Private Public Partnership Project (PPP)
Large-scale Integrated Project (IP)

D9.3.3: API IDE Support

Project acronym: FI-WARE  
Project full title: Future Internet Core Platform  
Contract No.: 285248  
Strategic Objective: FI.ICT-2011.1.7 Technology foundation: Future Internet Core Platform  
Project Document Number: ICT-2011-FI-285248-WP9-D.9.3.3  
Project Document Date: 2014-01-31  
Deliverable Type and Security: Public  
Author: FI-WARE Consortium  
Contributors: FI-WARE Consortium
1 Executive Summary

This version of the deliverable "API IDE Support" presents detailed information of the Tools Chapter in the Third Release. Moreover, this document reflects both the tools that have been added and the ones that have been updated from the previous version (including the redistribution of tools among the software deliverables of the Tools Chapter).

A description of the developed and selected tools is given in detail.
2 About This Document

The purpose of this document is to provide to the end users of the FI-WARE platform a set of tools that facilitate the discovery and the integration, in their FIApps, of the Generic Enablers Implementations provided by FI-WARE. To complement the traditional development environment here are proposed a set of well-known IDE extensions that have been selected to improve the integration with other external tools (e.g. version control systems, ticketing system) and the collaboration/communication among the development team members.

2.1 Intended Audience

This document and the tools described are mainly oriented to developers and system architects. In general, any developer or software architect that wants to take benefits from working with a FI-WARE Platform for developing Future Internet applications.

2.2 Chapter Context

The Tools chapter, in the context of FI-WARE project, is in charge of providing a set of tools and practices in order to address requirements and support the FI Application developers to manage the development of their Future Internet applications based on a FI-WARE Platform instance. More precisely, Future Internet applications will benefit from the services made available by the FI-WARE Platform (Generic Enablers instances).

To support the various activities, which compose the lifecycle of Future Internet applications, such set of tools provides support for collaboration, development, for training and for testing applications running on top of a Core Platform Instance.

2.3 Structure of this Document

The document is generated out of a set of documents provided in the public FI-WARE wiki. For the current version of the documents, please visit the public wiki at http://wiki.fi-ware.eu/

The following resources were used to generate this document:

- **D.9.3.3 API IDE Support front page**
  - **Tools.REST Client Generator**
    - An Eclipse plug-in that allows the developer to easily create (wizard style UI) a Java client to interact with RESTful services described by means of a WADL.

- **Tools.PROSA-IDE**
  - An Eclipse plug-in for the PROSA tool. PROSA is a tool for the QoS runtime monitoring and testing of the composed services during their operation in service compositions.

- **Tools.SoPeCo-IDE**
  - Software Performance Cockpit is a framework for systematic performance evaluations of software systems, based on systematic measurements, statistical methods, and machine learning.

- **Tools.FusionForge connector**
  - The FusionForge connector supports the Tasks and Tickets management functionality for the Eclipse Mylyn plug-in.

- **Tools.Eclipse Communication Framework**
The scope to adopt this plug-in is to support developer to share the same piece of code and edit it simultaneously.

**Tools.Wiki Edit support**

This component adds to the FI-WARE IDE the support of writing documents using the wiki paradigm.

**Tools.Source Code Management**

This client allows developers to easily interact with the source code repository directly from within the FI-WARE IDE.

**Tools.Library and Dependency Management**

The proposed composition of tools provides a comprehensive support to the developer on managing the project's libraries and dependencies in general.

### 2.4 Typographical Conventions

Starting with October 2012 the FI-WARE project improved the quality and streamlined the submission process for deliverables, generated out of the public and private FI-WARE wiki. The project is currently working on the migration of as many deliverables as possible towards the new system.

This document is rendered with semi-automatic scripts out of a MediaWiki system operated by the FI-WARE consortium.

#### 2.4.1 Links within this document

The links within this document point towards the wiki where the content was rendered from. You can browse these links in order to find the "current" status of the particular content.

Due to technical reasons not all pages that are part of this document can be linked document-local within the final document. For example, if an open specification references and "links" an API specification within the page text, you will find this link firstly pointing to the wiki, although the same content is usually integrated within the same submission as well.

#### 2.4.2 Figures

Figures are mainly inserted within the wiki as the following one:

```
[[Image:....|size|alignment|Caption]]
```

Only if the wiki-page uses this format, the related caption is applied on the printed document. As currently this format is not used consistently within the wiki, please understand that the rendered pages have different caption layouts and different caption formats in general. Due to technical reasons the caption can't be numbered automatically.

#### 2.4.3 Sample software code

Sample API-calls may be inserted like the following one.

```
http://[SERVER_URL]?filter=name:Simth*&index=20&limit=10
```
2.5 Acknowledgements

The current document has been elaborated using a number of collaborative tools, with the participation of Working Package Leaders and Architects as well as those partners in their teams they have decided to involve.

2.6 Keyword list

FI-WARE, PPP, FI-CoDE, Future Internet, Catalogue, Drupal, Generic Enabler, Library Management, Dependency Management, Maven, Eclipse, REST, RESTful, WADL

2.7 Changes History

<table>
<thead>
<tr>
<th>Release</th>
<th>Major changes description</th>
<th>Date</th>
<th>Editor</th>
</tr>
</thead>
<tbody>
<tr>
<td>v1</td>
<td>First draft</td>
<td>14 Jan 2014</td>
<td>E-IIS, UDE, SAP</td>
</tr>
<tr>
<td>v2</td>
<td>Version ready for submission</td>
<td>7 Feb 2014</td>
<td>E-IIS</td>
</tr>
</tbody>
</table>
Table of Contents

1 Executive Summary ..................................................................................................................2

2 About This Document ..............................................................................................................3
   2.1 Intended Audience ................................................................................................................3
   2.2 Chapter Context .....................................................................................................................3
   2.3 Structure of this Document ..................................................................................................3
   2.4 Typographical Conventions ................................................................................................4
   2.5 Acknowledgements ...............................................................................................................5
   2.6 Keyword list ..........................................................................................................................5
   2.7 Changes History ....................................................................................................................5

3 REST Client Generator ............................................................................................................8
   3.1 Introduction ..........................................................................................................................8
   3.2 Installation ............................................................................................................................8
   3.3 How to use ............................................................................................................................9
   3.4 Test REST client ..................................................................................................................13

4 PROSA-IDE ............................................................................................................................16
   4.1 Introduction ..........................................................................................................................16
   4.2 Architecture .........................................................................................................................16
   4.3 PROSA Integrated Development Environment (IDE) ......................................................17
   4.4 Installation and Usage .........................................................................................................26

5 SoPeCo-IDE ............................................................................................................................27
   5.1 Introduction ..........................................................................................................................27
   5.2 Architecture ........................................................................................................................27
   5.3 Installation ............................................................................................................................28
   5.4 Usage ....................................................................................................................................30

6 FusionForge connector ............................................................................................................31
   6.1 Introduction ..........................................................................................................................31
   6.2 Installation ............................................................................................................................31
   6.3 Usage ....................................................................................................................................33
   6.4 Support ................................................................................................................................51
   6.5 Appendix ..............................................................................................................................51

7 Eclipse Communication Framework .........................................................................................53
   7.1 Introduction ..........................................................................................................................53
   7.2 Installation ............................................................................................................................53
   7.3 Architecture ........................................................................................................................54
   7.4 How To ................................................................................................................................54
   7.5 Improvements ......................................................................................................................58

D.9.3.3 API IDE Support
D.9.3.3 API IDE Support
3 REST Client Generator

3.1 Introduction

The FI-CoDE platform offers a plug-in for the FI-CoDE IDE that allows to automatically generate a Java client for a given RESTful service. The main goal of this component is to simplify the integration of GEs in FI applications developed in Java. All the services published with a RESTful interface (here are included the Generic Enabler implementations) that provides a formal description of the service, using a WALD, file are directly supported by the REST Client Generator.

The detailed documentation, binaries and source code are available from the official download site reported below.

For the list of known issues or to provide feedback on this plug-in, use the on-line Bug tracker and select "REST Client Generator" as Asset option.

3.1.1 Information

Name: REST Client Generator
Version: 1.0.0
License: EPL v1.0
Download: http://forge.fi-ware.eu/frs/?group_id=15#title_rest-client-generator (Binaries and Source Code)
Update Site: http://www.fi-ware.eu/tools/updates
Scope: Create REST client from WADL

This document refers to the REST Client Generator plug-in release 1.0.0. Since in the FI-CoDE context, the previous plug-in (release 0.0.1) and the FIA plug-in provided by ATOS have to use Jersey API, a new plug-in was created in order to contain the common JARs that both plug-ins can use and on which both of them depend. The new plug-in is named "Rest Access Client". The functionalities of REST Client Generator plug-in (release 1.0.0) are the same of the previous version.

3.2 Installation

There are two way to install the REST Client Generator plug-in:

- from update-site
- to copy JARs in to the eclipse/plugins folder

3.2.1 Installation from update-site

This plug-in can be installed from update-site URL at link http://www.fi-ware.eu/tools/updates/
Fusion Forge Plug-ins

To install REST Client Generator plug-in, you can check FI-WARE REST client generator.

3.2.2 Installation with JARs

Another way to install this plug-in consists of in to copy the plug-in JARs from link https://forge.fi-ware.eu/frs/?group_id=15 in to the eclipse/plugins folder of the running Eclipse IDE:

- eu.fiware.ficode.clientgenerator_[version.datetime].jar
- eu.fiware.ficode.rest_[version.datetime].jar

Restart the Eclipse IDE and check whether the plug-in is actually installed in: Help > About Eclipse > Installation Details > Plug-ins.

3.3 How to use

To use REST Client Generator plug-in you must open the Eclipse IDE and select the path: File > New > Other...

The Menu will show you the "REST Client Generator" folder listing two wizards to generate your REST client (see figure below):

1. "generate client from file" allows you to generate a client using the WADL file
2. "generate client from URL" allows you to generate a client using an endpoint

REST Client Generator menu

3.3.1 Generate client from file for Java Project

If you want to add your REST client to a Java Project, having already your service’s WADL file, select the option "Generate client from file" wizard.
When you click on the “Finish” button a new class (HelloClientFromFile) is created. You can note that:

- the “Dependencies” field allows you to add dependencies to your project. As shown in the figure above, if you add a “jersey-bundle-1.16.jar” library, this .jar will be used by your new REST client class.
- in the “File wadl” field must be selected a file having a .wadl extension.

### 3.3.2 Generate client from file for Maven Project
If you want to add your REST client to a Maven Project, having already your service's WADL file, select the option “Generate client from file” wizard.

In this case, you have to set the “Dependencies” field and enable pom.xml option.
When you click on the “Finish” button a new class (HelloClientFromFile) is created. You can note that:

- the “Dependencies” field allows you to add dependencies to your project. As shown in the figure above, if you add a “jersey-apache-client4” artifact, this dependency will be added in your pom.xml file.

3.3.3 Generate client from URL for Java Project

If you want to add your REST client in a Java Project, having already your service's endpoint, you can choose the “Generate client from URL” wizard.
Client from URL for Java Project

When you click on the “Finish” button the new class (HelloClientFromURL) is created. You can note that:

- in this case the “Dependencies” field is left on the default option, which is “no dependency”, since the dependency was already set for the Java Project.

3.3.4 Generate client from URL for Maven Project

If you want to add your REST client to a Maven Project, already having your service’s endpoint, you can choose the “Generate client from URL” wizard.
When you click on the “Finish” button the new class (HelloClientFromURL) is created. You can note that:

- in this case the "Dependencies" field is left on the default option, which is "no dependency", since the dependency was already set for the Maven Project.

### 3.4 Test REST client

To test your REST client, the Test class was created having the following main method:

```java
public static void main(String[] args) {
    String param = "FI-WARE from File";
    HelloClientFromFile helloFromFile = new HelloClientFromFile();
    String resultFromFile = helloFromFile.hello().param(param).getAsTextPlain(String.class);
    System.out.println("HelloClientFromFile -> " + resultFromFile);
}
```
param = "FI-WARE from URL";
HelloClientFromURL helloFromURL = new HelloClientFromURL();
String resultFromURL = helloFromURL.hello().param(param).getAsTextPlain(String.class);

System.out.println("HelloClientFromURL -> " + resultFromURL);
}

If you run the above code, you will see the following result in the output:

HelloClientFromFile -> Jersey say : FI-WARE from File
HelloClientFromURL -> Jersey say : FI-WARE from URL

The actual REST service class (HelloWorldService) is shown in the following code:

@Path("/hello")
@Produces(MediaType.TEXT_PLAIN)
public class HelloWorldService {

    @GET
    public String sayHello() {
        return "Hello Jersey";
    }

    @GET
    @Path("/{param}")
    public Response getMsg(@PathParam("param") String msg) {
        String output = "Jersey say : " + msg;
        return Response.status(200).entity(output).build();
    }
}

3.4.1 Settings proxy and endpoint
You can note that when you use the instance of your client class, you must call the hello() method (see in the above code) to get the "Hello" object. There are three available hello() methods (overload):
1. `hello()`  
2. `hello(Client client)`  
3. `hello(Client client, URI baseURI)`  

where `client` is a `com.sun.jersey.api.client.Client` object and `baseURI` is a `java.net.URI` object.

If you need to set a proxy, you can use your custom client (you must use ApacheHttpClient API): for example you can use the following code to get the client object:

```java
private static DefaultApacheHttpClient4Config getConfig(){
    DefaultApacheHttpClient4Config config = new DefaultApacheHttpClient4Config();
    config.getProperties().put(ApacheHttpClient4Config.PROPERTY_PROXY_URI, "http://" + "proxy_url" + ":" + "proxy_port");
    config.getProperties().put(ApacheHttpClient4Config.PROPERTY_PROXY_USERNAME, "username");
    config.getProperties().put(ApacheHttpClient4Config.PROPERTY_PROXY_PASSWORD, "password");
    return config;
}
```

And to get the `client` and `URI` object, you can type:

```java
Client client = ApacheHttpClient4.create(getConfig());
URI baseURI = new URI("http://your.service.com/");
helloFromFile.hello(client, baseURI);
```
4 PROSA-IDE

4.1 Introduction

PROSA is an online testing framework targeted to ensure the constant availability of QoS monitoring data for a given service (used by a FI App). The runtime solution monitors the (real) usage of a given service and additionally allows performing additional service invocations (online tests) where necessary. PROSA thereby allows users to ensure that a minimum set of QoS data points is available for the given service for every time interval.


To allow easy integration with FI-CODE IDE, the PROSA tool is provided as an Eclipse plug-in.

4.1.1 Information

Name: PROSA
Version: 3.2.0
License: EPL v1.0
Download: http://forge.fi-ware.eu/frs/?group_id=15#title_prosa
Scope: QoS Testing and Monitoring

4.2 Architecture
The High Level Architecture of PROSA

The above figure shows the high level architecture of PROSA. The key components of PROSA are the Client, the Online Testing Module, and the Monitoring Module. In this deliverable we focus only on the Client.

The Client is an Eclipse plug-in with the purpose to allow the user interact with the tool. For example using the client, the user can connect PROSA with service composition engines; SCE; e.g., Apache ODE and Activiti, and retrieve the deployed processes in those execution engines, and select the process/service/task of interest.

Through the Client, the user can view the Monitoring Data (i.e., QoS Data) available from both past and current executions of the process instances and their constituent services/tasks.

If for a particular service additional QoS Data is required (e.g., for performing more sound analysis), the user can use the PROSA Client to specify the online testing configuration (e.g., service, input, and test frequency), and test the service accordingly. The QoS Data resulting from testing is also visualized using the Client. To facilitate the evaluation of the QoS Data for a particular service, the PROSA Client also views the QoS Data from both monitoring the service as well as testing the service in one single graph.

PROSA stores the collected QoS Data permanently using the Monitoring Module. The user may want to retrieve the QoS Data for a particular instance, service, or task that was collected in a certain time period. The Client allows the user to export the available QoS Data for instances, services and tasks, available from a certain time period, and save the results in a csv file.

4.3 PROSA Integrated Development Environment (IDE)

In this section, we present the PROSA Client. We highlight the several features of the PROSA tool and how they are enabled by the PROSA Client.

4.3.1 PROSA Perspective

As an Eclipse plug-in, PROSA has its own perspective. To open the PROSA perspective, the user can go to: "Window"--"Open Perspective"--"Other", and then choose the "Prosa Perspective".
The PROSA Perspective

Once the PROSA Perspective is selected, the following main page will appear.
4.3.2 PROSA Menu

After opening the Eclipse instance in which the PROSA Plug-in is installed, the user can see the menu "PROSA" for the PROSA Client. The menu has two options: "Start" and "Export QoS Data".
As the labels of the options indicate, the "Start" option is for starting the PROSA tool and the "Export QoS Data" option is for exporting the available QoS data for instances, services or tasks.

4.3.2.1 **The "Start" Option**

When the user selects the "Start" option from the PROSA menu, the "Deployed Processes" view will appear as shown below.
Starting PROSA - The "Deployed Processes" View

On the left-hand side, the "Deployed Processes" includes a place holder for listing the deployed processes when the tool is connected to the Apache ODE and Activiti execution engines. At the bottom-left corner, the "Deployed Processes" has the "Update" button. When the user wants to retrieve the deployed processes, the user has to click on the "Update" button. To facilitate the navigation through the deployed processes, the PROSA Client employs a tree structure for listing the deployed processes and their constituent services or tasks as shown below.
Deployed Processes in Apache ODE and Activiti

The user can click on any of the processes, services or tasks. The following views then appear.

The "QoS History for Process Instances" View

This view is devoted for displaying the available QoS Data of the process instances and the services or tasks invoked in the instances. The view includes an interactive "x-y plot" (or chart) with two axis: "Invocation time" (i.e., when the instance was invoked) and "Response time (ms)" (i.e., the time duration until we get the response from the instance). Whenever a new instance of the process is invoked, the plot displays its response time as soon as the instance’s execution is finished (i.e., real time). The blow figure shows the response times of the instances of an example process.
The PROSA "QoS History for Process Instances" View

A "point" in the plot maps the response time to a particular instance. The user can click on the "point" to see a more detailed value of the response time.

The plot allows interactive interaction with the user, such that, the user can select part of the plot and zoom-in and out to better focus on the "points" in the selected areas. This is especially helpful when there are many "points" shown in a short time period.

The user can also see the QoS data of the services/tasks that were used in a particular instance. This can be performed by clicking on the respective "point" of an instance and the results will be shown in a separate below plot in the same view, where on the x-axis the services/tasks of the instance are shown, and on the y-axis the respective response time is shown. Each "point" in the plot maps a response time value for a service/task.

The "Web Service QoS History" View

Once the user clicks on a service/task in the tree (belonging to a process), the "Web Service QoS History" view opens automatically. This view is devoted for displaying (in an x-y plot) all the available QoS data for the service that is clicked on. The QoS data of a service/task can origin from monitoring the execution of the service while invoked in different instances of the process it belongs to. Additionally, the QoS data of a service, used in a process deployed in Apache ODE, can be available from online testing the service using the PROSA tool (see the PROSA "Web Service Testing" view). The plot used in the "Web Service QoS History" view is also interactive, displays in real time both the QoS data of the service from monitoring and from testing together. The below figure shows the PROSA "QoS Historical Data" view.
The PROSA "Web Service QoS History" View

The "Web Service Testing" View

The below figure shows the "Web Service Testing" view which also automatically opens when a service is clicked on. This view includes three parts. The first part shows information about the service under test (name, URL, and operation). The second part is dedicated for specifying the online testing parameters (i.e., test inputs and test frequency). The third part is devoted for the x-y plot which will display the QoS data resulting from testing the service. The plot type is same as the one used for displaying the QoS history of the instances and services/tasks.
4.3.2.2 **The "Export QoS Data" Option**

The second option of the PROSA menu is the "Export CSV Data". Once this option is selected, the "Export CSV Data" view automatically opens. This view allows the user specify the QoS data to be exported. For example, it includes a menu which lists all the available processes, date and time intervals for specifying the date and time of the "First Execution" and "Last Execution", a menu for listing the available operations used in the selected process, another menu for selecting the source of the QoS data (i.e., Monitoring, Testing, or Monitoring and Testing together), and finally a field for providing the full path of the file where the results should be save (see the below figure). The "Export CSV Data" view also includes a part for displaying the QoS data that will be exported as specified by the user. This allows the user to check that s/he made a correct selection before exporting the QoS data to a file.
4.4 Installation and Usage

The installation and usage of the PROSA tool is documented in deliverable 9.4.3. There we provided a step by step guide for installing and using PROSA. Please refer to this document for further details.
5 SoPeCo-IDE

5.1 Introduction

The Software Performance Cockpit (SoPeCo) is a framework for systematic performance evaluations of software systems, based on systematic measurements, statistical methods, and machine learning. It enables developers to do systematic goal-oriented measurement scenario during early stages of the creation of their artifacts and thereby reduces costs and increases time to market. Being improved and adapted for FI-WARE, SoPeCo plays an essential role in FI-WARE by enabling the performance-aware development of GEs and FI Applications including performance-based comparison and selection of GE implementations. In this chapter, the focus will be on the UI and IDE aspects of SoPeCo.

5.1.1 Information

**Name:** Software Performance Cockpit  
**Version:** 5.0.2  
**License:** BSD  
**Website:** [http://www.sopeco.org/](http://www.sopeco.org/)  
**Catalogue:** [http://catalogue.fi-ware.eu/enablers/sopeco](http://catalogue.fi-ware.eu/enablers/sopeco)  
**Download:** [http://forge.fi-ware.eu/frs/?group_id=15#title_sopeco](http://forge.fi-ware.eu/frs/?group_id=15#title_sopeco)  
**Scope:** Measurement-based Performance Testing and Analysis

5.2 Architecture

A SoPeCo setup consists of the following main components shown in the figure below.
SoPeCo components (high level)

At the bottom, we have the service- or system under test (SUT), which represents the artifact a developer wants to analyze. From SoPeCo’s point of view, this artifact is considered to be a black box. That might run on the developer’s local machine or some remotely accessible location.

In order to observe and interact with the SUT, the developer has to implement a SoPeCo Measurement Controller (MEController). This MEController acts as an adapter connecting the SUT with the SoPeCo Engine. It can act as a workload driver or coordinate an external workload driver. Also the measurements and observations are done by the MEController itself or coordinated in case of using external tools (like SAR etc.). In order to implement the MEController, the developer leverages the SoPeCo core library to implement the single class that covers the logic. This is done in Java based on a lightweight approach which provides the developer with enough flexibility to choose his preferred development environment. In the course of this document, we will focus on the Eclipse IDE.

On top of the MEController, the SoPeCo Engine is responsible for coordinating the actual experiment run and to analyze the results. It steers the MEController through the defined series of experiments using the defined exploration strategies for varying the test parameters and dimensions. As such, it acts as the core of the SoPeCo framework. Building on top of it, the SoPeCo Web UI provides browser based interaction in order to improve ease of use as well as productivity. Providing a publicly available instance of the SoPeCo Web UI, the users are freed from any installation needs and use the tool instantly. Also sharing for instance the same project specific account, they can use the hosted instance as a central location for collecting performance results.

5.3 Installation

In this chapter we will focus on the user interface relevant parts of the SoPeCo framework. In particular we can consider two tasks and their related components: First we have the development of the MEController which happens within an IDE. The second part then is the SoPeCo Web UI which enables developers to use SoPeCo via the browser.

5.3.1 MEController development

The development of an MEController is a straightforward process and centered around the idea of providing a Java library that provides the interfaces and heavy lifting in order to drastically lower implementation efforts. From a technical point of view, this means that you need to download the SoPeCo core JAR file from the tools section of the FI-WARE forge at http://forge.fi-ware.eu/FRS/?group_id=15#title_sopeco

This library has to be added to a new Java project that you can implement in your preferred Java development environment. Given FiCoDE has standardized on the Eclipse IDE (Indigo release), we made sure that the SoPeCo library integrates well with that environment. In the picture below you see an example of an initial MEController implementation in Eclipse with automated stubs.
5.3.2 SoPeCo Web UI

In the current implementation of SoPeCo, the Engine and the Web UI are closely connected. This is why they are installed together in one bundle:

1. Download and install Java 6 or later (JDK).
2. Download and install Apache Derby which acts as a database for persisting SoPeCo experiment run results.
3. Download and install Apache Tomcat 7 or later as application server.
4. Download the WAR file of the SoPeCo Web UI from the tools section of the FI-WARE forge at http://forge.fi-ware.eu/frs/?group_id=15#title_sopeco
5. Start the derby database with `java -jar <PATH-TO-APACHE-DERBY-INSTALL-DIR>/lib/derbyrun.jar server start`
6. Start the application server
7. Place the WAR file into the ./webapps/ folder of the application server
8. Open a browser and access the SoPeCo Web UI at http://localhost:8080/sopeco/

Congratulation, you will now see the login screen of the SoPeCo Web UI and be able to define and run performance experiments.
login screen

Another alternative to the steps described before is the usage of the publicly available instance at http://app.sopeco.org/ where you can directly start using SoPeCo without any prior installation work.

5.4 Usage

The usage of the SoPeCo framework is documented in deliverable 9.4.3. There we provided a step by step guide for using SoPeCo to performance test the Repository GE provided by FI-WARE. Please refer to this document for further details.
6 FusionForge connector

6.1 Introduction

This user manual is for providing information on how to install and use the FusionForge connector. This connector is designed to work within the Eclipse IDE \[1\] and it's based on the Mylyn \[2\] plug-in. The development is done in the context of the FI-WARE \[3\] project. This manual refers to the FusionForge connector release 1.1.0 and it supports the Task and Ticket (Tracker) management functionality. Together with the binary packages, also the source code page is available. Note that the FusionForge connector is released under the open source licence EPL v1.0 \[4\].

6.1.1 Information

Name: FusionForge Connector
Version: 1.1.0
License: EPL v1.0
Download: Binaries and Source Code
Update Site: www.fi-ware.eu/tools/updates
Scope: Ticket and Task management

6.2 Installation

The installation of the FusionForge connector can be done either manually, as described below, or using the update site reported in the Information section.

6.2.1 Tested Environments

- Eclipse Indigo 64bit (Linux, Windows)
- Java JDK 1.6

6.2.2 Dependencies

The FusionForge connector requires the Mylyn Commons SDK package. This can be installed from the Eclipse IDE.

- open the wizard: Help – Install New Software …
- select the Mylyn update site from the Work with combo-box
- check the flag for Mylyn Commons SDK
- continue until the end of the Install steps.
Figure 1

Restart the Eclipse IDE and check that the Mylyn Commons SDK is correctly installed. You must check it in Help – About Eclipse – Installation Details – Installed Software.
6.2.3 Install the packages
Copy the plug-in packages into the `eclipse/plugins` folder of your local Eclipse installation.

- eu.fiware.fusionforge.client_1.1.0.201210111152.jar
- eu.fiware.mylyn.fusionforge.core_1.1.0.201210111152.jar
- eu.fiware.mylyn.fusionforge.ui_1.1.0.201210111152.jar

Restart the Eclipse IDE and check that the plug-in is correctly installed:

- Help – About Eclipse – Installation Details – Plug-ins

6.3 Usage
This chapter describes how to use the FusionForge connector following the Mylyn user interface.

6.3.1 Task Repository
To configure a FusionForge task repository follow the steps below.

D.9.3.3 API IDE Support
**Step 1**
Open the Task Repositories view

![Show View](image)

**Figure 4**

**Step 2**
Select *Add Task Repository* from the contextual menu

![Task Repositories](image)

**Figure 5**

**Step 3**
Select FusionForge and proceed with *Next*. 
Figure 6

Step 4
Enter the forge credentials and click *Validate Settings* to test the connection.
When you press Finish you are ready to create a new query, task or ticket.

6.3.2 Create Query
The connector retrieves from the forge all the information you need, according to your user.

Click on FusionForge connector and choose a “New Query...” to start a query wizard.
When you create a query, you can choose to create a task or ticket query. Task query retrieves only tasks in the forge while ticket query only tickets. For both, task and ticket, you can choose two query types:

- **custom query**: you can use a form where it's possible to customize the query
- **predefined filters query**: you can use some predefined filters for your query

### 6.3.2.1 Custom task query

Select the query type: *Create task query using form*
Click \textit{Next} to create a new query task. Define the custom query you want to execute: give it a name and select the values for the available fields.
When you press Finish the query is stored and the connector starts to retrieve the tasks that satisfy the filter.

6.3.2.2 *Predefined task query*

Select the predefined filter query: *Predefined task filter for selected project.*
Figure 12

You must choose the project (or All Projects) and a typologies to enable the *Finish* button.

6.3.2.3  **Custom ticket query**

Select the query type: *Create ticket query using form*.
When you press Finish the query is stored and the connector starts to retrieve the tasks that satisfy the filter.

6.3.2.4 **Predefined ticket query**
Select the predefined filter query: Predefined ticket filter for selected project.
You have to select the project (or All Projects) and a typology to enable the Finish button.

6.3.3 Create a new task
Click on FusionForge connector and choose a “New Task...” to start a task wizard.
Select the Project you want to use and click on *Finish* button.
6.3.4 Create a new ticket
Click on connector and choose a “New Ticket...” to start a ticket wizard.

Select the Project you want to use and click on Finish button.
6.3.5 Task List View
In the Task List View panel you can see your query as a folder that contains the tasks and tickets.

![Task List View](image)

You can use the Task List View to open queries, tasks and tickets for editing. From the Task List contextual menu it's possible to create new queries and new tasks or tickets.

6.3.6 Task Detail
Double click on a task in the Task List View and you get the task detail on the main eclipse panel.
From this panel it's possible to edit the task fields and submit them to the server.

6.3.7 Ticket Detail
Double click on a ticket in the Task List View and you get the ticket detail on the main eclipse panel.
In this view, there are some custom fields (for instance SelectBox, RadioBox, Textfield, ShortDetails).
6.3.8 Contextual Menu

A new feature that comes with this new release is contextual menu. To use the contextual menu you can select a text from an open file and click the mouse’s right button as in the next figure.

![Contextual Menu Example](image)

From “test.txt” file, it has been selected the “test” word and you can start the search inside Task and Ticket. The Forum option is not yet implemented. The search operation is the same for tasks and tickets. Let's retrieve information from tasks, the following figure shows the search wizard.
Figure 23

Choose your repository and push the “Start Search” button to retrieve all tasks that have in summary or description the “Query text” field. The following view shows the result of the search. In this case you can open the task from the list and edit it, or you can add a new task simply pushing the “Add new Task” button.
If you choose to add a new task, follow the wizard and a new task in edit mode is open for you with your selected text in the summary and in description. The search for ticket has the same behaviour.

Note that the search in the tasks or tickets can takes time, depending on the number of tasks, or tickets, stored into the repository. To improve this performance issue, the first time all tasks or tickets are stored in a local cache, so the next time the search is faster. In this case any ticket or task added or changed into the repository will not be taken into account.

A timeout period is used to refresh the cache content. You can set the cache timeout from Window – Preference – Fusion Forge – Cache settings. The default timeout is 30 minutes.
6.4 Support

If you want to notify a bug or a proposal for improvement please refer to the public tracker at: https://forge.fi-ware.eu/tracker/?atid=208&group_id=15&func=browse

6.5 Appendix

6.5.1 FusionForge patch

It has been developed a patch for the FusionForge platform in order to integrate it with the software tools developed in FI-WARE, and especially with the "FusionForge Connector" and "FusionForge User Project Management". The code contained in this patch is already contributed to the FusionForge community.
6.5.2 OSLC-CM

In parallel with the development of the dedicated connector for FusionForge, it has been taken into account the possibility to adopt a connector based on OSLC-CM specifications. The aim of these specifications is to provide a common API for ALM tools, and in the case of managing tickets from the IDE, this is reflected in the possibility to have one single (Mylyn) connector and not one per tool. Monitoring the development activities of this OSLC-CM connector, and from what is reported in the official web site (Mylyn OSLC Connectors), there are still some feature missing that prevent it's adoption to support the entire process of managing tickets.
7 Eclipse Communication Framework

7.1 Introduction

The scope to adopt this plug-in is to support developer to share the same piece of code and edit it simultaneously. The scenarios covered by this feature are for example: collaborative development, where two developers can write on the same file at same time; support/teaching, when it would be easier from a member (of a distributed team) to ask and receive support on a piece of code written by another member; peer review, as supporting tool that improve the effectiveness of this practice. The effective adoption of this plug-in depends then on the specific context in which the project is developed.

Next chapters, briefly, describes the installation steps and gives a more detailed idea of the typical usage of the plug-in. At the end is proposed an extension of the Collaborative Development Environment in order to have a better and more effective integration with this collaborative development service.

7.1.1 Information

Name: Eclipse Communication Framework
Version: 3.7.1
License: EPL v1.0
Download: Eclipse Marketplace
Documentation:

- Official Website
- DocShare Plugin

Scope: The main goal for adopting this plug-in is to support the developers in sharing the same piece of code and edit it in a real-time and collaborative way.

7.2 Installation

This plug-in can be installed directly from the Eclipse IDE opening the wizard at Help -> Eclipse Marketplace ... and searching for Eclipse Communication Framework (see image). Install the ECF component following the guided steps.

![Eclipse Marketplace](image-url)
Eclipse Marketplace ...

The official web page of this plug-in is: Eclipse Communication Framework

7.2.1 Requirements

The users that want to work with ECF plug-in have to be registered to an Instant Messaging service (IM service) compatible with the ones supported by ECF. A user can start a Collaborative Session with the users on his/her contacts list that have ECF installed.

7.3 Architecture

The architecture diagram (see image) represents the main components involved in order to have the collaborative development feature available. The Instant Messaging service (IM service) is considered part of the CDE even if it's provided by third parties.

The two users involved in the collaboration session shares the same code interacting with their own Eclipse IDEs. The information exchange is controlled by the IM service through which the users are connected.

7.4 How To

This chapter contains operative indications on how to execute specific tasks.

7.4.1 Configure IM account

The first step to have the plug-in features available (after the installation), is to configure an existing IM account. From the Eclipse IDE open the Communications Perspective and again open the Contacts View panel.
XMPPS New Connection

According to the selected IM type the proper fields are prompted to the user (see image for an example). Press Finish to retrieve the user contact list (see image for an example).

7.4.2 Shared editing

The Eclipse Communication Framework allows to share the editing of a file with another person. This feature comes out of the box once the plug-in is correctly installed and configured. The preconditions to enable the shared editing of the same file are:

- both parties
  - are connected to the same IM service;
  - are running Eclipse and have the ECF properly installed and configured;
- each one belongs to the contact list of the other;
- have the same project on the workspace.

User #1 open the file to share with User #2. From the contextual menu (on the opened file) select Share Editor With ... and from the sub-menus select the contact person you what to share the edit with. Once selected, the contact, User #2, receives a notification message to accept or reject the request from the User #1. If the User #2 accepts, a temporary copy of the shared file is opened on User #2 editor, and both, User #1 and User #2, can start typing simultaneously. All the changes on the text that are done by one party are reflected in real time to the other party, included cursor position and text selections. The sequence diagram Shared Editing represents the interaction steps between the users and the IM service.

D.9.3.3 API IDE Support
Once the connection is established, the two users can chat using the Message View. (see the image below)
Future Internet Core Platform

D.9.3.3 API IDE Support

**Shared Editing - Chat Messages**

When User #1 requests for a shared editing session, User #2 receives a pop-up message with that request. (see the image below)

**Shared Editing - Request**

When the User #2 accepts the request to share the editing of a file, the two users can write on the same file and receive the updates in real-time, cursor position and text highlight, included. (see the image below)
Shared Editing - Real-time

7.5 Improvements

This chapter proposes how to improve the Collaborative Editing integration into the FI-CoDE. The sequence diagram represents a complete process that goes from the user subscription (to a project) to sharing a file for collaborative editing with another member of the project.

The entire diagram can be split in two main parts (referring to step numbers):

from 1 to 3
- additional steps to extend the process of joining a project

from 4 to 12
- usual process steps to share and edit a file with a contact.
The proposed improvement consists in some additional steps executed when a request to join a project on the CDE (e.g. forge) from a user is accepted. These additional steps, as reported on the sequence diagram, consist of the automatic subscription of a new member to the contact lists of the other team members. The new member contact list will be also updated with the contacts of the projects' members already subscribed. If necessary, the IM service account for the user is created.

The main benefits drawn from this extension are basically:
- all team members of a project receive an IM account;
- all team members can directly get in contact with each other;

D.9.3.3 API IDE Support
Future Internet Core Platform

- define one contact group per project (*);
- use one single IM service avoiding different accounts and tools;
- share a chat room dedicated to the project (*);
- keep the log of all the conversation as additional knowledge base;
- include into the cross search service the conversation logs.

(*) Depending on the specific implementation of the IM service it's possible to include into the extension process the automatic set up of a chat-room shared by all the team members and a new contact group that contains all the project members.
8 Wiki Edit support

8.1 Introduction

In the context of the project it's proposed to adopt a specific plug-in, Mylyn WikiText, that adds to the Future Internet project under development the support of writing documents using the wiki paradigm. At the moment a relevant set of wiki dialects are already supported (MediaWiki, Textile, Confluence, TracWiki and TWiki), anyway, the architecture of the plug-in allows to develop additional ones and extend the list.

8.1.1 Information

Name: WikiText
Version: 1.9.0
License: EPL v1.0
Download: Eclipse Marketplace
Documentation: Mylyn WikiText User Guide
Scope: To enable the FIApp project the writing of documentation using the wiki paradigm.

8.2 Installation

This plug-in can be installed (or updated) directly from the Eclipse IDE opening the wizard at Help -> Eclipse Marketplace ... and searching for WikiText (see the image). Select and install the Mylyn WikiText, by following the steps listed in the wizard.
8.2.1 Requirements
The WikiText requires some Mylyn components that are automatically resolved and installed during the guided install procedure.

8.3 How To
The needed operative information on how to use WikiText for writing wiki contents, can be found at the Getting Started section of the User Guide.

8.4 Use Case

8.4.1 Context
The management of the Wiki content using Eclipse WikiText plug-in.

8.4.2 Scenario
Conversion of MediaWiki content into Wiki content using the Eclipse editor.

8.4.3 Description
A FI-App developer wants to publish content in a Wiki, starting from available content already published to a MediaWiki. This can be done by using the Eclipse IDE having installed WikiText plug-in. The FI-App developer by using the WikiText plug-in creates a new MediaWiki file within his/her project. After that, the developer copies the content from the MediaWiki and inserts it into the MediaWiki file just created within the project. Then, the FI-App developer converts the file from MediaWiki into a Wiki file. This way, the WikiText plug-in generates a new Wiki file and the FI-App developer can copy the content from the Wiki file thus created in the Wiki website to be published.

8.4.4 Use Case Diagram

<table>
<thead>
<tr>
<th>Use Case</th>
<th>Reuse MediaWiki content</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor</td>
<td>FI-App developer</td>
</tr>
<tr>
<td>Actions</td>
<td>• Create a new MediaWiki file</td>
</tr>
<tr>
<td></td>
<td>• Insert content into the MediaWiki file</td>
</tr>
<tr>
<td></td>
<td>• Convert file from MediaWiki into Wiki</td>
</tr>
<tr>
<td>Subject</td>
<td>Eclipse + WikiText plug-in</td>
</tr>
<tr>
<td>Post condition</td>
<td>Content converted from MediaWiki to Wiki</td>
</tr>
</tbody>
</table>
8.5 Improvements

A possible improvement to be implemented into the Mylyn WikiText plug-in is the possibility to connect to a remote wiki installation and edit those wiki pages directly from the Eclipse IDE. A relationship with the Mylyn community of developers is under development.
9 Source Code Management

9.1 Introduction

One crucial aspect to be addressed when dealing with collaborative development is the management of the source code. To help on this, it's possible to adopt a solution identified as Source Code Management (SCM) system. There are various solutions available on the market, at the right moment (e.g. Subversion, Git); the one that has been selected, as first reference implementation, is Subversion together with one of its Eclipse clients, Subclipse. This client allows developers to easily interact with the source code repository directly from within the Eclipse IDE. Apart from the basic features (check out, commit, update), also the most complex ones (manual conflict resolution, branch, tag, merge) are supported, out of the box, with graphical and intuitive user interface.

Starting from the assumption that the selected reference implementation of the forge solution is FusionForge, we consider that the Subversion server is already available (see FusionForge and Subversion installation manuals respectively).

9.1.1 Information

Name: Subclipse
Version: 1.10.x
License: EPL v1.0
Download: Eclipse Marketplace
Documentation: Official Website
Scope: Source Code Management

9.2 Installation

Before to proceed with plug-in installation it's important to obtain the version of the Subversion server that will be used. This is useful in order to select the proper version of the Subclipse client.

This plug-in can be installed (or updated) directly from the Eclipse IDE opening the wizard at Help -> Eclipse Marketplace ... and searching for Subclipse (see image). Install the Subclipse component following the guided steps.
After the installation it may be useful to take a look at the FAQ page for additional information on specific usage or constraints (e.g. svn+ssh protocol).

**Note for Linux Users:** To avoid a error related to JavaHL library that may be not available, after the installation, go to the Preferences for SVN and change the SVN Interface - Client option to SVNKit (Pure Java).

### 9.3 Architecture

The architecture diagram (see image) represents the main components involved and their interactions. The plug-in provides to the developer a single contact point to interact with the source code manager system.
9.4 How To

Here are reported the features that are mostly used during a traditional development process. Additional features and details are reported on the official documentation.

9.4.1 Connect to SVN Repository

Once the plug-in is installed, the first action to do is to configure an SVN Repository to connect to. This allows to browse the repository content and checkout projects or contents in general.

9.4.2 Synchronize the code

Update and Commit operations are well supported by the plug-in. Thanks to the Team Synchronizing perspective, and the Synchronization view, it's easy to check what are the resources that has been modified locally and remotely (by other users). It highlights the conflicts detected in order to take care of those and solve them before committing the updates.

9.4.3 Tag, Branch, Merge

All these three basic operations are fully supported by the Subclipse plug-in and it's possible to execute them with dedicated wizards. It's worth to note that the Merge operation can benefits from the "Revision graph" graphical representation in order to simplify the identification of the right revisions and ranges to merge.

9.5 Use Case

9.5.1 Context

Storage and sharing of the source code by using Eclipse Subversion plug-in.

9.5.2 Scenario

Source code management is delegated to specific tools that manage the repository, for instance svn or cvs. These software tools are used for the storage and the management of the source code, according to given policies. These management tools allow and manage, for example, code sharing between multiple users, update and versioning of the code, resolution of conflicts in the source code, management of branches, tags, or trunks.

9.5.3 Description

A FI-App developer uses his/her Eclipse IDE having installed the Subversion plug-in to interface with the svn repository. Once connected and authenticated to the repository, the developer has given permissions in order to manage content within it. The FI-App developer connects to a given code into a given directory, then synchronizes and uses that code and, possibly, updates and commits it into the repository to be shared.

9.5.4 Use Case Diagram

| Use Case | Conflict in the source code |

D.9.3.3 API IDE Support
<table>
<thead>
<tr>
<th>Actor</th>
<th>FI-App developer 1, FI-App developer 2, SVN Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actions</td>
<td></td>
</tr>
</tbody>
</table>
| FI-App developer 1  
|   - synchronize to the repository  
|   - modify the file “X”  
| FI-App developer 2  
|   - synchronize to the repository  
|   - modify the file “X”  
| FI-App developer 1  
|   - commits the changes done in file “X”  
| FI-App developer 2  
|   - try to commit the changes done in file “X”  
| SVN server  
|   - check for conflicts: return error message  
| FI-App developer 2  
|   - manages the source code to resolve conflicts |
| Subject | Eclipse 1 + Subversion plug-in, Eclipse 2 + Subversion plug-in, SVN server |
| Precondition | FI-App developer 1 and FI-App developer 2 are logged in to SVN |
| Post condition | Conflicts in the source file “X” are solved |

Conflict in the source code
10 Library and Dependency Management

10.1 Introduction

The proposed composition of tools provides a comprehensive support for the library and dependency management topic. The selected reference tools are the Maven integration for Eclipse (m2e), on the user side, and Nexus for as a repository management, server, solution. With m2e, Maven is used within Eclipse in a natural and intuitive interface. A Nexus installation at FI-WARE Instance level facilitates the adoption by the developers of Future Internet applications of the artifacts made available by that FI-WARE Instance. This document highlights the most relevant features and benefits in the context of Future Internet application development based on the FI-WARE environment. Additional and detailed documentation on m2e and Nexus are available from the official web sites.

10.1.1 Information

10.1.1.1 Maven Integration for Eclipse
Name: Maven Integration for Eclipse
Version: 1.5
License: EPL v1.0
Download: Eclipse Marketplace
Documentation: Official Website
Scope: Dependency Management plug-in

10.1.1.2 Nexus
Name: Nexus
Version: 2.7.x
License: EPL v1.0
Download: Official Website
Documentation: Official Website
Scope: Artifact Repository Management

10.1.1.3 Apache Maven
Name: Apache Maven
Version: 3.1.x
License: Apache Licence v2
Download: Official Website
Documentation: Official Website
Scope: Dependency Management

10.2 Installation

10.2.1 m2e

This plug-in can be installed directly from the Eclipse IDE opening the wizard at Help -> Eclipse Marketplace ... and searching for Maven (see image). Install the "Maven Integration for Eclipse" component following the guided steps. All the requirements are resolved automatically.
10.2.2 Nexus
To download and install the Nexus solution follow the instructions described on the official web site, the dedicated page is [Download and Install Nexus](https://example.com).

10.3 Architecture
In this part it's described how the two solutions, m2e and Nexus, works together, highlighting the development use cases aspects. The proposed solution aims to take advantage from the most relevant features of both of them.
Architecture Overview

In a complete and integrated scenario, it's feasible to have GE/SE Providers that contribute libraries for a specific FIWARE Instance (Testbed in the specific case). These assets, developed for that FIWARE Instance, are stored into that instance, thanks to the adoption of a service for library and dependency management service. The reference implementation for this service is the Nexus solution. The choice of this one among the others is due to the fact the client component, installed into the Eclipse IDE, is the m2e (Maven integration for Eclipse) solution that are developed and maintained by the same company. Both of them represent a standard de facto for this sort of tools at support of software development. The GE/SE Provider thanks to the m2e plug-in is able to upload the assets into the repository directly from the Eclipse IDE.

Once the assets are made available into the Nexus service for that FIWARE Instance, a FIApp developer that wants to build an application on top of that FIWARE Instance is able to quickly and easily obtain all the assets of interests. This operation can be done browsing the repository directly from its native web interface or from the Eclipse IDE using the m2e plug-in.

10.4 How To

The sections of this How To consider mainly the developer point of view and the interaction within the Eclipse IDE environment. For a complete and comprehensive documentation on how to use and configure this set of tools refer to the official documentations:

- Nexus: Repository Management with Nexus
- m2e: FAQ
- Maven: Maven Users Centre

10.4.1 Configure the Repository

To install and configure the Nexus repository for a FIWARE Instance it's useful to follow the indications reported in Installation manual.

10.4.2 Add a library to the project

This is the typical usage of the Apache Maven tool. It can be done manually editing the POM files or more easily, and graphically, from the user interface provided by the m2e plug-in. The screenshot here aside, shows an example of a library search results. In fact from the m2e plug-in it's possible to query the repository for the desired library, get the list of the available results, select the one to add to the current project (POM).
10.4.3 Deploy an artifact to the Repository

To deploy an artifact to a remote repository is a typical Maven operation that is well documented on the official Getting Started Guide. This operation allows developers to make their assets available to all the other developers connected to that FIWARE Instance Repository.

10.5 Use Case

10.5.1 Context

Library and dependency management using Eclipse m2e plug-in.

10.5.2 Scenario

The library and dependency management is delegated to software tools, such as maven, which solve dependencies on other external libraries, including them into the local or remote repository.

10.5.3 Description

A FI-App developer uses the Eclipse IDE to import, in the IDE, a maven project that uses libraries not included in the IDE. After having imported the project, the m2e plug-in allows to download
libraries, required by the project, to the local computer. The execution of m2e plug-in allows to automatically solve all kinds of dependencies, by simply using the .xml file in which are defined all needed libraries to compile the project.

10.5.4 Use Case Diagram

<table>
<thead>
<tr>
<th>Use Case</th>
<th>Build a project through maven</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor</td>
<td>FI-App developer, m2e plug-in</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>FI-App developer</td>
</tr>
<tr>
<td>- import and open a maven project in eclipse IDE</td>
</tr>
<tr>
<td>m2e plug-in</td>
</tr>
<tr>
<td>- check and download needed libraries</td>
</tr>
<tr>
<td>FI-App developer</td>
</tr>
<tr>
<td>- build the project through the m2e plug-in</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Subject</th>
<th>Eclipse + m2e plug-in</th>
</tr>
</thead>
<tbody>
<tr>
<td>Post condition</td>
<td>The imported project is built by automatically solving all the by means of m2e</td>
</tr>
</tbody>
</table>

Build a project using maven

10.6 Improvements

Thanks to a traditional feature of maven, the archetypes, it will be possible to define some of those, in the context of the FI-WARE project, that will be useful for starting a new project for a FIApp or for a Generic Enabler. An archetype is a template that is used to setup the structure of a new project. This template structure can cover aspects that range from the nature of the project (web app, jar library, java application, mobile app) to predefined dependencies definitions.