Deliverable D3.7

Initial DECIDE OPTIMUS

Editor(s):	María José López
Responsible Partner:	TECNALIA
Status-Version:	Final – v1.0
Date:	27/11/2017
Distribution level (CO, PU):	CO

Project Number:	GA 726755
Project Title:	DECIDE

Title of Deliverable:	D3.7 Initial DECIDE OPTIMUS
Due Date of Delivery to the EC:	30/11/2017

Workpackage responsible for the Deliverable:	WP3 – Continuous Architecting
Editor(s):	TECNALIA
Contributor(s):	TECNALIA
Reviewer(s):	Gema Maestro (EXPERIS)
Approved by:	All Partners
Recommended/mandatory readers:	WP3, WP4, WP5

Abstract:	This software deliverable comprises the initial OPTIMUS simulation engine. The initial version will realize basic features that will be extended in the subsequent versions.
Keyword List:	Simulation, classification, Eclipse IDE, plugin, java
Licensing information:	This program and the accompanying materials are made available under the terms of the Eclipse Public License 2.0 which is available at https://www.eclipse.org/legal/epl-2.0/ The document itself is delivered as a description for the European Commission about the released software, so it is not public.
Disclaimer	This deliverable reflects only the author's views and views and the Commission is not responsible for any use that may be made of the information contained therein

Document Description

Document Revision History

Version	Date	Modifications Introduced	
		Modification Reason	Modified by
v0.1	15/11/2017	TOC & General sections	TECNALIA
v0.2	16/11/2017	First Draft version	TECNALIA
V0.3	23/11/2017	Version ready for internal review	TECNALIA
V1.0	27/11/2017/	Ready for submission	TECNALIA

Table of Contents

List of Figures

List of Tables

Terms and abbreviations

ACSmI	Advanced Cloud Service meta-Intermediator
DB	Database
EC	European Commission
GUI	Graphical User Interface
JSON	JavaScript Object Notation
MCSLA	Multi Cloud Service Level Agreement
NFR	Non-Functional Requirements
UI	User Interface

Executive Summary

This document contains the technical description of the DECIDE OPTIMUS tool in its first version. The DECIDE OPTIMUS tool provides the developer with the best possible theoretical deployment for his or her multi-cloud application. This outcome is based on the classification of the microservices that compose the application, the Non Functional Requirements involved, and the cloud services handled by the DECIDE framework (specifically the ACSmI tool). The application classification (for more information see [1]) is included in DECIDE OPTIMUS as the first step of it.

Moreover, the document includes sections about how to install, and use DECIDE OPTIMUS, and the license under it is published.

The details about the functionality and the technical aspects are described in the corresponding sections as well as the manual and the instructions to test the software.

Taking into account that there is no specific document describing OPTIMUS architecture, the general design of it is presented here, so that it allows the comparison to the several prototypes of OPTIMUS and its evolution. The general requirements and the functionalities are also described in this document, as well as the coverage provided by the M12 prototype, which is the original subject of the document.

Future versions of this document will present the evolution of the DECIDE OPTIMUS tool, the new features that the new prototypes will provide to the developer and the technical characteristics associated to it. These versions are planned for being ready in the months 24 and 30.

1 Introduction

This document presents the functionalities, design and development of the DECIDE OPTIMUS tool in its first version, as a prototype. The content is the corresponding one to the first version of prototype planned for month 12.

The global architecture and the main general functionalities, as well as all the requirements for the DECIDE OPTIMUS tool, are being presented in this document as the starting point of the current and future versions of the prototypes. There is no deliverable about the general characteristics of OPTIMUS, so this document will be the reference for the development in the 24 and 30 months releases.

1.1 Document structure

This document is composed of four (4) main sections:

1. General introduction about the content and structure of the document.
2. The implementation of the DECIDE OPTIMUS. The requirements that it has to cover, its architecture and the description of its main components, its technical development aspects and how it fits into the whole DECIDE architecture.
3. A section about how to install it, how to use it and any licensing information in order to test the prototype.
4. Conclusions arisen from its design and development, and future work.

2 Implementation

2.1 Functional description

DECIDE OPTIMUS deployment simulation tool will evaluate and optimize the characteristics of the multi-cloud application deployment from the developer's perspective considering a set of provided cloud resources alternatives.

DECIDE OPTIMUS will provide the best possible deployment application topology, based on the non-functional requirements set by the developer and the requirements of the multi-cloud application, automating the provisioning and selection of deployment schemas for multi-cloud applications.

Functionalities:

The main functionalities of the DECIDE OPTIMUS global tool are:

1. Multi-cloud application classification. This functionality consists of associating the components (microservices) that form the multi-cloud application to a group of Cloud Services where they could be deployed.

   For this purpose, the profiling of the microservices of the multi-cloud applications has to be considered as an input, to match the characteristics of those microservices with the group of Cloud Services features where they will be deployed.

   This classification will be based on the information provided by the developer and the information handled by "Types management" subcomponent.

2. Theoretical deployment generation. Once the classification is made, and the NFRs gathered, it will perform a process where it will prepare a request to invoke ACSmI and obtain the Cloud Services that fulfil the requirements of the microservices for their deployment.

   This request will be composed of generic Cloud Services, and the list of resources that the microservices need. This functionality requires interacting with the ACSmI API in order to obtain the sorted list of Cloud Services that meet the criteria requested.

3. Simulation. The combination of the different possibilities of deployment, performed by a complex algorithm, taking into account the theoretical deployment possibilities and the sorted list of Cloud Services (from ACSmI) that suit them, will be ranked in order to select the best of them. The Schema with the needed information for preparing the contract with the CSPs and setting up the deployment will be shown to the developer to confirm it, and stored into the historical repository managed by the App Controller.

DECIDE OPTIMUS will be developed following an incremental strategy with several releases in months 24 and 30 of the DECIDE project plan. Therefore in this M12 prototype, the functionalities covered are the following:

1. Multi-cloud application classification. **Partially covered**. This prototype will present the UI for introducing the details about the application and the microservices, such as the name of the multi-cloud application, the name of each microservice that it is composed of, if it has a detachable resource, and if this resource access to a DB or not. The Type associated will be presented in a combo with the value of this first version classification. The developer will be able to change this value.

2. Theoretical deployment generation. **Limited coverage**. A message with the type of the microservice and the group of Cloud Services over which they could be deployed and the NFRs that are stored in the Application Description JSON file, will be created.
3. Simulation. **Not covered**. There will not be simulation in this prototype, but the JSON file created in the previous steps, including the NFRs, will be shown in the tab of the Simulation process of the OPTIMUS UI.

Requirements:

The global requirements for the DECIDE OPTIMUS tool have been analyzed, reviewed and gathered in the WP3. The requirements planned for OPTIMUS and those which will be implemented by this prototype are described in the Table 1.

Table 1. Requirements covered by the M12 prototype.

Req. ID	Req. Description	Requirement coverage by the prototype
WP3-PROFI-REQ1	Load/read information about the application (components).	The prototype will request the developer for the information. Moreover, it will load the information if the JSON file already existed.
WP3-PROFI-REQ2	Classify the application, based on the "stereotypes of the components" that we defined in the design phase of the profiling tool, and comparing it with the information about the (component) application.	Not covered
WP3-PROFI-REQ3	Ask the developer to confirm the classification	Not covered
WP3-PROFI-REQ4	Store the information about classification made.	This prototype will save the information requested about application components and the classification made. This classification will consist in a combo with a first tentative of classification. This value can be changed by the developer.
WP3-PROFI-REQ5	Mechanisms for update the "stereotypes of the components" information	Not covered
WP3-OPTI-REQ1	The OPTIMUS tool shall be capable of reading the non-functional characteristics of the app from NFR DB	The NFRs will be stored in the Application Description JSON file, and this prototype will read them from there.
WP3-OPTI-REQ2	The OPTIMUS tool shall be capable of reading the classification of the app (or its components)	The first step of the OPTIMUS tool will store the classification in the Application Description JSON file (WP3-PROFI-REQ4), and OPTIMUS will be able to read it.

Req. ID	Req. Description	Requirement coverage by the prototype
WP3-OPTI-REQ3	OPTIMUS will analyse the app NFR and the classification (FR) in order to ask ACSmI for information about cloud services that cover the requirements (F/NF) of the multi-cloud application.	Not covered
WP3-OPTI-REQ4	For each component of the multi-cloud application, OPTIMUS engine builds the theoretical composition of services needed and prepares the process (various configuration parameters and deployment topology) to simulate (normal & stressful conditions) the behaviour of the component.	Not covered
WP3-OPTI-REQ5	OPTIMUS engine runs the simulations for each component of the multi-cloud application and ranks each of them	Not covered
WP3-OPTI-REQ6	OPTIMUS shall use algorithms such as genetic algorithms, Harmony search, or Dandelion codes to provide a set of potential combinations of cloud services that fulfil the established user requirements. This process will go after the theoretical deployment generation and will combine the results of each of the possibilities.	Not covered
WP3-OPTI-REQ7	OPTIMUS shall provide the developer with the information about the proposed deployment schema (those with the highest rank) for the application to cover the required NFR and FR, and the technological risk that each of these configurations imply, i.e. moving from an IaaS to a Pass, or move from one PaaS to another. This will show in the UI and will require confirmation from the developer.	Not covered

Req. ID	Req. Description	Requirement coverage by the prototype
WP3-OPTI-REQ8	OPTIMUS tool is able to define new schema from developer side (proactively) and from results coming from ADAPT (reactively) to set up a new deployment schema, if a malfunctioning of a deployed multi-cloud application occurs	Not covered
WP3-OPTI-REQ9	OPTIMUS shall provide a forecast on some important system characteristics such as performance, cost, or security that can motivate an optimization decision	Not covered
WP3-OPTI-REQ10	DECIDE OPTIMUS will provide automation of the provisioning resources and deployment schemas for multi-cloud native applications	Not covered

The prototype described in this document will perform the functionalities described above but due to the requirements that must be accomplished, the scope of those functionalities will be the following:

Table 2. Functionality covered by the M12 prototype**.**

Functionality	Scope in this prototype	Requirement associated
Multi-cloud application classification	The information about the application and its components will be requested to the developer. They will not be asked for all the information needed for classifying the components, but a couple of details, such as the name of the application and the name of each microservice. The component classification type will be presented to the developer for them to choose one of the available ones. This list will be the first version of "general applications profiling repository". The selected classification type and the information about the application and its components will be stored into the Application Description JSON file. If the Application Description JSON file already exists, the prototype will read from it in order to present the UI with the corresponding objects already created.	WP3-PROFI-REQ1 WP3-PROFI-REQ4
Theoretical deployment generation	The prototype will process the information about the classification and will read the NFRs about each microservice. These data will compose, in	WP3-OPTI-REQ1 WP3-OPTI-REQ2

Functionality	Scope in this prototype	Requirement associated
	this version, the requirements to perform a request to the corresponding service of ACSmI in order to obtain the Cloud Services that fit those requirements.	
Simulation	This functionality is not covered by this prototype version. Nevertheless, the request to the ACSmI described in the previous row it will be shown as the best theoretical deployment selected.	

2.1.1 Fitting into overall DECIDE Architecture

OPTIMUS is the tool that obtains, through a simulation process, the best theoretical deployment for each of the components or microservices that form the multi cloud application. The global DECIDE architecture with OPTIMUS in it, is shown in Figure 1

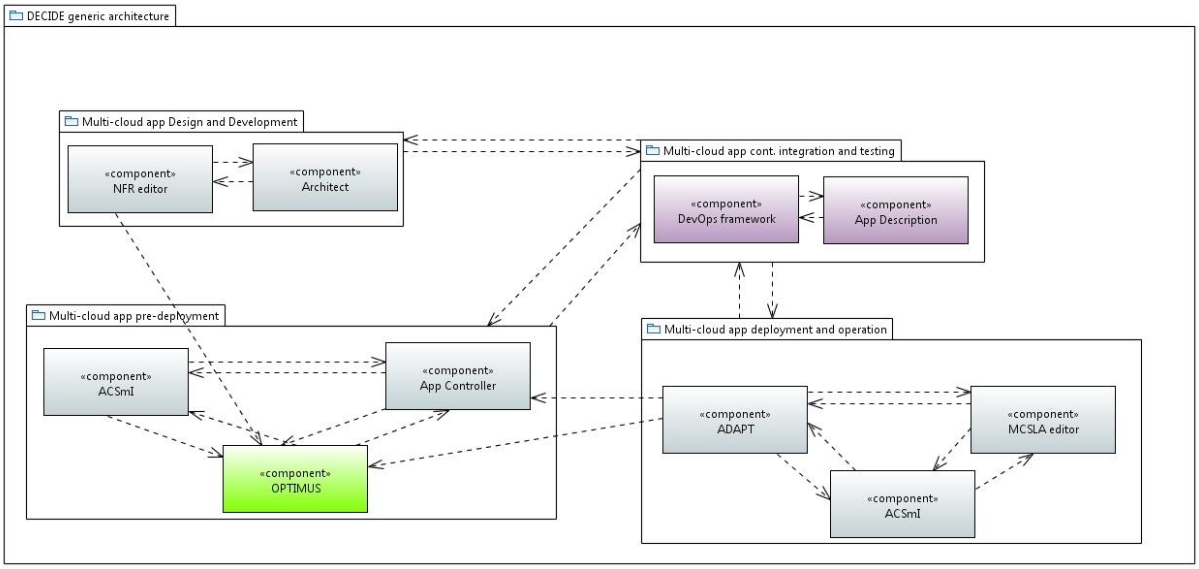

Figure 1. OPTIMUS in DECIDE architecture**.**

There are two entry points where OPTIMUS can be invoked:

1. When the developer is creating the DECIDE application, informing the characteristics of it in order to, first of all, develop it and then deploy it with the best possible deployment schema, regarding the characteristics of the multi cloud application.

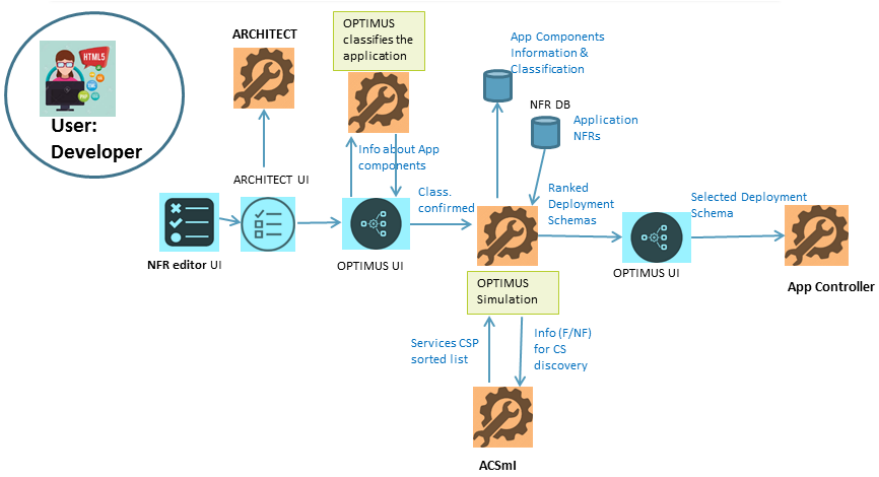

Figure 2. Developers role using OPTIMUS in the DECIDE workflow**.**

2. When ADAPT monitoring discovers a violation of the MCSLA. This means that some of the Cloud Services involved in the deployment are not fulfilling the agreement. The ADAPT tool ask to OPTIMUS for a new simulation, and OPTIMUS has to perform a new simulation for it.

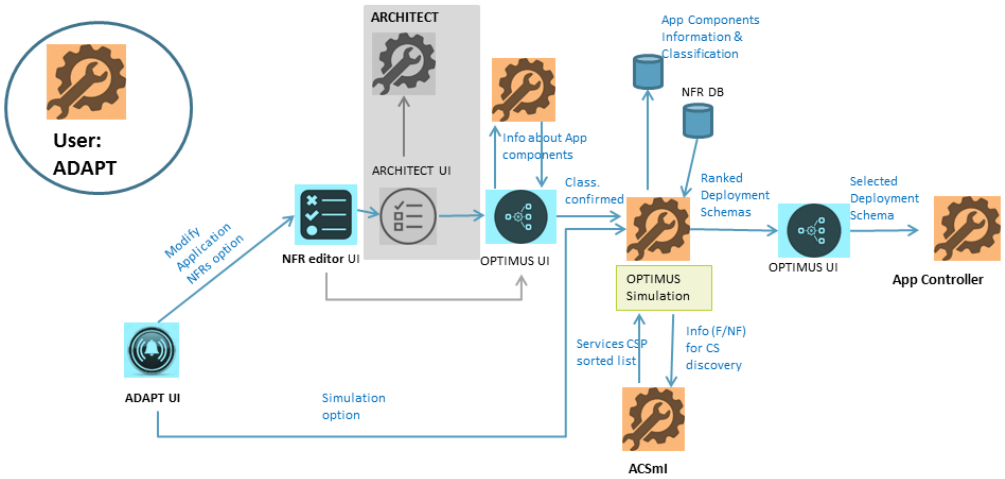

Figure 3. ADAPT role using OPTIMUS in the DECIDE workflow**.**

OPTIMUS interacts with the rest of the DECIDE tools as it is presented in the Figure 4.

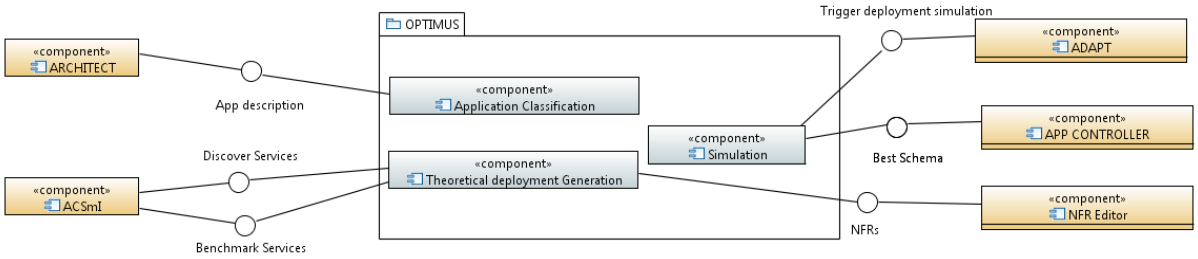

Figure 4. ACSmI Monitoring Architecture

- ARCHITECT: OPTIMUS is the tool that follows ARCHITECT in the global workflow of DECIDE [2]. The information that ARCHITECT obtains from the developer or by its own processes, will be stored in the Application Description JSON file. This file could be an input to OPTIMUS.

- ACSmI: When the theoretical deployment generation of OPTIMUS is performed, the first action is to call to Discover Services of ACSmI. OPTIMUS will ask for a list of the Cloud Services that fulfil the requirements associated to the microservices. ACSmI will send to OPTIMUS a benchmarking of the Cloud Services in a sorted list format, so that OPTIMUS can assign a score to each possibility of deployment.
- ADAPT: When one violation of the MCSLA is discovered, ADAPT request to OPTIMUS for a new best deployment schema.
- APP Controller: the best deployment schema obtained by OPTIMUS Simulation process is stored in a historic repository managed by APP Controller. This repository will be consulted in order to know if that best schema is already obtained, and in that case take the second as the best one.
- NFR Editor: Each microservice can specify some Non-Functional Requirements associated to it, and taking into account that it is OPTIMUS UI from where the information about those microservices is provided, the NFR Editor will be launched by OPTIMUS for the developer to enter those Non-Functional Requirements.

2.2 Technical description

In this section, it will be presented the technical aspects about the development of this version of the OPTIMUS prototype.

2.2.1 Prototype architecture

The general architecture planned for OPTIMUS tool is shown in Figure 5.

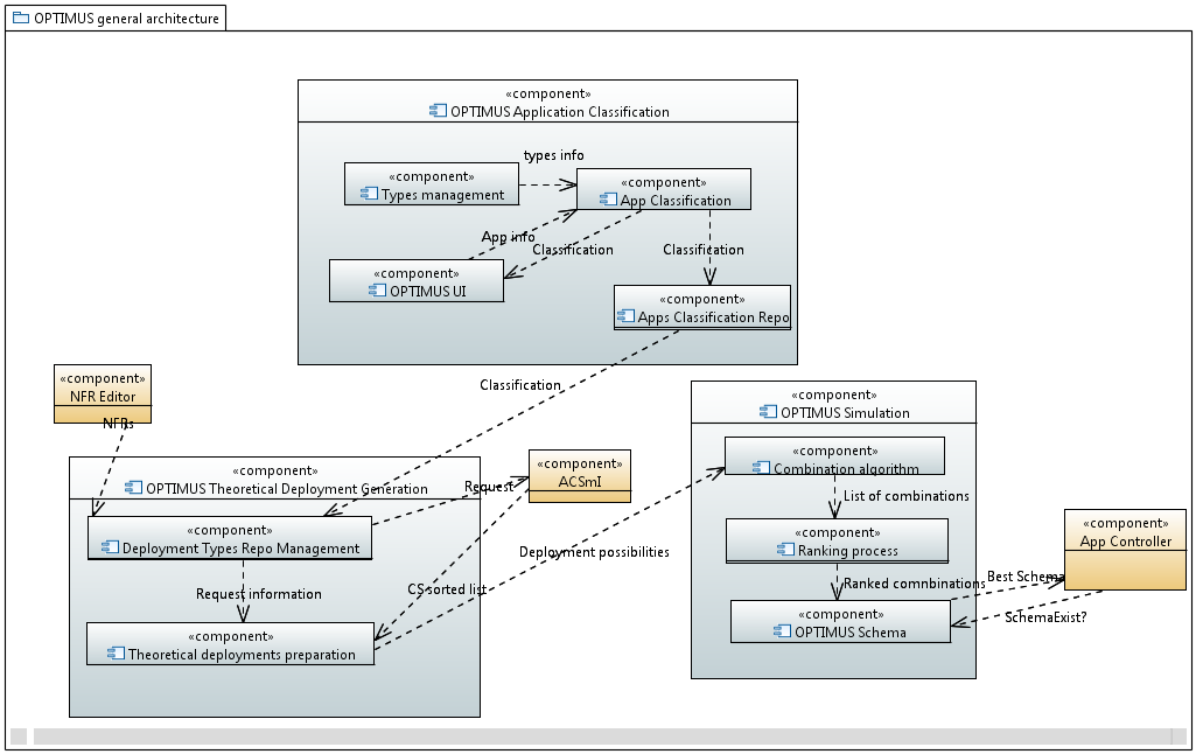

Figure 5. OPTIMUS High level architecture

The architecture of M12 OPTIMUS prototype will not include all the components of the general architecture, but just the ones that are involved in accomplishing the requirements planned for this release. The Figure 6 depicts the architecture of the M12 prototype.

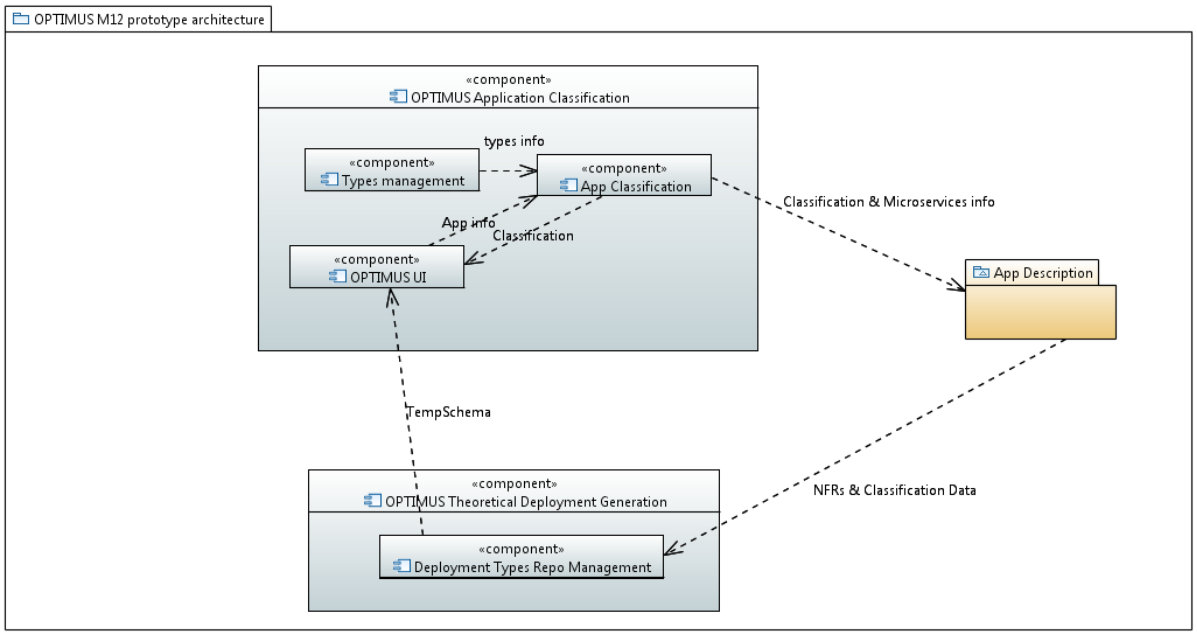

Figure 6. OPTIMUS M12 prototype architecture

2.2.2 Components description

The main components detailed in the OPTIMUS general architecture are represented in Figure 7.

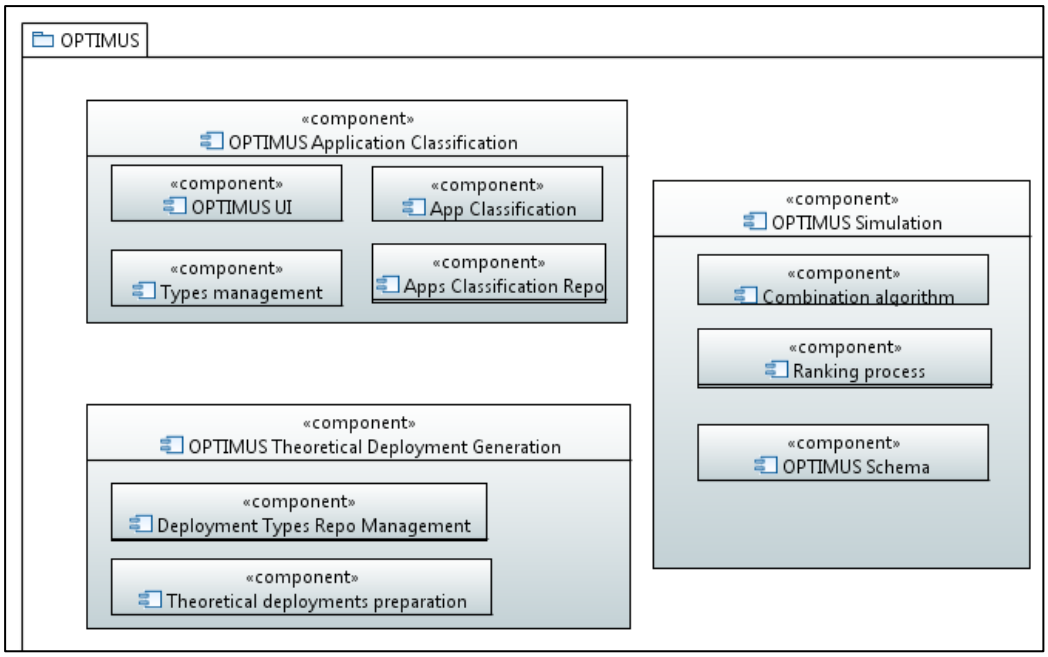

Figure 7. OPTIMUS component diagram

Application classification

The Application Classification subcomponent interacts with the developer through the *OPTIMUS UI*, from where he provides the information about the multi-cloud application that OPTIMUS needs to classify each of the microservices of the multi-cloud application. The *App*

Classification process will match the information stored about the **types of microservices**[1] (part of *Types management*) and the characteristics associated to each of the multi-cloud application microservices.

The output of the classification will be stored into the *Apps classification Repo*, physically located in the App Description JSON file.

The Types management subcomponent will manage the system knowledge about the Cloud Services that the microservices are going to need to be deployed.

More details about the classification process can be found on the DECIDE deliverable D3.4 Initial profiling and classification techniques [3]

Theoretical deployment generation

Once the classification is made, knowing the type of microservice OPTIMUS is capable to work out the Cloud Services that are suitable for the microservice. The *Deployment Types Repo Management* manages the equivalence among the microservices types and the CSPs in which they could be theoretically deployed.

Taking into account the classification, the characteristics of each microservice and the NFRs associated to them by the developer, the *Theoretical Deployment Preparation* will prepare a list of requirements to invoke to ACSmI Discovery Service and obtain the Cloud Services that meet them.

Simulation

Considering the obtained group of different possibilities for the deployment, a complex algorithm will perform a combination of all these possibilities and they will be scored in a list. The first theoretical deployment in that list will be the "best schema", and if this schema has not been selected before (checking the historical repo managed by App controller), it will be presented to the developer to confirm it. Once the confirmation is made, it will be sent to the App controller.

This Simulation phase could be triggered by DECIDE ADAPT when it detects a violation of the MCSLA.

Nevertheless, for the M12 prototype, not all the components are involved in the committed functionality, and therefore the component diagram changes. The component diagram for OPTIMUS M12 prototype can be observed in Figure 8.

[1] The concept "types of microservices" can be found in the OPTIMUS architecture and in the related references to the classification process. Each of the microservices needs a Cloud Service that fulfills the requirements for deploying it properly. It is named so, **Type of microservices,** as a group of characteristics that are required from a Cloud Service to be selected and matched with a specific microservice.

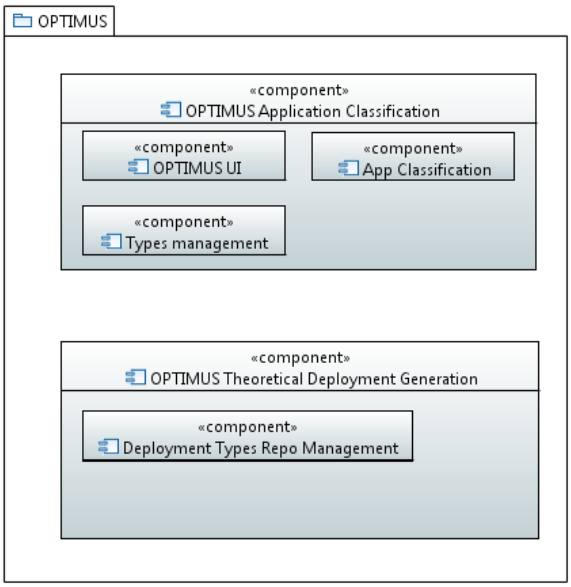

Figure 8. OPTIMUS M12 prototype component diagram

These components are a logical description of the architecture and functionality that DECIDE OPTIMUS tool provides. Nevertheless, at this stage of the development, the software is not exactly divided in elements of code.

Application classification

The subcomponent *OPTIMUS UI* will be the means for obtaining the details about the application and the microservices, in this prototype a few of them will be requested, for example, the name of the multi-cloud application, the name of each microservice that it is composed of, the detachable resource associated to it if any with its DB characteristic associated or not. The result of the classification process made by *App Classification* using the information managed by *Types management* will be presented to the developer in the form of a combo (element to show a list of values) with the value of this first version classification. The developer will be able to change this value.

More detail about the classification process and the planned values for it can be found on the DECIDE deliverable D3.4 Initial profiling and classification techniques [1].

Theoretical deployment generation

In this prototype, this module will create a message with the information provided by the modules that takes place before. That is the type of the microservice, the group of Cloud Services over which they could be deployed (managed by *Deployment Types Repo Management*), and the NFR that are stored in the Application Description JSON file.

The types of microservices (groups of Cloud Services) at this moment are “COMPUTING”, “Storage” and “Storage. DB”. To the group of COMPUTING will be associated the Virtual Machines and Containers services, and to the Storage group, all the services that provide Storage (space to store data) fully managed by the providers. This storage can be in a DB.

The information that will be presented in the Simulation tab to the developer, not as the result of the simulation but as the first step of that process, is the JSON file at that moment.

2.2.3 Technical specifications

The DECIDE OPTIMUS M12 prototype has been developed in the form of an Eclipse plugin, and with the structure of a multipage editor. So the tool has to be installed in an Eclipse framework for its use by the developer.

The multipage editor consists of three tabs. The first of them contains the *Application Description* JSON file, and will reflect the changes that the developer makes using the other tabs. The second tab is for the classification, where a group of objects can be added for each of the microservices that the multi-cloud application has.

For developing the graphical objet related to the OPTIMUS UI (multipage editor) it has been used the WindowBuilder eclipse plugin, developed to create Java GUI applications by dragging and dropping elements from a palette onto a design surface, in this case the tabs of the multipage editor.

The structure of the developed plugin has four (4) basic elements and each of them is a java class element.

- *optimus.java*: it is the main point to manage the whole OPTIMUS editor. It creates the tabs, names them and manages when there is any change among the different pages.
- *Classification.java*: it manages the appearance of the Classification tab and the processes assigned to each element on it.
- *Microservice.java*: it is the element that is responsible for creating the group of objects to gather the information about one microservice. Each time the developer pushes the "Add microservice" button, the same group of objects will be presented for him to fill them in the corresponding information.
- *Simulation.java*: it manages the Simulation tab. The optimus element will create this tab and will include the results of the simulation, that is, the best schema for the deployment.
- *OptimusContributor.java*: general tasks for basic actions related to the multipage editors

3 Delivery and usage

3.1 Package information

The structure of the OPTIMUS package in eclipse has the following structure:

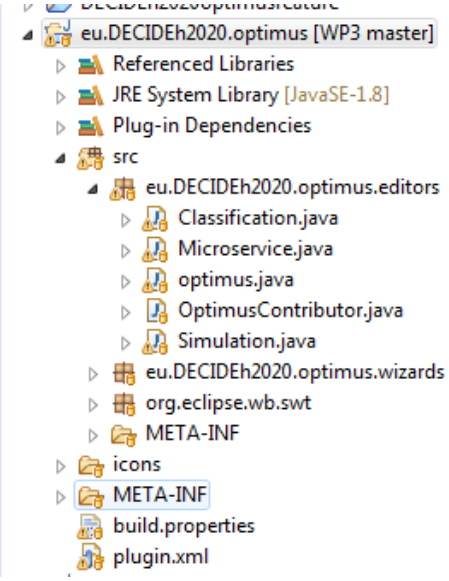

Figure 9. Eclipse Source folder structure of OPTIMUS component**.**

The package eu.DECIDEh2020.optimus.editors contains the source code for the whole OPTIMUS tool. There are included the several java classes that form the plugin.

For future versions, it has been decided that the simulation process will consist in a REST service that could be invoked from different places of the code. In the case of OPTIMUS tool, the button "Simulate" would invoke the service.

3.2 Installation instructions

As the DECIDE OPTIMUS tool is an Eclipse project and plugin, the installation requires the eclipse IDE.

The software needed for running OPTIMUS classification M12 prototype is available in this url:

https://git.code.tecnalia.com/DECIDE_Public/DECIDE_Components/tree/M12/OPTIMUS

For running OPTIMUS classification tool M12 prototype, via Eclipse java project:

a. Start eclipse IDE (eclipse oxygen) with the eu.DECIDEh2020.optimus project imported.
b. Once the project appears as a project in the Project Explorer tab, Run As an eclipse application.

3.3 User Manual

The developer creates a new OPTIMUS file clicking right mouse button and selecting New → Other -

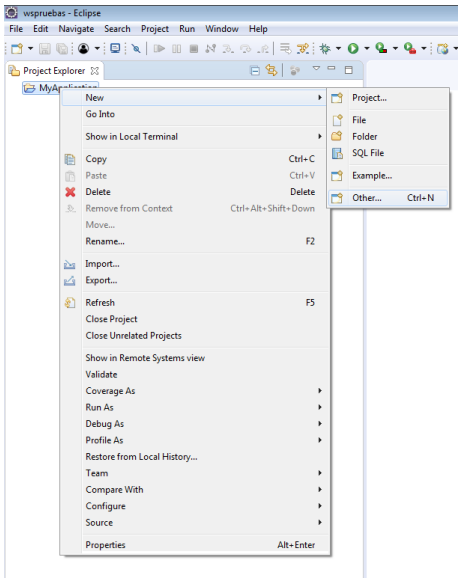

Figure 10. Step 1: Creation of a new file.

In the next window, select OPTIMUS Editor File from the OPTIMUS Wizards option

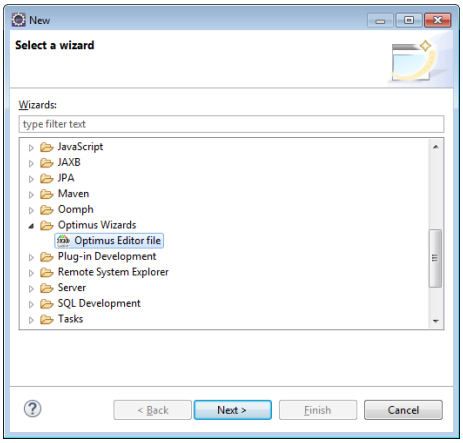

Figure 11. Step 2: Selection of the OPTIMUS Editor file.

Then, indicate the name and the container where it will be located.

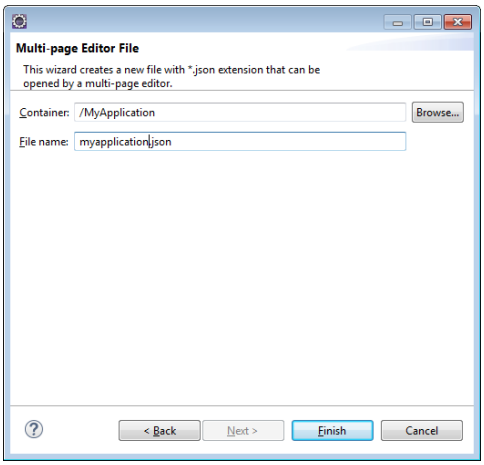

Figure 12. Step 3: Selection of the OPTIMUS file's name and container.

Open the file created, with the OPTIMUS Editor. So the extension JSON will be associated to this editor.

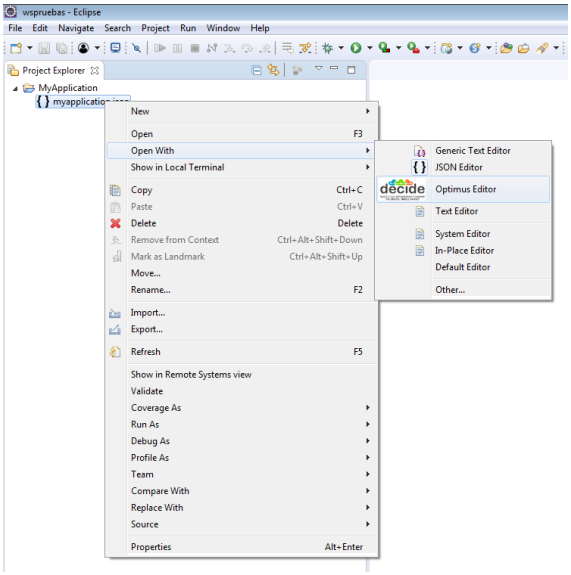

Figure 13. Step 4: Association of the Optimus Editor to the new file.

At this moment, the initial content of the file is shown in the first tab of the editor.

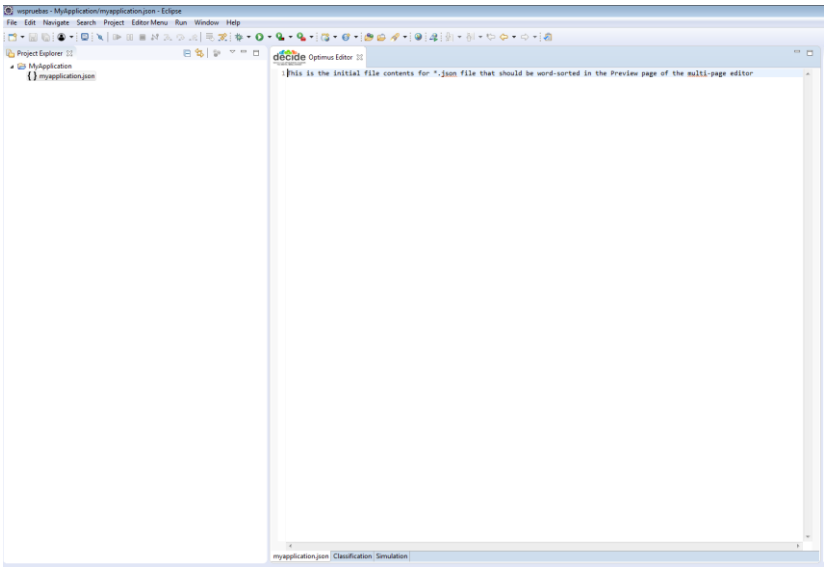

Figure 14. Step 5: Initial new file.

The second tab is the Classification tab, where the developer will enter the data about the application.

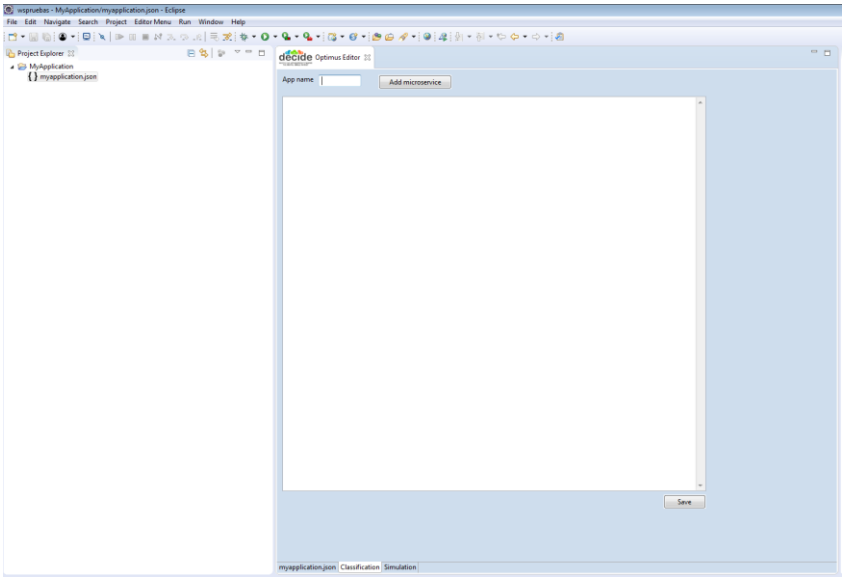

Figure 15. Step 6: Classification tab.

The last tab is the corresponding to the Simulation process.

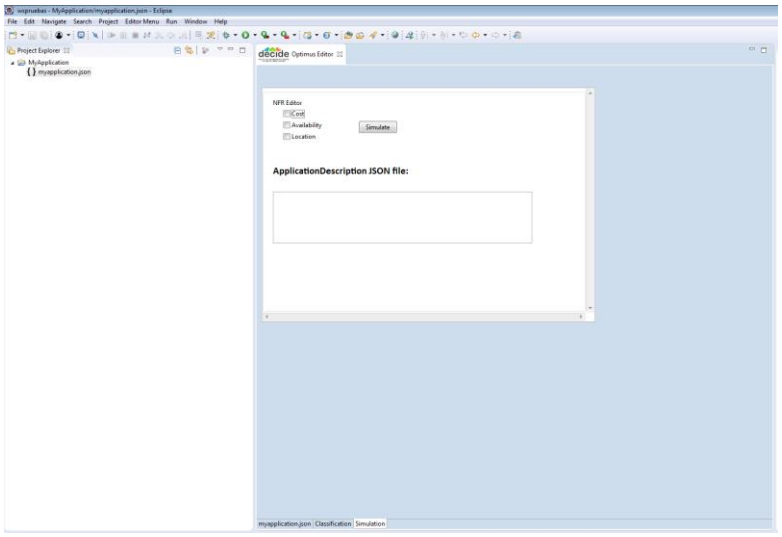

Figure 16. Step 7: Simulation tab.

For classifying an application, fill in the name of the application, and push the "Add microservice" as many times as microservices the application has. The combos to the right of the microservice name and the Detachable resource are the result of the classification process, and are based on the gathered information about them.

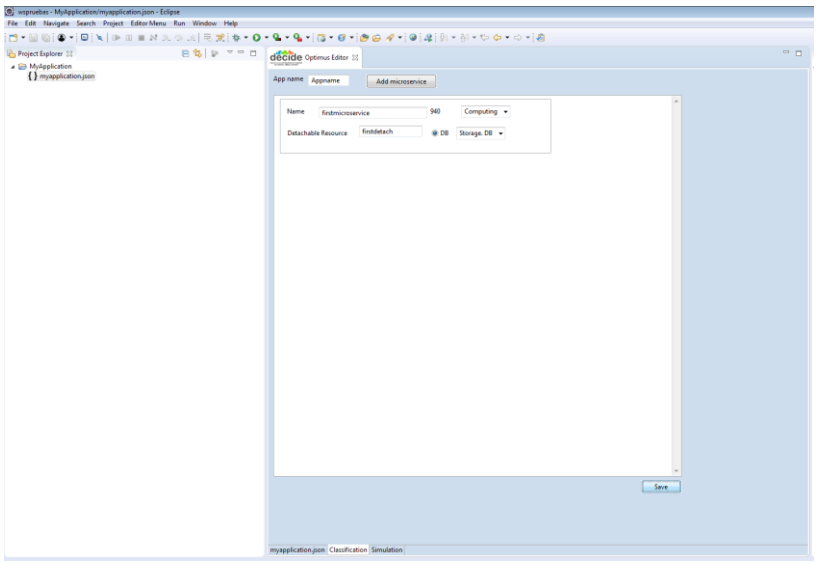

Figure 17. Step 8: New microservice in the Classification tab.

Once the Classification tab is completed, push the "Save" button and check the information in the first tab. This information will be also saved to disk.

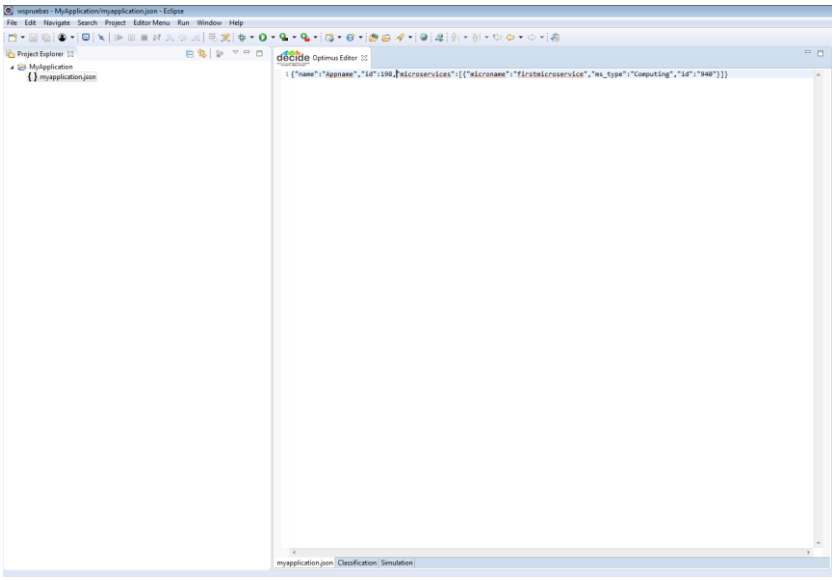

Figure 18. Step 8: Application Description saved.

In the Simulation tab, this prototype permits to assign the NFRs for the application. And pushing the "Simulate" button shows the new state of the Application Description JSON file, with the NFRs assigned.

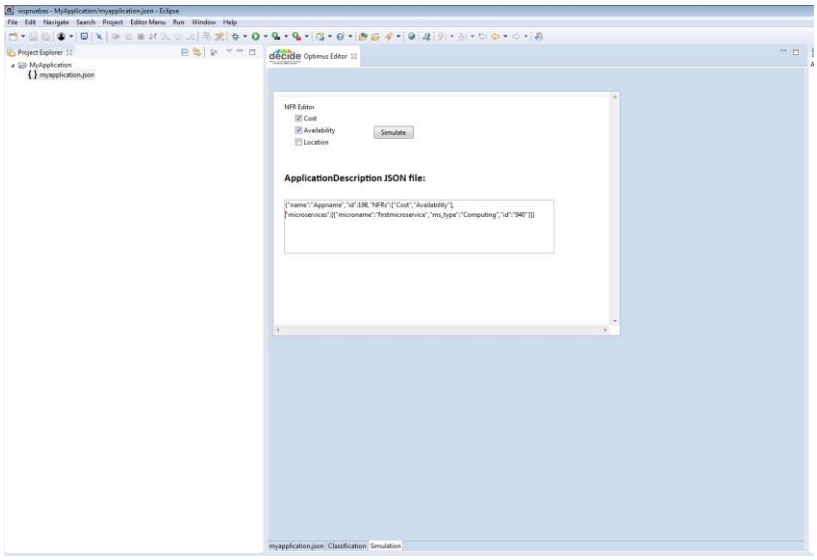

Figure 19. Step 9: Simulation result.

If the "Save" button in the Classification tab is pushed, the first tab will show the new Application Description JSON file.

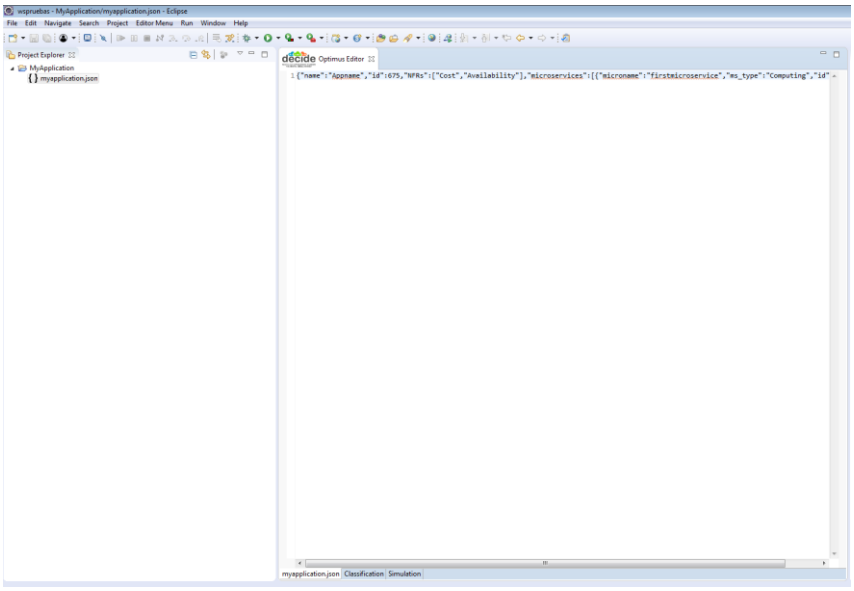

Figure 20. Step 10: New Application Description saved.

3.4 Licensing information

The information about the license under the software will be distribute, has been placed at the header of all the code files (*.java files).

These headers are composed of the following lines:

```
/*******************************************************************************
* Copyright (c) 2017 Tecnalia.
*
* This program and the accompanying materials are made
* available under the terms of the Eclipse Public License 2.0
* which is available at https://www.eclipse.org/legal/epl-2.0/
*
* SPDX-License-Identifier: EPL-2.0
* Contributors (in alphabetical order):
* Gorka Benguria                              Tecnalia
* Iñaki Etxaniz                               Tecnalia
* Juncal Alonso                               Tecnalia
* Leire Orue-Echevarria                       Tecnalia
* Maria Jose Lopez                            Tecnalia
* Marisa Escalante                            Tecnalia
* Initially developed in the context of DECIDE EU project www.DECIDE-h2020.eu
*******************************************************************************/
```

3.5 Download

The eclipse java source code of the OPTIMUS project is available in the DECIDE open git repository, accessing this URL:

https://git.code.tecnalia.com/DECIDE_Public/DECIDE_Components/tree/M12/OPTIMUS

This code is the project with the source code corresponding to the DECIDE OPTIMUS classification tool M12 prototype. This project should be imported from an Eclipse IDE.

4 Conclusions

In this deliverable, the first prototype of the DECIDE OPTIMUS tool is presented. Future versions of this deliverable will include the new features that the new prototypes will provide to the developer.

The details about the functionality and the technical aspects are described in the corresponding sections as well as the manual and the instructions to test the software. These details will evolve as well as the relationships among the different DECIDE tools evolve. The new versions of this report will picture the modifications and why are they caused.

The evolution of the application description, could lead to modifications in the prototypes of all DECIDE tools and their interactions, and consequently OPTIMUS will have to adapt to the new schema. Future versions of the prototypes will cover more aspects of the general architecture, more modules will be developed and the functionality covered by OPTIMUS will be increased, especially the Simulation module.

References

[1] DECIDE Consortium, “D3.4 – Initial profiling and classification techniques,” 2017.

[2] DECIDE, “D2.1 Detailed Requirements Specification,” 2017.

[3] DECIDE, “D3.4 Initial profiling and classification techniques.,” 2017.

[4] DECIDE Consortium, "D5.1 ACSmI requirements and technical design," 2017.

[5] DECIDE Consortium, “D4.7 Initial multi-cloud application monitoring,” 2017.

[6] Windows Azure, “SLA summary for Azure services,” Microsoft, 2017. [Online]. Available: https://azure.microsoft.com/en-us/support/legal/sla/summary/. [Accessed 08 November 2017].

[7] AWS, “Amazon S3 Service Level Agreement,” [Online]. Available: https://aws.amazon.com/s3/sla/. [Accessed November 2017].

[8] JHipster, “The JHipster Registry,” 2017. [Online]. Available: http://www.jhipster.tech/jhipster-registry/. [Accessed November 2017].

[9] Angular, “Angular Docs,” 2017. [Online]. Available: https://angular.io/. [Accessed November 2017].

[10] Bootstrap , “Bootstrap: The most popular HTML, CSS, and JS library in the world.,” 2017. [Online]. Available: https://getbootstrap.com/. [Accessed November 2017].

[11] Netflix OSS, “Netflix Open Source Software Center,” 2017. [Online]. Available: https://netflix.github.io/. [Accessed November 2017].

[12] Elastic, “The Open Source Elastic Stack,” 2017. [Online]. Available: https://www.elastic.co/products. [Accessed November 2017].

[13] Docker, “Docker,” [Online]. Available: https://www.docker.com/. [Accessed November 2017].

[14] Gulp, “Gulp: Automate and enhance your workflow,” [Online]. Available: https://gulpjs.com/. [Accessed November 2017].

[15] Apache Maven Project, “Welcome to Apache Maven,” [Online]. Available: https://maven.apache.org/. [Accessed November 2017].

[16] Gradle Inc., “Gradle Build Tool,” [Online]. Available: https://gradle.org/. [Accessed November 2017].

[17] JHipster, “JHipster,” [Online]. Available: http://www.jhipster.tech/. [Accessed November 2017].

[18] AWS, “Amazon CloudWatch,” AWS, [Online]. Available: https://aws.amazon.com/cloudwatch/. [Accessed November 2017].

[19] AWS, “Amazon EC2 Service Level Agreement,” AWS, [Online]. Available:

https://aws.amazon.com/ec2/sla/. [Accessed November 2017].

[20] Rails, "Ruby on Rails: A web-application framework that includes everything needed to create database-backed web applications according to the Model-View-Controller (MVC) pattern.," 2017. [Online]. Available: http://rubyonrails.org/. [Accessed November 2017].

[21] SQlite, "SQLite Home Page," 2017. [Online]. Available: https://www.sqlite.org/. [Accessed November 2017].

[22] nginx, "nginx," [Online]. Available: https://nginx.org/en/. [Accessed November 2017].

[23] Phusion B.V, "Passenger - Enterprise grade web app server for Ruby, Node.js, Python," 2017. [Online]. Available: https://www.phusionpassenger.com/. [Accessed November 2017].

[24] Spring, “Spring Boot,” 2017, [Online]. Available: https://projects.spring.io/spring-boot/. [Accessed November 2017].

[25] Telegraf, "Telegraf is the Agent for Collecting & Reporting Metrics & Data," [Online]. Available: https://www.influxdata.com/time-series-platform/telegraf/. [Accessed November 2017].

[26] Telegraf, “GitHub telegraf/plugins/inputs/sqlserver,” [Online]. Available: https://github.com/influxdata/telegraf/tree/master/plugins/inputs/sqlserver. [Accessed November 2017].

[27] Yeoman, “Yeoman: The web's scaffolding tool for modern webapps,” [Online]. Available: http://yeoman.io/. [Accessed November 2017].

[28] Webpack, “Webpack Module Bundler,” [Online]. [Accessed November 2017].