|  |
| --- |
| **OCL conf checker**  **SOFTWARE USER MANUAL** |

|  |  |  |
| --- | --- | --- |
| **Prepared by** | ***Date*** | ***Signature*** |
|  |  |  |
| **Patricia SEGONDS – Atos**  **Anne HAUGOMMARD - Atos** | 12/03/2021 |  |
|  |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| **Accepted by** | ***Date*** | ***Signature*** |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| **Approved by** | ***Date*** | ***Signature*** |
|  |  |  |
|  |  |  |
|  |  |  |

**TABLE OF CONTENTS**

[1. Introduction 3](#_Toc66457082)

[1.1. Objective of the document 3](#_Toc66457083)

[1.2. Document organisation 3](#_Toc66457084)

[2. Applicable and reference documents 4](#_Toc66457085)

[2.1. Applicable Documents 4](#_Toc66457086)

[2.2. Reference Documents 4](#_Toc66457087)

[3. Terms, definitions and abbreviated terms 5](#_Toc66457088)

[4. OCL conf checker - Presentation 6](#_Toc66457089)

[4.1. Tool context 6](#_Toc66457090)

[4.2. Checktool interfaces 7](#_Toc66457091)

[5. OCL conf checker - Installation 8](#_Toc66457092)

[5.1. Eclipse installation 8](#_Toc66457093)

[5.2. Suggestion of Eclipse plugins 8](#_Toc66457094)

[6. OCL checker - Instanciation on new architecture 10](#_Toc66457095)

[6.1. EMF project creation from XSD schema 10](#_Toc66457096)

[6.2. Model generation from genmodel 14](#_Toc66457097)

[6.3. (optional) Add plugin extension to use your metamodel with multiple metamodels 16](#_Toc66457098)

[6.4. Creation of a new feature 17](#_Toc66457099)

[6.5. Creation of a new UPDATE SITE 19](#_Toc66457100)

[7. OCL conf checker – Usage (GUI) 25](#_Toc66457101)

[7.1. Execution environment 25](#_Toc66457102)

[7.2. Creation of an execution project 25](#_Toc66457103)

[7.3. Initialisation of an OCL rule file 27](#_Toc66457104)

[7.4. Initialisation of a ruleset file 28](#_Toc66457105)

[7.5. Configuration check 29](#_Toc66457106)

[8. OCL Checker and Multiple metamodels 32](#_Toc66457107)

[9. APPENDIX 1: Batch mode usage on Windows and Linux 33](#_Toc66457108)

[10. APPENDIX 2: Procedure to release an update site for a new version of XSD 36](#_Toc66457109)

[11. APPENDIX 3 : Visualize metamodel as a diagram 46](#_Toc66457110)

# Introduction

## Objective of the document

This document is the OCL configuration checker User Manual.

The document describes recommendations and guidance on how to use the tool.

## Document organisation

This document is divided into main sections:

* Presentation
* Installation
* Instantiation on new configuration
* Usage

# Applicable and reference documents

## Applicable Documents

None

## Reference Documents

|  |  |
| --- | --- |
| **Reference** | **Title** |
| RD1 | Object Constraint Language  <https://www.omg.org/spec/OCL/> |

# Terms, definitions and abbreviated terms

|  |  |
| --- | --- |
| **Acronym** | **Meaning** |
| EMF | Eclipse Modeling Framework |
| OCL | Object Constraint Language |
| XSD | XML Schema Definition |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# OCL conf checker - Presentation

OCL Checker tool is a tool used in the development process of IMA space projects.

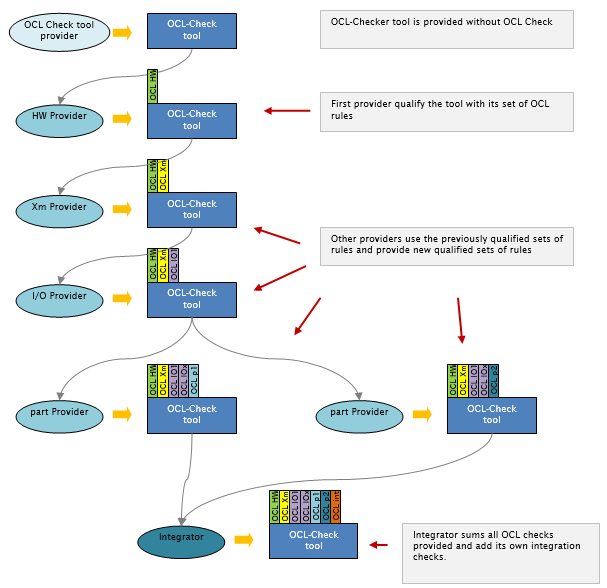
## Tool context

The provider will deliver the tools with a configuration of checks implemented by OCL rules and XSD(s) files.

The user will use the tool in the configuration delivered by the provider and performs the checks on its configuration data.

A user can also be a provider for the next users.

Global interactions of different user/providers are illustrated in the following diagram:



This diagram highlights an enrich rules pattern. Details of this pattern are:

1. Get the OCL Checker with OCL rules updated by the previous provider
2. Add your part of the product (a partition)
3. Check that OCL rules provided are checked
4. Define your own set of OCL rules relative to the new part
5. Qualify the new set of OCL rules
6. Deliver the new development, new configuration file and the qualified new set of OCL rules

## Checktool interfaces

XML Description

XSD Description

Check Report

OCL

rules

OCL conf checker

Input definition

* XML description: one or several XML files defining model elements

/!\ CAUTION /!\

* Only one namespace is supported, the same must be used in all XSD.
* Defined types must be unique (no duplication authorized), an XSD specific ‘types.xsd’ can be used.
* Complex types referenced in an xsd must be present in the xsd.

Output Definition

* Check Report: Result of the check

# OCL conf checker - Installation

As a prerequisite Java 1.8 must be installed.

The tool is running on following Eclipse environment. Following steps are necessary to run it:

* Eclipse installation
* [optional] Suggestion of Eclipse plugins

## Eclipse installation

|  |  |  |
| --- | --- | --- |
| **Environment** | **Version** | **Source** |
| Eclipse | 2020-03-modeling-tools | Download zip file from:  https://www.eclipse.org/downloads/packages/release/2020-03/r/eclipse-modeling-tools |

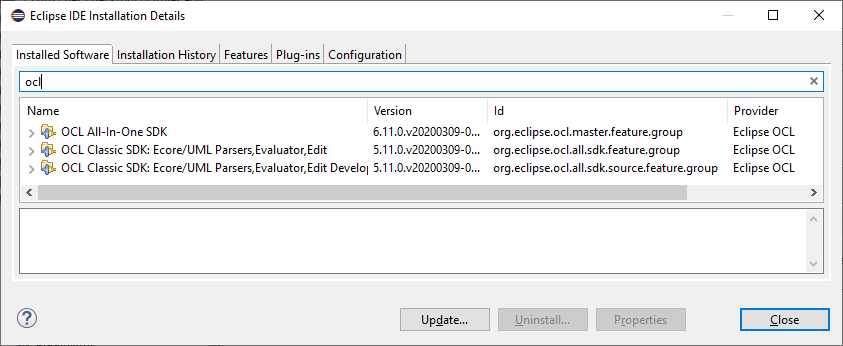
1. Download zip file
2. Unzip it in a required application directory
3. Launch eclipse.exe

## Suggestion of Eclipse plugins

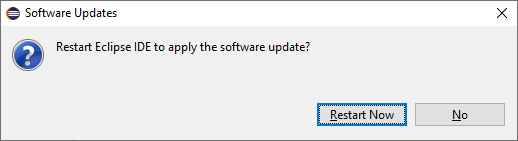
Install OCL Editor

Following elements must also be installed:

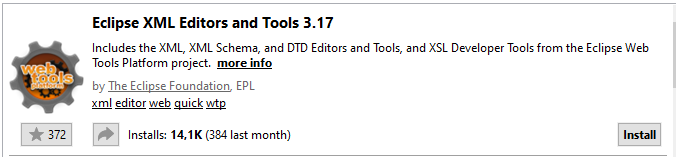
1. Go on Help > Install new software
2. From repository <http://download.eclipse.org/releases/2020-03>, install OCL following elements:



1. Click on Next, wait for installation
2. Click on Next, accept License text
3. Click on Finish
4. Restart Eclipse as requested:

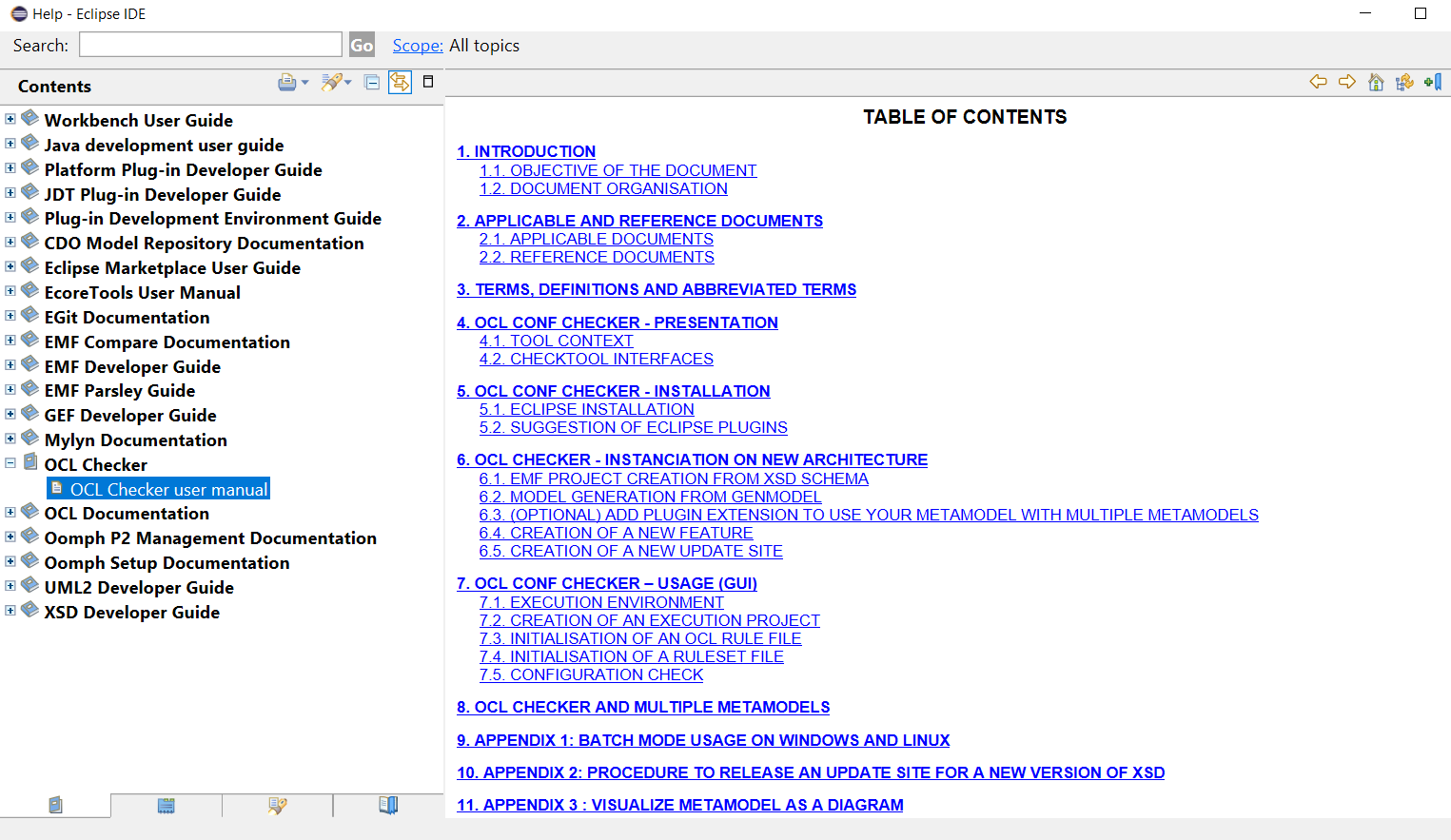


Other suggestion :



## Online documentation inside Eclipse

After installation on Eclipse, this OCL Checker User Manual document is also available within the tool, and accessible after tool installation on Eclipse Help> Help contents



Note : The OCL Documentation is also available within Eclipse.

# OCL checker - Instanciation on new architecture

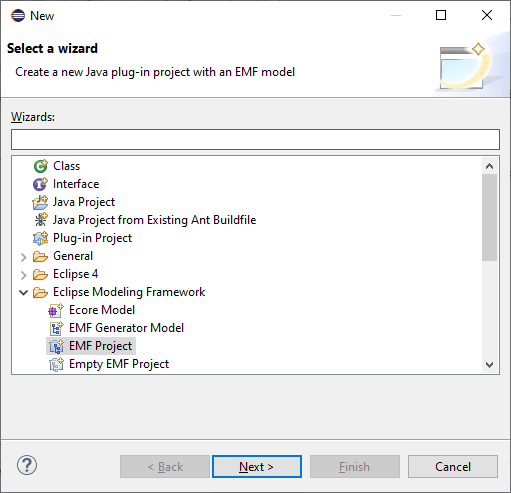
To check an OCL configuration on a new architecture, specific XSD schema configuration files defining this architecture must be used in the tool regarding the following 3 steps procedure:

1. EMF project creation from XSD schema
2. Model generation from genmodel (of previous created project)
3. Creation of a new feature (to package all previous created elements)
4. Create a dedicated update site (with the previous created feature) to enable the installation of the metamodel into an Eclipse environment
5. Then this update site can be installed into the environment and the XSD and associated metamodel is known from OCLChecker tool.

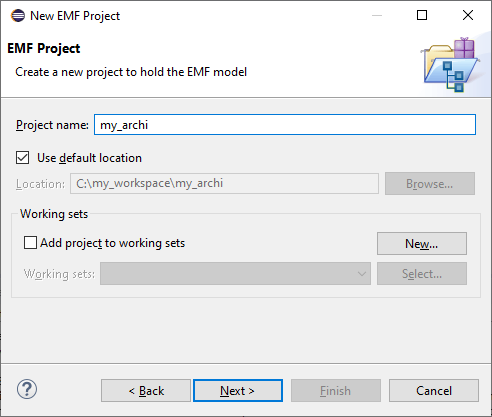
Hereafter are described all those steps.

## EMF project creation from XSD schema

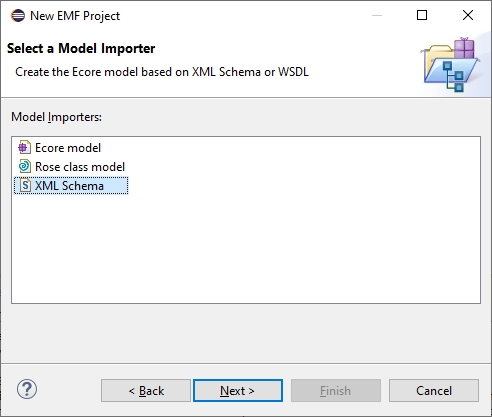
1. Go on File 🡪 New 🡪 Other… in the main menu
2. Select EMF Project:



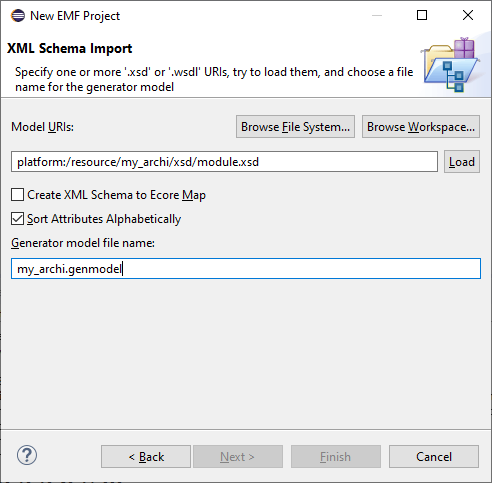
1. Click on Next button
2. Give a name to the new project:



1. Click on Next button
2. Select XML Schema as Model Importer:



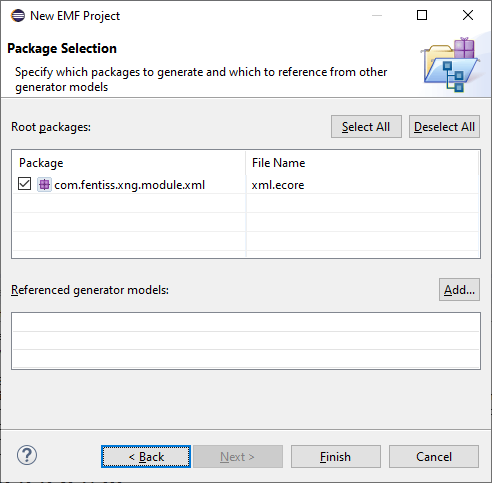
1. Click on Next
2. Click on Browse File System and select the XMS schema, and give a name to the new generator model file:



Note:

Please note that in this step a check is done on the XML schema, this check can also be done by clicking on the Load button.

1. Click on Next. The ecore filename can be modified at this step.



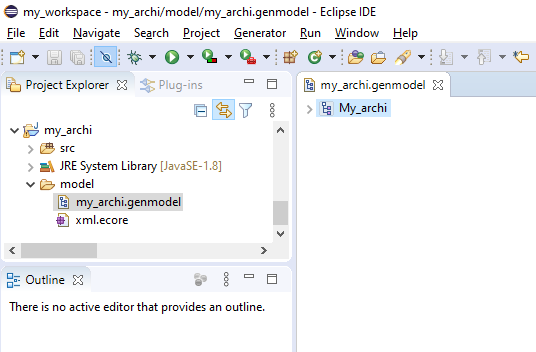
1. Click on Finish
2. The new project has been created and can be seen in the project view explorer.

As an additional feature interesting to mention, you can use Eclipse plugins to visualize the generated Ecore as a diagram (see 11 - APPENDIX 3 : Visualize metamodel as a diagram).

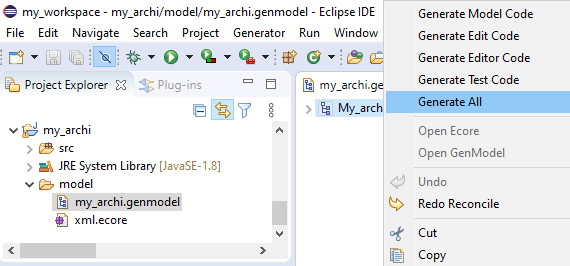
## Model generation from genmodel

From previously created project

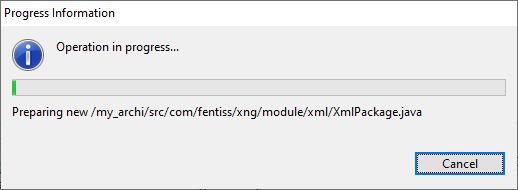
1. Double-click on the .genmodel file in the model folder to open the .genmodel:



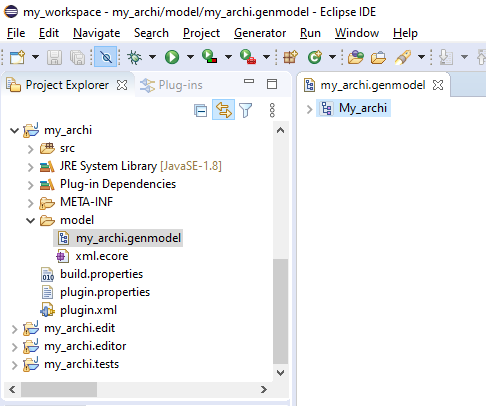
1. Right-click on the root node and select Generate Model, Generate Edit and Generate Editor :



1. Wait for generation:



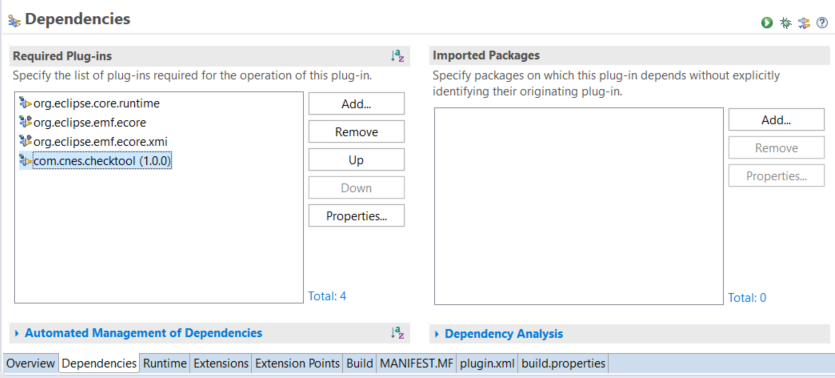
1. 3 new projects are created:
   * <project\_name>.edit
   * <project\_name>.editor
   * <project\_name>.tests



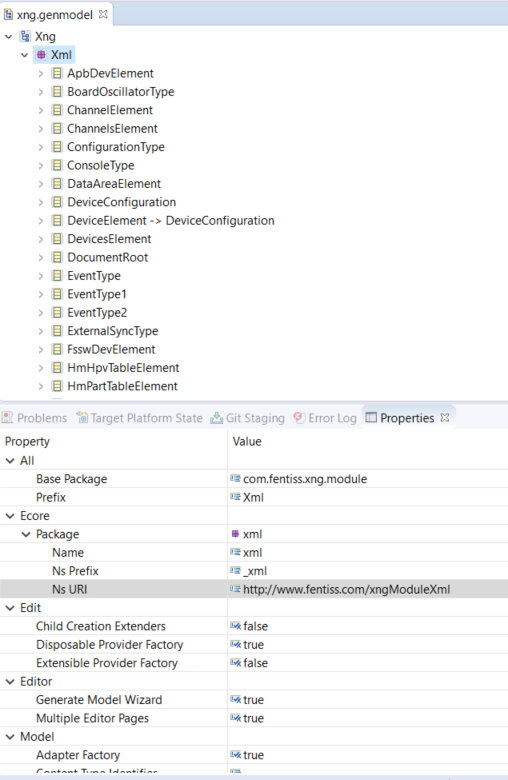
## (optional) Add plugin extension to use your metamodel with multiple metamodels

If you plan to use your XML files along with other metamodels / XML depending on other XSD described in section 8 - OCL Checker and Multiple metamodels below , you need to add an OCL Checker extension to your plug-in.

1. Open file plugin.xml from plugin <project-name>
2. On tab “Dependencies”, click on “Add..” and select “com.cnes.checktool”



1. Keep plugin.xml opened
2. On the same project, open the .genmodel with the editor  
   For each ePackage ( node with  symbol), select it and open Properties view to get property Ecore>Package>NsURI



1. For each EPackage Ns URI found on the .genmodel,   
   add one extension in plugin.xml, before last line </plugin> with the following content :

<extension

point="com.cnes.checktool.metamodel.epackage">

<EPackage

EPackageURI="<<EPackage URI>>">

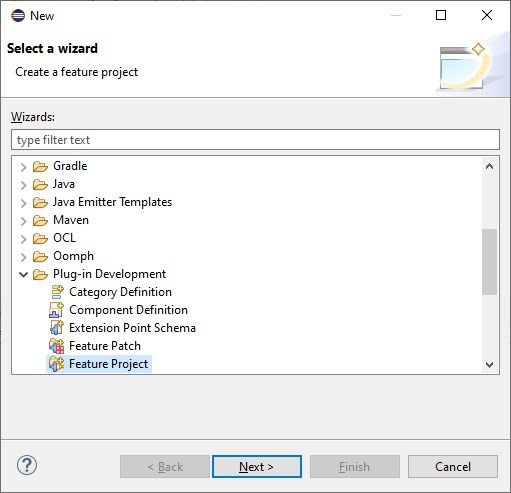
</EPackage>

</extension>

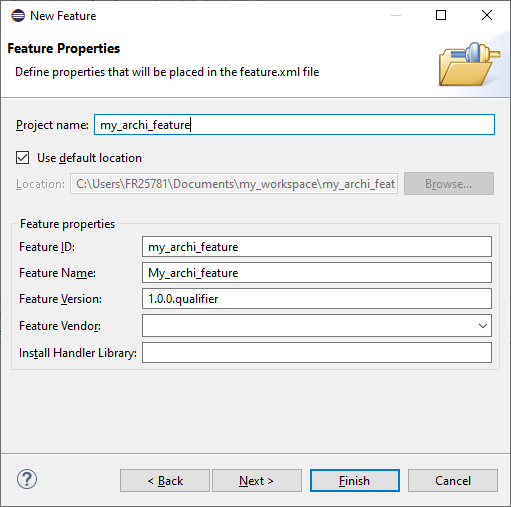
1. Save and close plugin.xml

## Creation of a new feature

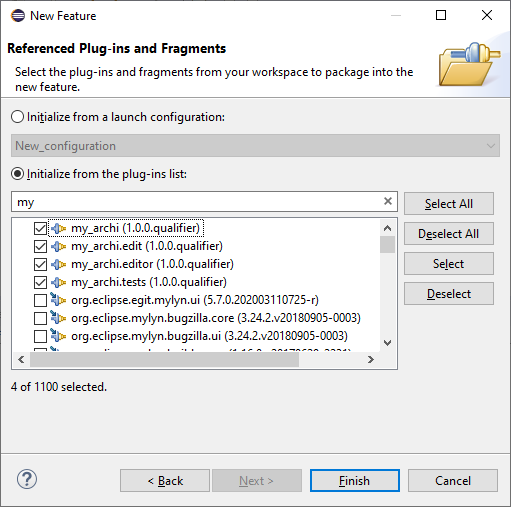
1. Go on File 🡪 New 🡪 Other… in the main menu
2. Select Feature Project:



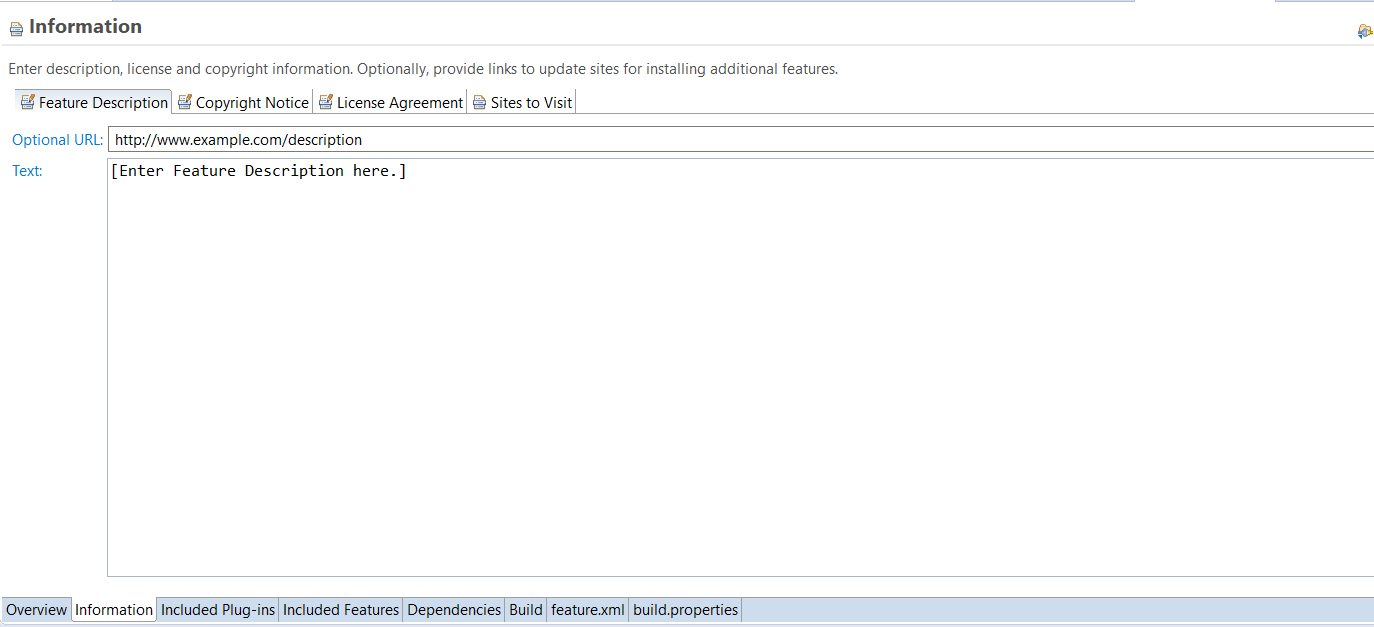
1. Click on Next
2. Enter project name. Suggestion: <project\_name>.feature



1. Click on Next
2. Select all related projects:



1. Click on Finish
2. New feature project appears in the project explorer.
3. Open file “feature.xml”, on “Information” tab



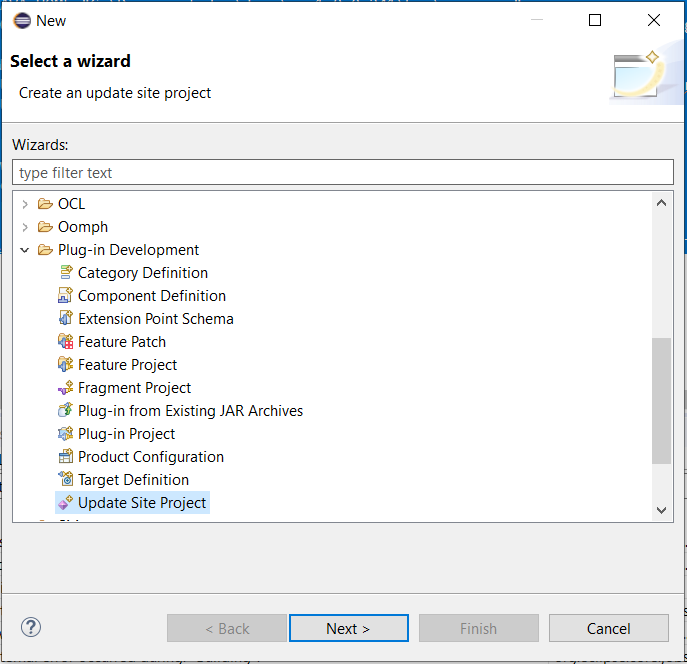
1. Fill “Feature Description”, “Copyright notice” and “License Agreement”

Note : the “License Agreement” will be displayed to any user installing this feature, for acceptation.

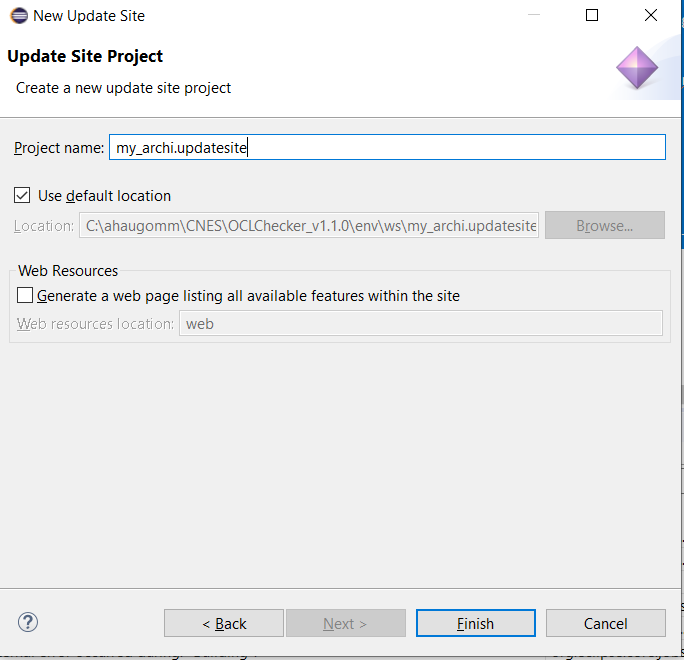
It is HIGHLY recommended to fill these three tabs.

## Creation of a new UPDATE SITE

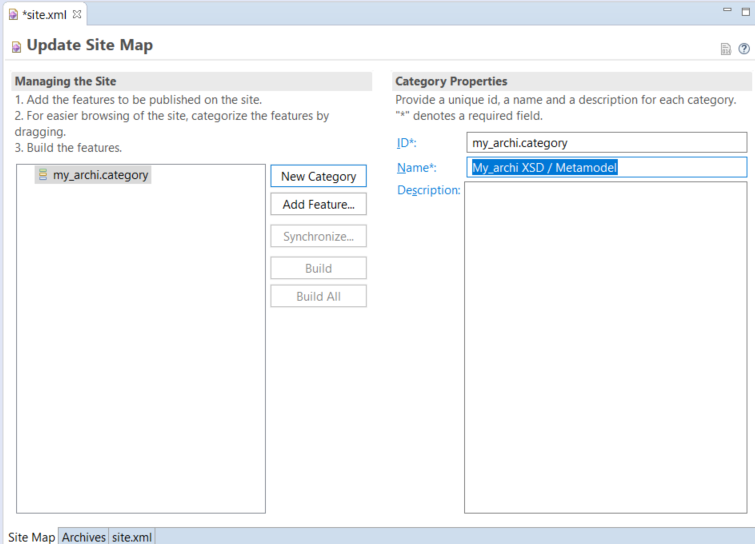
1. Go on File 🡪 New 🡪 Other… in the main menu
2. Plug-in development 🡪 Update site project



1. Fill the name on next page “<project-name>.updatesite”

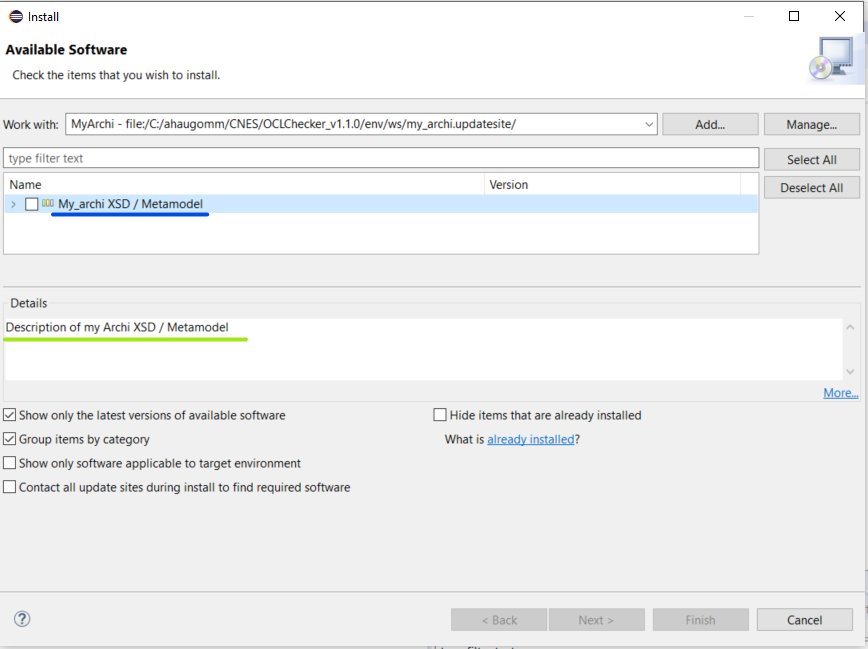


1. Open site.xml file and select “New Category”

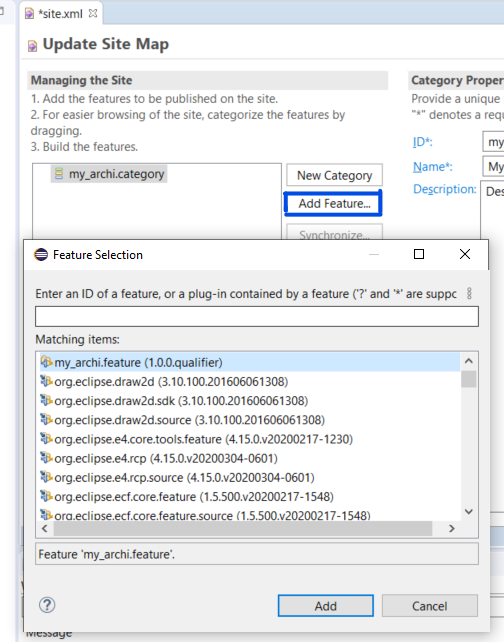


1. Fill ID with <project-name>.category, and Name with something meaningful for the user to know what is being installed. Description can be filled to add more details on the category.

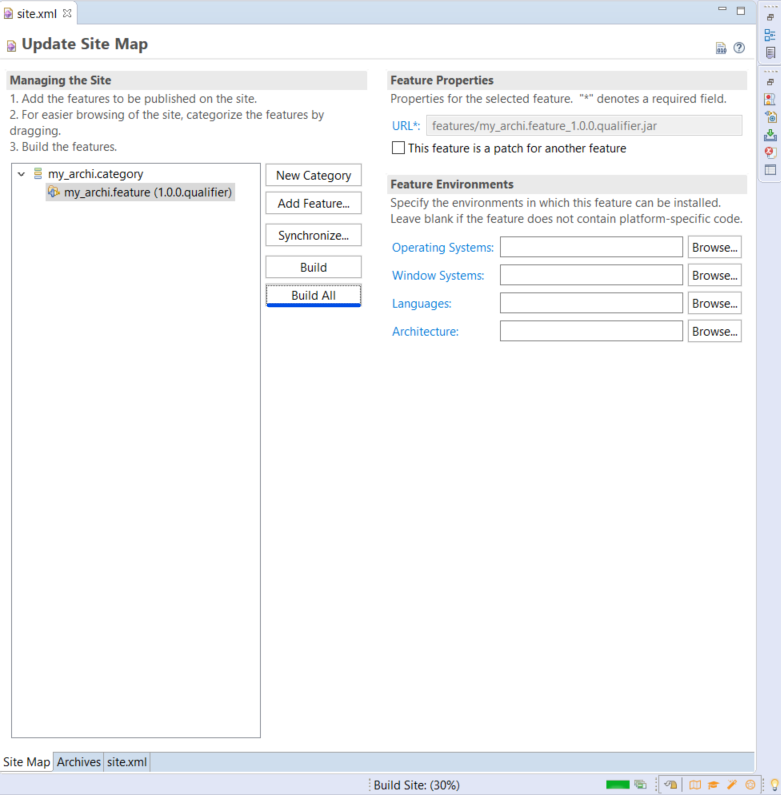
Note : when users will be installing content from this update site, the category is displayed as a group of items to be installed (in blue) and the description appears below (green).



1. In site.xml, select category and click on “Add feature” and select <project-name>.feature (version 1.0.0.qualifier) from the list.



1. Save site.xml and click on “Build all”

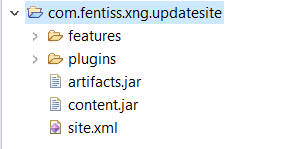


1. Wait until Build is 100%

Before build, you have only one file



After build, you have 2 folders and 3 files

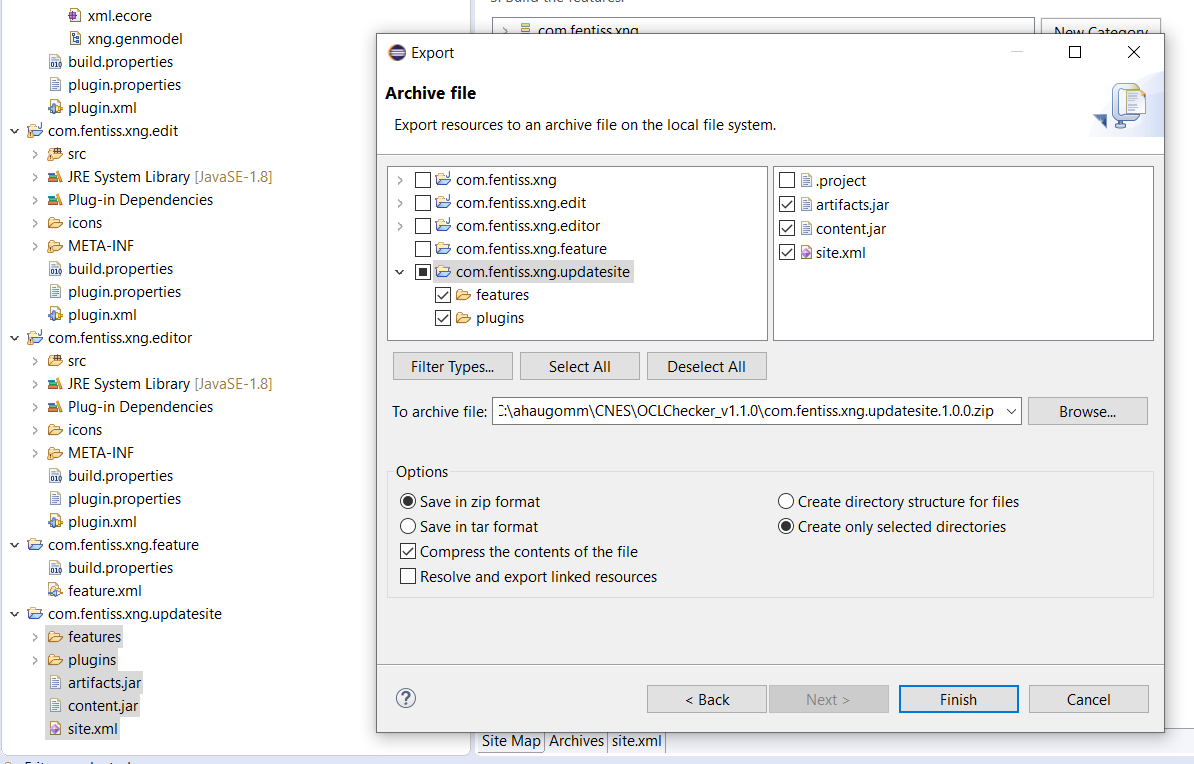


1. In order to share this update site, create an archive of all files in the project

WARNING : The archive must contain directly the site.xml file, no upper level folder.

If you create the archive from Eclipse

* + Select the files (site.xml, artifacts.jar, contents.jar, features folder , plugins folder) and   
    click File > Export > Archive file > Next
  + Check option “Create only selected directories”



* + Finish

1. Open the resulting archive to ensure that you have site.xml at top level.
2. Test the installation from your eclipse platform
   * Help > Install new software > Add.. > Archive ..
   * Select archive of the new update site
   * Check that the two names in the check boxes correspond to
     + For highest level , the label of the category from site.xml in update site project
     + For lower level, the feature name (in feature.xml)

# OCL conf checker – Usage (GUI)

The tool must be used in Eclipse running environment.

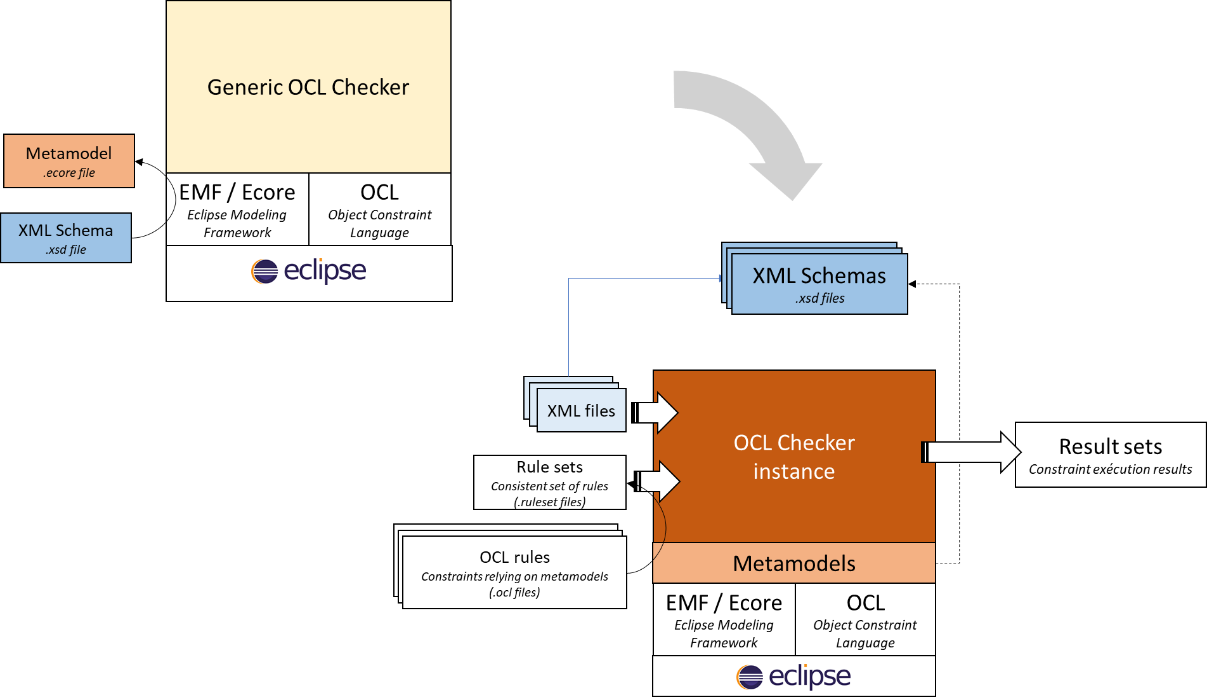
Only ruleset files can be executed on xml files to validate rules.

To use the tool the following steps must be followed:

1. Running environment setup
2. Creation of an execution project
3. Initialisation of a rule file
4. Initialisation of a ruleset file
5. Configuration check

Hereafter are described all those steps.

## Execution environment



As presented in section 4 - OCL conf checker - Presentation above, in order to use OCL Checker tool with XML files compliant with a specific XSD , you need to “install this XSD as a metamodel in the Eclipse environment”.

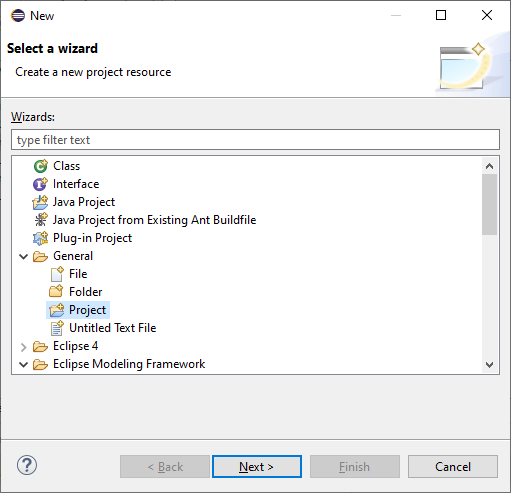
This corresponds to :

* Install “Generic” OCL Checker on top of the Eclipse installation (see section 5 - OCL conf checker - Installation)
* Install required metamodels (from XSD) for the XML models to check in the same Eclipse environment (see section 6 - OCL checker - Instanciation on new architecture)

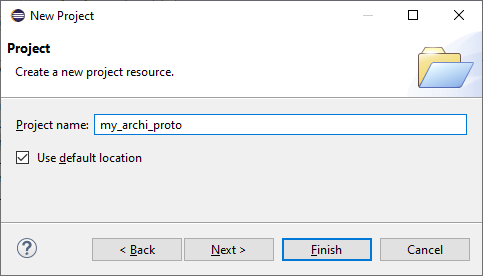
Then the OCL Checker knows how to handle such XML models for check.

## Creation of an execution project

1. Go on File 🡪 New 🡪 Other… in the main menu
2. Select General Project:



1. Click on Next
2. Precise name project



1. Click on Finish
2. New empty project appears in the project view explorer.

## Initialisation of an OCL rule file

A rule file is an OCL file associated to the namespace from an ecore file.

/!\ CAUTION /!\

Please be aware that creation of OCL file from another manner can lead to a bad ocl file (due to a bad first import line in the created file).

1. In your project, click on New File
2. Precise the name of your new ocl rule file, with .ocl extension
3. Once created, fill in the file with the minimum content:

|  |
| --- |
| **import** *'http://www.fentiss.com/xngModuleXml'*  **package** *xml*  -- OCL rules  **endpackage** |

Here completion, with CTRL + J, can be used to precise the namespace to be imported (corresponding to the ecore file).

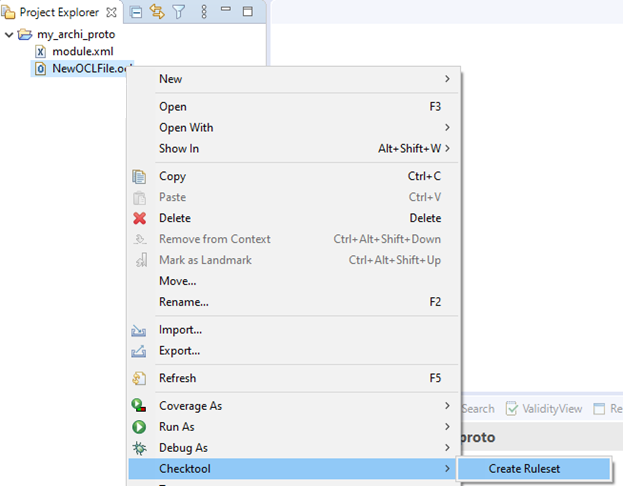
1. Fill in the file with the relevant ocl rules.

Note : The full OCL Syntax is available at the following location : <https://www.omg.org/spec/OCL/2.4/PDF> .

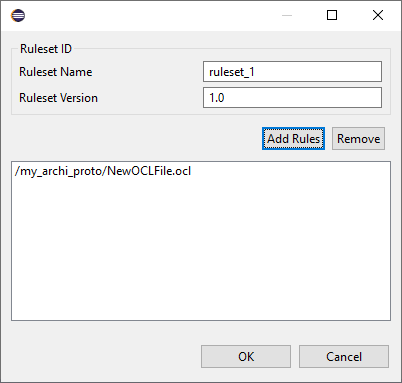
## Initialisation of a ruleset file

A ruleset file is set of rules issued from one or several ocl files.

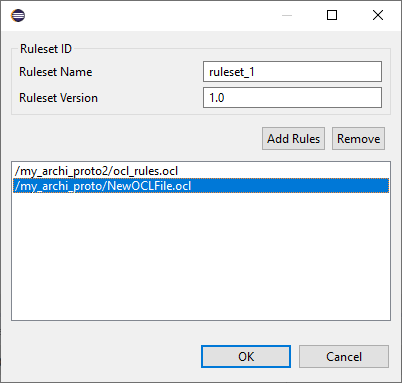
1. Click on an ocl file.
2. Right click on it and select Checktoool 🡪 Create Ruleset



1. Precise the ruleset filename:



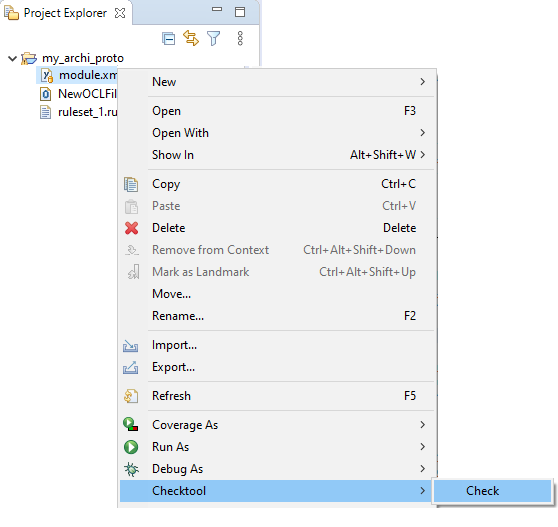
Several ocl files can be added through Add Rules:



1. Click on OK
2. Ruleset file is created

## Configuration check

1. Import XML files in the execution project.
2. Click on XML configuration file to be checked
3. Right click on it and select Checktool 🡪 🡪 Check

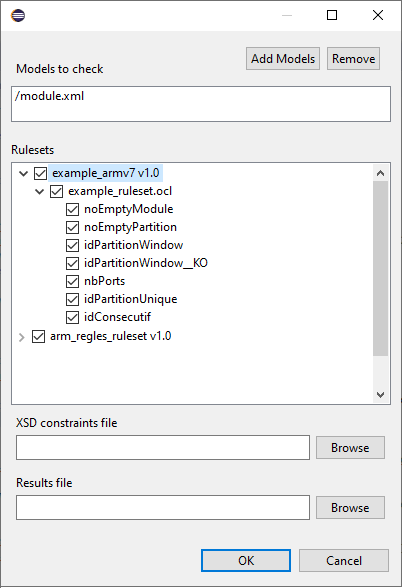


1. Select the ruleset

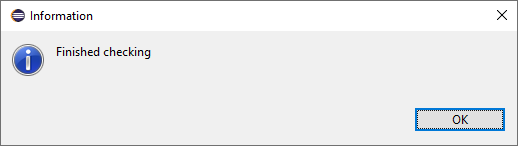
And click on Browse to precise result file:

/!\ CAUTION /!\

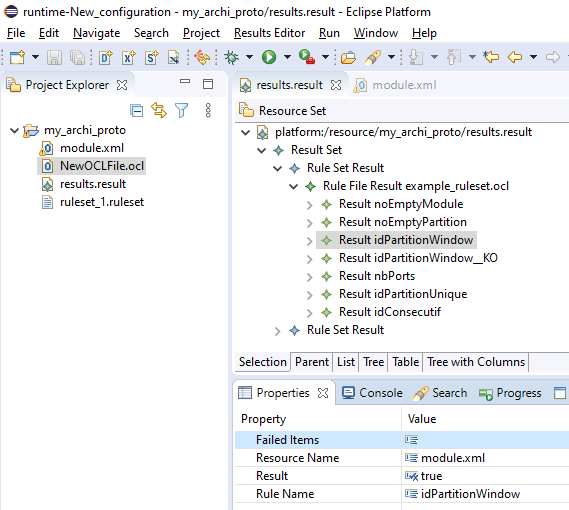
Output result file must be filled to launch execution.



1. Click on OK to launch the check, wait for execution end



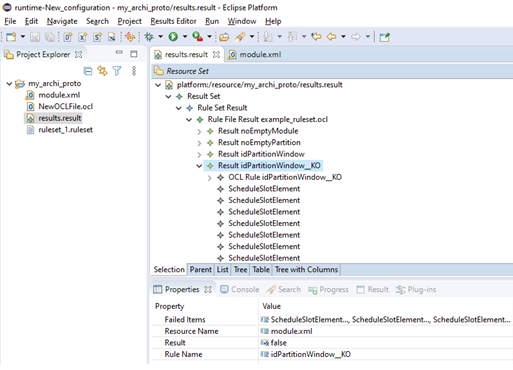
1. The result file has been created.



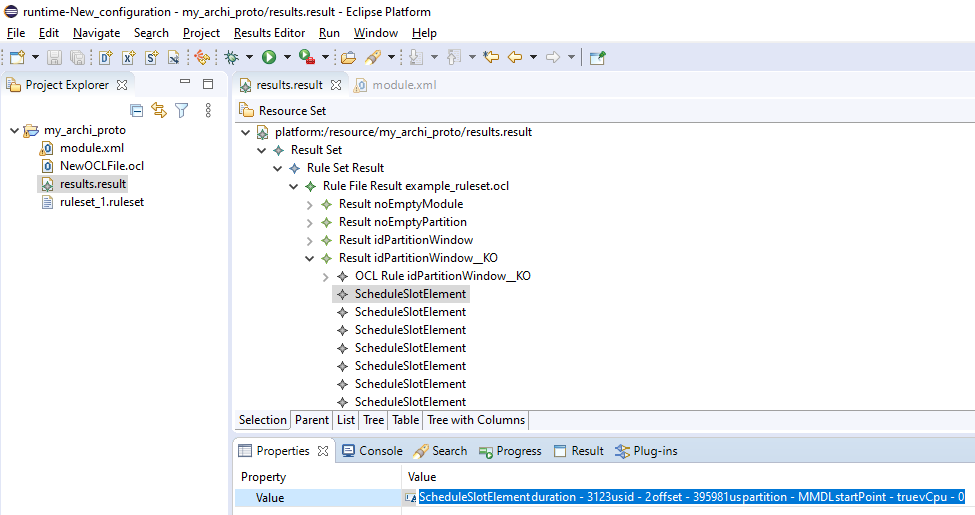
1. Click on it and expand the resource set in the right view.

For each rule you have its result: true or false (see the arrow above).

In case of a rule has failed, as follow, the list of failed items can be accessed:



Select each element in the upper view to display its properties (no element is displayed when the rule is OK). Double click on ‘Value’ in Properties tab to display it:



Those information leads to identify the element who caused the failure.

# OCL Checker and Multiple metamodels

With OCL evaluation used inside OCLChecker, there is a limitation : only one metamodel declaration is allow on the first lines of .ocl files.

Thus the ocl file to use rules with multiple metamodel should look like the following :

**import** *'http://www.fentiss.com/xngModuleXml'*

*-- import ‘metamodel2 NS\_URI’*

**package** *xml*

**context** *PartitionElement*

**inv** my\_rule : <<combine elements from NS\_URI 1 and NS\_URI 2>>

**endpackage**

The second metamodel is imported through the extension point declared in section 6.3 - (optional) Add plugin extension to use your metamodel with multiple metamodels .

# APPENDIX 1: Batch mode usage on Windows and Linux

Tool can be used in batch mode, as an Eclipse application with a set of arguments .

java   
-cp *<<Eclipse Equinox launcher>>*   
org.eclipse.core.launcher.Main   
-data *<<Folder to use as a workspace>>*   
-application **com.cnes.checktool.application**

**--provider**=*<<Folder to use as workspace>>* **--rulesets**=*<<RuleSet files>>* **--ocls**=*<<OCL rule Files>>*  **--xmlmodels**=*<<XML models to evaluate>>*  **--output**=*<<Result file path>>*

Hereafter are tool batch arguments for application “com.cnes.checktool.application” :

|  |  |  |
| --- | --- | --- |
| Argument | Required | Description |
| **provider** | yes | Folder to use as a workspace (Batch mode launches an Eclipse instance).  Contains all ruleset files and XSD constraints. |
| **rulesets** | no | The path, relative to the provider workspace, of the rulesets to be applied. |
| **xsdpath** |  | The path, relative to the provider workspace, of the selected XML schema file. |
| **xmlmodels** | yes | The absolute path of all xml models to be evaluated, separated by ; Use quotes “” around the full parameter value if your are using multiple values.  --xmlmodels=”path/to/m1.xml;path/to/m2.xml;path/to/m3.xml” |
| **ocls** | No | The path, relative to the provider workspace, of the selected constraint file. |
| **output** | yes | The absolute path of the file containing the results. |

Note:

Tool will not run if rulesets, ocls and xsdpath are both null.

* Sample command line (windows) with ruleset file (replace blue values by local paths):

@echo off

set **JAVA\_HOME**="C:\Program Files\Java\jre1.8.0\_241\bin\java.exe"

set **EQUINOX\_LAUNCHER\_PATH**="C:\ahaugomm\CNES\OCLChecker\_v1.1.0\env\eclipse-modeling-2020-03-R-win32-x86\_64\eclipse\plugins\org.eclipse.equinox.launcher\_1.5.700.v20200207-2156.jar"

set **TEST\_WORKSPACE\_PATH**=C:\ahaugomm\CNES\OCLChecker\_v1.1.0\env\ws

set **XML\_MODELS**=%TEST\_WORKSPACE\_PATH%\OCLChecker\_Tests\OCLCHECKER-TV-030-001\file1.xml;%TEST\_WORKSPACE\_PATH%\OCLChecker\_Tests\OCLCHECKER-TV-030-001\file2.xml

set **RULE\_SETS**=\OCLChecker\_Tests\OCLCHECKER-TV-030-001\OCLCHECKER-TV-030-001\_v1.0.ruleset

set **OUTPUT\_FILE**=%TEST\_WORKSPACE\_PATH%\OCLChecker\_example\res.result

echo Executing CheckTool

echo with rulesets: %RULE\_SETS%

echo on models: %XML\_MODELS%

echo result in: %OUTPUT\_FILE%

**%JAVA\_HOME% -cp %EQUINOX\_LAUNCHER\_PATH% org.eclipse.core.launcher.Main -data %TEST\_WORKSPACE\_PATH%** -application com.cnes.checktool.application --provider**=%TEST\_WORKSPACE\_PATH%** --rulesets**=%RULE\_SETS%** --xmlmodels**=%XML\_MODELS%** --output**=%OUTPUT\_FILE%**

* Sample command line (windows) with OCL files directly (replace blue values by local paths): :

|  |
| --- |
| @echo off  set **JAVA\_HOME**="C:\Program Files\Java\jre1.8.0\_241\bin\java.exe"  set **EQUINOX\_LAUNCHER\_PATH**="C:\ahaugomm\CNES\OCLChecker\_v1.1.0\env\eclipse-modeling-2020-03-R-win32-x86\_64\eclipse\plugins\org.eclipse.equinox.launcher\_1.5.700.v20200207-2156.jar"  set **TEST\_WORKSPACE\_PATH**=C:\ahaugomm\CNES\OCLChecker\_v1.1.0\env\ws  set **XML\_MODELS**="%TEST\_WORKSPACE\_PATH%\OCLChecker\_Tests\OCLCHECKER-TV-040-001\file1.xml;%TEST\_WORKSPACE\_PATH%\OCLChecker\_Tests\OCLCHECKER-TV-040-001\file2.xml"  set **OCL\_RULES**="/OCLChecker\_Tests/OCLCHECKER-TV-040-001/OCLCHECKER-TV-040-001.ocl;/OCLChecker\_Tests/OCLCHECKER-TV-040-001/NewOCLFile.ocl"  set **OUTPUT\_FILE**=%TEST\_WORKSPACE\_PATH%\OCLChecker\_Tests\OCLCHECKER-TV-040-001\OCLCHECKER-TV-040-001.result  echo Executing CheckTool  echo with ocls: %OCL\_RULES%  echo on models: %XML\_MODELS%  echo result in: %OUTPUT\_FILE%  **%JAVA\_HOME%** -cp **%EQUINOX\_LAUNCHER\_PATH%** org.eclipse.core.launcher.Main -data **%TEST\_WORKSPACE\_PATH%** -application com.cnes.checktool.application --provider=**%TEST\_WORKSPACE\_PATH%** --ocls=**%OCL\_RULES%** --xmlmodels=**%XML\_MODELS%** --output=**%OUTPUT\_FILE%** |

* Sample command line (linux) with ruleset file (replace blue values by local paths)::

#!/bin/bash

# This script launches OCL Checker tool on models

# Usage : ./oclChecker\_launcher.sh

export **JAVA\_HOME**=/usr/bin/java

export **EQUINOX\_LAUNCHER\_PATH**="/home/lv/Tools/Eclipse/eclipse/plugins/org.eclipse.equinox.launcher\_1.5.700.v20200207-2156.jar"

export **WORKSPACE\_PATH**=/home/lv/Tools/Eclipse/ws\_test

export **XML\_MODELS**="$WORKSPACE\_PATH/example/example\_sparcv8/module.xml;$WORKSPACE\_PATH/example/example\_sparcv8/module2.xml;$WORKSPACE\_PATH/example/example\_sparcv8/module3.xml"

export **RULE\_SETS**="/example/example\_sparcv8/example\_sparcv8.ruleset;/example/example\_sparcv8/another.ruleset"

export **OUTPUT\_FILE**=$WORKSPACE\_PATH/example/example\_sparcv8/output.result

echo Executing CheckTool

echo with rulesets: $RULE\_SETS

echo on models: $XML\_MODELS

echo result in: $OUTPUT\_FILE

**$JAVA\_HOME** -cp **$EQUINOX\_LAUNCHER\_PATH** org.eclipse.core.launcher.Main -data **$WORKSPACE\_PATH** -application com.cnes.checktool.application --provider=**$WORKSPACE\_PATH** --rulesets=**$RULE\_SETS** --xmlmodels=**$XML\_MODELS** --output=**$OUTPUT\_FILE**

* Sample output of the command line

Executing CheckTool

with ocls: "/OCLChecker\_Tests/OCLCHECKER-TV-040-001/OCLCHECKER-TV-040-001.ocl;/OCLChecker\_Tests/OCLCHECKER-TV-040-001/NewOCLFile.ocl"

on models: "C:\ahaugomm\CNES\OCLChecker\_v1.1.0\env\ws\OCLChecker\_Tests\OCLCHECKER-TV-040-001\file1.xml;C:\ahaugomm\CNES\OCLChecker\_v1.1.0\env\ws\OCLChecker\_Tests\OCLCHECKER-TV-040-001\file2.xml"

result in: C:\ahaugomm\CNES\OCLChecker\_v1.1.0\env\ws\OCLChecker\_Tests\OCLCHECKER-TV-040-001\OCLCHECKER-TV-040-001.result

10:07:36 INFO Started checking process

10:07:36 INFO Preparing temporary working space

10:07:37 INFO Importing rulesets

10:07:37 INFO Importing model files

10:07:37 INFO Checking files

10:07:37 INFO Evaluating rule file OCLCHECKER-TV-040-001.ocl

10:07:42 INFO Evaluating rule file NewOCLFile.ocl

10:07:42 INFO Done

[WARN] Problem while deactivating CDOViewProviderRegistryImpl

java.lang.InterruptedException

at java.util.concurrent.locks.AbstractQueuedSynchronizer.acquireSharedInterruptibly(Unknown Source)

at java.util.concurrent.Semaphore.acquire(Unknown Source)

at org.eclipse.net4j.util.lifecycle.Lifecycle.lock(Lifecycle.java:310)

at org.eclipse.net4j.util.lifecycle.Lifecycle.internalDeactivate(Lifecycle.java:118)

at org.eclipse.net4j.util.lifecycle.ShareableLifecycle.internalDeactivate(ShareableLifecycle.java:52)

at org.eclipse.net4j.util.lifecycle.Lifecycle.deactivate(Lifecycle.java:168)

at org.eclipse.net4j.util.lifecycle.LifecycleUtil.deactivate(LifecycleUtil.java:235)

at org.eclipse.net4j.util.lifecycle.LifecycleUtil.deactivate(LifecycleUtil.java:225)

at org.eclipse.net4j.util.lifecycle.LifecycleUtil.deactivate(LifecycleUtil.java:251)

at org.eclipse.emf.internal.cdo.bundle.Activator$Implementation.doStop(Activator.java:129)

at org.eclipse.emf.internal.cdo.bundle.Activator$Implementation.stop(Activator.java:99)

at org.eclipse.osgi.internal.framework.BundleContextImpl$4.run(BundleContextImpl.java:899)

at org.eclipse.osgi.internal.framework.BundleContextImpl$4.run(BundleContextImpl.java:1)

at java.security.AccessController.doPrivileged(Native Method)

at org.eclipse.osgi.internal.framework.BundleContextImpl.stop(BundleContextImpl.java:891)

at org.eclipse.osgi.internal.framework.EquinoxBundle.stopWorker0(EquinoxBundle.java:1029)

at org.eclipse.osgi.internal.framework.EquinoxBundle$EquinoxModule.stopWorker(EquinoxBundle.java:370)

at org.eclipse.osgi.container.Module.doStop(Module.java:658)

at org.eclipse.osgi.container.Module.stop(Module.java:520)

at org.eclipse.osgi.container.ModuleContainer$ContainerStartLevel.decStartLevel(ModuleContainer.java:1885)

at org.eclipse.osgi.container.ModuleContainer$ContainerStartLevel.doContainerStartLevel(ModuleContainer.java:1760)

at org.eclipse.osgi.container.SystemModule.stopWorker(SystemModule.java:275)

at org.eclipse.osgi.internal.framework.EquinoxBundle$SystemBundle$EquinoxSystemModule.stopWorker(EquinoxBundle.java:202)

at org.eclipse.osgi.container.Module.doStop(Module.java:658)

at org.eclipse.osgi.container.Module.stop(Module.java:520)

at org.eclipse.osgi.container.SystemModule.stop(SystemModule.java:207)

at org.eclipse.osgi.internal.framework.EquinoxBundle$SystemBundle$EquinoxSystemModule$1.run(EquinoxBundle.java:220)

at java.lang.Thread.run(Unknown Source)

Note : The large warning message (in blue above) comes from another plugin and is systematically displayed. It does not correspond to OCLChecker execution , please ignore it.

# APPENDIX 2: Procedure to release an update site for a new version of XSD

5 projects should be available in the ed

* 3 plugins for the metamodel
  + com.xx.yy *<= containing the metamodel*
  + com.xx.yy.edit *<= generated from metamodel*
  + com.xx.yy.editor *<= generated from metamodel*
* One feature (= installable unit, referencing the plugins)
  + com.xx.yy.feature *<= description of the feature to install (list of plugins, versions, dependencies, copyright, license)*
* One update site (= installer)
  + com.xx.yy.updatesite <= *Eclipse installation unit. To be built and distributed for a new release of the metamodel.*

For each evolution of the XSD , a new update site of the metamodel should be released and install into ECLIPSE in order to be taken into account in OCL rules used by OCL Checker.

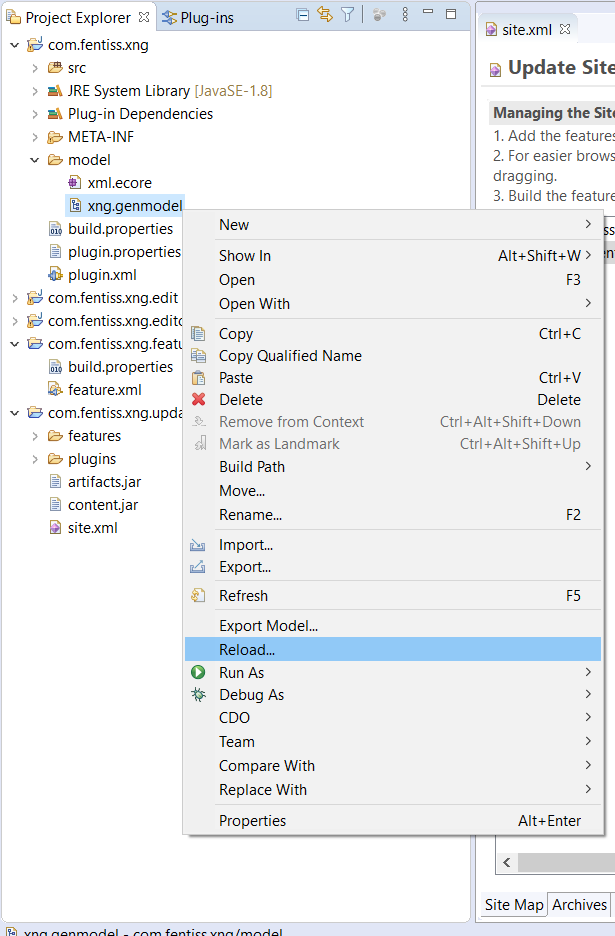
This update site build is done is several steps :

* Reload the metamodel from the new version of XSD
* Regenerate associated code
* Increase all version numbers (plugins, feature, update site)
* Re-build the update site for new version
* Export of the update site

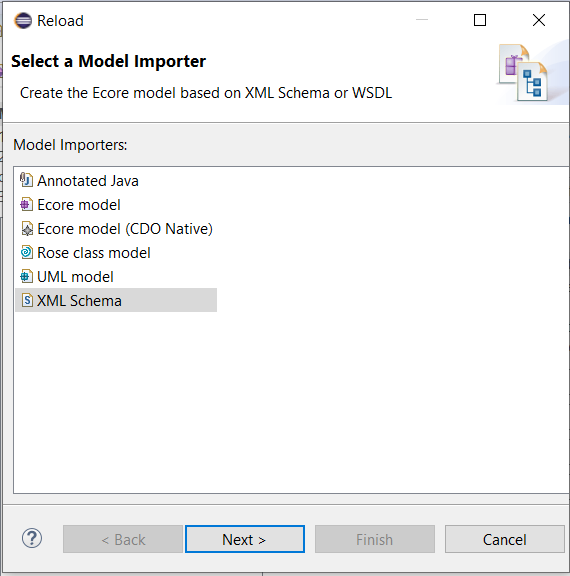
These steps are detailed below.

* **Import the updated XSD into the .genmodel file**

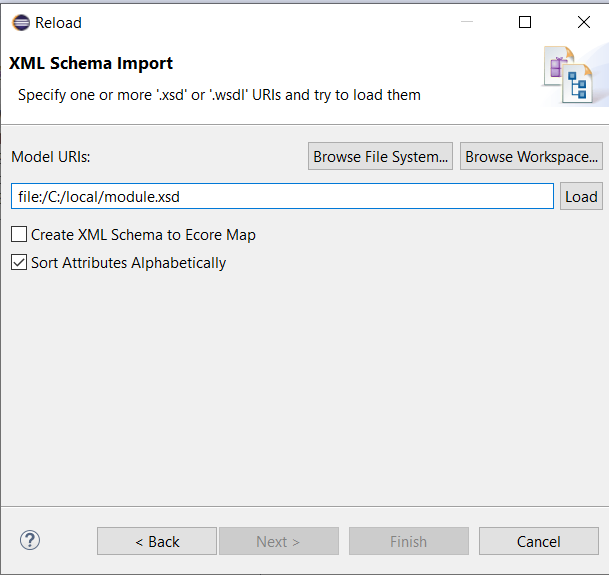
1. From the plugin containing the ecore and genmodel file : right click on .genmodel file > Reload



1. Select XML schema > Next

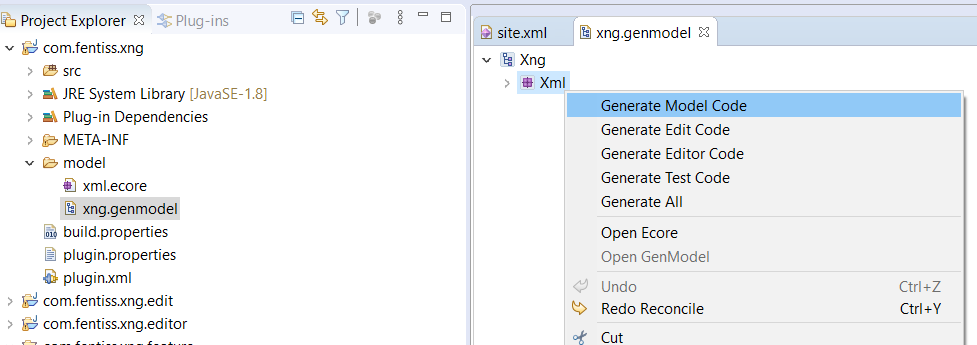


1. Browse your local updated XSD file  
   Click Load, then Next > Finish



* **Regenerate code from .genmodel file**

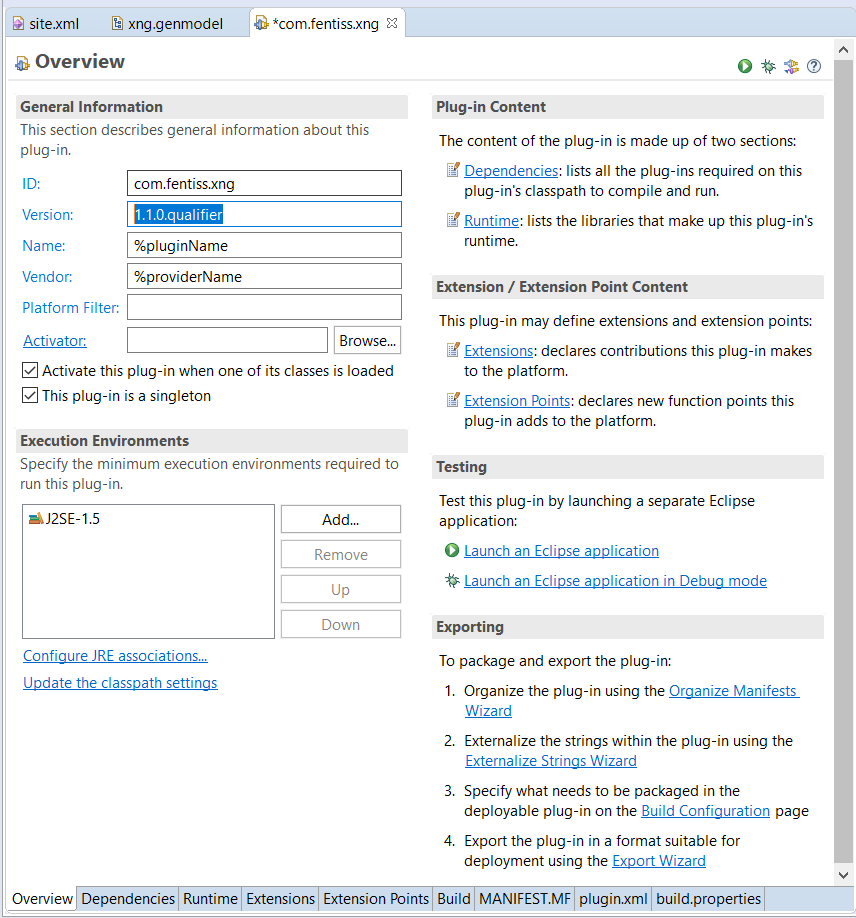
1. Open .genmodel file. From main ePackage > right click and press successively :
   * Generate model code,
   * Generate Edit code,
   * Generate editor code.

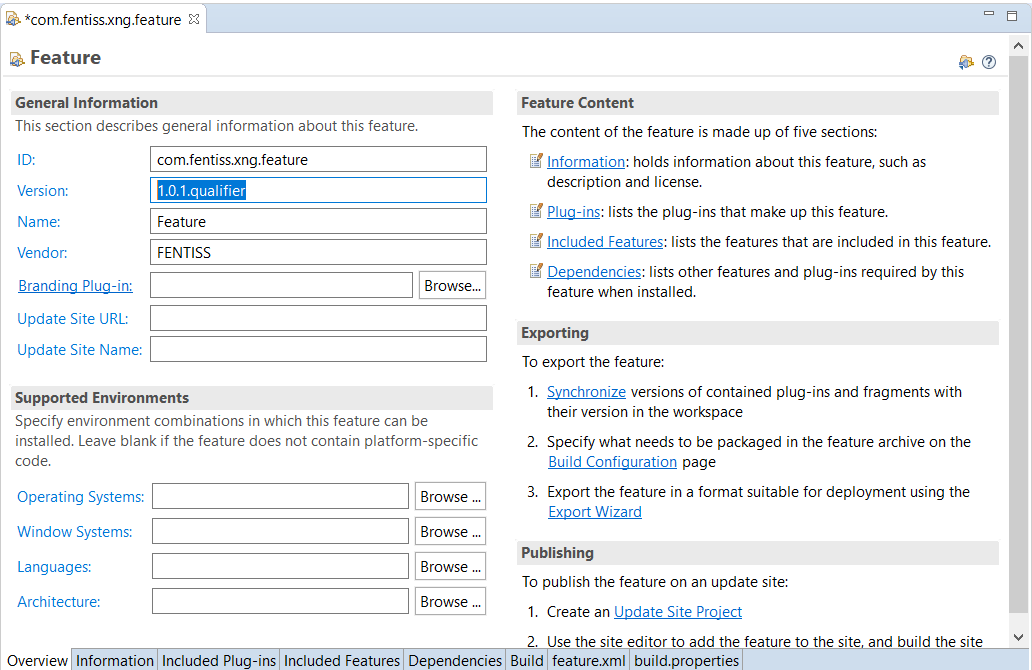
****

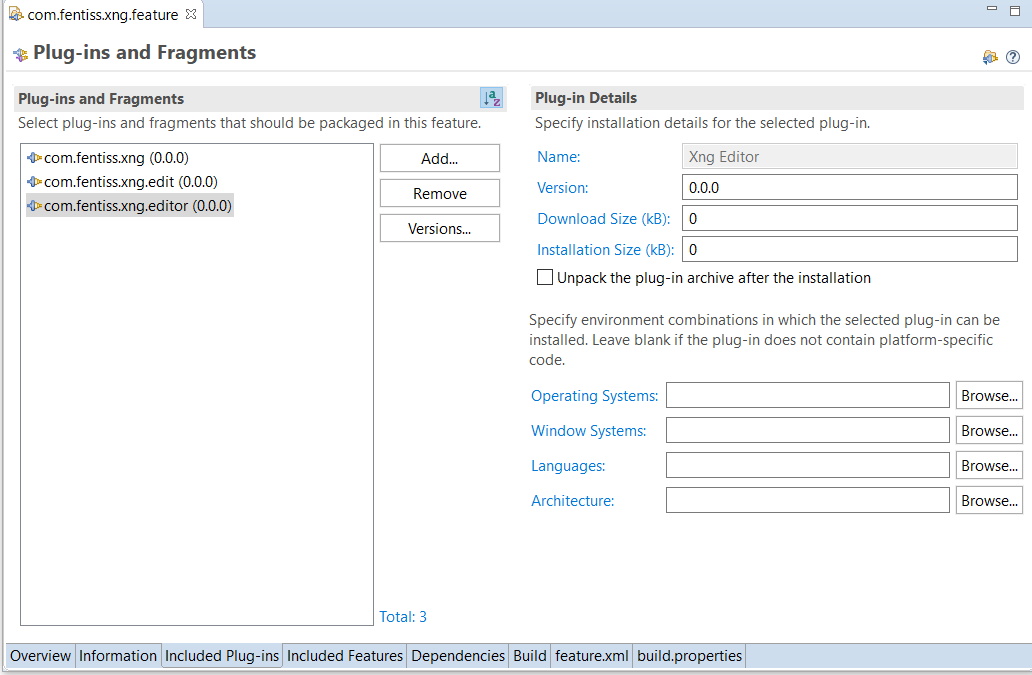
* **Increase version numbers in plugins, feature and update site**

1. On each of the 3 plugins XX, XX.edit and XX.editor :

open plugin.xml file

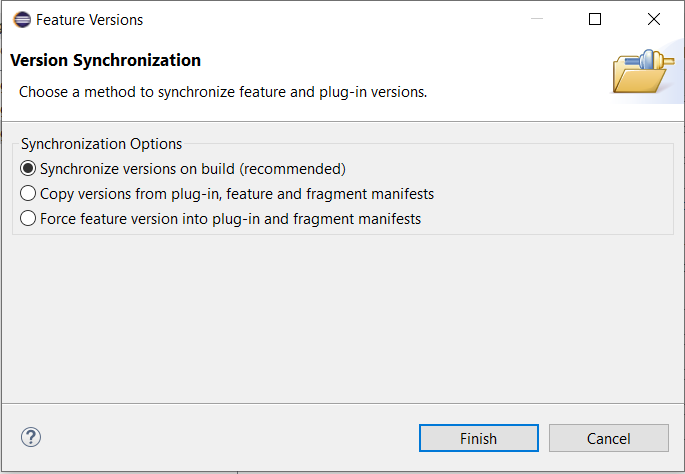
In “Overview” tab, update version number and save  


1. Open feature.xml file in the feature project
   * In “Overview” tab, increase version number  
     
   * In “Included Plug-ins” tab, check that no version is mentioned



In case the version is not 0.0.0 for each plugin,

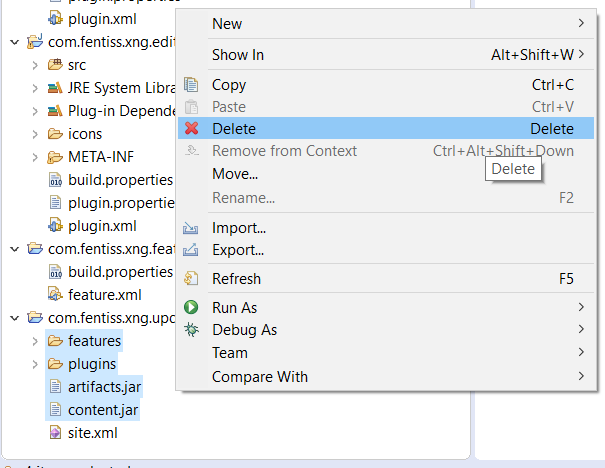
* + - Select each plugin in the list
    - Click “Versions…”
    - Select “Synchronize versions on build”



* + - Finish

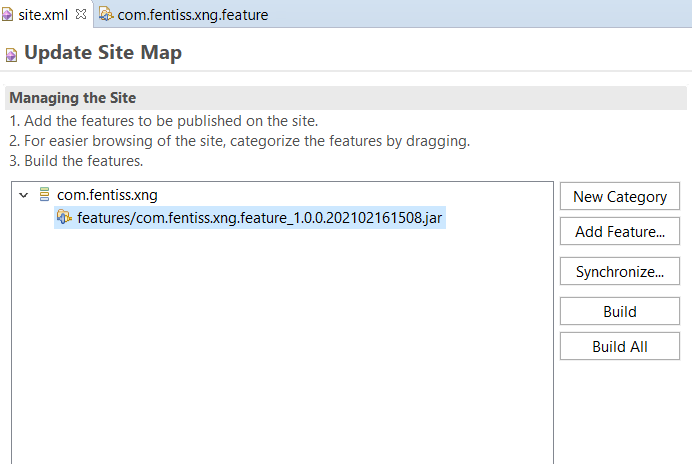
1. From the update site project created previously

* If the update site project contains more than site.xml file, delete all other files (they are generated during BUILD)

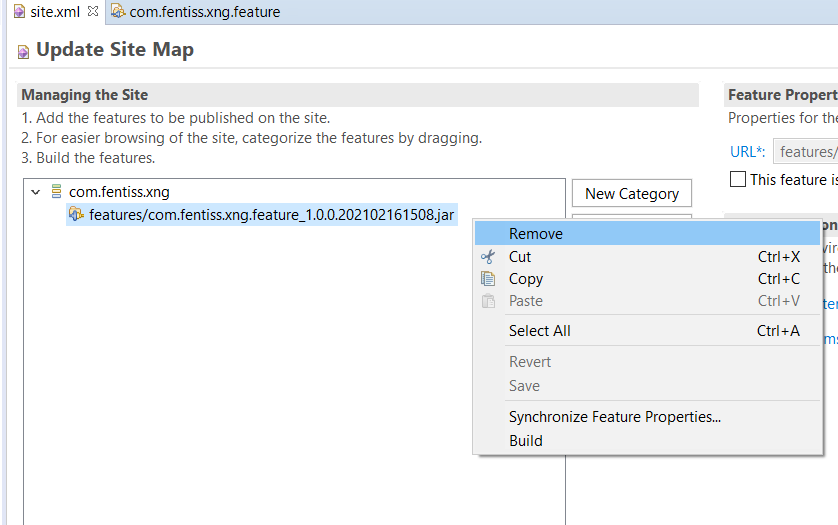


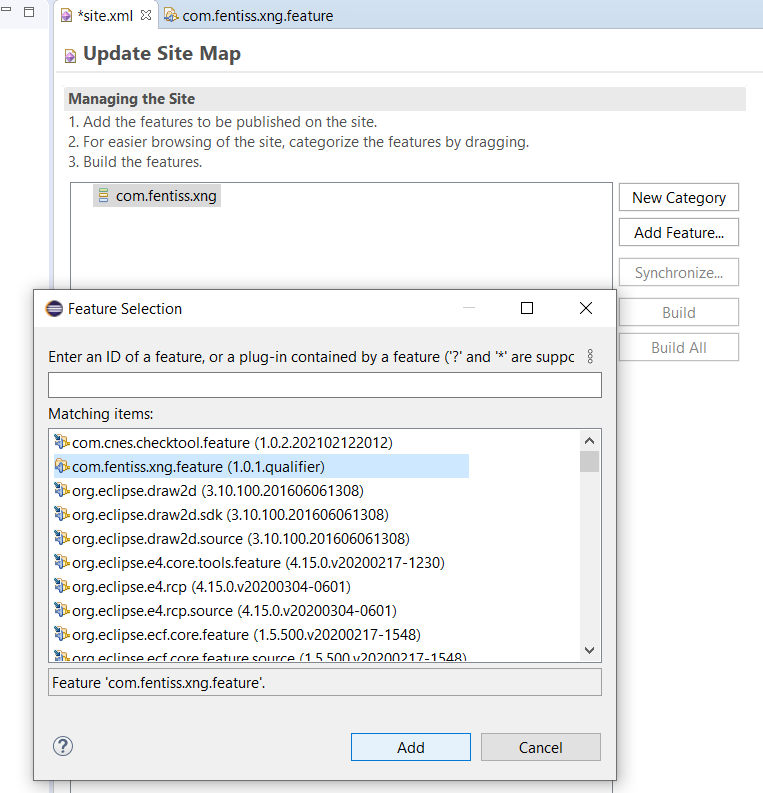
* Open site.xml

The content should look like the following : a category, and a feature with a version number in which the “.qualifier” is replaced by the build date (year month day time).

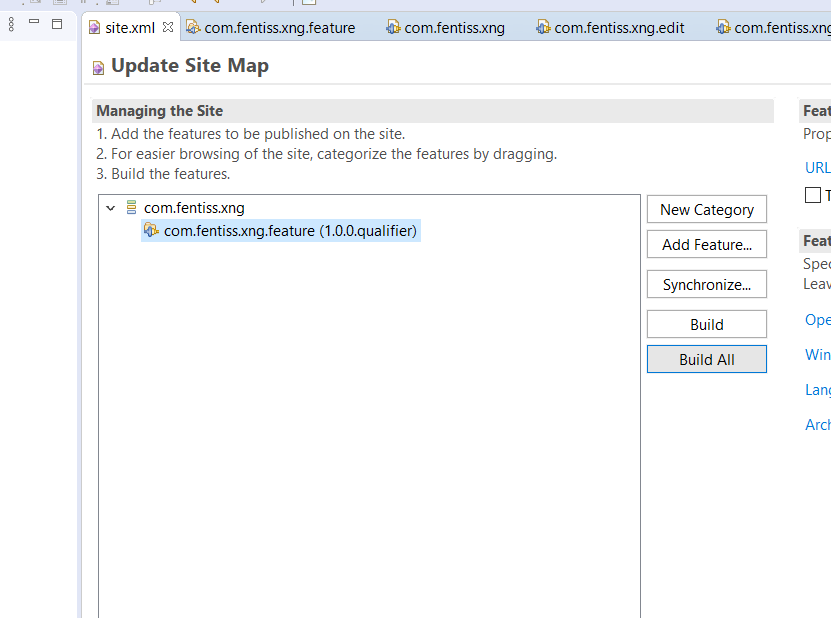


* Select the feature and right click > remove (to remove this previous version of the feature from last build)



* Select the category and click “Add feature…” to select the new version of the feature (with suffix .qualifier)  
  
* Save the site.xml
* If you store in version control system , this is the version to be stored (before BUILD)
* **Build the update site and release**

1. Open site.xml from the update site project
2. Click “Build All”



1. Wait until the progress is 100% (on the bottom of Eclipse main window )

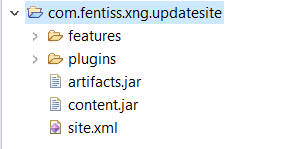


1. Look at update site project in ProjectExplorer view

Before build



After build

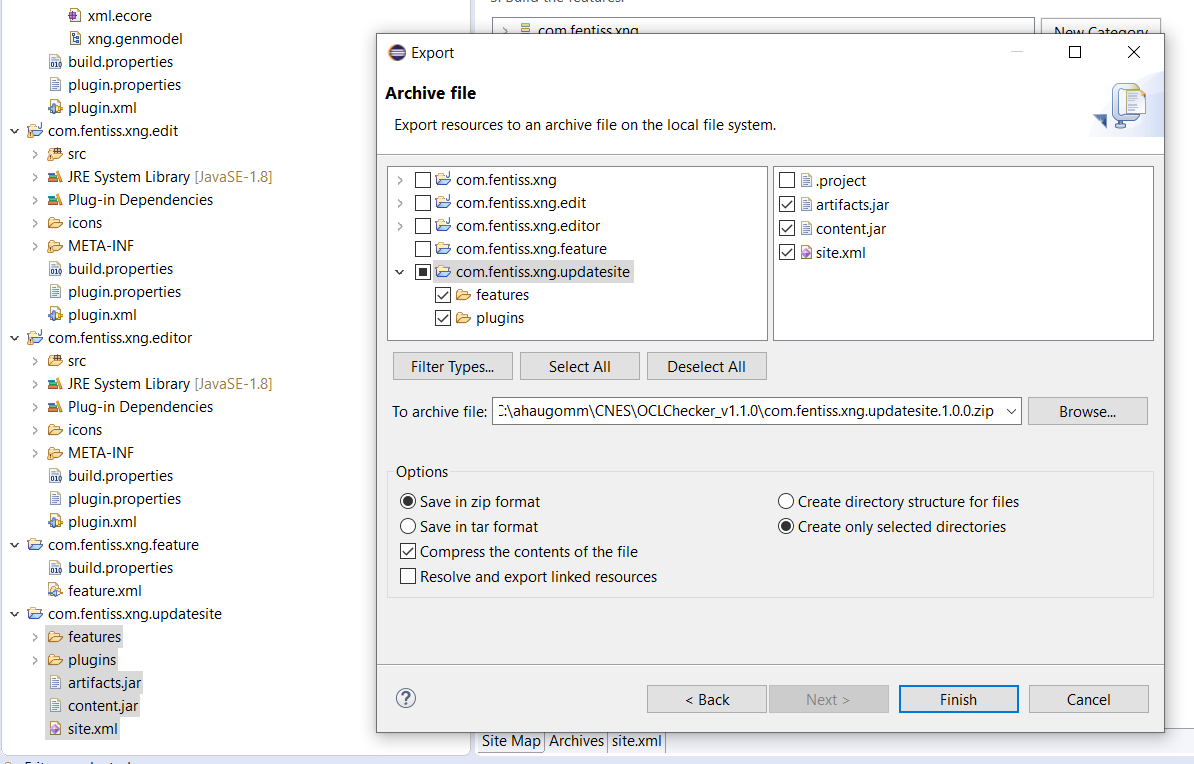


1. In order to share this update site, create an archive of all files in the project

WARNING : The archive must contain directly the site.xml file, no upper level folder.

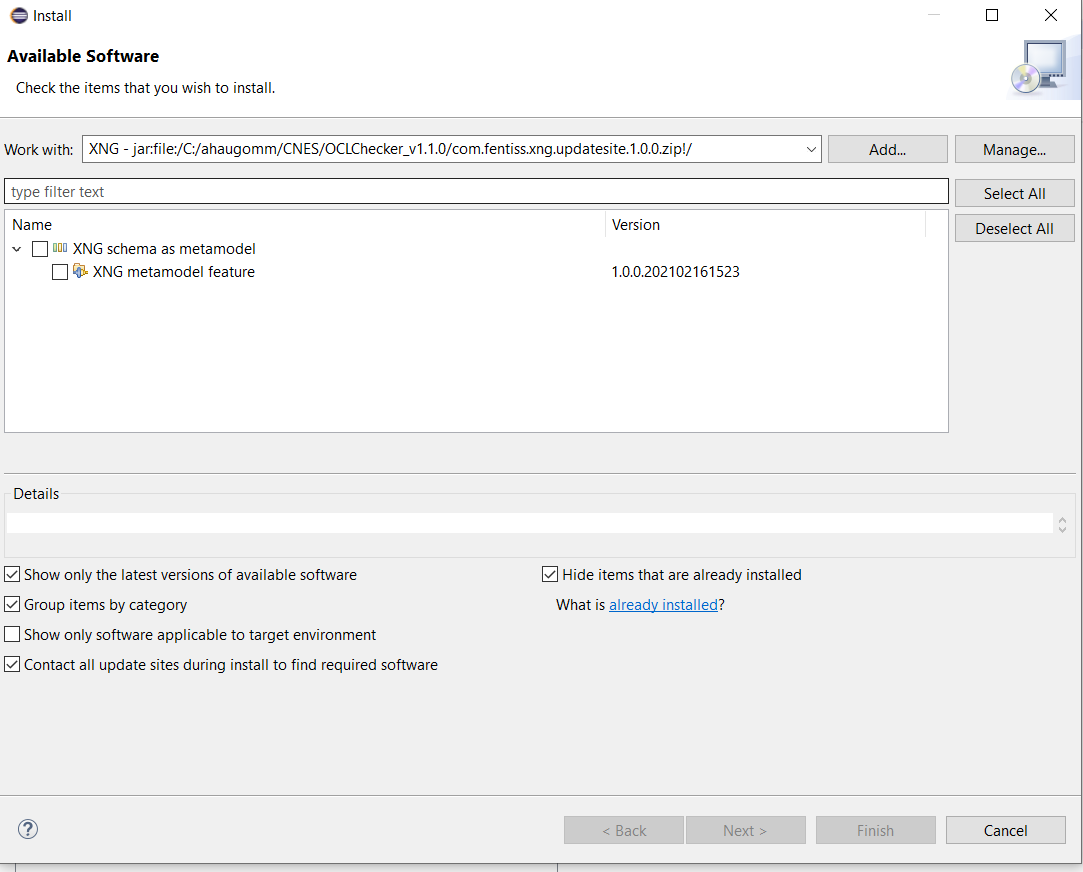
If you create the archive from Eclipse

* + Select the files (site.xml, artifacts.jar, contents.jar, features folder , plugins folder) and   
    click File > Export > Archive file > Next
  + Check option “Create only selected directories”



* + Finish

1. Open the resulting archive to ensure that you have site.xml at top level.
2. Test the installation from your eclipse platform
   * Help > Install new software > Add.. > Archive ..
   * Select archive of the new update site
   * Check that the two names in the check boxes correspond to
     + For highest level , the label of the category from site.xml in update site project
     + For lower level, the feature name (in feature.xml)



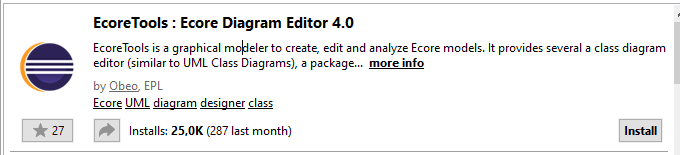
* + Click next twice
  + The text from the license correspond to what has been provided in feature.xml > information tab.

1. Share the archive file of the update site to users.

Note : the update site can also be shared as a URL.   
Extract the archive in a folder and provide the URL pointing to this folder (again, the folder containing directly site.xml file)

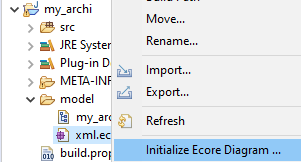
# APPENDIX 3 : Visualize metamodel as a diagram

*Prerequisite:*

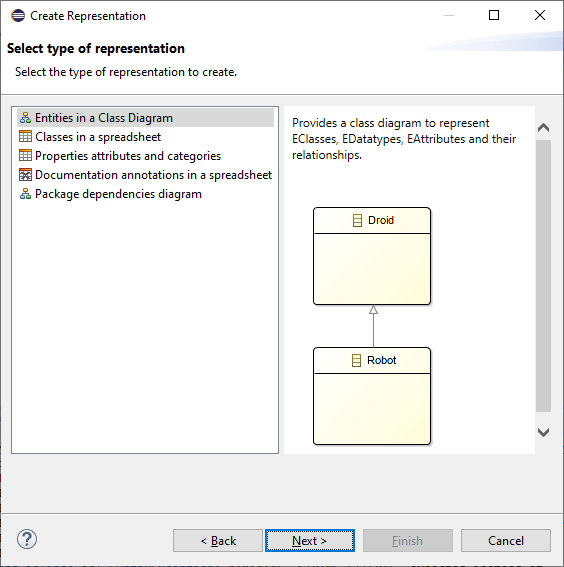


Here is the plugin documentation: <https://www.eclipse.org/ecoretools/doc/>.

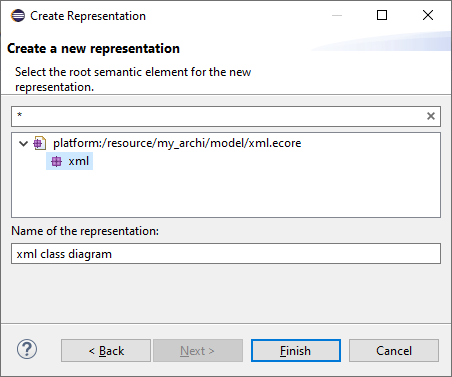
1. Select the .ecore file
2. Right click on it and go on Initialize Ecore Diagram …



1. Enter diagram name
2. Click on Next
3. Select Entities in a Class Diagram

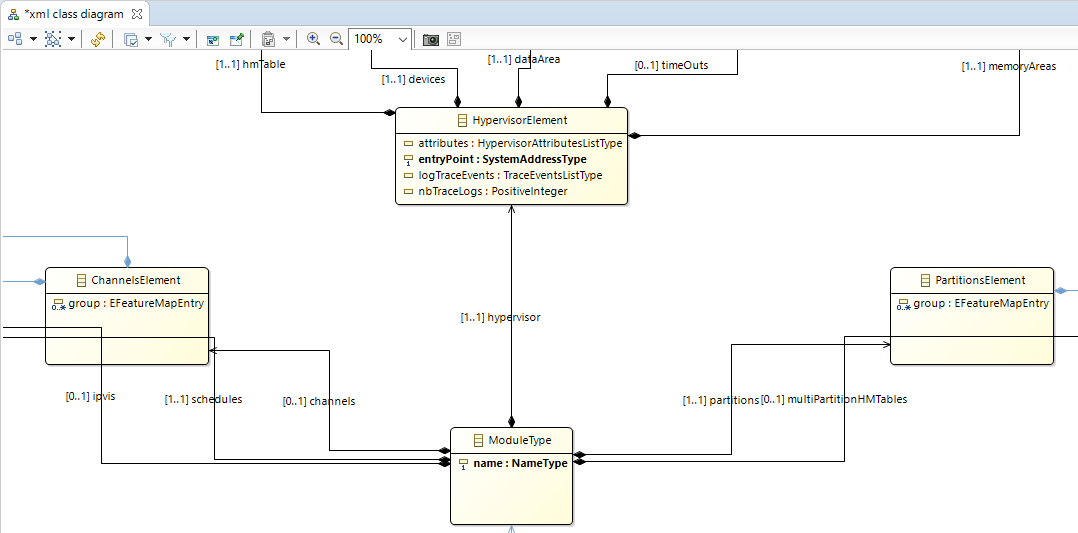


1. Click on Next
2. Select the related ecore

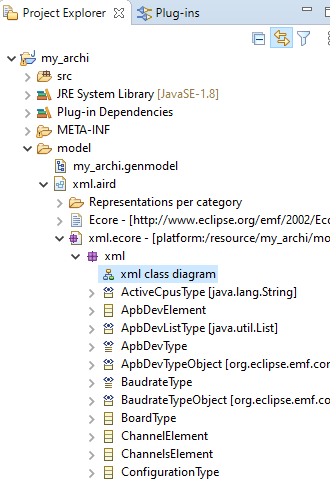


1. Click on Finish
2. A new .aird file appear in the project, click on it.
3. Double-click in the tab to initialize diagram, wait. Move in the diagram to see different elements.

Here is a diagram extract from an example:



Navigation in the different elements is also available in the project explorer:



**VERSIONS**

| *Issue* | *Date* | *Relevant Information* |
| --- | --- | --- |
| 1.0 | 2020-10-12 | * First issue for the OCL conf checker tool User Guide |
| 1.1 | 2021-03-12 | OCL Conf checker v1.1 (update installation prerequisites, appendices) |
|  |  |  |
|  |  |  |