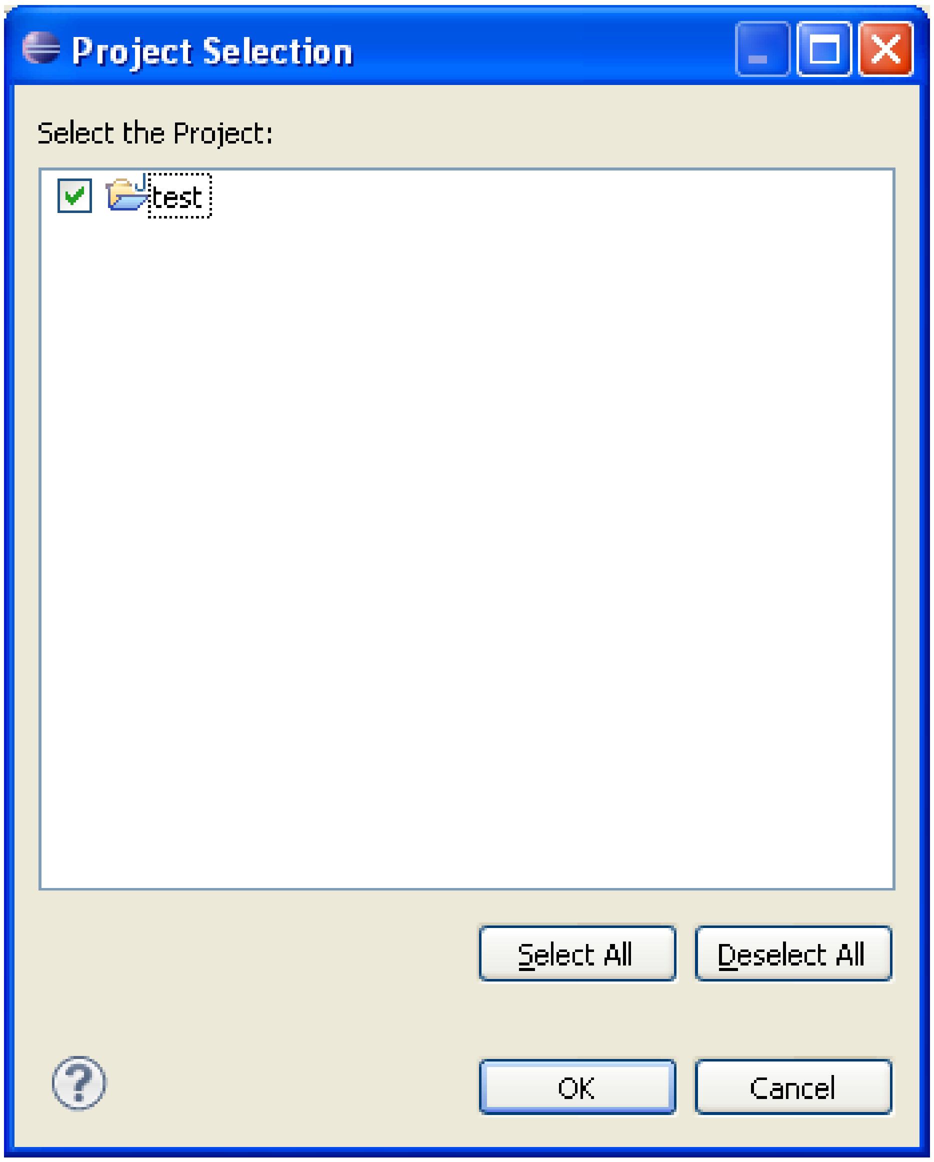
MyProject2
MyProject3
MyProject1

My project 2

output
CSM.xml
sources
.project
TWProfile.xml
readme.txt

Checkout the source


Secure repository browser
Projects
MyProject2
MyProject1
MyProject3
My project 1 description
output
CSM.xml
sources
.project
TWProfile.xml
readme.txt
Cancel
Validate

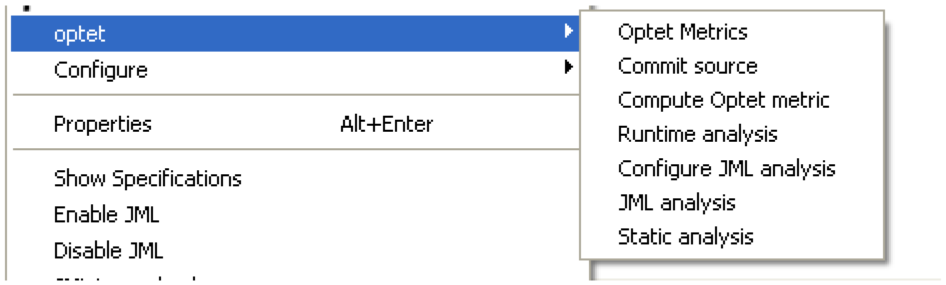

Project Selection
Select the Project:
test
Select All
Deselect All
OK
Cancel

**3. Java Development

After the creation of the java project, the developer write the source code of the application, he integrates different new components in order to be able to build the application he has to deliver. In this section, we consider that the developer has skills to write properly java source code

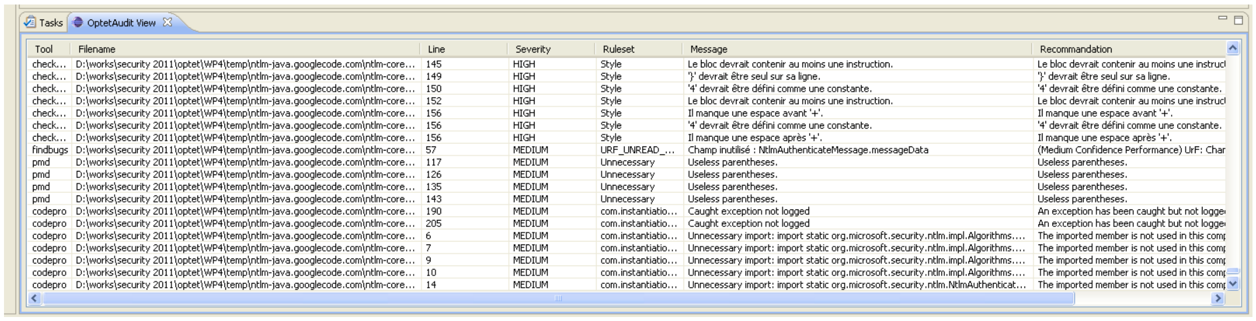
**4. The analysis phase

During the development of the component, the developer can run, at any time, the analysis (static and runtime) in order to evaluate the quality of his code with respect to the predefined trustworthiness requirements selected in the secure SVN repository. In order to launch the analysis project, the developer must use the contextual menu and select the desired analysis:



**4.1. The static analysis

When the developer selects the static analysis, all the sources are checked regarding the trustworthiness requirements associated with the project. As results, all the malformations detected into the code are displayed. The display of this is realized into the OptetAudit view as follows:



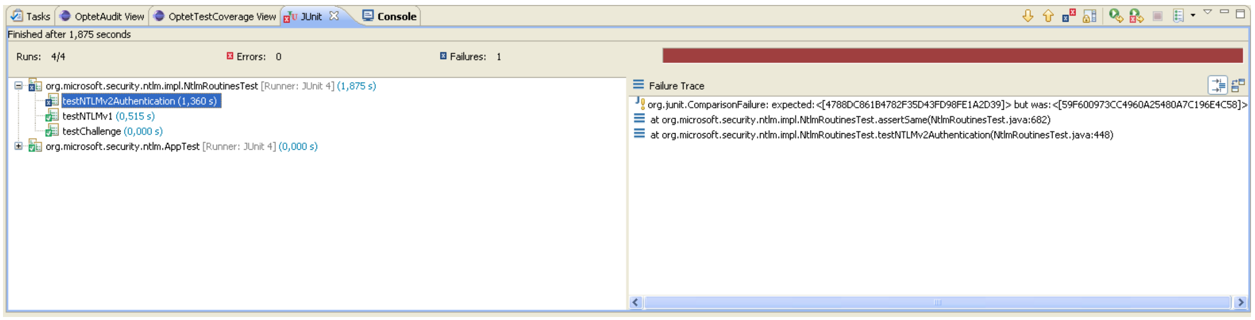
Tool	Filename	Line	Severity	Ruleset	Message	Recommandation
check…	D:\works\security 2011\optet\WP4\temp\ntlm-java.googlecode.com\ntlm-core…	145	HIGH	Style	Le bloc devrait contenir au moins une instruction.	Le bloc devrait contenir au moins une instruct
check…	D:\works\security 2011\optet\WP4\temp\ntlm-java.googlecode.com\ntlm-core…	149	HIGH	Style	'}' devrait être seul sur sa ligne.	'}' devrait être seul sur sa ligne.
check…	D:\works\security 2011\optet\WP4\temp\ntlm-java.googlecode.com\ntlm-core…	150	HIGH	Style	'4' devrait être défini comme une constante.	'4' devrait être défini comme une constante.
check…	D:\works\security 2011\optet\WP4\temp\ntlm-java.googlecode.com\ntlm-core…	152	HIGH	Style	Le bloc devrait contenir au moins une instruction.	Le bloc devrait contenir au moins une instruct
check…	D:\works\security 2011\optet\WP4\temp\ntlm-java.googlecode.com\ntlm-core…	156	HIGH	Style	Il manque une espace avant '+'.	Il manque une espace avant '+'.
check…	D:\works\security 2011\optet\WP4\temp\ntlm-java.googlecode.com\ntlm-core…	156	HIGH	Style	'4' devrait être défini comme une constante.	'4' devrait être défini comme une constante.
check…	D:\works\security 2011\optet\WP4\temp\ntlm-java.googlecode.com\ntlm-core…	156	HIGH	Style	Il manque une espace après '+'.	Il manque une espace après '+'.
findbugs	D:\works\security 2011\optet\WP4\temp\ntlm-java.googlecode.com\ntlm-core…	57	MEDIUM	URF_UNREAD_…	Champ inutilisé : NtlmAuthenticateMessage.messageData	(Medium Confidence Performance) UrF: Char
pmd	D:\works\security 2011\optet\WP4\temp\ntlm-java.googlecode.com\ntlm-core…	117	MEDIUM	Unnecessary	Useless parentheses.	Useless parentheses.
pmd	D:\works\security 2011\optet\WP4\temp\ntlm-java.googlecode.com\ntlm-core…	126	MEDIUM	Unnecessary	Useless parentheses.	Useless parentheses.
pmd	D:\works\security 2011\optet\WP4\temp\ntlm-java.googlecode.com\ntlm-core…	135	MEDIUM	Unnecessary	Useless parentheses.	Useless parentheses.
pmd	D:\works\security 2011\optet\WP4\temp\ntlm-java.googlecode.com\ntlm-core…	143	MEDIUM	Unnecessary	Useless parentheses.	Useless parentheses.
codepro	D:\works\security 2011\optet\WP4\temp\ntlm-java.googlecode.com\ntlm-core…	190	MEDIUM	com.instantiatio…	Caught exception not logged	An exception has been caught but not logge
codepro	D:\works\security 2011\optet\WP4\temp\ntlm-java.googlecode.com\ntlm-core…	205	MEDIUM	com.instantiatio…	Caught exception not logged	An exception has been caught but not logge
codepro	D:\works\security 2011\optet\WP4\temp\ntlm-java.googlecode.com\ntlm-core…	6	MEDIUM	com.instantiatio…	Unnecessary import: import static org.microsoft.security.ntlm.impl.Algorithms.…	The imported member is not used in this comp
codepro	D:\works\security 2011\optet\WP4\temp\ntlm-java.googlecode.com\ntlm-core…	7	MEDIUM	com.instantiatio…	Unnecessary import: import static org.microsoft.security.ntlm.impl.Algorithms.…	The imported member is not used in this comp
codepro	D:\works\security 2011\optet\WP4\temp\ntlm-java.googlecode.com\ntlm-core…	9	MEDIUM	com.instantiatio…	Unnecessary import: import static org.microsoft.security.ntlm.impl.Algorithms.…	The imported member is not used in this comp
codepro	D:\works\security 2011\optet\WP4\temp\ntlm-java.googlecode.com\ntlm-core…	10	MEDIUM	com.instantiatio…	Unnecessary import: import static org.microsoft.security.ntlm.impl.Algorithms.…	The imported member is not used in this comp
codepro	D:\works\security 2011\optet\WP4\temp\ntlm-java.googlecode.com\ntlm-core…	14	MEDIUM	com.instantiatio…	Unnecessary import: import static org.microsoft.security.ntlm.NtlmAuthenticat…	The imported member is not used in this comp

In this view, the developer can see:

- The tool which detects the problem;
- The file involved;
- The line where the problem is;
- The severity of the problem;
- The rule set;
- The message of the problem;
- The recommendation in order to solve it.

**4.2. The runtime analysis (Due to a license problem, the runtime analysis is desactivated)

When the developer selects the runtime analysis, the result is the execution of the unit tests present in the project. This allows the computation of some metrics like the coverage. The data is displayed using the result of the JUnit execution:

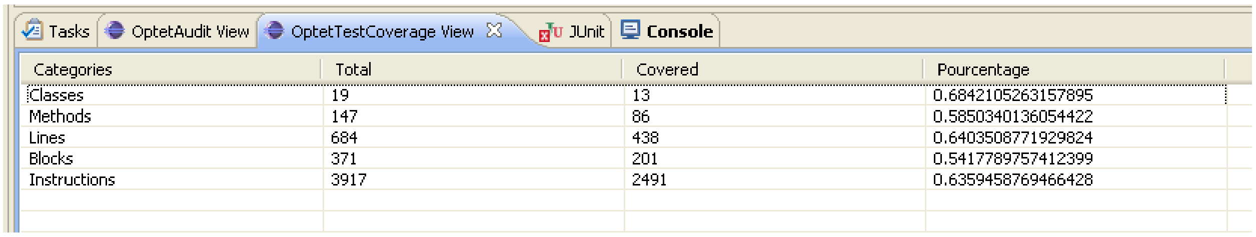


```
Finished after 1,875 seconds
Runs:   4/4          Errors:   0          Failures:   1

org.microsoft.security.ntlm.impl.NtlmRoutinesTest [Runner: JUnit 4] (1,875 s)
    testNTLMv2Authentication (1,360 s)
    testNTLMv1 (0,515 s)
    testChallenge (0,000 s)
org.microsoft.security.ntlm.AppTest [Runner: JUnit 4] (0,000 s)

org.junit.ComparisonFailure: expected:<[4788DC861B4782F35D43FD98FE1A2D39]> but was:<[59F600973CC4960A25480A7C196E4C58]>
 at org.microsoft.security.ntlm.impl.NtlmRoutinesTest.assertSame(NtlmRoutinesTest.java:682)
 at org.microsoft.security.ntlm.impl.NtlmRoutinesTest.testNTLMv2Authentication(NtlmRoutinesTest.java:448)
```

For the coverage part, the results regarding the number of classes, methods, lines, etc... are displayed in another view, the OptetTestCoverage view:



Categories	Total	Covered	Pourcentage
Classes	19	13	0.6842105263157895
Methods	147	86	0.5850340136054422
Lines	684	438	0.6403508771929824
Blocks	371	201	0.5417789757412399
Instructions	3917	2491	0.6359458769466428

**4.3. Optet Metric computation phase

During this phase, a full analysis (static and runtime) is realised and computed regarding the required TWAttribute needed for this development. The output of the different analysis are computed using the method defined in the [7](To resume, the different evidence have weight with predefined values in order to calculate the TWAttribute value).

The evidences detected by the factory have the following forms:

- The unit test coverage : based on the execution of the unit test, the ratio number of method tested/ the total of all the methods are computed
- The comment ratio : use the code analysis , the ratio of comments to the number of lines of code is computed
- The Unit test ration : The number of test OK / The total of Tests is computed after the junit tests
- The Rules : For a dedicated TWAttributes, a list of rules configures the static analysis plugins (PMD, Checkstyle and findbugs). When a rule is violated by the code, this rule is considered as failed. The computation is realised by using the number of violated rules over the total number of rules checked for the TWAtrribute.

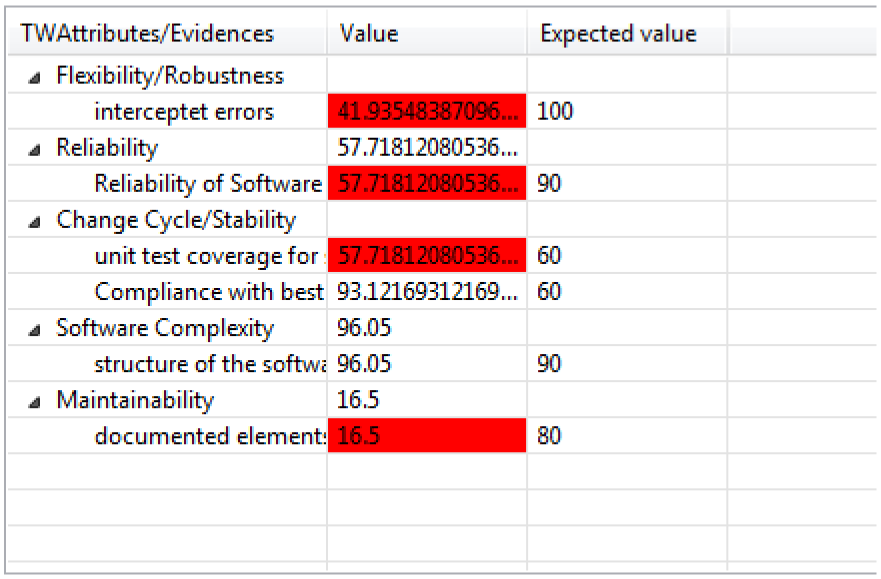
Using these inputs, the computation could be realised by using the weight assigned to the different evidence. The configuration of the different evidence weight is configurable by using a specific configuration file. However, nobody must change it. The result of this computation is display in the dedicated view ?OptetMetric view? like this:

The specific case of the Availability is due to no evidence are found in order to compute the result.

**5. Commit phase

When the developer considers that his code is compliant with the trustworthiness requirements, he stores his code into the secure SVN repository. For this, he selects the ?commit source? entry present into the contextual menu. Using this command, the commit is realised in the sources directory of the Optet project (in the secure SVN repository associated). In order to just commit the required files, this view is displayed:

With this view, the developer could select each file on which make a commitment.



TWAttributes/Evidences	Value	Expected value
◢ Flexibility/Robustness		
interceptet errors	41.93548387096...	100
◢ Reliability	57.71812080536...	
Reliability of Software	57.71812080536...	90
◢ Change Cycle/Stability		
unit test coverage for	57.71812080536...	60
Compliance with best	93.12169312169...	60
◢ Software Complexity	96.05	
structure of the softwa	96.05	90
◢ Maintainability	16.5	
documented element:	16.5	80

2.5 Certification of a Java application

**1. Selection of the project to be certified

The certifier needs to select a project from the secure SVN repository in order to certify it. In this case, the operation to load the project is the same as for the developer part (the case ?From an existing SVN project?). This operation is the same for all kinds of project (Java, Web, mobile)

**2. Certification process

To certify a component, a dashboard is dedicated in order to guide the certifier in the process. This dashboard is called using contextual menu of the project.

The following view appears:

For each step, a dedicated view helps the certifier to enter the required data. All the data is required for the successful completion of the certificate generation.

**2.1. The software description

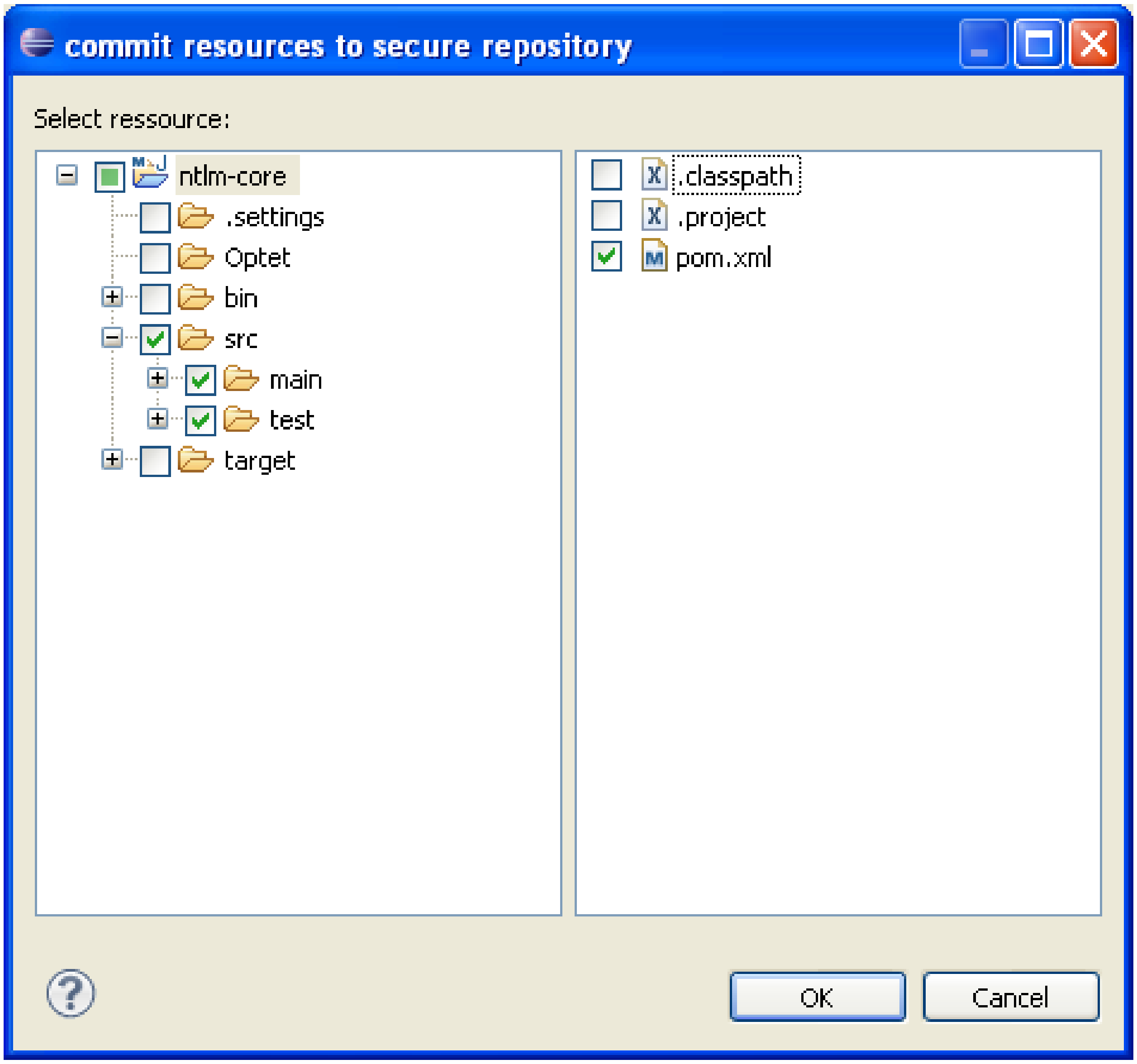
This view helps the certifier to describe the system and to define the certification perimeter:

In this view, from the root element, we can create a subcomponent and associate either other components, or an attribute or a stakeholder to this component.

- The Component: a component could be associated with another component, some attribute and some stakeholder.
- The attributes: For the attributes, the certifier may select their type as InputParameter or OutputParameter and choose if this attribute is selected for the evaluation;
- The Stakeholder: For the stakeholder, the certifier may select the type as End-user, System or Service. All the defined stakeholders are displayed in the stakeholders list for reuse if needed.

**2.2. The Trustworthiness problem definition

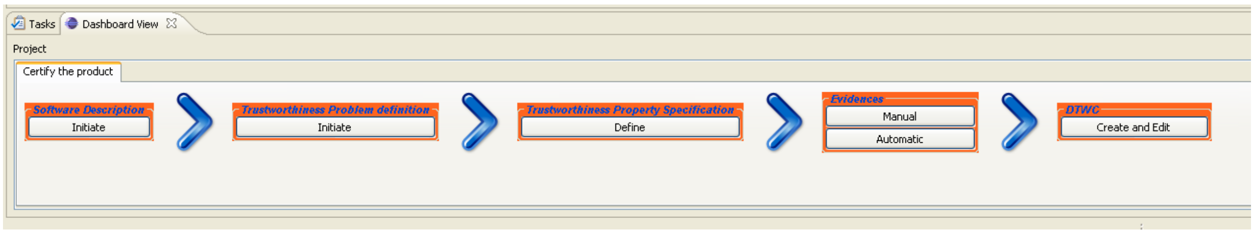
In this second phase, the certifier associates each asset present in the TOE with the potential problems to which the asset might be exposed.

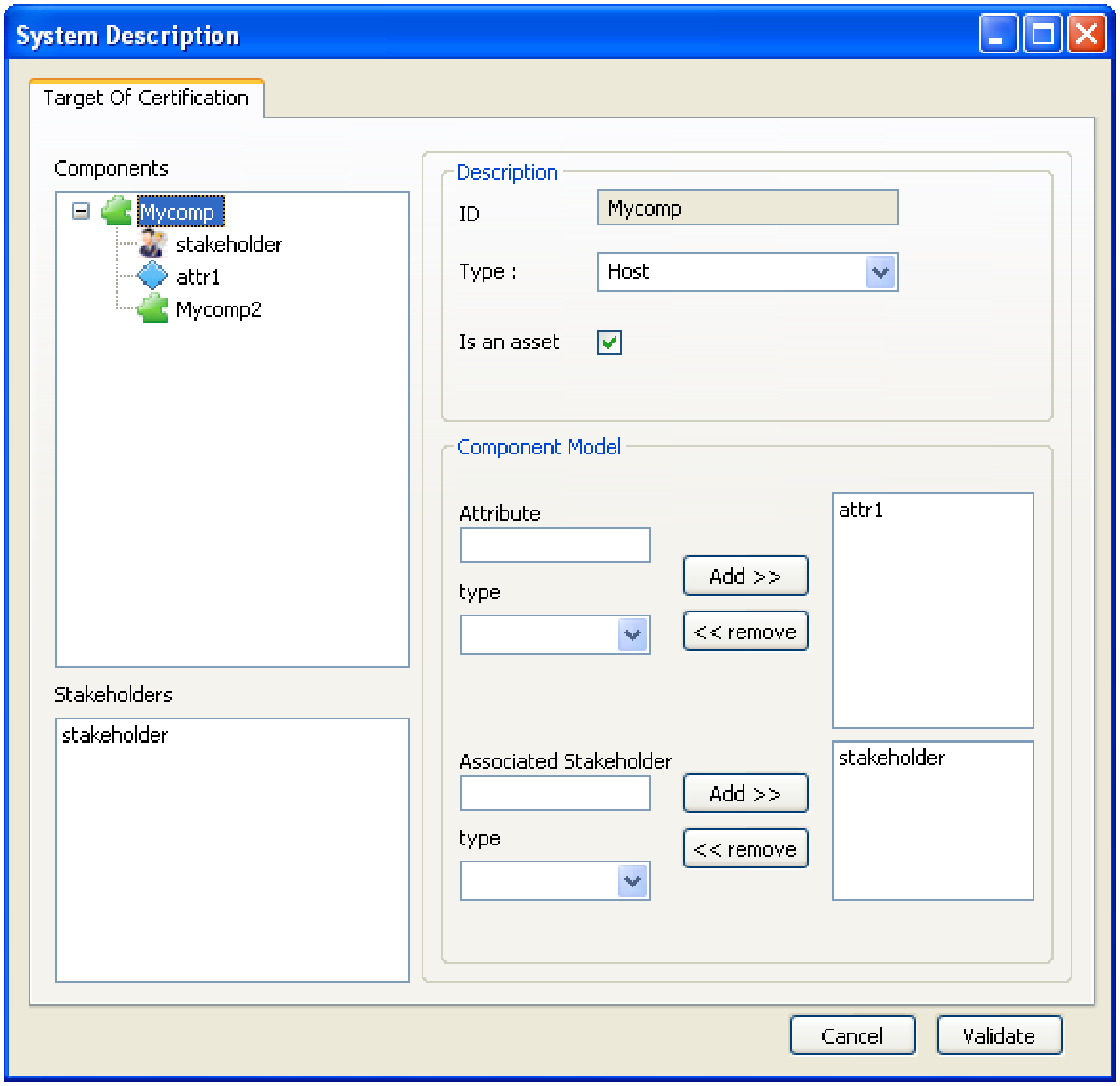


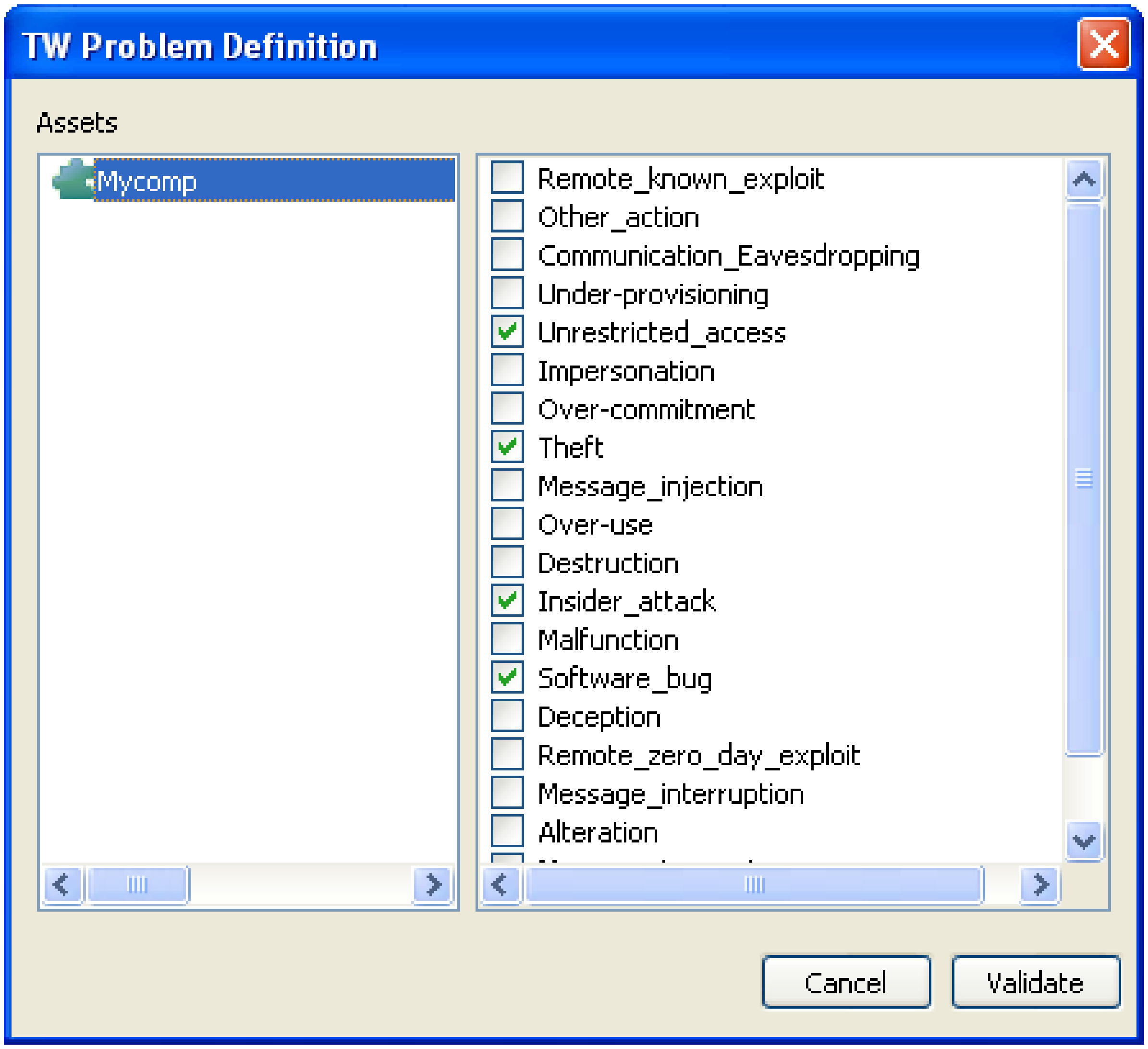
Select ressource:

- ntlm-core
  - [ ] .settings
  - [ ] Optet
  - [ ] bin
  - [✓] src
    - [✓] main
    - [✓] test
  - [ ] target

- [ ] .classpath
- [ ] .project
- [✓] pom.xml




System Description
Target Of Certification
Components
Mycomp
stakeholder
attr1
Mycomp2
Stakeholders
stakeholder
Description
ID
Mycomp
Type :
Host
Is an asset
✓
Component Model
Attribute
type
Add >>
<< remove
attr1
Associated Stakeholder
type
Add >>
<< remove
stakeholder
Cancel
Validate

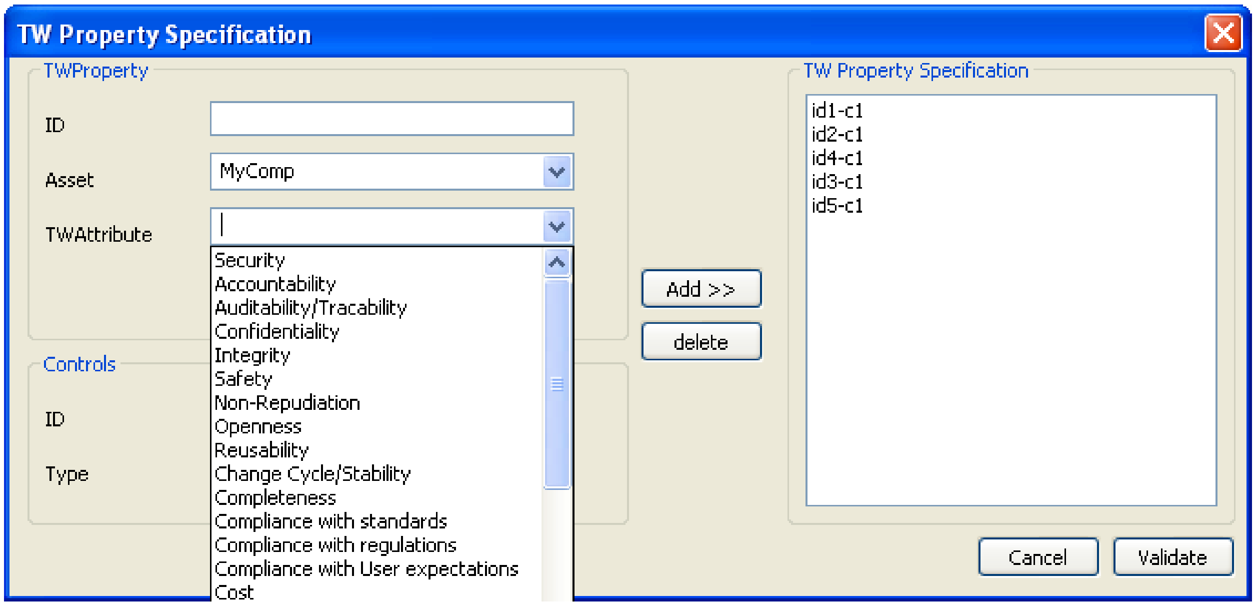


- [ ] Remote_known_exploit
- [ ] Other_action
- [ ] Communication_Eavesdropping
- [ ] Under-provisioning
- [✓] Unrestricted_access
- [ ] Impersonation
- [ ] Over-commitment
- [✓] Theft
- [ ] Message_injection
- [ ] Over-use
- [ ] Destruction
- [✓] Insider_attack
- [ ] Malfunction
- [✓] Software_bug
- [ ] Deception
- [ ] Remote_zero_day_exploit
- [ ] Message_interruption
- [ ] Alteration

For the selected asset, the associated problems checked by the certifier will be also published in the DTWC. (The list of the possible problems is not already defined).

**2.3. The Trustworthiness property specification

In the next step, the certifier associates each asset present in the TOE with a list of TWAttributes defined in the system:



For this version the possible TWAttributes are:

- Flexibility/Robustness (Intercepted errors)
- Reliability (Reliability of Software)
- Maintainability (Documented functions)
- Change Cycle/Stability (unit test coverage for stability, Compliance with best programing practices)
- Software Complexity (structure of the software)

On the right side, all the TW property specification created by the certifier appears. These TWproperties can be deleted if there is a need to.

**2.4. The evidence

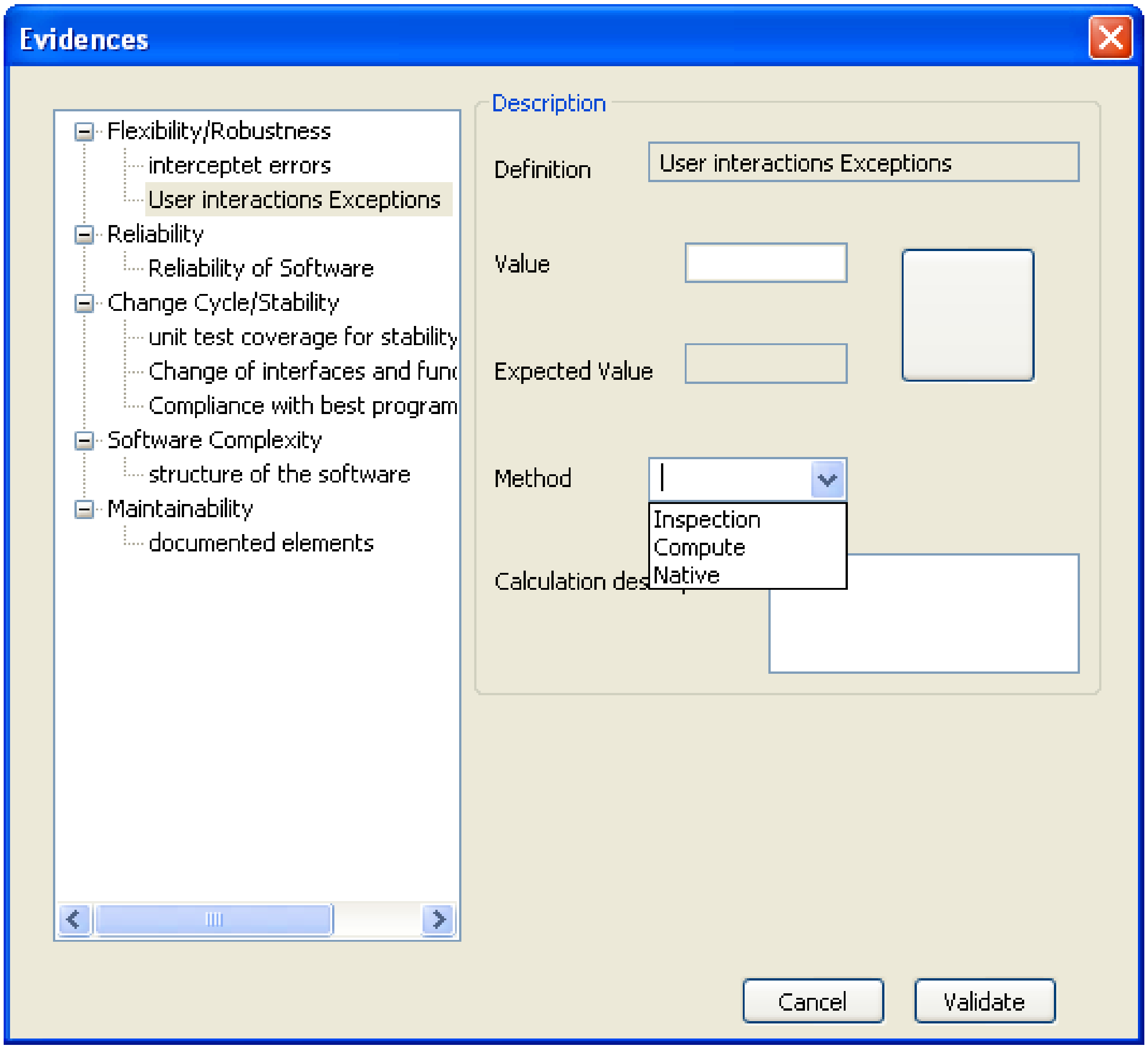
At this stage, the certifier checks the code in order to find evidence to fill the data related to the trustworthiness attributes defined for the system in the evaluation report. The certifier has two possibilities:

- Manual evaluation: In this case, the certifier must enter manually all the metrics values required by the certification process
- Automatic evaluation: In this case, the certifier run automatically the static and runtime analysis in order to extract evidences. The computation of all the evidences found will use to fulfill the required TWAttributes metrics

**2.4.1. Manual evaluation

When the certifier selects manual evaluation, the following view appears:

For each TWAttribute and the associated metrics, the user sets a value and the method used to find this value. The possible methods are:



Definition: User interactions Exceptions

Value

Expected Value

Method

Calculation des

- Compute: if the certifier uses an external tool to evaluate the metric;
- Inspection: if the certifier explores all the files manually in order to evaluate the metric;
- Native: if the language properties used to develop the application imposes the metric.

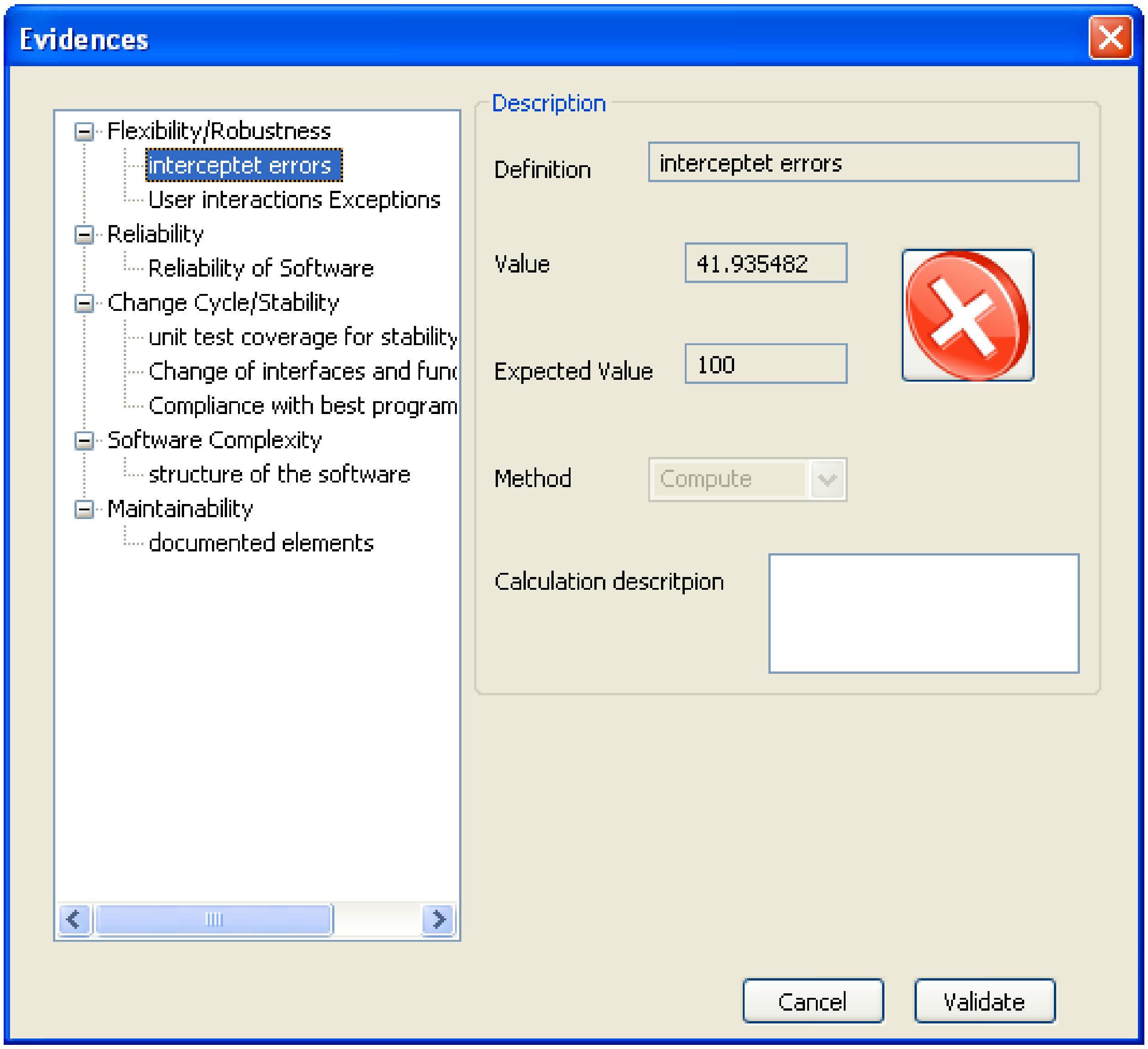
The certifier must set all the values required.

In this view, the expected value for the metric is displayed indicating the required value. A specific icon helps to show rapidly if the computed value is under/over the expected value.

**2.4.2. Automatic evaluation

In this case, the same view appears but with some not modifiable values. The values are coming from the automatic evaluation made when the certifier clicks on this ?Automatic? button.

When the values are coming from the automatic evaluation, the certifier can?t modify them. But, if the automatic evaluation can?t fulfill some attribute due to missing information, the certifier must enter the value manually.


Evidences

- Flexibility/Robustness
  - interceptet errors
  - User interactions Exceptions
- Reliability
  - Reliability of Software
- Change Cycle/Stability
  - unit test coverage for stability
  - Change of interfaces and func
  - Compliance with best program
- Software Complexity
  - structure of the software
- Maintainability
  - documented elements

Description

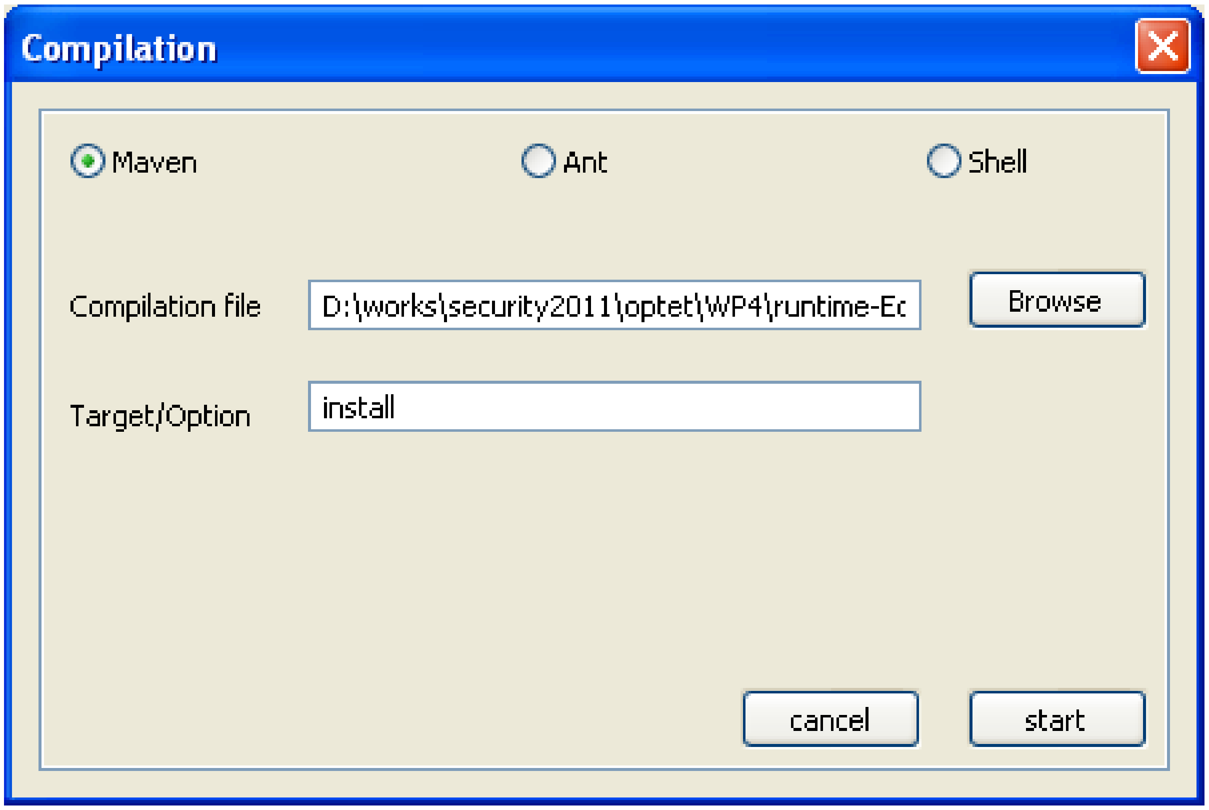
Definition	interceptet errors
Value	41.935482
Expected Value	100
Method	Compute
Calculation descritpion	

**2.5. The DTWC[2] generation

This last step is carried out when all the previous steps are done. During this phase:

- The product is compiled

The compilation of the product is realised using the following interface



Compilation file: D:\works\security2011\optet\WP4\runtime-Ec

Target/Option: install

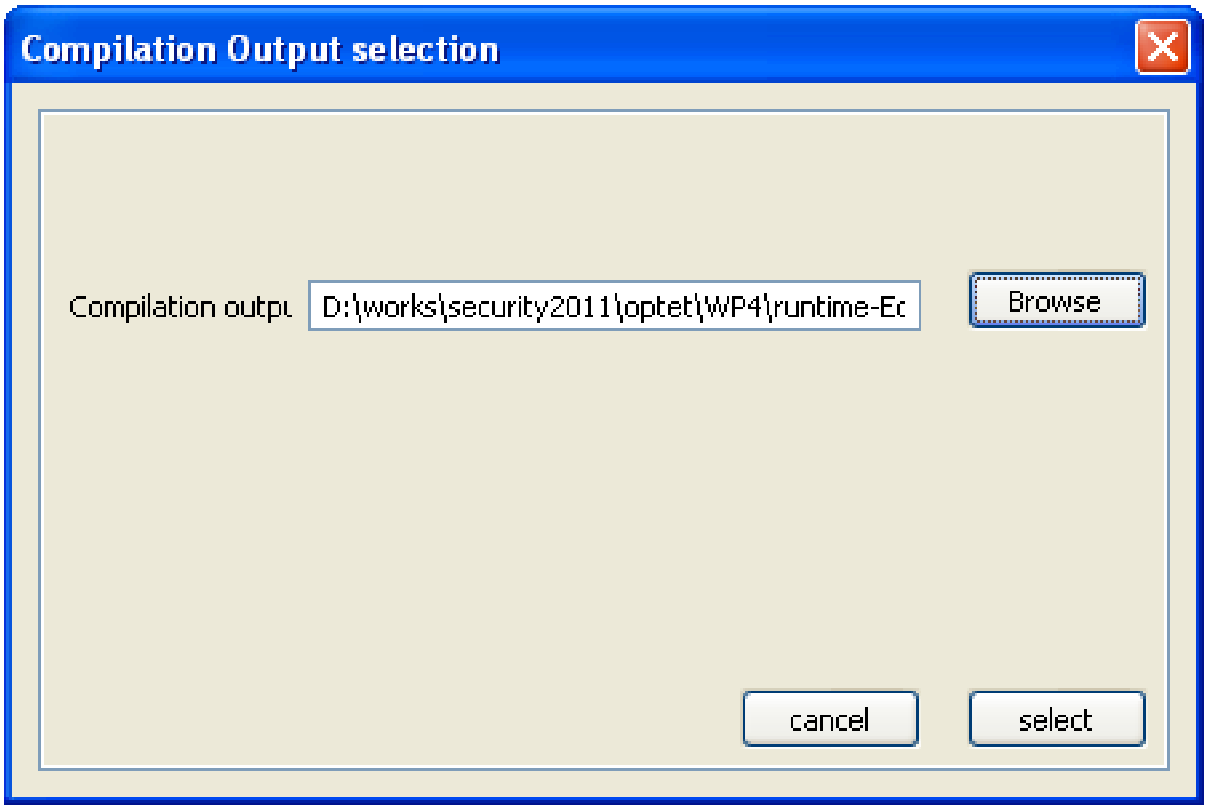
Using this interface, the certifier can select 3 possibilities to build the product: Maven, Ant or a Shell script. These three kinds of build are the common way to build a product into eclipse.

Then, the certifier must select the build file using the browse button and indicate the target option defined into the build file.

When the build is finished, the certifier must select the compiled product.

This product will be the element pushed by the technology provider to the marketplace:

- The DTWC is generated using the data collected during the certification workflow, the hash of the product is inserted into the certificate and signed with the certifier?s certificate defined in the preference pages of the IDE;
- The DTWC is transferred to the web service responsible to host all the certificate files.



Compilation outpu D:\works\security2011\optet\WP4\runtime-Ec

CHAPTER 3

FIWARE Trustworthy Factory - Installation and Administration Guide

3.1 Trustworthy Factory Installation

This guide tries to define the procedure to install the Trustworthy Factory in a machine, including its requirements and possible troubleshooting that we could find during the installation.

3.1.1 Requirements

The pre-requisite for the java factory is a Windows OS and a JDK1.7. The JAVA_HOME and the PATH must be set up according to the JDK installation directory.

3.2 Installation

Two possible installation procedures could be used to have the Trustworthy Factory ready to use

3.2.1 The Zip installation

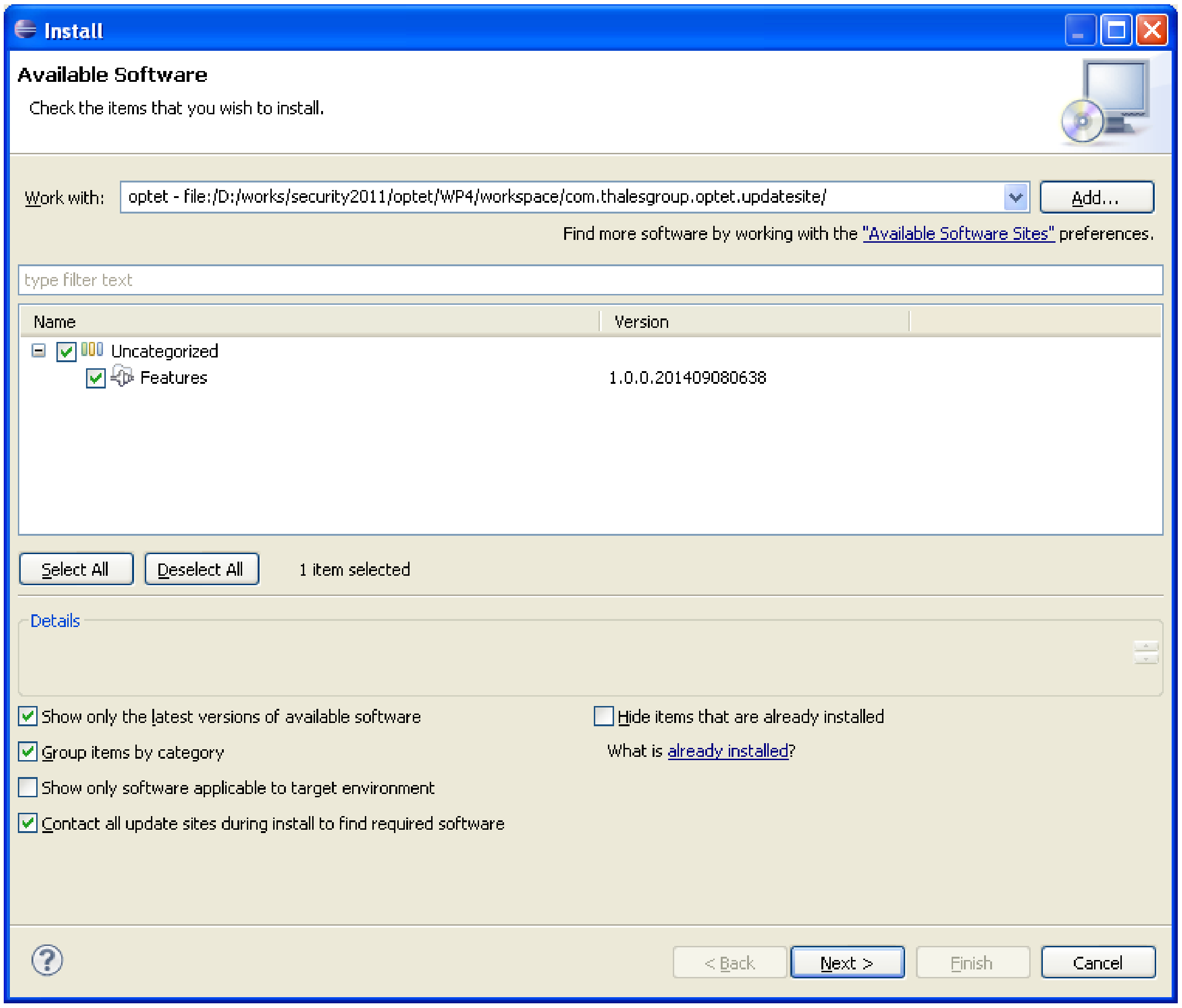
The simplest solution is to take the Zip installation provided to extract the complete factory already configured for the development of Java application. For that, recover the Zip archive and unzip it. The factory is ready to use.

3.2.2 The Eclipse update

This installation is based on a virgin eclipse installation where all the plugins must be deployed. For that, an Kepler eclipse version must be used. The process is to use the update site of all the plugins in order to build the java factory. In eclipse HMI, the software installation view must be opened : Help>Install new software

The installation of a new plugin is realised by following the 3 steps:

- For each plugin, one must create a new update site entry using the "Add" button and fulfill the required information:
- The name of the update site
- The URL of the update site



Available Software

Check the items that you wish to install.

Work with: optet - file:/D:/works/security2011/optet/WP4/workspace/com.thalesgroup.optet.updatesite/

Find more software by working with the "Available Software Sites" preferences.

Name	Version
✓ Uncategorized	
✓ Features	1.0.0.201409080638

1 item selected

- ✓ Show only the latest versions of available software
- ✓ Group items by category
- ☐ Show only software applicable to target environment
- ✓ Contact all update sites during install to find required software
- ☐ Hide items that are already installed

What is already installed?

For the selected update site, we must select the package to install. Depending on the installed plugin, a selection of features is displayed. The user must select the features of the plugin required for the installation. In the Trustworthy Factory context, the selection required is described later for each plugin. # After the selection, the usage licence must be accepted in order to complete the installation. In the Trustworthy Factory case, the following plugin must be installed:

Tool	Update site
checkstyle	http://eclipse-cs.sf.net/update/
PMD	http://sourceforge.net/projects/pmd/files/pmd-eclipse/update-site/
findbugs	http://findbugs.cs.umd.edu/eclipse
Subversive	http://download.eclipse.org/technology/subversive/2.0/update-site/
Subversive	http://community.polarion.com/projects/subversive/download/eclipse/4.0/update-s
maven	http://download.eclipse.org/technology/m2e/releases/1.3/1.3.1.20130219-1424
Eclemma	http://update.eclemma.org/

Then, the Trustworthy Factory update site must be installed to complete the installation. The installation is realised using the URL update site of the trustworthy factory (https://github.com/TrustworthyFactory/UpdatesiteFactory).

At the end of the installation procedure, the eclipse.ini file must be modified. The line -product org.eclipse.platform.ide must be added/replaced in this file. The eclipse platform must be restarted at the end of the installation in order to activate all the options.

3.3 Diagnosis Procedures

The factory is based on Eclipse which provides error managements and diagnostic element. The factory follows this philosophy providing :

- Error Dialogs: The factory displays messages using message dialogues to display information to the user (errors or informations)
- **Log information: the factory has a log file where problems are recorded. The log file can be found in a couple of places:**
  - Workspace log - This is the most common location for the log file, It is stored in your workspace in the meta-data directory. Check out workspace/.metadata/.log.
  - Configuration log - Sometimes information is record in the configuration log file instead of the workspace log. (especially if the workspace hasn't been created yet, there isn't one, or it cannot be created) Check your configuration area for a configuration log file. (configuration/<timestamp>.log)
- Console errors: Occasionally problems happen in the system really early before there is a workspace and before there is a configuration area. This means that there is nowhere to write the log file so information is written to the console. For that launch the factory using the eclipsec.exe executable instead of eclipse.exe.

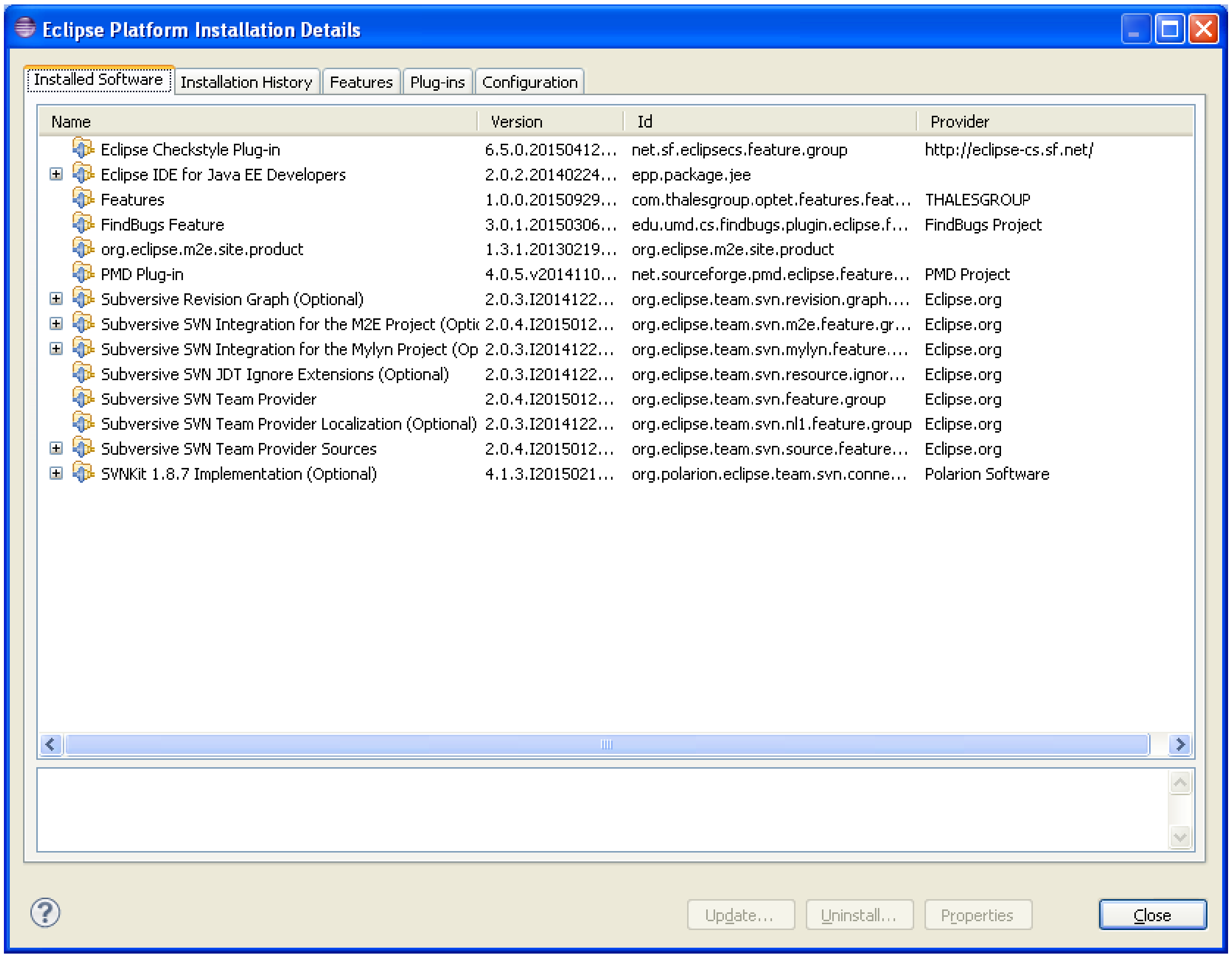
3.4 Sanity check procedures

The Sanity Check Procedures are the steps that a System Administrator will take to verify that an installation is ready to be tested. This is therefore a preliminary set of tests to ensure that obvious or basic malfunctioning is fixed before proceeding to unit tests, integration tests and user validation.

3.4.1 End to End testing

In the case of the Optet plugins, the different elements are signed in order to keep the consistency of the different elements, especially the configuration elements.

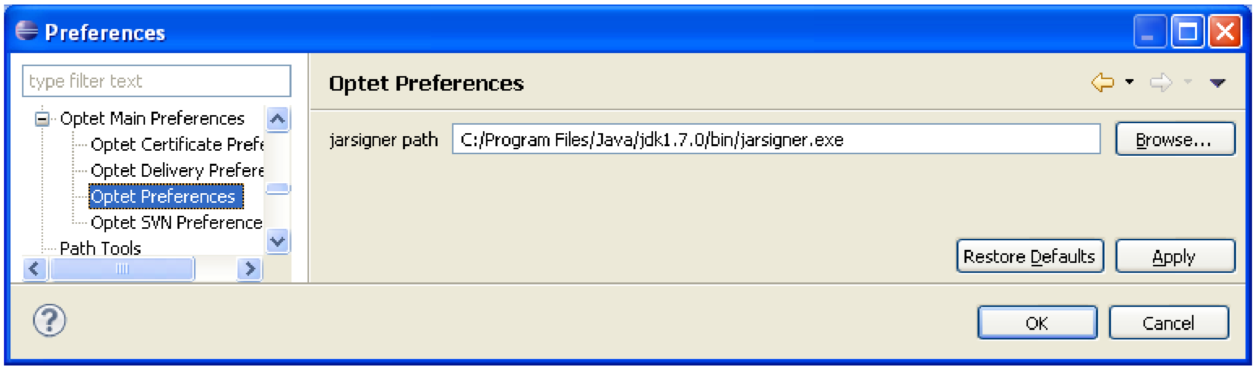
1. Installation inspection Le following plug-ins list must be check to in the factory to check the completude of the installation. For that, in the "About eclipse platform", check is the folowing mandatory plug-ins are present:



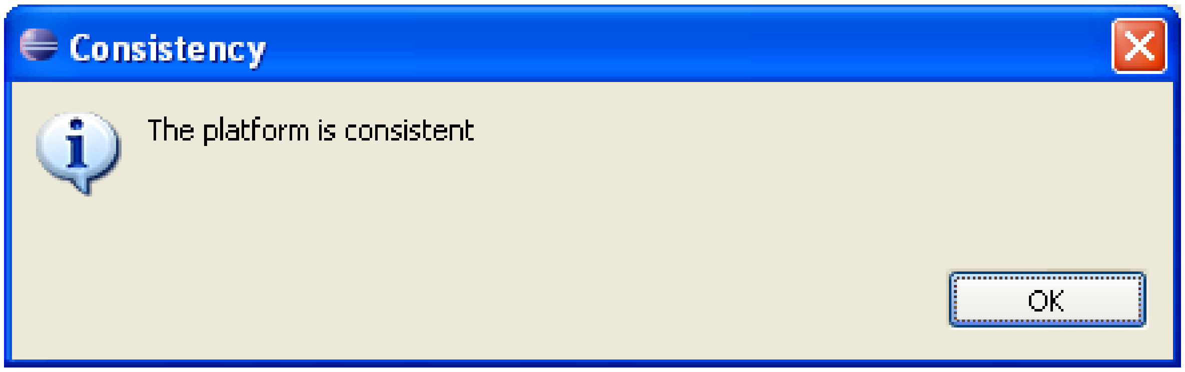
Name	Version	Id	Provider
Eclipse Checkstyle Plug-in	6.5.0.20150412...	net.sf.eclipsecs.feature.group	http://eclipse-cs.sf.net/
Eclipse IDE for Java EE Developers	2.0.2.20140224...	epp.package.jee	
Features	1.0.0.20150929...	com.thalesgroup.optet.features.feat...	THALESGROUP
FindBugs Feature	3.0.1.20150306...	edu.umd.cs.findbugs.plugin.eclipse.f...	FindBugs Project
org.eclipse.m2e.site.product	1.3.1.20130219...	org.eclipse.m2e.site.product	
PMD Plug-in	4.0.5.v2014110...	net.sourceforge.pmd.eclipse.feature...	PMD Project
Subversive Revision Graph (Optional)	2.0.3.I2014122...	org.eclipse.team.svn.revision.graph....	Eclipse.org
Subversive SVN Integration for the M2E Project (Optic	2.0.4.I2015012...	org.eclipse.team.svn.m2e.feature.gr...	Eclipse.org
Subversive SVN Integration for the Mylyn Project (Op	2.0.3.I2014122...	org.eclipse.team.svn.mylyn.feature....	Eclipse.org
Subversive SVN JDT Ignore Extensions (Optional)	2.0.3.I2014122...	org.eclipse.team.svn.resource.ignor...	Eclipse.org
Subversive SVN Team Provider	2.0.4.I2015012...	org.eclipse.team.svn.feature.group	Eclipse.org
Subversive SVN Team Provider Localization (Optional)	2.0.3.I2014122...	org.eclipse.team.svn.nl1.feature.group	Eclipse.org
Subversive SVN Team Provider Sources	2.0.4.I2015012...	org.eclipse.team.svn.source.feature...	Eclipse.org
SVNKit 1.8.7 Implementation (Optional)	4.1.3.I2015021...	org.polarion.eclipse.team.svn.conne...	Polarion Software

2. Configuration consistency check tool The consistency check is based on the verification of the signature of the different plugins. In order to check the signature, he can use the jarsigner process provided by the JDK. A specific preference page is provided into the Optet configuration to configure the path of this jarsigner.

3. Run the consistency check In the Optet Presentation page, a button called "check consistency" can be used to check the consistence of the eclipse installation. The check realised is the signature verification of all the signed plugins present into the eclipse installation directory. If the consistency is Ok, the following message appears:



jarsigner path C:/Program Files/Java/jdk1.7.0/bin/jarsigner.exe



The platform is consistent

Otherwise, the error message will be displayed.

3.4.2 List of Running Processes

Using the windows Task Manager, check that an “eclipse” process in runing

3.4.3 Network interfaces Up & Open

The eclipse platform must be configure to access internet in order to install plugins and run compilation. For that, use the eclipse tutorial in order to configure the internet access

3.4.4 Databases

No Database

3.5 Diagnosis Procedures

The Diagnosis Procedures are the first steps that a System Administrator will take to locate the source of an error in a GE. Once the nature of the error is identified with these tests, the system admin will very often have to resort to more concrete and specific testing to pinpoint the exact point of error and a possible solution. Such specific testing is out of the scope of this section.

3.5.1 Resource availability

The resource availability in the node should be at least 4Gb of RAM and 8GB of Hard disk in order to prevent enabler's bad performance in both nodes. This means that bellow these thresholds the enabler is likely to experience problems or bad performance.

3.5.2 Remote Service Access

This GE can't be access remotly.

3.5.3 Resource consumption

The factory is vbased on eclipse which is well know to have a bad memory management. The memory consumption can vary from 250M to 1G Ram. The CPU consuption is depending of the usage (édition, compilation, code generation, execution, etc...)

3.5.4 I/O flows

Input flow: No input flow must arrived to the factory

output flows: The factory uses internet connection in order to access plugin repository and libraries repository. The required port are 80 and 443. Following the user usage, some specific repository must be access and the port is dependant of the repository specification