FI-WARE DevComE technical description front page

Large-scale Integrated Project (IP)

D9.5: FI-WARE DevComE technical description

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-D9.5a

Project Document Date: 30 May 2012

Deliverable Type and Security: Public

Author: Matteo Melideo (ENG)

Contributors: Davide Dalle Carbonare (ENG), Roozbeh Farahbod (SAP), Alessio Graziani (ENG), Jesús Gorroñogoitia Cruz (ATOS), Tim Jonischkat (UDE), Fredrik Pettersson (EAB), Osama Sammodi (UDE), John Sandberg (EAB), Pasquale Vitale (ENG), Marcel Zalmanovici (IBM-IL)

Abstract: The primary goal is to offer a multi-functional development environment enabling the development and management of the applications built to address the needs of the Future Internet and based on the adoption and integration of the FI-WARE project results.

Contents

Keyword list: FI-WARE, PPP, DevComE, use cases, sequence diagrams, behavioral scenarios, forge, collaboration, development environment, conceptual architectures, IDE tools, deployment suite

FI-WARE DevComE Technical Description

Version: **2.0**

Introduction

The goal of this document is to describe at different levels of detail what the FI-WARE **Developer Community and Tools** (DCT) will deliver. The primary goal is to offer a multi-functional development environment enabling the development and management of the applications built to address the needs\challenges of the Future Internet and based on the adoption and integration of the FI-WARE project results. In this respect, exploiting the experiences and knowledge that many developers worldwide already have with Software Development Kits (SDKs), Integrated Development Environments (IDEs) and Collaborative Development Environments (CDEs), this time these environments will not be offered in isolation but as an integrated one. The challenge, then, will be to provide a single access point to the Future Internet developers offering at the same time:

- a powerful, agile and complete development suite;
- the possibility to rely on the support of the community;
- the possibility to test, deploy and monitor the final results anywhere and anytime.

This will be the FI-WARE DevComE: a set of tools, code samples, documentation, compilers, headers and libraries the developers can use to create Future Internet applications and services based on the FI-WARE results (e.g. FI-WARE Compliant Platform Products).

Relevant Information

In this paragraph is reported the relevant information that is used or taken into account for the definition of the solutions that will be produced by the DCT. All that information is extracted from the Description of Work (DoW) or from the FI-WARE High Level Description (HDL) where it's possible to find the whole set of definitions of the most frequent terms used in the project. Here they are reported only extracts with the most significant aspects highlighted and necessary for the comprehension of the document itself.

Roles

Here they are reported the main roles to be considered for the definition of the high level scenarios (DevComE Use Cases).

- **FI-WARE Application/Service Provider**: A company or organization which develops FI applications and/or services based on FI-WARE GE APIs and deploys those applications/services on top of FI-WARE Instances. [...]
- **FI-WARE GE Provider**: Any implementer of a FI-WARE GE. The open and royalty-free nature of FI-WARE GE specifications will allow parties other than partners in the FI-WARE consortium to develop and commercialize products that are in compliance with FI-WARE GE specifications.
- **FI-WARE Instance Provider**: A company or organization which deploys and operates a FI-WARE Instance [...].

Assets

The assets that have been identified and that are relevant for the DCT are listed below.

- **FI-WARE Generic Enabler (GE)**: A functional building block of FI-WARE. Any implementation of a Generic Enabler (GE) [...] supports a concrete set of Functions and provides a concrete set of APIs and inter-operable interfaces that are in compliance with open specifications published for that GE.
  - Signature and behavior of operations linked to APIs (Application Programming Interfaces) that the GE should export. Signature may be specified in a particular language binding or through a RESTful interface.
  - In charge of GE Providers.
- **FI-WARE Compliant Platform Product**: A product which implements, totally or in part, a FI-WARE GE or composition of FI-WARE GEs in compliance with open specifications published for that GE.
  - In charge of GE Providers.
  - It contains the client component that allows the interaction with the FI-WARE Compliant Platform Product itself. This component is, programming, language dependent and may be specific for that particular FI-WARE Compliant Platform Product implementation.
- **FI-WARE Instance**: FI-WARE Instances are built by means of integrating a concrete set of FI-WARE compliant Platform Products, and typically a number of complementary products to gain differentiation on the market or to enable monetization (e.g. Specific Enablers, integration with own Operating Support Systems, Billing or Advertising systems).
  - In charge of FI-WARE Instance Provider.
- **FI-WARE Testbed**: A concrete FI-WARE Instance operated by partners of the FI-WARE project that is offered to Use Case projects within the FI-PPP Program, enabling them to test their proof-of-concept prototypes. The FI-WARE Testbed is also offered to third parties to test their Future Internet Applications although support to them is provided on best-effort basis.
  - Provided by *Experimentation Support, Integrated Testing and Validation* activity.
- **Future Internet Application**: An application that is based on APIs defined as part of GE Open Specifications. A Future Internet Application should be portable across different FI-WARE Instances that implement the GEs that Future Internet Application relies on, no matter if they are linked to different FI-WARE Instance Providers.

It is worth to note that inside the rest of the document the term *asset* will be used to avoid repeating one, or more, of the previously listed elements. This choice goes in the direction of simplifying the readability of the document itself.

Major Outcomes

The major outcomes in terms of tools and methods are listed below with a brief description. Also these items may be included in the meaning of **asset** inside the rest of this document.

- **Integrated Development Environment (IDE)**: The main tool adopted by the developers to produce their solutions. It will include a set of predefined plug-ins to extend its functionalities.
- **Catalogue**: The identified entry point where to publish and search for FI-WARE Compliant Platform Products.
- **Deployment Tools**: The tools that will be in charge of the Future Internet Application deployment into the selected FI-WARE Instance.
- **Test & Validation tools**: The tools that will be in charge of the Future Internet Application testing and validation along the application life-cycle.
- **Forge**: The integrated environment that provides the most common collaboration features (e.g. version control system, task list, document management, release management, forum). The forge tool is in charge of providing the Collaborative Development Environment1 (CDE) for the FI- WARE GE Providers and FI Application/Service Providers roles activities.

- **Methodology**: Given the complexity and the novelties associated to implementing FI Apps based on FI-WARE, DevComE should come with a set of methods which will support the developers to build such applications.

These basic set of tools will be completed case by case by complementary elements (e.g. supporting scripts, documentation, compilers and methods).

To describe the main functionalities provided by this environment we provide Use Case diagrams and related scenarios together with architectural and technological views supported by textual descriptions for each specific architectural component.

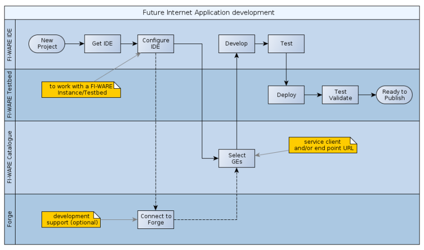
DevComE Use Cases

Introduction

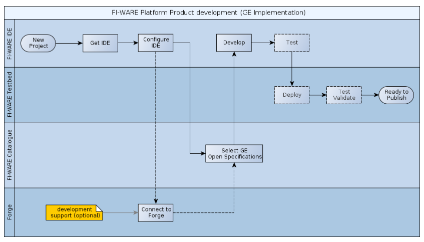
The high level scenarios that will be supported by FI-WARE DevComE are the development of FI-WARE Compliant Platform Products and of Future Internet Applications and Services. The next two figures display the main aspects of both the scenarios.

Legend:

- dashed line: optional path
- dashed boxes: not provided by the FI-WARE DevComE



Future Internet Application development



FI-WARE Compliant Platform Product development

Actors and Roles

The development cycle presented in the two high level scenarios need to follow a set of operations available to specific user roles. Given the definition in chapter Roles, below they are provided more contextualized descriptions of the roles envisaged in FI-WARE and specifically tailored to the identified DCT use cases diagrams:

- **FI-WARE GE Provider**: is in charge of managing from a technical point of view a project to implement FI-WARE Compliant Platform Products. Contributing to the project development in terms of code and documentation. Provide assets to the Catalogue and supports the community of users.
- **FI Application/Service Provider**: is in charge of managing from a technical point of view a project to implement FI Applications. Contributing to the project development in terms of code, documentation, test, deployment and

analysis.

- **FI-WARE Instance Provider**: is in charge of the management and maintenance of the FI-WARE Instance Catalogue operational aspects. Administers in terms of configuration policy, governing rules, configuration sw/hw the whole FI-WARE DevComE.

Use Cases

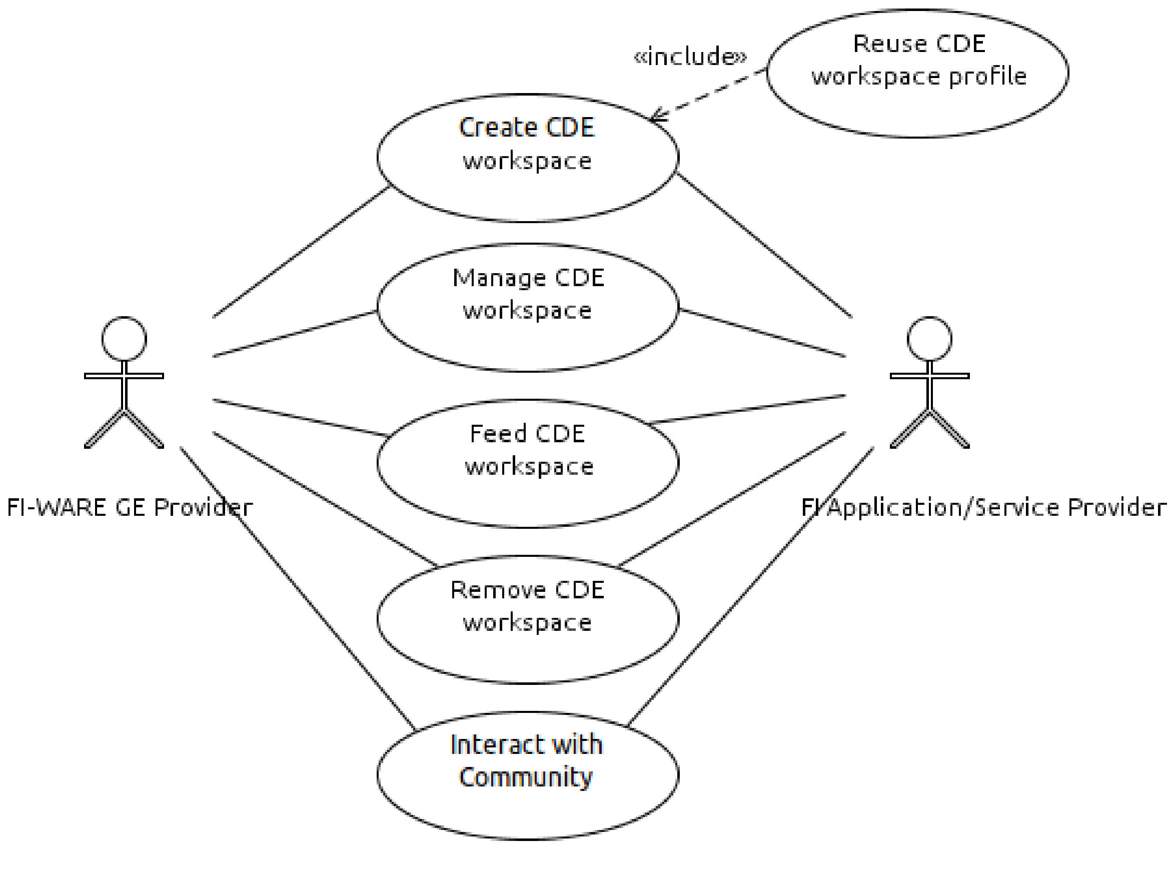
The presented high level scenarios have been elaborated in the next chapters in a more detailed view that allows identifying better specific features together with the user role they refer to. In this respect the main functionalities, provided by the FI-WARE DevComE and for a matter of readability, are grouped depending on which topic they most refer to:

- Collaboration Development Environment (CDE)
- Project (Asset)
- Catalogue.

It is worth to note that having a running FI-WARE Instance is not a prerequisite to all the defined use cases, in fact the development of a FI-WARE Compliant Platform Product may rely only on GE Open Specifications. In any case when the prerequisite is to have a running FI-WARE Instance, the use case involves the FI-WARE Instance Provider role.

Use Cases - CDE

This Use Case diagram (Figure: CDE adoption Use Cases) will be enriched by a textual description for each case in the next sections. These textual descriptions will also be used to inspire the User Stories necessary for the development of the various system functionalities.



CDE adoption Use Cases

Create a CDE workspace

Name	Create a CDE workspace
Initiator	FI-WARE GE Provider, FI Application/Service Provider
Goal	Create a workspace within the identified CDE. The CDE instance could live stand alone but it could also be associated with an existing software project (from the IDE).

Steps

1. The *initiator* starts the CDE creation process within the IDE
2. The *initiator* provides a project name
3. The IDE wizard is initiated.
4. The IDE wizard shows three different typologies:
   1. Forge: Types of available CDEs (e.g. Fusion Forge)
   2. Tools: Types of available and predefined collaborative tools (SVN, Trac, Mantis, ...)
   3. Custom: Custom environment
5. The *initiator* selects a typology
6. The *initiator* selects an instance according to the chosen typology:
   1. Forge: the list of the available forge instances is presented
   2. Tools: the list of the available tools instances is presented (by type)
   3. Custom: the drivers for the custom environment need to be provided to the IDE
7. Include "Register"
8. The *initiator* creates an empty project within the selected CDE instance
9. Include "Manage CDE"

Extension points

6.2 The *initiator* reuses existing CDE profiles (Reuse CDE Profiles);

1. The IDE wizard shows the list of available predefined CDE profiles;
2. The *initiator* selects the preferred CDE profile;
3. The *initiator* continues with step 7 and exit the wizard as the profile contains also the configuration of the CDE tools;

8. The project name already exists

1. The *initiator* is prompted for a new name.

Non-Functional requirements

List of non-functional requirements that the use case must meet. The non-functional requirements are listed in the form: <keyword> : < requirement>. Non-functional keywords include, but are not limited to Performance, Reliability, Fault Tolerance, Frequency, Usability, and Priority. Each requirement is expressed in natural language or an appropriate formalism.

Manage a CDE workspace

To complete this Use Case, input from the other project chapters are needed

Name	Manage a CDE workspace
Initiator	FI-WARE GE Provider, FI Application/Service Provider
Goal	Anyway related to added value services directly manageable from the IDE and not related to the usual configurations done directly into the CDE. For instance: 1. as soon as the project is created the initiator will be prompted by the IDE with a form to invite already registered people to join the just created project or to send request of invitations (e.g. link LinkedIn, FB, …); 2. it will be possible to graphically associate tools with roles (e.g. PERMIS/RBAC); 3. it will be possible to connect the contents, from different tools, with each other to allow a sharing of information among them (e.g. mind map to display contents relationships); 4. to enable different views for some tools providing only a minimal set of functionalities necessary for the specific function useful within the IDE; 5. save existing CDE profiles containing all the configurations;

Steps *(3rd example)*

1. open the context view starting from a task or ticket
2. browse for the resources to link
    1. both from the context view
    2. one from the context view and another one
3. set the connection
4. save the context
5. commit the context together with the task

Extension points

1. *To be detailed*

Non-Functional requirements

List of non-functional requirements that the use case must meet. The non-functional requirements are listed in the form: <keyword> : < requirement>. Non-functional keywords include, but are not limited to Performance, Reliability, Fault Tolerance, Frequency, Usability, and Priority. Each requirement is expressed in natural language or an appropriate formalism.

Feed a CDE workspace

To complete this Use Case, input from the other project chapters are needed

Name	Feed a CDE workspace
Initiator	FI-WARE GE Provider, FI Application/Service Provider
Goal	This functionality wants to offer added value services to the developers to interact with the selected CDE workspace directly from the IDE but without replacing the existing functionalities already offered by the selected collaborative tools. For instance: 1. post a message in a forum by means of a contextual menu that interact with the forum directly in the CDE (e.g. on the usage of a specific GE method just by text selection of this method in the IDE) 2. select any text in the IDE and make a cross search into the CDE workspace contents 3. from any text selection post a bug into the issue tracker using the IDE contextual menu

Steps (3rd example)

1. browse the list of task or tickets
2. select one
3. activate the context mode (Mylyn)
    1. search into the CDE for the selected text
    2. post a new task/ticket starting from the selected text
4. work on the source code and resource files

5. commit the changes into the Version Control System
6. update the task/ticket information

Extension points

1. *To be detailed*

Non-Functional requirements

List of non-functional requirements that the use case must meet. The non-functional requirements are listed in the form: <keyword> : < requirement>. Non-functional keywords include, but are not limited to Performance, Reliability, Fault Tolerance, Frequency, Usability, and Priority. Each requirement is expressed in natural language or an appropriate formalism.

Remove a CDE workspace

Name	Remove a CDE workspace
Initiator	FI-WARE GE Provider, FI Application/Service Provider
Goal	For any reason the initiator (In this case the initiator may be the simple CDE member) is not anymore assigned to this project and, as a consequence, to the management/contribution of the supporting community. As a consequence he/she could not have any more the interest in having links or references to the CDE workspace in his FI-WARE IDE.

Steps

1. The *initiator* selects the project (to whom the CDE workspace is linked to);
2. Include "Register"
3. The *initiator* requests to remove any reference/link to the CDE workspace within the project;
4. FI-WARE IDE sends a notification to the CDE;
5. CDE checks there are not pending tasks or issues assigned to the *initiator*
6. CDE notifies the "Ok to remove" to the FI-WARE IDE
7. FI-WARE IDE removes all the references (e.g. contextual menus) from the IDE to the CDE

Extension points 6. CDE notifies the “Stop to remove” to the FI-WARE IDE

1. FI-WARE IDE notifies the *initiator* about the still assigned tasks/issues
2. The *initiator* solves or assigns to other registered users this issues/tasks
3. Restart from point 3.

Non-Functional requirements

List of non-functional requirements that the use case must meet. The non-functional requirements are listed in the form: <keyword> : < requirement>. Non-functional keywords include, but are not limited to Performance, Reliability, Fault Tolerance, Frequency, Usability, and Priority. Each requirement is expressed in natural language or an appropriate formalism.

Interact with Forge Community

Name	Interact with Forge Community
Initiator	FI-WARE GE Provider
Goal	Increase usage of a particular asset, make sure developers and users keep coming back to the community to learn about new things happening, take care of feedback regarding the asset

Steps

1. The *initiator* logs into the Forge
2. The *initiator* reads the contributions
    1. Forum entries
    2. Tickets entries
    3. Discussion entries
    4. Etc. as defined
3. The *initiator* handle entries where appropriate
4. The *initiator* creates requirements to the backlog of asset development

Extension points

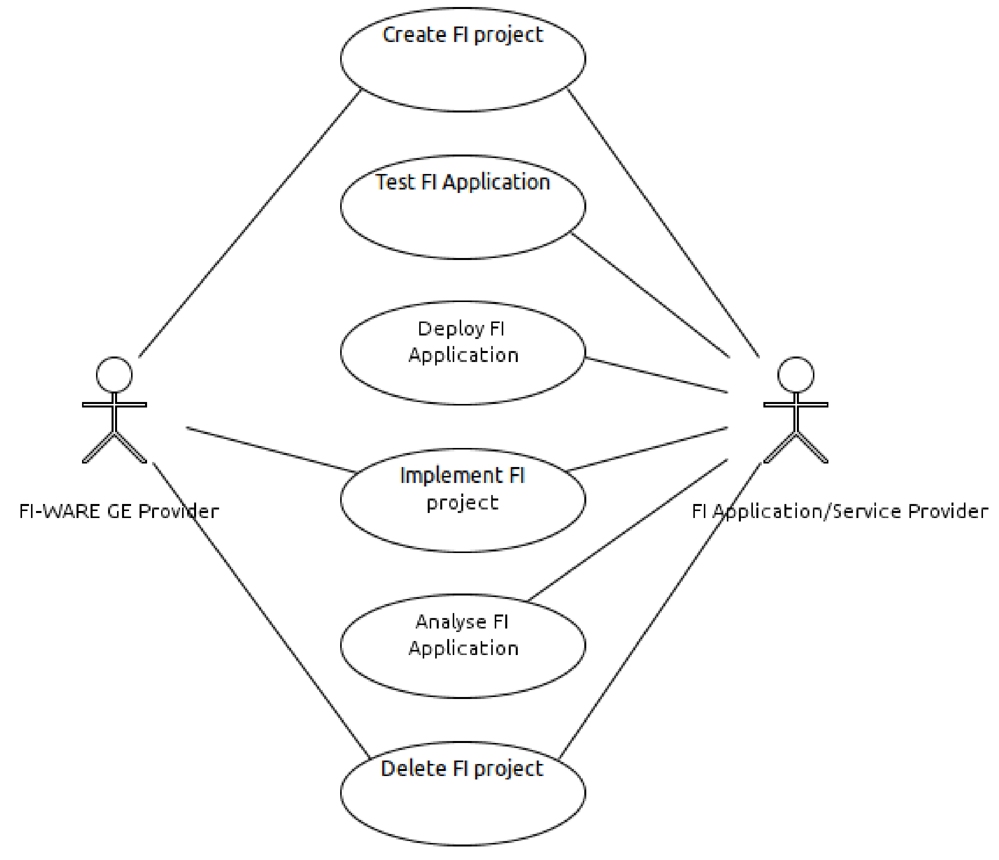
1. The *initiator* can logs into the Forge from:
    1. the web interface
    2. the IDE interface

Non-Functional requirements

List of non-functional requirements that the use case must meet. The non-functional requirements are listed in the form: <keyword> : < requirement>. Non-functional keywords include, but are not limited to Performance, Reliability, Fault Tolerance, Frequency, Usability, and Priority. Each requirement is expressed in natural language or an appropriate formalism.

Use Cases - FI Project/Application

This Use Case diagram (Figure: FI Project/Application Use Cases) will be enriched by a textual description for each case in the next sections. These textual descriptions will also be used to inspire the User Stories necessary for the development of the various system functionalities. In these Use Cases the term FI project is used represent both FI-WARE Compliant Platform Products and FI Applications.



FI Project/Application Use Cases

Create FI project

Name	Create FI project
Initiator	Project Manager
Goal	Create a new project environment in the FI-WARE IDE

Steps

1. Project Manager opens the FI-WARE IDE
2. Project Manager names a new FI project
3. FI-WARE IDE checks if the FI project names is syntactically correct
4. FI-WARE IDE checks if the FI project names is not existing
5. Project Manager creates a new FI project
6. Project Manager starts a new FI project from scratch

Extension points

3. The name is not syntactically correct

- a. FI-WARE IDE notifies the Project Manager the FI project name is not correct

- b. Project Manager provides a new name following the FI-WARE IDE syntactical indications

4. The name is already existing

- a. FI-WARE IDE notifies the Project Manager the FI project name is already existing

5. Project Manager creates a new FI project

- a. The Project Manager lunches the wizard to create the CDE
- b. ref. “Create a CDE” Use Case

6. Project Manager wants to start from existing FI-WARE artifacts

- a. ref. “Browse the catalogue” Use Case

Non-Functional requirements

tbd

Test FI Application

Perform functional test on FI Application

TO BE DETAILED FOR FUNCTIONAL TESTS

Name	Test FI Application
Initiator	Project Manager, Developer
Goal	The developer tests a FI project

Steps

1. The developer opens the FI-WARE IDE
2. The developer selects the FI project
3. The developer selects a set of tests to perform (**TO BE FURTHER DETAILED**)
4. The developer starts execution of the selected tests
5. The developer observes the test results

Extension points

None at the moment

Non-Functional requirements

tbd

Create a Performance Test Suite for a GE Specification

Name	Create a Performance Test Suite for a GE Specification
Initiator	Developer
Goal	GE specification writers should be able to develop a Performance Test Suite for any GE Specification. The performance suite defines GE-specific performance criteria and will serve as a basis for analysis and comparison of GE implementation performance results.

Steps

(The GE Specification is available and its API is accessible in SoPeCo.)

1. The user creates a SoPeCo adapter descriptor defining the configuration parameters for the measurements.
2. The user creates a configuration model including a measurement specification that defines the experiment series.
3. The user creates a SoPeCo Software Adapter for the target GE Specification that can access the GE API, perform measurements, and finally capture the target metrics.
4. The software adapter is added as a plugin to the SoPeCo framework.

5. The user will create a new Satellite Starter plugin in the SoPeCo framework which can start an RMI server and publish a corresponding satellite controller for the target GE Specification in order to provide access the software adapter through RMI.
6. The user will create a Cockpit Starter for the target GE Specification which can create an instance of SoPeCo and initiate the measurements.
7. The user will create a new SoPeCo Test Suite for the target GE Specification with the previously created SoPeCo components.
8. She then publishes the SoPeCo Test Suite in the repository for future access by GE developers.

(A SoPeCo test suite is available for future performance analysis of GE instances that adhere to the GE Specification.)

Extension points

None at the moment

Non-Functional requirements

tbd

Perform Performance Measurements on a GE Implementation

Name	Perform Performance Measurements on a GE Implementation
Initiator	Developer
Goal	Platform product developers should be able to perform performance tests on their platform products (implementation of GE Specifications) and publish the performance results along with the products.

Steps

(A GE Implementation is running and accessible and a SoPeCo Test Suite package is available for the GE Specification of the target GE Implementation.)

1. The GE developer starts the target GE Implementation, selects the corresponding SoPeCo Test Suite and binds the SoPeCo Test Suite to the running GE Implementation.
2. The GE developer consequently starts the SoPeCo Satellite Starter and Cockpit Starter for the target GE Implementation to perform the measurements.
3. The GE developer analyzes the data by feeding the previously generated measurement data into the SoPeCo visualization tool.
4. The GE developer publishes the results of her/his performance analysis on the target GE Implementation in the repository accessible by App or GE developers.

(The results of performance analysis are accessible in the repository.)

Extension points

None at the moment

Non-Functional requirements

tbd

Retrieve the latest published performance analysis data for a GE

Name	Retrieve the latest published performance analysis data for a GE
Initiator	Developer
Goal	The application developer should be able to retrieve and validate the latest performance analysis results of a given GE implementation (platform product).

Steps

(The results of performance analysis for a given GE Implementation are available in the repository.)

1. The App developer searches for a specific GE Implementation, and chooses to view the results of performance analysis on that specific instance.

Extension points

None at the moment

Non-Functional requirements

tbd

Perform Performance Measurements on a FI Application

Name	Perform Performance Measurements on a FI Application
Initiator	Developer
Goal	FI Application developers should be able to define and perform performance tests on their applications and view the performance results.

Steps

1. The user performs the Use Case Create a Performance Test Suite for a GE Specification for a FI Application instead of a GE.
2. The user performs the Use Case Perform Performance Measurements on a GE Implementation for a FI Applications instead of a GE.

Extension points

None at the moment

Non-Functional requirements

tbd

Setup of a FI Application Monitoring with Trace Analyzer

Name	Setup of a FI Application Monitoring with Trace Analyzer
Initiator	Developer
Goal	FI Application developers should be able to select tools and monitoring options for their performance tests.

Steps

(The FI Application is running in its target environment.)

1. The user opens the FI-WARE IDE, selects the specific FI Application project and chooses the environment to be set up.
2. The user opens the Trace Analyzer configuration
3. Out of the list of the available monitoring tools, the user selects a target tool and sets up the monitoring options for it
4. The user repeats step 3 for all the target monitoring tools

5. The user saves the changes. The FI-WARE IDE updates the changes in the runtime environment of the FI Application.
6. The user closes the FI-WARE IDE.

(The FI Application on the specified environment is being monitored by Trace Analyzer through the specified tools.)

Extension points

None at the moment

Non-Functional requirements

tbd

Analysis of service response times

Name	Analysis of service response times
Initiator	Developer
Goal	FI Application developers should be able to view the monitoring results and apply filtering and aggregation to reduce the amount of data.

Steps

(The FI Application on the specified environment was monitored to collect service responsiveness data (service name, request time, response parameters, optional: request parameters, response status).)

1. The Trace Analyzer parses the monitoring data.
2. The Trace Analyzer aggregates and associates related events (e.g., corresponding request and response events)
3. The Trace Analyzer presents the user with a sequence of charts showing number of concurrent requests handled and their durations. The charts show overall state as well as state by request type.
4. The user applies optional filters to visualize only requests with certain parameters or requests with duration above the user-set threshold.
5. The user goes to the statistics view to view metrics summary for the trace.
6. The user identifies additional data to be collected for more in-depth analysis (e.g., more detailed monitoring for certain service types, or system monitoring for cross-cutting problems.

(The monitoring data available for the target run is presented for the user analysis.)

Extension points

None at the moment

Non-Functional requirements

tbd

Analysis of causes for long service response times

Name	Analysis of causes for long service response times
Initiator	Developer
Goal	FI Application developers should be able to easily observe important characteristics such as response times.

Steps

(The FI Application on the specified environment was monitored to collect service responsiveness data (service name, request time, response parameters, optional: request parameters, response status) as well as additional system or application data (queue states, synchronization data, IO activity, etc.))

1. The Trace Analyzer parses the monitoring data.
2. The Trace Analyzer aggregates and associates related events (e.g., corresponding request and response events)
3. The Trace Analyzer presents the user with charts combining request duration data (optionally filtered by the type of request and its parameters) together with the additional monitoring data (visualization depending on the type of additional data; for example: threads in blocking states, cpus and their activity breakdown, etc.)
4. The user turns on the options to highlight the time intervals where the service times were beyond the user-defined threshold.
5. The user consults the Performance Anti-Patterns view identifying possible issues within the collected data (e.g., hot locks, blocking IO within locks, etc.)
6. The user identifies whether the metrics collected are related to the cause of the unsatisfactory response times.

(The monitoring data available for the target run is presented for the user analysis.)

Extension points

None at the moment

Non-Functional requirements

tbd

Deploy FI Application

Name	Deploy FI Application
Initiator	Project Manager
Goal	Deploy FI project into the testbed or target environment

Steps

1. The project manager opens the FI-WARE IDE
2. The project manager selects the FI project
3. The FI-WARE IDE checks that no open issues exist
4. The FI-WARE IDE returns no issues exist
5. The project manager deploys the project
6. deployment-time tests can be executed
   - a. The developer observes the automated selection of deployment-time tests based on the service descriptions
   - b. The developer observes the execution of deployment-time tests

Extension points

4. FI-WARE IDE returns issues

- a. The Project Manager notifies the developers about the existing issues in the issue tracker in the CDE
- b. Ref. “Test a FI Project”

Non-Functional requirements

tbd

Implement FI project

Name	Implement FI project
Initiator	FI Application/Service Provider
Goal	Implement a FI Application using the IDE

Steps

1. the initiator opens the IDE
2. the initiator creates the new project by selecting the desired project type
   1. Generic Enabler
   2. FI Application/Service
3. the initiator follows the defined methodology to develop the project

Extension points

tbd

Non-Functional requirements

tbd

Analyse FI Application

Name	Analyse FI Application
Initiator	Project Manager, Developer
Goal	Gain information about the QoS of a running FI project

Steps

1. The project manager or developer opens the FI-WARE IDE
2. The project manager or developer selects the FI project
3. The project manager or developer requests monitoring data from a testbed or target environment the project was deployed to before
4. The project manager or developer uses analyze the data with an appropriate tool, e.g. “Trace Analyzer”

Extension points

1. If the monitoring data is insufficient for some services or apps

- a. The project manager or developer requests the testbed or target environment to analyze the service invocation frequency
- b. The project manager or developer advises the testbed or target environment to perform appropriate test invocations
- c. After an appropriate waiting time, the project manager or developer continues with step 4

Non-Functional requirements

tbd

Delete FI project

Name	Delete FI project
Initiator	Project Manager and Developer
Goal	Delete a project environment in the FI-WARE IDE

This use case provides a two-fold functionality: a soft and hard deletion. A soft deletion will unregister the project from the IDE workspace list of active projects, but its content is not deleted. A hard deletion will unregister the project and delete the project content from the IDE workspace.

Steps

1. Project Manager opens the FI-WARE IDE
2. Project Manager selects the FI project
3. Project Manager deletes the FI Project.
4. FI-WARE IDE checks if the FI project is linked to or has reference with a CDE;
5. FI-WARE IDE notifies the existence of links or references to a CDE;
6. Include “Remove a CDE”;
7. FI-WARE IDE asks whether to erase project content from the IDE workspace (hard deletion) or not (soft deletion);
8. Project Manager selects soft deletion;
9. FI-WARE IDE unregisters the project from its workspace;

Extension points

5. FI-WARE IDE notifies no existing links or references to any CDE

- a. Skip point 6 above and go directly to 7.

8. Project Manager selects hard deletion

- 9. FI-WARE IDE unregisters the project from its workspace
- 10. FI-WARE IDE deletes the project content from the IDE workspace

Non-Functional requirements

tbd

Use Cases - Deployment

This section of the document defines the functionalities provided by the Deployment Tool. Functionalities are described by using Use Cases Model. Deployment Tool uses cases are organized in a hierarchical structure and high level use cases are described by a UML Use Case Diagram. For each use case a detailed textual description is available.

General Description

The main purpose of the Deployment Tool is to provide deployment functionalities to developers adopting FI-WARE development tools. In a standard development environment, developers deploy applications by using specific functionalities provided by the adopted IDE. The IDE is not able to deploy any kind of application within whatever application container. It is only able to deploy certain kind of applications (for instance *web applications* or *web service*) within well know application container (for instance *Tomcat* or *JBoss*). Another strong constraint of this approach is that the application container has to run locally and can't be located on a remote machine, and it is very difficult to think that a developer machine may have the same performance / characteristics of a customer production environment machine. At the end, this approach can be considered valid only during development and preliminary testing phases (as it's easy and fast to use), while it cannot be considered reliable during pre-release test, or to certificate some adopted software solutions to the customer target environment.

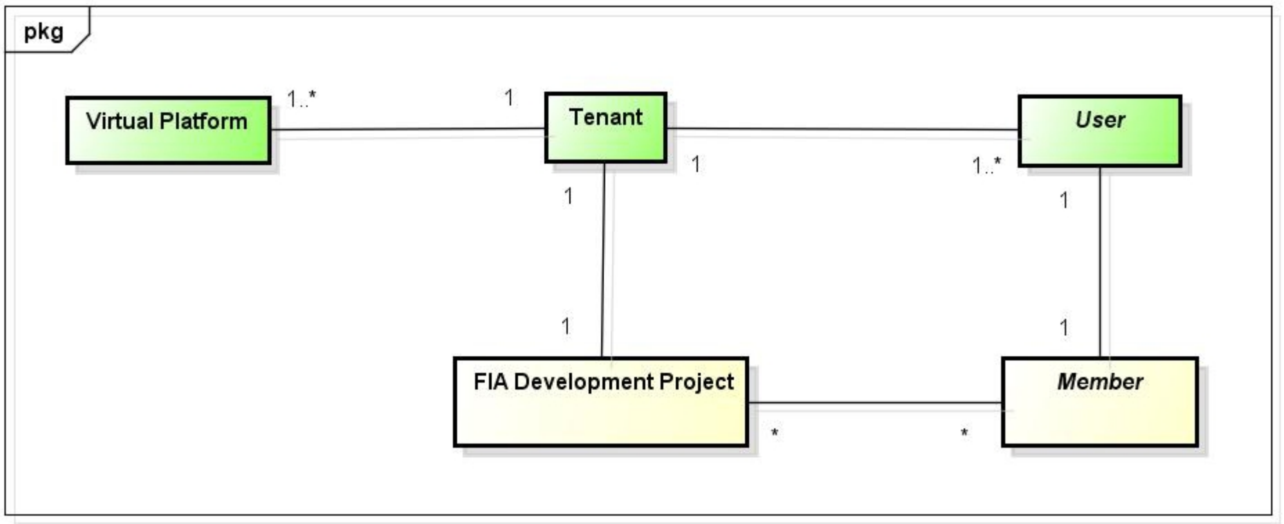
The idea behind the FI-WARE Deployment Tool, is to offer to developers a different approach to application deployment by integrating (within the IDE) traditional deployment tools (like the one described above) with the virtualization capabilities provided by the *Cloud Generic Enabler*. These capabilities may be used to define (per project) a set of Virtual Machines used to deploy and test a FI Application under different (hardware and software) conditions, providing, in this way, a complete tool, integrated in the IDE, useful for developers as well as for testing teams. Furthermore, the Deployment Tool thus conceived meets what is stated in the document *The Open Virtual Machine Format White Paper Specification V0.9* [OVF Whitepaper 0.9] regarding the concept of Virtual Appliance:

Packaging an application together with the operating system on which it is certified, into a virtual machine that can be easily transferred from an ISV, through test and development and into production as a pre-configured, pre-packaged unit with no external dependencies, is extremely attractive. Such pre-deployed, ready to run applications packaged as virtual machines (VMs) are called virtual appliances.

The Deployment Tool contains all the information (application plus Virtual Platform) needed to adopt the OVF language specifications in order to build portable Virtual Appliances.

Conceptual Entity Model

The diagram in figure *Deployment Tool Conceptual Entity Diagram* shows the main conceptual entities of the Deployment Tool and their relationships (green entities belongs to Cloud Environment, yellow entities belong to FI-WARE IDE):



Deployment Tool Conceptual Entity Diagram

Please note that in certain points this description / diagram contains also some technical indications useful to better understand the ideas behind the deployment tool, and not to define technical solutions that is out of the scope of this document.

1. **FIA Development Project**: is the most important entity, it represents a development project, in the FI-WARE IDE, of a FI Application. It has relationships with the following entities:
   1. **Member**: a member (regardless of the role) of the development team of a FI Application.
   2. **Virtual Platform**: a Virtual Machine plus all its infrastructure services, used to host the application.
2. **Tenant**: an entity of the Cloud Environment used to logically group virtual environments. It has a relationship (1 to 1) with the entity *FIA Development Project*. A *FIA Development Project* corresponds to a *Tenant* created in the Cloud Environment.

3. **User**: a user of the Cloud Environment. It has a relationship (1 to 1) with the entity *Member*. A *Member* of a development corresponds to a *User* created in the Cloud Environment.

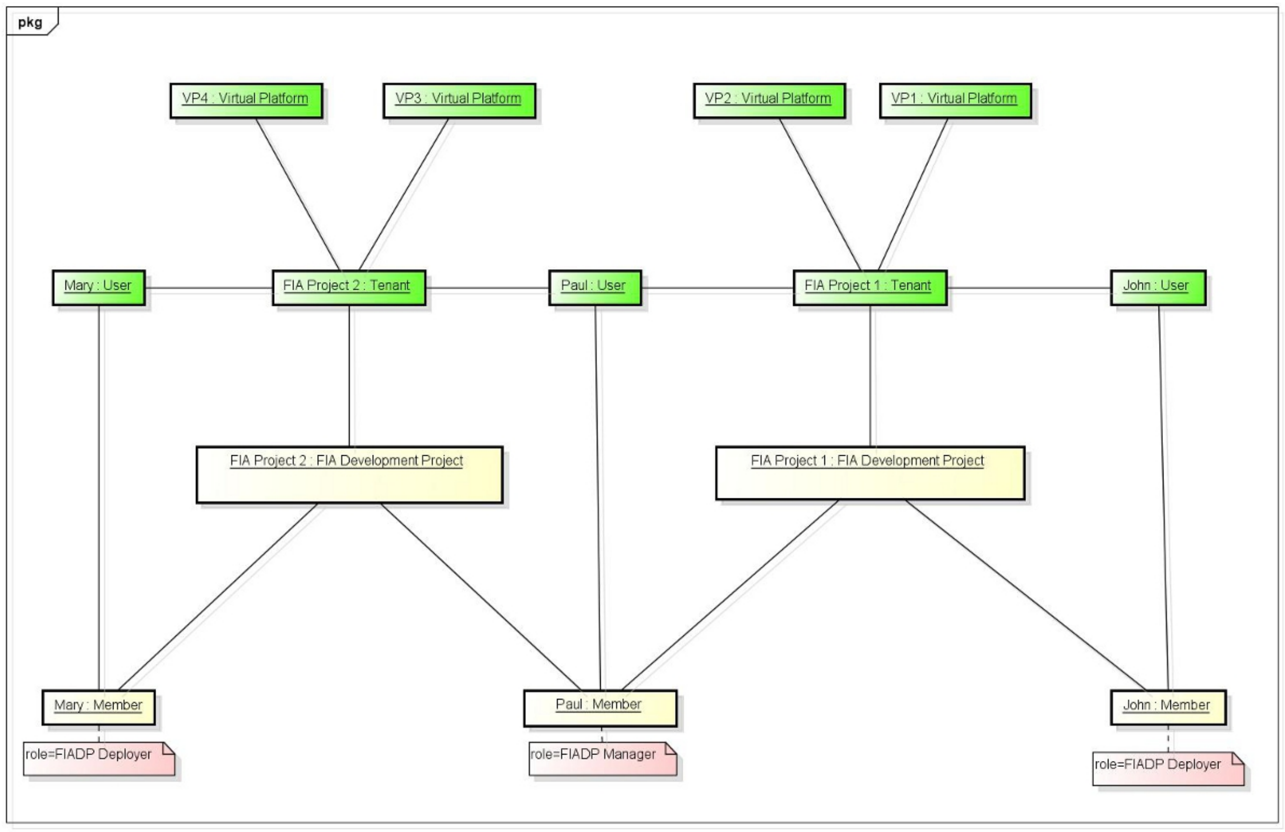
Deployment Actors and Roles

For what concern the roles defined for the **Member** entity, three roles (subclass of the Actor *FI Application / Service Provider*) have been identified:

- **Cloud Administrator**: is a member of the development team in charge of defining Tenants and Users of the Cloud environment.
- **FIA Development Project Manager (FIADP Manager)**: is a member of the development team. Usually he is the technical leader of the project. He is in charge of managing the Virtual Platforms for a specific FIA Development Project.
- **FIA Development Project Deployer (FIADP Deployer)**: is a member of the team in charge of developing a FI Application. It is very likely that a Deployer is also a developer contributing to the project. In the specific context of this document, only deployment aspects are described.

Practical Application

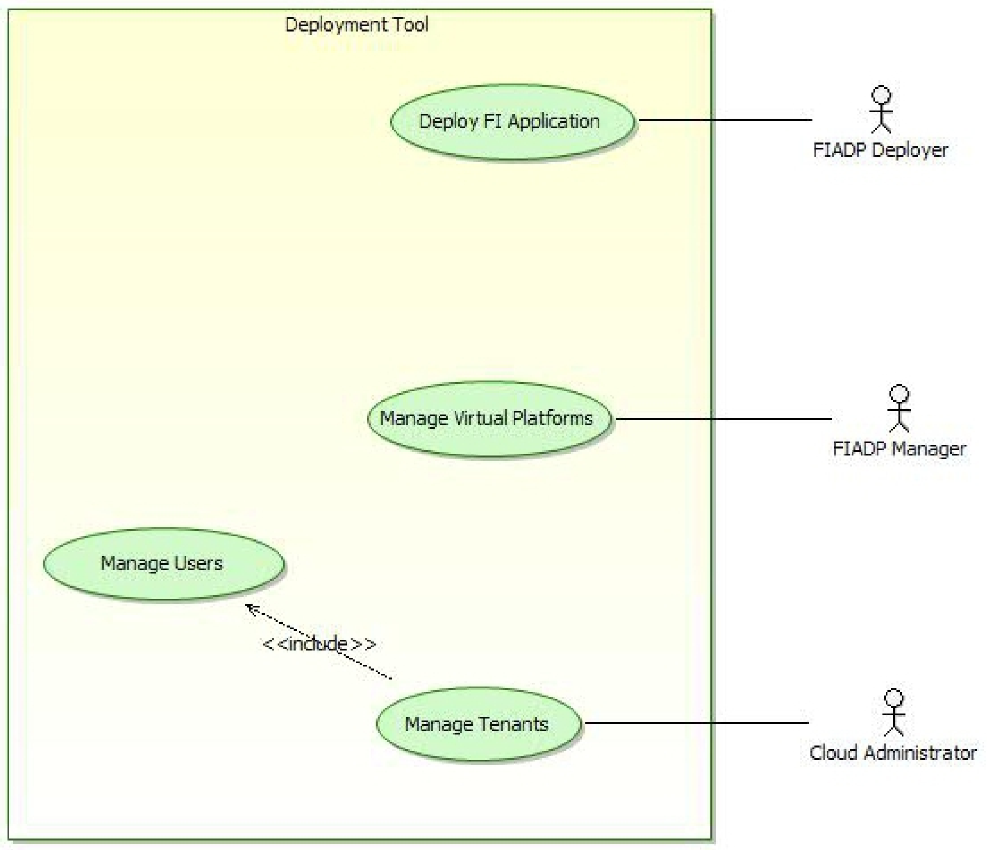
The diagram *Deployment Tool Conceptual Entities Practical Application* shows a practical application of the conceptual diagram described in the previous section. The diagram shows two development projects (*FIA Project 1* and *FIA Project 2*) with their respective development team (*Paul* and *John* for FIA Project 1 and *Paul* and *Mary* for FIA Project 2) and defined Virtual Machines:



Deployment Tool Conceptual Entities Practical Application

Deployment Tool Macro Use Cases

In this section we provide a general view of the macro use cases provided by the Deployment Tool. Each macro use case will be further decomposed into several and more detailed use cases described later in this document



Deployment Tool Main Use Case Diagram

Manage Virtual Platforms

Name	Manage Virtual Platforms
Initiator	FIA Development Project Manager
Goal	Create and manage Virtual Platforms for FIA Development Projects.
Description	This use case groups all the basic functionalities that an administrator needs to properly manage Virtual Platforms of a FIA Development Project: 1. Instantiate Virtual Platform 2. Modify Virtual Platform 3. Remove Virtual Platform 4. View Virtual Platform Section Manage Virtual Platforms Use Cases provides detailed description of the use cases concerning the management of Virtual Platforms.

Manage Tenants

Name	Manage Tenants
Initiator	Cloud Administrator
Goal	Manage Tenants defined in a Virtual Environment. Each Tenant of a Virtual Environment represents a specific FIA Development Project.
Description	This use case groups all the basic functionalities that an administrator needs to properly manage Tenants: 1. Define a Tenant 2. Modify a Tenant 3. Remove a Tenant 4. View a Tenant The Deployment Tool employees the Tenant mechanism, tipical of cloud computing, to create dedicated environments for specific FIA Development Project. Section Manage Tenants Use Cases provides detailed description of the use cases concerning the management of Tenants.

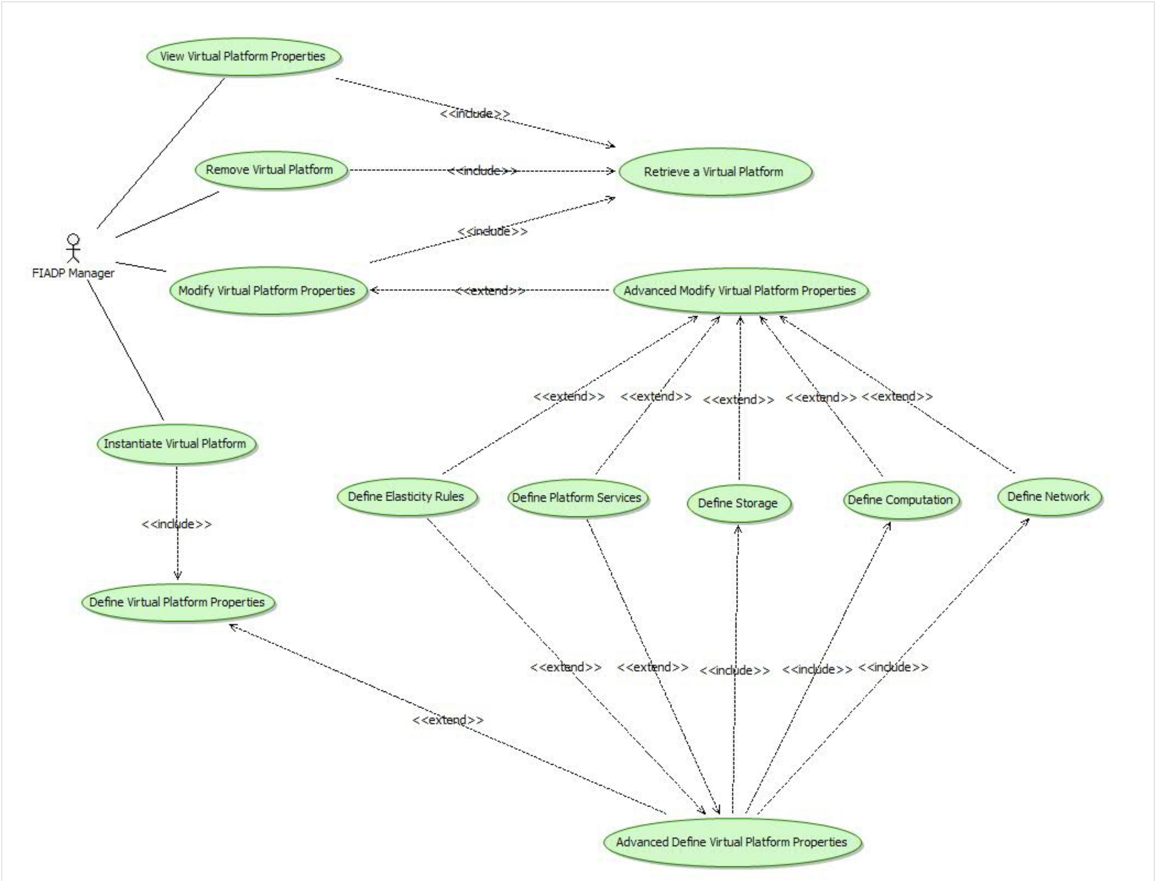
Manage Users

Name	Manage Users
Initiator	Cloud Administrator
Goal	Defining users, profiling them, and assigning them to a Tenant. Each user of a Tenant is a member participating to the development of FIA Development Project.
Description	This use case groups all the basic functionalities that an administrator needs to properly manage Users: 1. Define a User 2. Modify a User 3. Remove a User 4. View a User Section Manage Users Use Cases provides detailed description of the use cases concerning the management of Users.

Deploy Future Internet Application

Name	Deploy Future Internet Application
Initiator	FIA Development Project Deployer
Goal	Deploy on a specific Virtual Platform a FI Application developed within the FI-WARE IDE.
Description	This use case groups all the basic functionalities that a deployer / developer needs to properly deploy a Future Internet Application over one of its associated Virtual Platforms. Section Deploy FI Application Use Cases provides detailed description of the use cases concerning the deployment of a FI Applicayion.

Manage Virtual Platforms Use Cases


View Virtual Platform Properties
Remove Virtual Platform
Retrieve a Virtual Platform
FIADP Manager
Modify Virtual Platform Properties
Advanced Modify Virtual Platform Properties
Instantiate Virtual Platform
Define Elasticity Rules
Define Platform Services
Define Storage
Define Computation
Define Network
Define Virtual Platform Properties
Advanced Define Virtual Platform Properties
<<include>>
<<extend>>

Manage Virtual Platforms Use Cases

Instantiate Virtual Platform

Name	Define Virtual Platform
Initiator	FIA Development Project Manager
Goal	Instantiate a Virtual Platform.
Description	This is the use case that the FIA Development Project Manager uses to create a new Virtual Platform for a specific Tenant.

Steps

1. The *initiator* starts the process of instantiation of a new Virtual Machine within the IDE
2. The system shows to *initiator* the list of the available Tenants.
3. The *initiator* select the desired Tenant.
4. The *initiator* performs the use case Define Virtual Platform Properties to define all the properties of the Virtual Platform.
5. The *initiator* request to the system the instantiation of a Virtual Platform.
6. The system instantiate the new Virtual Machine.
7. The system shows to *initiator* a successfull message.

Extension points

4 The *initiator* performs the use case Advanced Define Virtual Platform Properties to define all the properties of the Virtual Platform.

1. The *initiator* request to the system the instantiation of a Virtual Platform.

2. The system provides adequate feedback about the requested operation.

7 The system shows to *initiator* a failure message.

Define Virtual Platform Properties

Name	Define Virtual Platform Properties
Initiator	FIA Development Project Manager
Goal	Define, by using a Wizard, all the fundamental properties of a Virtual Platform.
Description	This functionality allows a FIA Development Project Manager to define all the characteristics of a Virtual Platform avoiding, at the same, the complexities of such operation. The FIA Development Project Manager is guided by a Wizard to identify the fundamental requirements of the Virtual Platform.

Steps

1. The *initiator* starts, within the IDE, the wizard for the definition of the properties of a Virtual Machine.
2. The system shows to *initiator* the list of possible Operating Systems.
3. The *initiator* select the desired Operating System.
4. The system shows to *initiator* the list of predefined configurations.
5. The *initiator* select the configuration that better fits his needs.
6. The system shows to the *initiator* the overall features of the selected configuration.
7. The system ask the *initiator* to accept the proposed configuration.
8. The *initiator* accepts the proposed configuration.

Extension points

8 The *initiator* does not accept the proposed configuration.

1. The system restarts the Wizard from step number 2.

Advanced Define Virtual Platform Properties

Name	Advanced Define Virtual Platform Properties
Initiator	FIA Development Project Manager
Goal	Define all the fundamental properties of a Virtual Platform.
Description	This functionality allows a FIA Development Project Manager to define all the characteristics of a Virtual Platform. The FIA Development Project Manager is explicitly requested by the system to manually provide a value for each characteristic.

Steps

1. The *initiator* starts, within the IDE, the process for the advanced definition of the properties of a Virtual Machine.
2. The *initiator* performs use case Define Storage to sets the storage features values.
3. The *initiator* performs use case Define Computation to sets the computational features values.
4. The *initiator* performs use case Define Network to sets the network features values.
5. The system shows to the *initiator* the overall features of the selected configuration.
6. The *initiator* confirms the operation.

Extension points

8 The *initiator* does not accept the proposed configuration.

1. The system restarts the from step number 2.

5 The *initiator* performs use case Define Elasticity Rules to sets the elasticity rule set fot the Virtual Platform.

1. The system shows to the *initiator* the overall features of the selected configuration.
2. The *initiator* confirms the operation.

5 The *initiator* performs use case Define Platform Services to sets the additional services to install on the Virtual Platform.

1. The system shows to the *initiator* the overall features of the selected configuration.
2. The *initiator* confirms the operation.

Define Storage

Name	Define Storage
Initiator	FIA Development Project Manager
Goal	Sets all the storage values for a Virtual Platform.
Description	This functionality allows a FIA Development Project Manager to sets all the features concerning the storage aspects of a Virtual Platform.

Steps

1. The *initiator* starts, within the IDE, the process for the definition of the storage of a Virtual Machine.
2. The system shows to the *initiator* the list of possible storage values.
3. The *initiator* selectes the one that best fits his needs.
4. The *initiator* confirms the operation.

Define Computation

Name	Define Computation
Initiator	FIA Development Project Manager
Goal	Sets all the computational values for a Virtual Platform.
Description	This functionality allows a FIA Development Project Manager to sets all the features concerning the computational aspects of a Virtual Platform.

Steps

1. The *initiator* starts, within the IDE, the process for the definition of the computation of a Virtual Machine.
2. The system shows to the *initiator* the list of possible computation values.
3. The *initiator* selectes the one that best fits his needs.
4. The *initiator* confirms the operation.

Define Network

Name	Define Network
Initiator	FIA Development Project Manager
Goal	Sets all the network values for a Virtual Platform.
Description	This functionality allows a FIA Development Project Manager to sets all the features concerning the network aspects of a Virtual Platform.

Steps

1. The *initiator* starts, within the IDE, the process for the definition of the network properties of a Virtual Machine.
2. The system shows to the *initiator* the list of possible network configurations.
3. The *initiator* selectes the one that best fits his needs.
4. The *initiator* confirms the operation.

Define Elasticity Rules

Name	Define Elasticity Rules
Initiator	FIA Development Project Manager
Goal	Sets the elasticity ruleset for a Virtual Platform.
Description	This functionality allows a FIA Development Project Manager to define the elasticity rules for the evolution of a specific Virtual Platform. Elasticity Rules allows a Virtual Machine to scale up / scale down in order to maintain the requested level of service.

Define Platform Services

Name	Define Platform Services
Initiator	FIA Development Project Manager
Goal	Define the platform services of a Virtual Platform.
Description	This functionality allows a FIA Development Project Manager to define platform services (HTTP Server, RDBMS, etc. etc) needed to run a FIA Development Project in the context of a Virtual Platform.

Steps

1. The *initiator* starts, within the IDE, the process for the definition of the platform services of a Virtual Machine.
2. The system shows to *initiator* the list of the Platform services compatible with the Virtual Machine.
3. The *initiator* select the desired platform services.
4. The *initiator* asks to the system to save the selection.
5. The system shows to the *initiator* the overall selected platform services.
6. The system asks the *initiator* to confirm the selection.
7. The *initiator* confirms.

Extension points

8 The *initiator* does not confirm the proposed configuration.

1. The system restarts from step number 3.

Modify Virtual Platform Properties

Name	Modify Virtual Platform Properties
Initiator	FIA Development Project Manager
Goal	Modify, by using a Wizard, all the fundamental properties of a Virtual Platform.
Description	This functionality allows a FIA Development Project Manager to modify all the features of an existing Virtual Platform avoiding, at the same, the complexities of such operation. The FIA Development Project Manager is guided by a Wizard to modify the fundamental properties of the Virtual Platform.

Steps

1. The *initiator* starts, within the IDE, the process for the modification of the properties of a Virtual Machine.
2. The *initiator* performs the use case Retrieve A Virtual Platform to select the Virtual Machine to be modified.
3. The system shows to *initiator* the list of predefined configurations compatible with the actual state of the Virtual Machine.
4. The *initiator* select the configuration that better fits his needs.
5. The system shows to the *initiator* the overall features of the selected configuration, stressing the modified features.
6. The system ask the *initiator* to accept the proposed configuration.

7. The *initiator* accepts the proposed configuration.

Extension points

8 The *initiator* does not accept the proposed configuration.

1. The system restarts the Wizard from step number 3.

Advanced Modify Virtual Platform Properties

Name	Advanced Modify Virtual Platform Properties
Initiator	FIA Development Project Manager
Goal	Modify all the fundamental properties of a Virtual Platform.
Description	This functionality allows a FIA Development Project Manager to modify all the features of an already defined Virtual Platform. The FIA Development Project Manager is explicitly requested by the system to manually provide all the features values.

Steps

1. The *initiator* starts, within the IDE, the process for the advanced definition of the properties of a Virtual Machine.
2. The *initiator* performs the use case Retrieve A Virtual Platform to select the Virtual Machine to be modified.
3. The system shows to The *initiator* the list of the available operations he can perfom:
   1. use case Define Storage to sets the storage features values.
   2. use case Define Computation to sets the computational features values.
   3. use case Define Network to sets the network features values.
   4. use case Define Elasticity Rules to sets the elasticity rule set fot the Virtual Platform.
   5. use case Define Platform Services to sets the additional services to install on the Virtual
4. The *initiator* performs the desired use case.
5. The system asks to the *initiator* if he confirms the requested modification.
6. The *initiator* confirms the operation.
7. The system executes the operation and shows operation feedback to the *initiator*.

Extension points

6 The *initiator* does not confirm and request additional modification.

1. The system restarts from step number 2.

Remove Virtual Platform

Name	Remove Virtual Platform Properties
Initiator	FIA Development Project Manager
Goal	Remove a previously defined Virtual Platform.
Description	This functionality allows a FIA Development Project Manager to remove an existent Virtual Platform.

Steps

1. The *initiator* starts, within the IDE, the deletion of a Virtual Machine.
2. The *initiator* performs the use case Retrieve A Virtual Platform to select the Virtual Machine to be removed.
3. The *initiator* requests to the system the deletion of the selected virtual Platform.
4. The system asks a confirmation to the *initiator*.
5. The *initiator* confirms the operation.
6. The system executes the operation and shows a feedback message to the *initiator*.

Extension points

5 The *initiator* does not confirm the removal.

1. The *initiator* performs again the use case Retrieve A Virtual Platform to select the Virtual Machine to be removed.

View Virtual Platform Properties

Name	View Virtual Platform Properties
Initiator	FIA Development Project Manager
Goal	Shows to the user all the properties of a specific Virtual Platform.
Description	This functionality allows a FIA Development Project Manager to view the state of an existent Virtual Platform.

Steps

1. The *initiator* starts, within the IDE, the view of a Virtual Machine.
2. The *initiator* performs the use case Retrieve A Virtual Platform to select the Virtual Platform to be shown.
3. The system shows to the *initiator* the state (the set of attributes that defines a Virtual Platform) of the requested Virtual Platform.

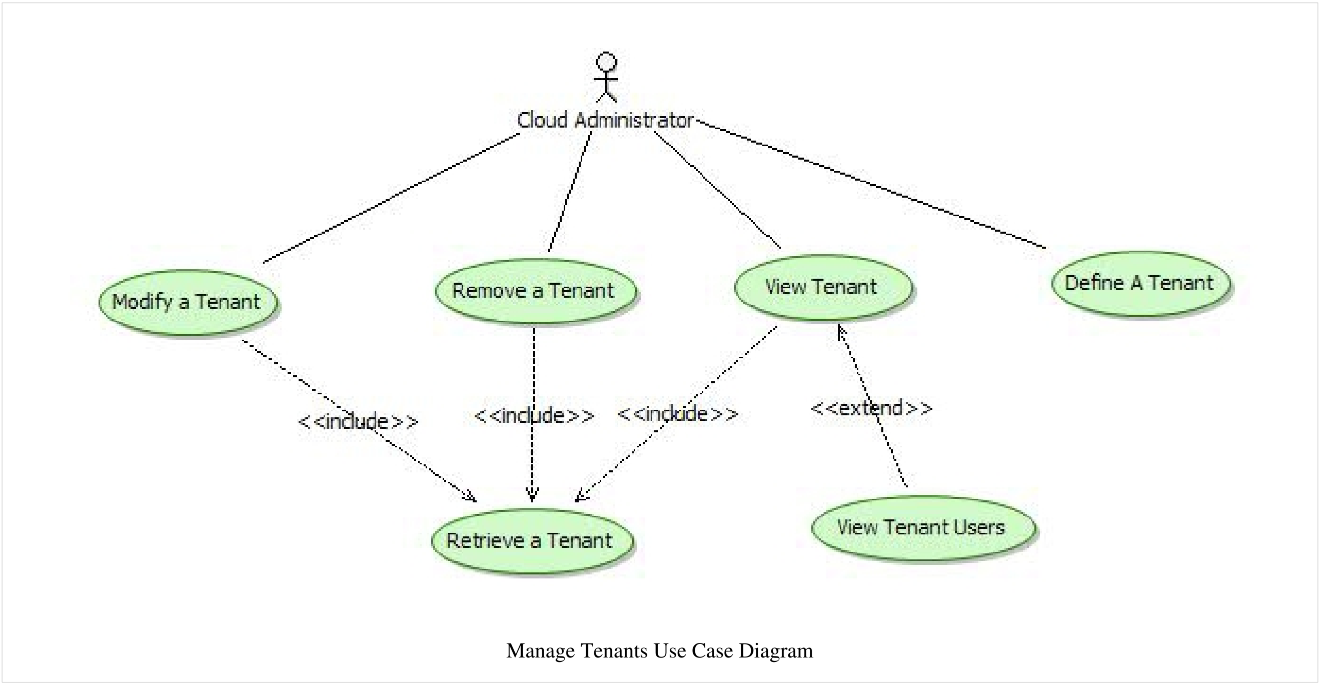
Retrieve a Virtual Platform

Name	Retrieve a Virtual Platform
Initiator	FIA Development Project Manager, FIA Development Project Developer
Goal	Retrieve a specific Virtual Platform.
Description	This functionality allows a FIA Development Project Manager to search and retrieve an existent Virtual Platform.

Steps

1. The *initiator* starts, within the IDE, the retrieval of a Virtual Machine.
2. The system shows to the *initiator* the list of the Tenants the User belongs to.
3. The *initiator* selects one or more Tenants.
4. The *initiator* requests to the system to search for Virtual Platforms defined within the context of the selected Tenants.
5. The system performs the search.
6. The system shows to the *initiator* the list of the Virtual Platforms found.
7. The *initiator* selects the desired Virtual Platform.

Manage Tenants Use Cases



Manage Tenants Use Case Diagram

Define a Tenant

Name	Define a Tenant
Initiator	Cloud Administrator
Goal	Define a new Tenant of the Cloud Environment.
Description	This functionality allows a Cloud Administrator to define a new Tenant of the Cloud Environment.

Steps

1. The *initiator* starts, within the IDE, the process for the creation of a new Tenant.
2. The *initiator* provides to the system the name of the new Tenant.
3. The *initiator* asks to the system to execute the operation.
4. The system asks the *initiator* to confirm the operation.
5. The *initiator* confirms.

Extension points

8 The *initiator* does not confirm the operation.

1. The system restarts from step number 2.

Modify a Tenant

Name	Modify a Tenant
Initiator	Cloud Administrator
Goal	Modify an existing Tenant of the Cloud Environment.
Description	This functionality allows a Cloud Administrator to modify an existing Tenant of the Cloud Environment.

Steps

1. The *initiator* starts, within the IDE, the process for the modification of an existing Tenant.
2. The *initiator* performs the use case Retrieve a Tenant to identify and select the Tenant to be modified.
3. The *initiator* provides to the system the modified Tenant name.
4. The system asks the *initiator* to confirm the operation.
5. The *initiator* confirms.

Extension points

5 The *initiator* does not confirm the operation.

1. The system restarts from step number 2.

Retrieve a Tenant

Name	Retrieve a Tenant
Initiator	Cloud Administrator
Goal	Retriev an existing Tenant of the Cloud Environment.
Description	This functionality allows a Cloud Administrator to search and retrieve an existing Tenant of the Cloud Environment.

Steps

1. The *initiator* starts, within the IDE, the process of research of a Tenant.
2. The system shows to the *initiator* the list of all the defined Tenants.
3. The *initiator* select the desired Tenant.

Extension points

3 The *initiator* select the desired Tenant.

1. The *initiator* provides to the system a filtering criteria based on the attribute *name* of the Tenant entity.
2. The *initiator* asks to the system to execute the filtering operation.
3. The system performs the filtering operation using the provided filter criteria.
4. The system shows the filtered results to the *initiator*.
5. The *initiator* select the desired Tenant.

Remove a Tenant

Name	Remove a Tenant
Initiator	Cloud Administrator
Goal	Remove an existing Tenant of the Cloud Environment.
Description	This functionality allows a Cloud Administrator to remove an existing Tenant of the Cloud Environment.

Steps

1. The *initiator* starts, within the IDE, the process for the deletion of an existing Tenant.
2. The *initiator* performs the use case Retrieve a Tenant to identify and select the Tenant to be deleted.
3. The system asks the *initiator* to confirm the operation.
4. The *initiator* confirms.
5. The system deletes the Tenant and informs the *initiator* about the successfull operation.

Extension points

4 The *initiator* does not confirm the operation.

1. The system restarts from step number 2.

5 The system cannot delete the Tenant because it contains active users.

1. The system informs the *initiator* with an adequate message.
2. The system restarts from step number 2.

View a Tenant

Name	View a Tenant
Initiator	Cloud Administrator
Goal	View all the information of an existing Tenant.
Description	This functionality allows a Cloud Administrator to view all the information regarding an exisiting Tenant of the Cloud Environment.

Steps

1. The *initiator* starts, within the IDE, the process for the viewing the state of an existing Tenant.
2. The *initiator* performs the use case Retrieve a Tenant to identify and select the Tenant to be shown.
3. The system shows to the *initiator* the basic data of the selected Tenant.

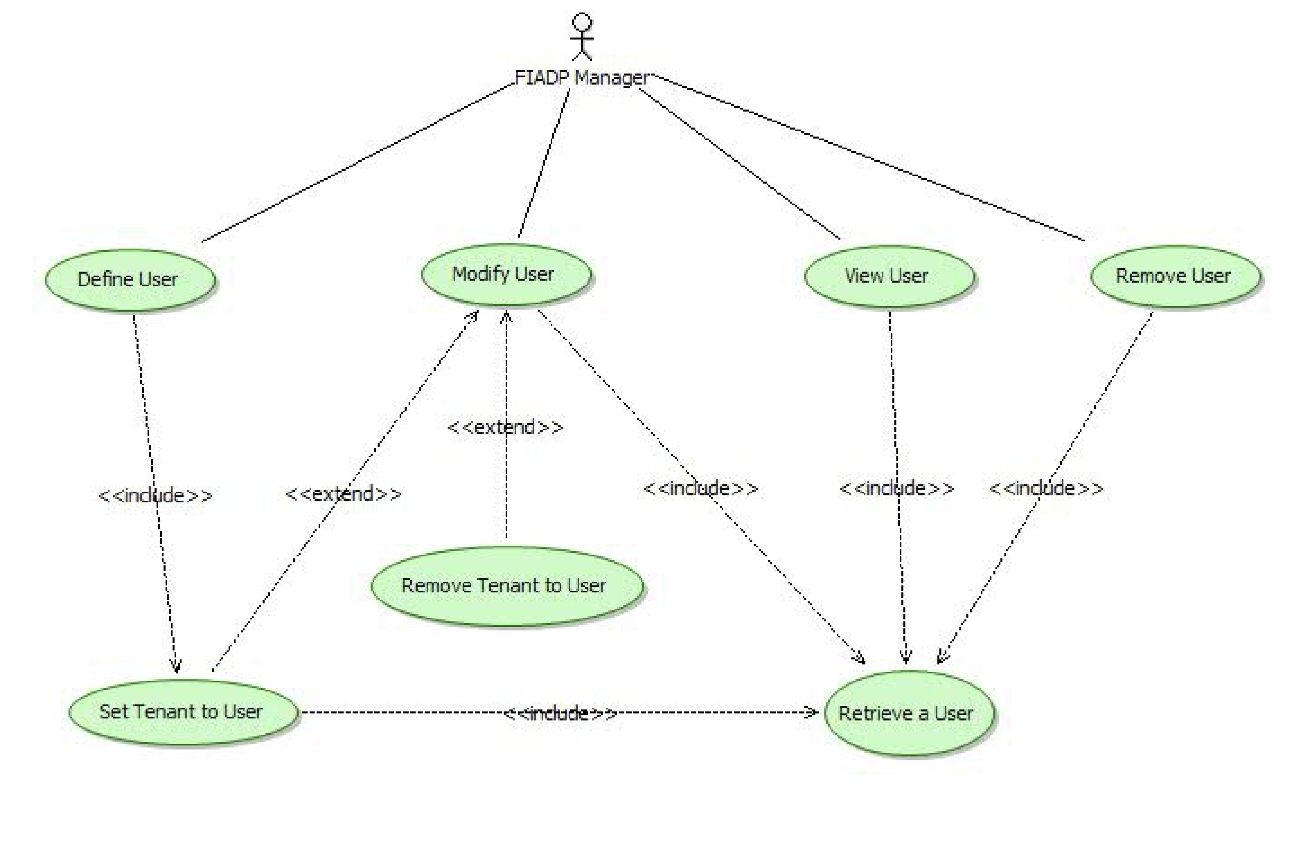
View Tenant Users

Name	View Tenant Users
Initiator	Cloud Administrator
Goal	View all Users of an exisiting Tenant.
Description	This functionality allows a Cloud Administrator to view all the Users whose Tenant is the one he selected.

Steps

1. The *initiator* performs the use case View a Tenant to see the basic data of a specific Tenant.
2. The *initiator* asks to the system to view the users belonging to the selected Tenant.
3. The system shows to the *initiator* the list of the users whose Tenant is the selected one.

Manage Users Use Cases



Manage Users Use Case Diagram

Define User

Name	Define User
Initiator	FIA Development Project Manager
Goal	Define a new User of a Tenant.
Description	This functionality allows a FIA Development Project Manager to define a new User of a Tenant.

Steps

1. The *initiator* starts, within the IDE, the process for the creation of a new User.
2. The *initiator* provides to the system the name of the new User.
3. The *initiator* provides to the system the mail of the new User.
4. The *initiator* performs the use case Set Tenant To User to assign the new User to a specific Tenant.
5. The system asks the *initiator* to confirm the operation.
6. The *initiator* confirms.
7. The system creates a temporary password to the User.
8. The system sends an email to the User containing all the necessary data for accessing the system.

Extension points

6 The *initiator* does not confirm the operation.

1. The system restarts from step number 2.

Set Tenant To User

Name	Set Tenant To User
Initiator	FIA Development Project Manager
Goal	Set a Tenant to a User.
Description	This functionality allows a FIA Development Project Manager to sets a Tenant to a User.

Steps

1. The *initiator* starts, within the IDE, the process for the setting a Tenant to a User.
2. The *initiator* performs the use case Retrieve A User to select the user.
3. The system shows the list of existing Tenants.
4. The *initiator* select the requested Tenant.
5. The *initiator* requests to the system to perform the assignment operation.
6. The system asks to the *initiator* to confirm the operation.
7. The *initiator* confirms the operation.
8. The system performs the assignment operation.

Extension points

7 The *initiator* does not confirm the operation.

1. The system restarts from step number 3.

Modify User

Name	Modify User
Initiator	FIA Development Project Manager
Goal	Modify an existing User of a Tenant.
Description	This functionality allows a FIA Development Project Manager to modify the state (attributes set) of an existing User.

Steps

1. The *initiator* starts, within the IDE, the process for modifying the state (attributes set) of an existing User.
2. The *initiator* performs the use case Retrieve A User to select the User.
3. The system shows the state of the User.
4. The *initiator* modifies the state of the User.
5. The *initiator* requests the system to save the new state.
6. The system asks to the *initiator* to confirm the operation.
7. The *initiator* confirms the operation.
8. The system performs the save operation.

Extension points

7 The *initiator* does not confirm the operation.

1. The system restarts from step number 3.

4 The *initiator* modifies the state of the User.

1. The *initiator* performs the use case Set Tenant To User to set a Tenant to the User.

4 The *initiator* modifies the state of the User.

1. The *initiator* performs the use case Remove Tenant To User to remove a Tenant to the User.

Retrieve a User

Name	Retrieve a User
Initiator	FIA Development Project Manager
Goal	Retrieve an existing User.
Description	This functionality allows a FIA Development Project Manager to search and retrieve an existing User.

Steps

1. The *initiator* starts, within the IDE, the process for of search of a User.
2. The *initiator* provides to the system the username of the new User to be searched or even just a part of the username of the User.
3. The *initiator* asks to the system to perform a search.
4. The system shows to the User the search results.
5. The *initiator* selects the requested User.

Extension points

5 The *initiator* selects the requested User.

1. The *initiator* performs another searche and restart from step number 2.

Remove Tenant To User

Name	Remove Tenant To User
Initiator	FIA Development Project Manager
Goal	Remove a Tenant to a User.
Description	This functionality allows a FIA Development Project Manager to remove a Tenant to a User.

Steps

1. The *initiator* starts, within the IDE, the process for the setting a Tenant to a User.
2. The *initiator* performs the use case Retrieve A User to select the user.
3. The system shows the list of existing Tenants.
4. The *initiator* select Tenant to be removed to the User.
5. The *initiator* requests to the system to perform the remove operation.
6. The system asks to the *initiator* to confirm the operation.
7. The *initiator* confirms the operation.
8. The system performs the remove operation.

Extension points

6 The system asks to the *initiator* to confirm the operation.

1. The system informs the *initiator* that the Tenant cannot be removed because it is the only one assigned to the User.

7 The *initiator* does not confirm the operation.

1. The system restarts from step number 3.

Remove User

Name	Remove User
Initiator	FIA Development Project Manager
Goal	Delete an existing User.
Description	This functionality allows a FIA Development Project Manager to delete an existing User.

Steps

1. The *initiator* starts, within the IDE, the process for removing a User.
2. The *initiator* performs the use case Retrieve A User to select the user.
3. The *initiator* requests to the system to perform the remove operation.
4. The system asks to the *initiator* to confirm the operation.
5. The *initiator* confirms the operation.
6. The system performs the remove operation.

Extension points

5 The *initiator* does not confirm the operation.

1. The system restarts from step number 3.

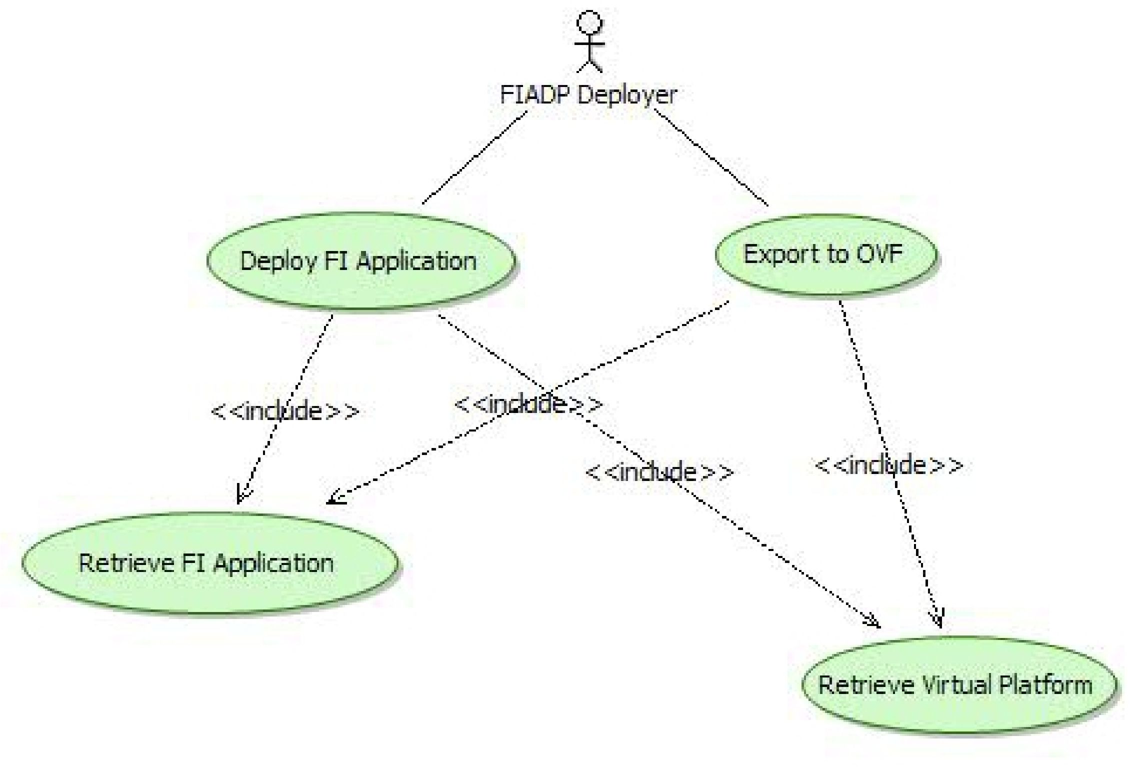
View User

Name	View User
Initiator	FIA Development Project Manager
Goal	View all the information of a specific User.
Description	This functionality allows a FIA Development Project Manager to access all teh information regarding a specific User.

Steps

1. The *initiator* starts, within the IDE, the process for viewing the state of a User.
2. The *initiator* performs the use case Retrieve A User to select the user.
3. The system shows to the User the complete state of a User.

Deploy FIA Development Project Use Cases



Deploy FIA Development Project Use Case Diagram

Deploy FIA Development Project

Name	Deploy FIA Development Project
Initiator	FIA Development Project Deployer
Goal	Deploy a specific FIA Development Project on a Virtual Platform.
Description	This functionality allows a FIA Development Project Deployer to deploy a FIA Development Project on one of the Virtual Platform ad hoc defined for this FIA Development Project.

Steps

1. The *initiator* starts, within the IDE, the process for deploying a FIA Development Project.
2. The *initiator* performs the use case Retrieve FIA Development Project to select the FIA Development Project to be deployed.
3. The *initiator* performs the use case Retrieve A Virtual Platform to select the Virtual Platform where the FIA Development Project need to be deployed.
4. The *initiator* selects the *ad-hoc* created script-file containing specific deployment instructions.
5. The *initiator* requests to the system to deploy the selected FIA Development Project in the selected Virtual Platform.
6. The system asks a confirmation to the *initiator*.
7. The *initiator* confirms the operation.
8. The system performs the deployment operation and provides to the *initiator*' a feedback message.

Extension points

5 The *initiator* does not confirm the operation.

1. The system restarts from step number 2.

Export to OVF

Name	Export to OVF
Initiator	FIA Development Project Deployer
Goal	Create an OVF package.
Description	This functionality allows a FIA Development Project Deployer to create an OVF package in order to deploy the FIA Development Project as a Virtual Appliance.

Steps

1. The *initiator* starts, within the IDE, the process for creating an OVF package.
2. The *initiator* performs the use case Retrieve FIA Development Project to select the FIA Development Project to be deployed.
3. The *initiator* performs the use case Retrieve A Virtual Platform to select the Virtual Platform.
4. The *initiator* requests to the system to deploy the selected FIA Development Project in the selected Virtual Platform.
5. The system asks a confirmation to the *initiator*.
6. The *initiator* confirms the operation.
7. The system performs the deployment operation and provides to the *initiator*' a feedback message.

Extension points

5 The *initiator* does not confirm the operation.

1. The system restarts from step number 2.

Retrieve FIA Development Project

Name	Retrieve a FIA Development Project From the Catalogue
Initiator	FIA Development Project Deployer
Goal	Retrieve FIA Development Project From the Catalogue.
Description	This functionality allows a FIA Development Project Deployer to select a specific FIA Development Project from the set of projects he has been assigned to.

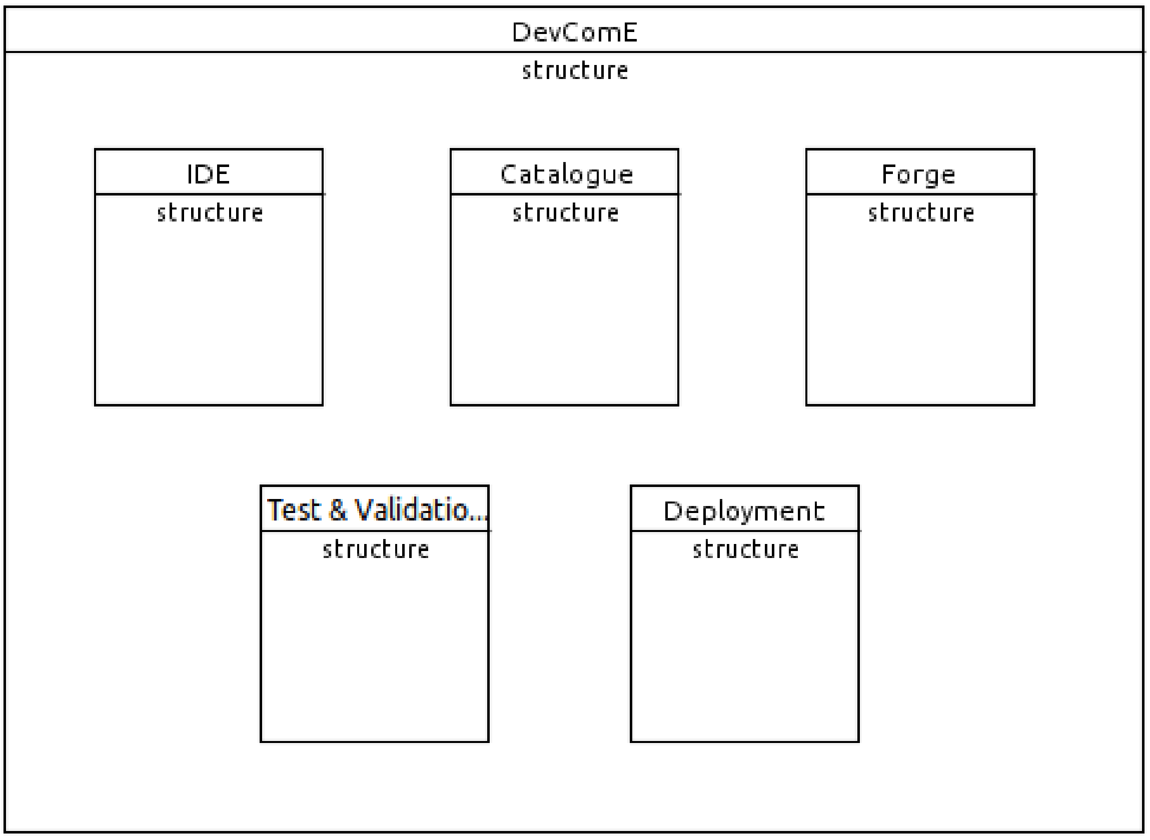
Steps

1. The *initiator* starts, within the IDE, the process for retrieving a FIA Development Project from a Catalogue.
2. The shows to the *initiator* the list of FIA Development Project he has been assigned to.
3. The *initiator* selects a specific FIA Development Project.

DevComE Architecture

FI-WARE will provide an harmonized set of tools and methods named FI-WARE DevComE (Figure: FI-WARE DevComE) composed by both clients and servers components, separately available. Specifically, FI-WARE will directly provide:

- FI-WARE IDE;
- FI-WARE Catalogue;
- Forge;
- Test & Validation Tools
- Deployment Tool.



FI-WARE DevComE

Assumptions

To implement and validate the proposed DevComE, the technology baseline identified is composed by:

- Java programming language (e.g. Oracle JDK);
- Eclipse IDE + Plug-ins (more details on a dedicated section);
- Forge solutions:
  - FusionForge (as FI-WARE reference implementation)
  - QualiPSo Factory (development still in progress)
  - GitHub (to be evaluated)
- Drupal content management system (as FI-WARE Catalogue reference implementation)
- All FI-WARE Compliant Platform Products are assumed to be e-Services [1] (e.g. Web Services, REST, …).

Forge

The Forge is the tool in charge of providing a Collaborative Development Environment [2] (CDE) to support a software development project. A Forge usually provides in a unique environment the following tools:

- version control system;
- bug tracking system;
- to-do list;
- mailing list;
- document management system;
- forum.

FI-WARE project adopts the FusionForge [3] solution as first reference implementation, to be integrated into the DevComE, while other solutions (GitHub, QualiPSo Factory) will be evaluated as additional options.

Anyway, the development of the CDE is not a goal of the *Developer Community and Tools* activities.

The choice of an open source solution, for a Forge, allows to get in contact with the community of developers and leaders, to understand the code base and, if/when it's the case, to contribute some updates that are driven by the FI-WARE needs and, at the same time, are general enough to be included into the official development line.

It's worth to note that a comprehensive CDE may be provided by a single Forge solution or thanks to the composition of specific tools, each one able to cover one (or more) feature of a CDE. In the following are listed some of these tools that are usually adopted by the people working in the open source environment:

- version control system (e.g. Subversion, Git)
- bug tracking system / to-do list (e.g. Bugzilla, Mantis, Trac, Jira)
- mailing-list (e.g. Mailman)
- document management system (e.g. Alfresco, Nuxeo)
- forum (e.g. JForum, Vanilla)

FI-WARE IDE

The approach to the development of the FI-WARE IDE is based on the idea that the user (developer, architect, project manager, etc) can improve the productivity by having a direct contact, directly from the IDE, with the other tools that are part of the whole DevComE (e.g. a Forge). The proposed solution is designed to reduce the distance and the possible misalignment between the project manager and the developers that are working together on the same project.

The approach to the development of the FI-WARE IDE is based on the idea that the user (developer, architect, project manager, etc) can improve the productivity by having a direct contact, directly from the IDE, with the other tools that are part of the whole DevComE (e.g. a Forge). The proposed solution is designed to reduce the distance and the possible misalignment between the project manager and the developers that are working together on the same project. The integration involves:

- CDE provided by the Forge;
- Catalogue;
- Test and Validation tools;
- Deployment tools.

The target users of this IDE are mainly the software developers working on the FI-WARE Compliant Platform Products and the community of software developers that will build Future Internet Applications and Services on top of FI-WARE Instances.

The proposal is to produce a FI-WARE IDE package that is based on the Eclipse IDE and already contains a set of, available, plug-ins selected according to the partners experience. In addition to these , other plug-ins, tailored to the needs of FI-WARE, are going to be developed and included into the final FI-WARE IDE package.

During the initial configuration phase, the FI-WARE IDE, will be arranged to work with a specific FI-WARE Instance. The benefit of this choice is to have directly available the list of the Generic Enablers provided by that FI-WARE Instance and be directly bound to the Forge it eventually provides.

The decision to adopt the Eclipse IDE is because of the assumption of considering Java based solutions and because it has a modular architecture (plug-ins based) as well. An additional and non-technical reason is, of course, its consolidated stability, reputation and community dimension.

The integration of the IDE with the DevComE tools allows the developer to remain better focused on the current task avoiding the continuous switching between different interfaces. On the other side the project manager may have the option to interact, for example, with the CDE features from the IDE and at the same time have a closer view to the developed artifacts.

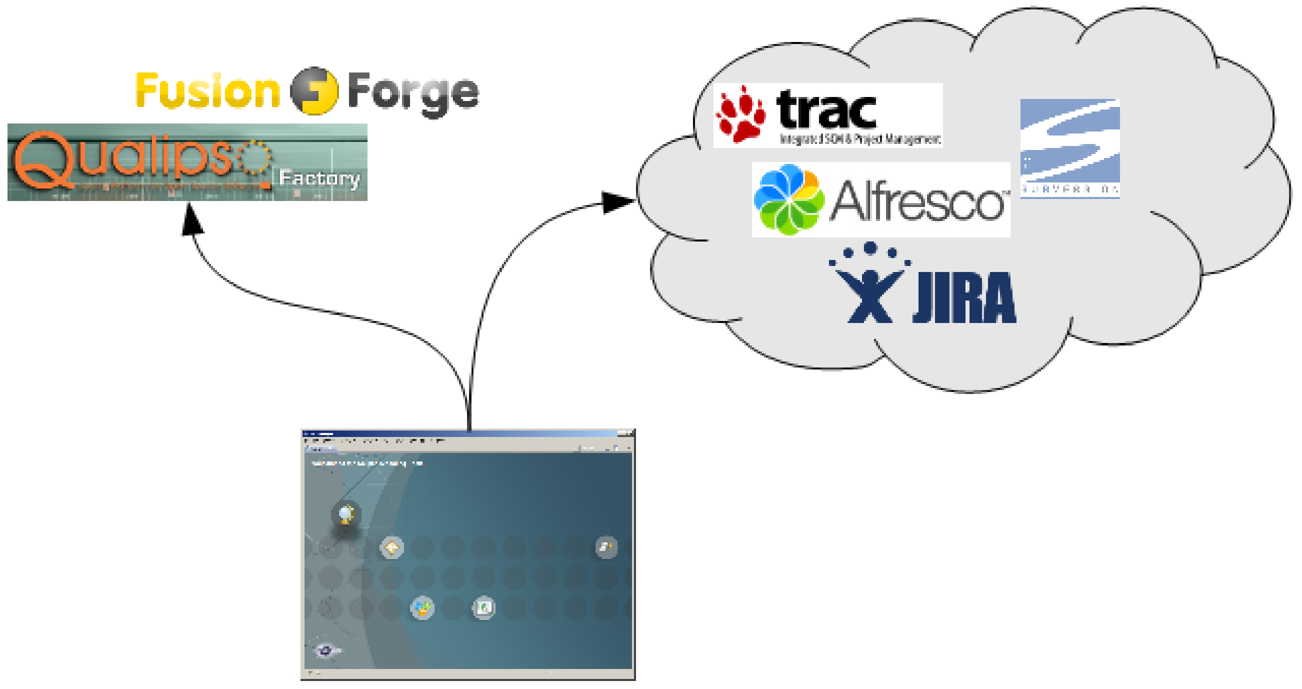
The goal is to provide through the IDE only the basic information and features of the other DevComE tools and not to completely replicate the user interface of these tools inside the IDE. This allows concentrating more on the added value features that will be agreed later on according to the users needs and requirements.

Next sub-chapters will explain in more details the characteristics of the integration between the IDE and the other tools composing the DevComE.

IDE-Forge integration: The FI-WARE MetaForge

This section explains the proposed integration between the IDE and the CDE (provided by the forge).

The IDE will connect to the CDE features whether they are provided by as a single forge solution or as a composition of specialized tools (see Figure: CDE composition).



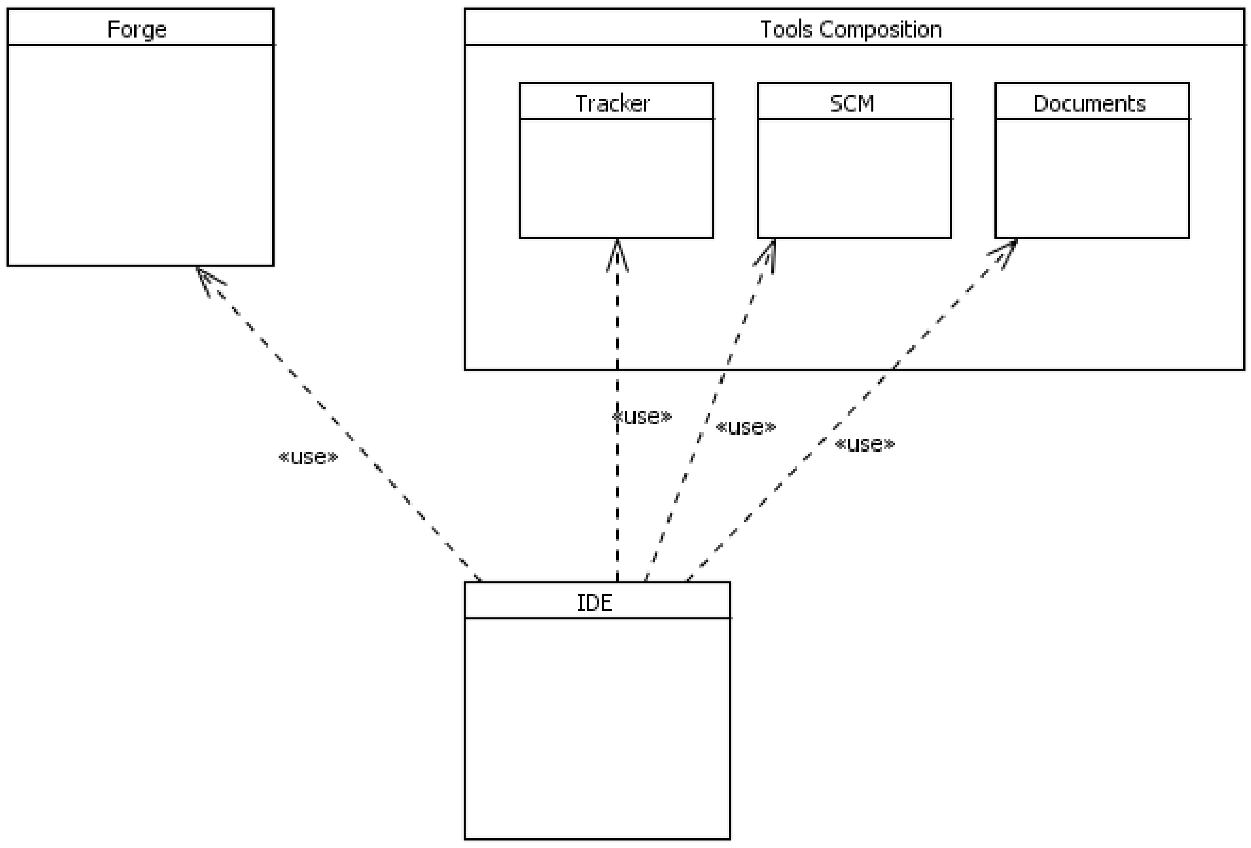
CDE composition

Main features provided by the integration (In *Italic*, the functionalities that are completely or partially provided by third party tools - e.g. Mylyn, Eclipse Plugin, etc. - and assessed during the development phases):

- *submit a new bug*
- *submit a new task*
- *associate a file to a bug/task*

- cross search: this allows to query different tools, for the same terms, and present the results in an homogeneous way.
- contextual menu: add a contextual menu that, from the IDE, lets the user, starting from an highlighted text (code, docs, etc) to:
- relationship management: (to be detailed after the architectural approval)
  - create relations between resource items
  - display relations between the resources in a mind-map style

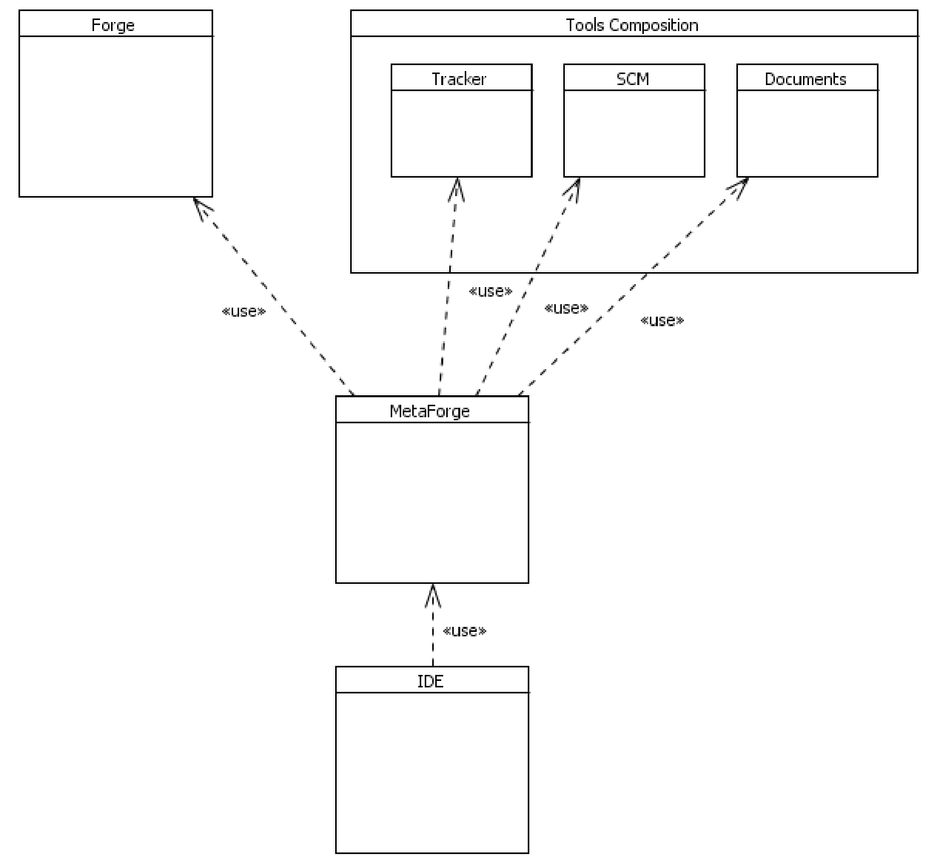
The Figure: IDE - CDE interaction highlights the two different types (Forge vs Tools Composition) of CDE that are envisaged for the integrated architecture:



IDE - CDE interaction

In order to make the solution as more flexible as possible in terms of variety of forges/tools that can be supported by the IDE integration, an additional layer is inserted between the IDE and the tools. This layer, the FI-WARE MetaForge (Figure: IDE - MetaForge - CDE), is designed to operate as an abstraction layer that, from one side, exposes the functionalities and the data using the IDE user interface, while, on the other side, it interacts with the actual installation of the tools.

This approach lets the IDE user interface and the functional logic to be independent from the tools actually installed in back-end.



IDE - MetaForge - CDE

The MetaForge component is divided in three main parts:

- Data Layer
- Functional Layer
- Driver Layer

Data Layer

All the data (or resources in general) handled by the MetaForge are defined in terms of classes (task, bug, document, etc) and relative attributes.

This is a chapter that will evolve along the next months and according to the elements that will be managed in the metaforge.

Functional Layer

This layer implements the functionalities made available by the MetaForge, based on the resources modeled by the Data Layer. Every functionality may replicate the respective one available from the actual tool (create, modify, delete, etc.) or may be a composition of those, also involving different tool types (e.g. cross search).

This layer implements the functionalities made available by the MetaForge, based on the resources modeled by the Data Layer. Every functionality may replicate the respective one available from the actual tool (create, modify, delete, etc.) or may be a composition of those, also involving different tool types (e.g. cross search) This layer is mainly responsible to host:

- the services APIs;
- the functional processes (as service composition)

This is a chapter that will evolve along the next months and according to the elements that will be managed in the metaforge.

Driver Layer

The real interaction between the IDE and the tools is implemented in a driver logic style. For instance, it can be seen as the one defined for the database systems:

- database server - collaborative tool
- database driver - tool driver
- jdbc APIs - MetaForge APIs

The execution of the operations requested by the MetaForge is implemented at the level of the tool driver.

This structure allows to select the desired forge, or set of tools, with the only constraint to have available (or develop) the implementation of the MetaForge APIs.

Main benefits obtained by this integration:

- same IDE user interface independently from the adopted forge solution (as bundle or as composition of tools), the IDE integration allows to manage the information using a common and uniform user interface;
- integrate different forges and tools only associating and configuring the driver implementation for that specific forge or tool; a tool can be integrated only providing the implementation of the required APIs.
- the focus is more on the resources and the processes that manage them, than on the tools actually installed.

This is a chapter that will evolve along the next months and according to the elements that will be managed in the metaforge.

IDE-API integration

The FI-WARE IDE supports the access to the FI-WARE Compliant Platform Products that are at the base of the development. The integration of these products into the Future Internet Applications and Services is done by means of a GE Client software component (e.g. client, driver, stub) that is specific for GE implementation, compliant to that GE Open Specifications and programming language dependent.

The GE Client (usually provided by the GE Provider) will include whatever needed (to be specified later on by Asset Suppliers) by the developer to complete an entire process of development, test and deploy. This will be further defined during the next steps of the project.

This is a chapter that will evolve along the next months and according to the inputs that will be received from the activities in charge of GE definition and development.

FI-WARE Catalogue

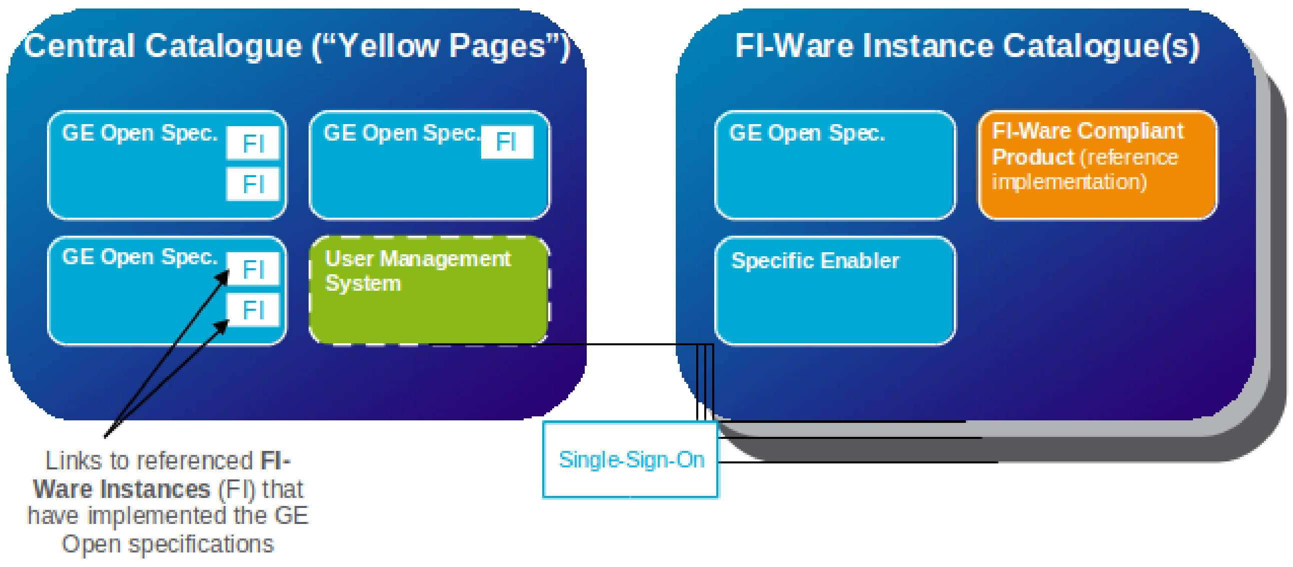
The catalogue is a container for assets published by asset providers and made available to developers that can browse the catalogue to find the specific assets they are interested in and download or use those assets. It provides features for asset publishing, browsing of assets, community for discussions around these assets and references to the appropriate support functionalities available for each asset.

Catalogue deployment view

There will be a need for two different instance types of the catalogue. First there is the instance catalogue, of which there will be one for each FI-WARE instance. The first such will be the catalogue for the FI-WARE testbed. This instance catalogue will contain:

- GE open specifications
- FI-WARE compliant platform products (including clients)
- Instance specific enablers and clients

Secondly there will be the central FI-WARE catalogue, which will contain the same GE open specifications as the instance catalogues. This catalogue instance type will be populated by each of the instance catalogues, such that it has a listing or database of each GE open specification that has been uploaded to an instance catalogue as well as references to the instance catalogues where the GE implementation can be found. As new generic enablers are added to an instance catalogue they will automatically be added to the central catalogue as well. Instance specific enablers are not automatically added to the central catalogue. For the initial deployment in the FI-WARE testbed we will most likely combine the two instance types above, and make the FI-WARE testbed catalogue act as both the central repository for GEs and as the testbed instance catalogue.


Central Catalogue ("Yellow Pages")
GE Open Spec.
FI
FI
GE Open Spec.
FI
GE Open Spec.
FI
FI
User Management System
Links to referenced FI-Ware Instances (FI) that have implemented the GE Open specifications
FI-Ware Instance Catalogue(s)
GE Open Spec.
FI-Ware Compliant Product (reference implementation)
Specific Enabler
Single-Sign-On

Catalogue Types

Proposed solution

The Catalogue architecture is based on the existing Ericsson Labs developer community and the experience from running that. However there will be some modifications made to provide a better fit to the needs of the FI-WARE Catalogue. The core of this solution is the Drupal CMS with extensions and adaptations made to better enable the specific functionality required for this system.

Catalogue functionality

The following chapters provide more details on distinct parts of the Catalogue functionality, including the asset publishing, community functions and user management.

Asset Publishing

The Catalogue environment is in charge of publishing and making available the various assets that can be used to build FI applications and new assets. More specifically:

- Generic Enabler Open Specifications
- FI-WARE compliant platform products
- Documentation
  - FAQ
  - Terms-of-use for:
    - Generic Enabler Open Specifications
    - FI-WARE Compliant Platform Products
    - Test clients

The FI-WARE Catalogue contains open specifications as well as references to the instance catalogues where an implementation of those specifications can be found. Publishing in the catalogue is done in the instance catalogue (as opposed to the central catalogue), and can be done either through the IDE or directly in the catalogue using web-based forms. All projects who wish to publish into the Catalogue need to align to specified requirements on:

- Technical readiness (test&verification; …)
- Documentation
- Categorization and keywords to enable search

The publishing process includes verifying that requirements are met. The process can be more or less automated; this will be specified in more detail later on. The FI-WARE Catalogue access policy is defined in order to regulate the access to the artefacts and related resources. Parts of the Catalogue will be visible to anyone, while parts (typically downloads and API key requests) are only available to registered users. The Catalogue will contain a document specifying the rules for publishing and access.

Support

There is a back-end tracker system to handle support requests, which distributes requests to the right GE provider. This is not considered a part of the catalogue and will have to be handled by separate external request tracker system. The catalogue will link to this system.

Developer Community

The catalogue also includes maintaining, monitoring and supporting the community of developers who use, develop or are just generally interested in the assets available in the Catalogue and/or the FI-WARE Instance as such. Keeping the community ‘alive’ and active is necessary in order to attract developers and drive usage. It cannot be left only up to the community itself, but some moderator activity needs to be continuously on-going. Developer community maintenance is largely not a technical issue, and so will be described in more detail in the methodology supporting the adoption of the FI-WARE SDK, here is defined only the technical tools to support it.

- News and blogs
- Developer fora/discussions
- Subscriptions

The catalogue will include linking to social media channels, to allow developers and the community responsible to share, promote and provide feedback.

User Management

The catalogue itself requires some kind of user management. For this we propose to use a central user management system that is based on Ericsson Developer Account, the user management system used for Ericsson Labs. The advantage of this central user management system that will be bundled with the central FI-WARE Catalogue is that it will provide single sign on functionality for users as they go between different catalogue instances as they browse. One typical use case is for a user to start out in the central catalogue, find a GE that he is interested in and then follow the links to the different FI-WARE Instance Catalogues where the GE is implemented to find an instance that seems interesting to work with. Having to register and log in to each separate instance would make the process much more complicated for the user. However, for the case where an FI-WARE instance provider wants to set up his own user management separate from that provided by the central FI-WARE Catalogue this is also supported. As the Drupal CMS is used as the foundation for the catalogue it is easy for the specific implementation to either pick another third party user management solution or to use the built-in Drupal user management.

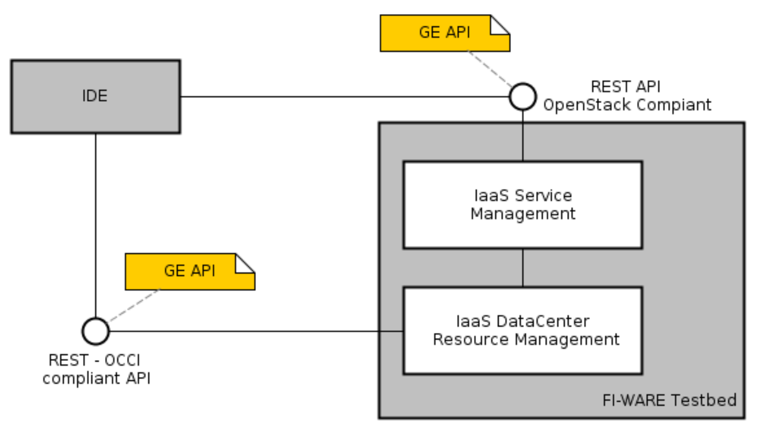
Optional catalogue components

This chapter contains optional catalogue components that have been discussed as features that we may or may not want to include as a part of the catalogue deliverable. As no firm decision has been made yet the components are included here as a reference, and once decision has been made the section will either be included in the previous sections or deleted from the document.

API key handling

The API key system as used in Ericsson Labs can be used by an API developer to allocate a key to each user of the API and provides facilities to limit the usage of the API to a certain number of requests or a certain request rate per API key. This is a service that can be provided by the Catalogue. It is optional for API developers to use the service or choose one of their own (this can be regulated in the publishing rules by the FI-WARE Instance owner). This would provide asset providers with a tool with which they can limit or throttle the access to their API. The reasons for doing this could include managing server load, blocking users that abuse the service or providing premium (paid for) services to high load users.

Deployment Tool



Architecture Overview

- IaaS DataCenter Resource Management
  - uses OpenStack as back-end solution
  - exposes a REST and OCCI compliant API
- IaaS Service Management
  - uses OpenStack as back-end solution
  - exposes OpenStack API
    - this API will be the one defined as GE API
    - the full implementation will be ready by Sept/Oct 2012
- IDE
  - interacts with RM for single VM mgmt
  - interacts with SM for multiple VM mgmt

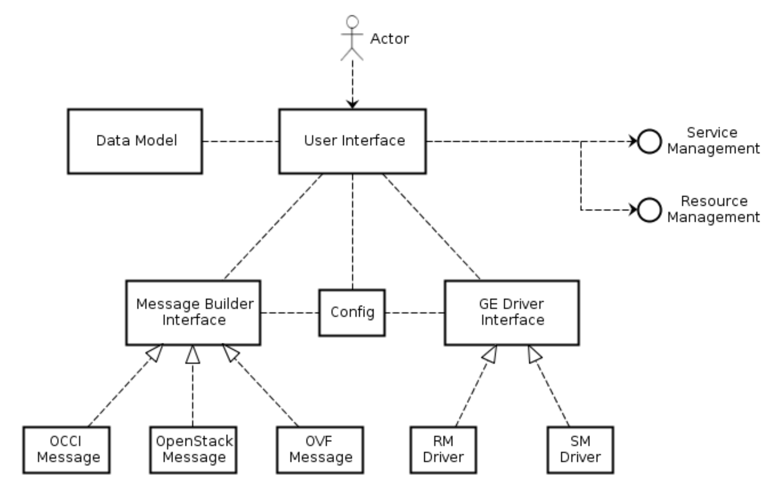
IDE Model

User Interface

the user interfaces, wizards and Eclipse views, used by the Actor to interact with the Cloud Hosting GEs. They use the *Message Builder Interface* and *GE Driver Interface* to interact with *SM* and *RM* end points.

Data Model

the component that stores the structured information managed by the user and exchanges with the GEs



Architecture Overview

Config

this part maintain the configuration that defines if the User Interface is interacting with a SM or a RM and the format of the exchanged messages.

Message Builder Interface

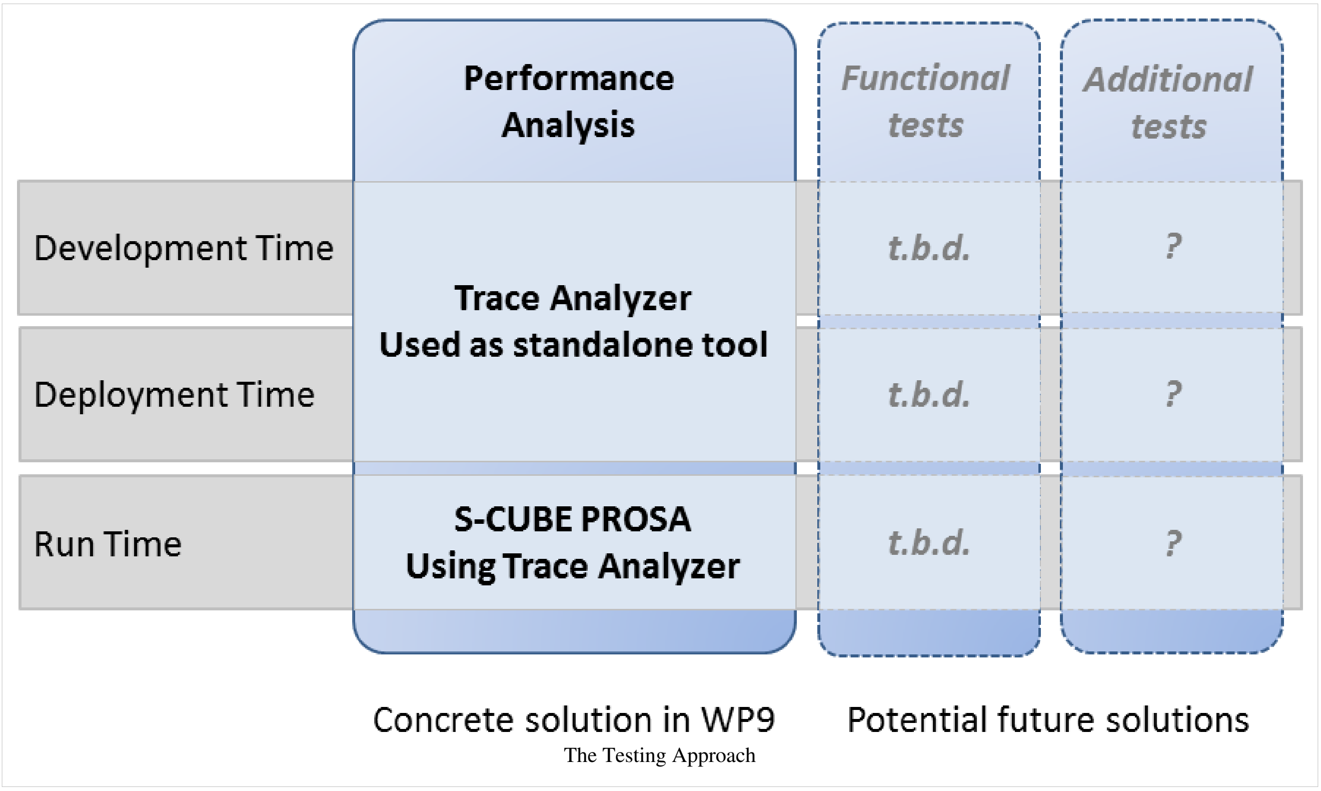
interface used to translate the information from (and to) the *Data Model* to (and form) the appropriate format defined in the *Config* component (may depend also on the GE Driver implementation). The implementations of this interface cover different formats such as OCCI, OpenStack and OVF.

GE Driver Interface

interface used to interact, in terms of operations to call, with the GE defined in the *Config* component. The abstract operations defined by the interface are mapped to the specific operations provided by the *SM* and *RM* GEs end points.

Testing and Validation Approach

While we can assume that state-of-the-art testing will be performed on FI-WARE applications under development, this activity will focus on testing activities that are specific to the nature and architecture of FI-WARE applications and instances. These are characterized by multiple distributed components and services that will not be complete before they are deployed. Thus, testing approaches suited for FI-WARE applications must span different stages of the development process: development time, deployment time, and run time. With these demands, different testing technologies can be defined in the context of FI-WARE as shown in Figure: *The Testing Approach*. In order to focus the activities performed by the DCT team, different testing approaches will be considered one after another to ensure substantial results for single approaches.



The Testing Approach

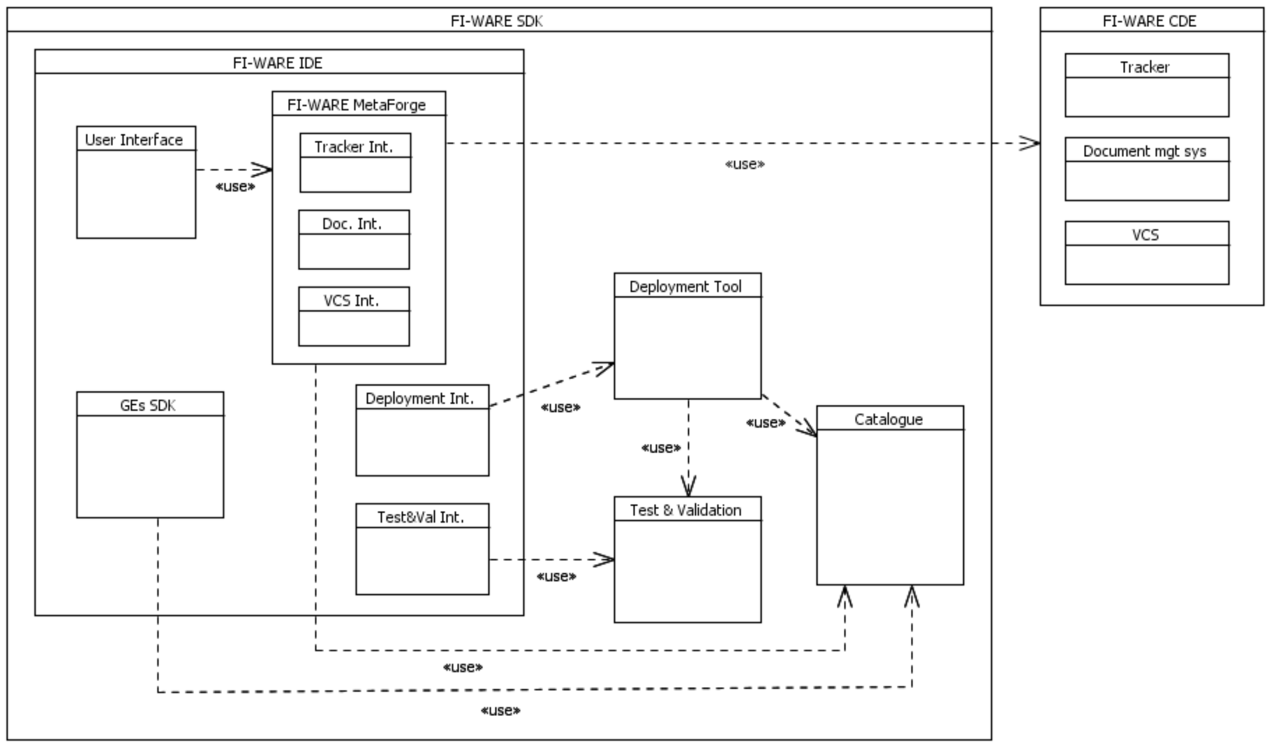
The following assets are provided by the partners for testing:

- Performance testing during development, deployment and run time will be supported by the **Trace Analyzer** tool [Biberstein et al, "Cell Broadband Engine processor performance optimization: Tracing tools implementation and use", IBM Journal of R&D], providing collection and analysis of data across different layers (h/w, os, middleware, services). The tool provides visualizations for the collected data, allows filtering and aggregating the collecting data, highlights performance anti-patterns, and identifies bottlenecks in the execution.
- The **Software Performance Cockpit** [SoftwarePerformanceCockpit] is a result of the EU project SLA@SOI. It allows for the definition and execution of complex test scenarios. Furthermore, it comes with rich mechanisms for analysing results by statistical means. Based on input from other project activities/teams and use cases, the tool will be extended to better cover distributed FI-scenarios.
- **PROSA** is a result of the EU project S-CUBE analyzing the quality of monitoring data gathered at run time and improves it if necessary by invoking tests.

Thus the DCT team will focus on performance and QoS analysis and test first. At development time and deployment time, Trace Analyzer will be provided for filtering, aggregating, and visualizing monitoring data gathered from testbeds and target environments. This will support developers in providing applications using resources in an optimal way. Furthermore, the Performance Cockpit supports the analysis of design decisions but also for planning feasible service level objectives for some given services or components. At run time, a continuous surveillance of the performance of all distributed services constituting a FI-WARE instance is desirable. The related data can also be handled by Trace Analyzer. In addition, it is necessary to consider the quality of the QoS-related data here. This is possible with S-CUBE PROSA which defines requirements for data points, monitors the availability of according data, and invokes tests in order to generate necessary data points through the testbeds or target environments.

Conclusions

The diagram below is a snapshot of all the elements that will take part to the community development environment we plan to develop. Each of the components below have been somehow described in the text above and will be further detailed in the next versions as soon as additional inputs will arrive from the other project participants and some of the open issues among our team will be sorted out.



Detailed view of the overall FI-WARE SDK environment

Annex

Here are reported the two questionnaires that have been used to collect requirements that are used to support the decision taken at Tools Chapter level and to define the road-map for the features to make available.

The FI-WARE WP9 Questionnaire is focused on gathering of information from the FI-WARE project partners. It has been produced in two subsequent versions (second version available here below) with a simplified second one that had the aim of collecting more contributions compared to the first call. The partners that haven't sent their contributions have been contacted in person during the Madrid plenary meeting in January 2012.

The FI-WARE IDE Survey is a specific questionnaire focused on collecting the priorities, based on personal viewpoint, regarding a set of proposed features to insert into the road-map for the development of the FI-WARE IDE. This survey has been useful also to collect additional features required by people directly involved in development aspects, also outside FI-WARE project partners and especially within partners corporate colleagues.

FI-WARE WP9 Questionnaire

Version 2.0

General

1. Is the document sufficiently clear in describing "the what" and "the how"? *(List the sections to improve specifying if it has to be improved the content or the readability)*
2. Do you have any specific standard or technological constraint?
3. Are the Use Cases descriptions sufficiently detailed and clear to understand what the functionality is expected to do?
4. Do you think additional functional and non-functional requirements are needed in the proposed environment? If yes, which ones?
5. How do you imagine your GE will be used?
6. How are GE Open Specifications defined? Do they define the interface of the GEs in a standard/commonly-accepted language?
7. If there is no standard/commonly-accepted language for defining GE specifications, HOW will YOU define your GE Open Specification?

IDE - CDE Interaction

1. Is it clear the purpose of the integration between the IDE and the CDE?
2. What are the most valuable features you would expect from the CDE facilities directly accessible from the IDE? (e.g. CRUD on tickets, ...)
3. If you were using the FI-WARE IDE what of the following functionalities you would expect to use directly from it? Please, prioritise (HA - highly appreciable, MA - Medium Appreciable, LA - Low Appreciable)
    1. Tickets *(manage the tickets of the forge from the Eclipse IDE, based on Mylyn)*
    2. Bugs *(the same as for Tickets)*
    3. Tasks *(the same as for Tickets)*
    4. Cross contents searches *(an IDE based search interface that lets to query different sources and displays aggregated results)*
    5. Contextual Menu: *(starting from a text/code selection)*
        - add a new Forum post
        - add a new Bug/Task/Ticket
        - start a collaborative session *(see Collaborative editing)*
    6. Contextual Chat *(start a chat directly addressed to a specific topic - e.g. method, lib, etc.)*
    7. Mailing list access and Mail notification
    8. User creation/authorization *(forge users management)*
    9. Policy settings
    10. Project creation
        - new project into the forge
        - selecting a specific type *(GE dev. - Application dev.)*
    11. Search for GE clients within the FI-WARE Catalogues *(a wizard that lets to browse the specific Catalogue and select the GE clients to use)*
    12. Enabling FI-WARE instance *(IDE configuration to work with a specific FI-WARE instance - supported by wizard)*
    13. Social networking interaction *(send and receive updates to/from the major social networks; e.g. twitter)*
    14. Relationship Service *(an IDE interface that enables to relate various items among each other - source code, bugs, tickets, forum, chat, etc.)*

15. Collaborative Editing *(edit simultaneously a source file with other members of the team)*

4. In your opinion are there functionalities currently not considered? If yes, which ones?

Generic Enablers

1. Do your GEs depend on other GEs in order to be used? If yes, synthetically explain these dependencies.

Catalogue

1. Do you see something important that's missing in the high level features of the Catalogue?
2. What are the most valuable features you would expect from the Catalogue?

Deployment / Test / Validation

Questions to all GE Providers

Performance testing

1. For what purposes do you need performance data? Please choose all the options that apply.
   - SLA monitoring
   - Monitoring performance at interfaces with other services
   - Monitoring performance of GE and its components
   - Bottleneck prediction
   - Other, please specify: ___________________________
2. What information are you interested in? Please specify the combinations you are interested in ("I want to see CPU and IO data together") and acceptable aggregation levels. Please choose all the options that apply.
   - CPU
   - IO
   - Network
   - Hypervisor activity (which?)
   - Guest OS activity (which?)
   - Performance of interfaces to other services
   - Inter-thread/process communication
   - Performance of user-defined internal interfaces
   - GE performance characteristics
   - Other, please specify: ___________________________
3. What are the most important aspects of your GEs that has to be monitored at run time? (response time, bandwidth, server resources, ...)
4. What do you perceive to be the most important challenges when testing software and/or services that rely on the capabilities offered by your chapter?

Deployment testing

1. Can the platform used to run your software be installed on common Linux machine? E.g. if your GE run on a mobile platform, There exist any simulation tool that make it possible to deploy the GE on a Linux machine and test from there?
2. How many distributed nodes do you expect to use in a typical deployment test for your software?
   - one (e.g. your sw is installed on the node and a testing script check everything is like expected)
   - two (e.g. client-server: the server is installed on a node and the client make some calls from a second node. This would allow to check also the network configuration of the deployment)
   - 3 to 5
   - more than 5. Please quantify: ___________________________

3. Can deployment tests be executed in batch mode (non-interactive mode, without human intervention)?

Questions to Service Composition Engine Provides (WP3)

1. Which information are being monitored by your service composition engines (business process engine, server-side mashup engine, ...) that are available from the outside?
    - Execution status of a service composition
    - Actual variable binding of a service composition
    - Performance information (expired runtime, call durations of external services, ...)
    - No monitoring available.
2. Do your service composition engines (business process engine, server-side mashup engine, ...) provide means for adaptation (e.g. replacement of a called service implementation at runtime)?
    - Yes, even for RUNNING instances!
    - Yes, but only on MODEL level (not for specific instances).
    - No adaptation possible at all.

Questions to Cloud Service Providers (WP4)

1. For what purposes do you need performance data? Please choose all the options that apply.
    - SLA monitoring
    - Infrastructure monitoring
    - Bottleneck prediction
    - Co-location optimization
    - Other, please specify
2. What information are you interested in? Please specify the combinations you are interested in ("I want to see CPU and IO data together") and acceptable aggregation levels. Please choose all the options that apply.
    - CPU
    - IO
    - Network
    - Hypervisor activity (which?)
    - Guest OS activity (which?)
    - Application performance characteristics
    - Other, please specify: ____________________________
3. What information can you expose to the GEs for application performance analysis? Please choose all the options that apply.
    - CPU
    - IO
    - Network
    - Hypervisor activity (which?)
    - Guest OS activity (which?)
    - Other, please specify: ____________________________
4. Which client-related performance information is of interest to identify possible bottlenecks
5. Which monitoring data about communication with clients can be provided for this purpose (identification of possible bottlenecks)?
6. What SLA information is available to Cloud Service Providers?
7. What kind of performance data does the monitoring GE provide for Cloud services? Please give some examples.

FI-WARE IDE Survey

The survey, to collect priorities information on proposed FI-WARE IDE features is reported here for convenience as it has been integrated into the FI-WARE WP9 Questionnaire.

Together with the proposed features here are reported also the results, in square brackets, as a number between 1 and 3. This value is computed as average of the votes provided (HA=3, MA=2, LA=1). The design and the planning of the Tools Chapter activities has been based also on the highest evaluated features obtained from the survey result.

If you were using the FI-WARE IDE what of the following functionalities you would expect to use directly from it? Please, prioritise (HA - highly appreciable, MA - Medium Appreciable, LA - Low Appreciable)

1. Tickets *(manage the tickets of the forge from the Eclipse IDE, based on Mylyn)* **[2,71]**
2. Bugs *(the same as for Tickets)* **[2,71]**
3. Tasks *(the same as for Tickets)* **[2,71]**
4. Cross contents searches *(an IDE based search interface that lets to query different sources and displays aggregated results)* **[2,18]**
5. Contextual Menu: *(starting from a text/code selection)*
    - add a new Forum post **[1,83]**
    - add a new Bug/Task/Ticket **[2,18]**
    - start a collaborative session *(see Collaborative editing)* **[2,27]**
6. Contextual Chat *(start a chat directly addressed to a specific topic - e.g. method, lib, etc.)* **[1,16]**
7. Mailing list access and Mail notification **[1,28]**
8. User creation/authorization *(forge users management)* **[1,83]**
9. Policy settings **[1,33]**
10. Project creation
    - new project into the forge **[2,06]**
    - selecting a specific type *(GE dev. - Application dev.)* **[2,22]**
11. Search for GE clients within the FI-WARE Catalogues *(a wizard that lets to browse the specific Catalogue and select the GE clients to use)* **[2,25]**
12. Enabling FI-WARE instance *(IDE configuration to work with a specific FI-WARE instance - supported by wizard)* **[2,07]**
13. Social networking interaction *(send and receive updates to/from the major social networks; e.g. twitter)* **[1,47]**
14. Relationship Service *(an IDE interface that enables to relate various items among each other - source code, bugs, tickets, forum, chat, etc.)* **[2,17]**
15. Collaborative Editing *(edit simultaneously a source file with other members of the team)* **[2,42]**

In addition to the features proposed, other requests has been pointed out and here are reported the most interesting ones:

1. Test management tool integration
2. Database design and reverse engineering
3. UML and ER modelling tools
4. License compatibility check

Some of these are already taken into account for the design and planning of the next activities (e.g. Test management tool integration).

References

[1] http://en.wikipedia.org/wiki/E-Services
[2] http://en.wikipedia.org/wiki/Collaborative_development_environment
[3] http://fusionforge.org