MULTICLOUD APPLICATIONS TOWARDS THE DIGITAL SINGLE MARKET

Deliverable D3.9

Final DECIDE OPTIMUS

Editor(s):	María José López
Responsible Partner:	TECNALIA
Status-Version:	Final – v1.0
Date:	31/05/2019
Distribution level (CO, PU):	CO

Project Number:	GA 726755
Project Title:	DECIDE

Title of Deliverable:	D3.9 Final DECIDE OPTIMUS
Due Date of Delivery to the EC:	31/05/2019

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

Abstract:	This software deliverable comprises the final OPTIMUS simulation engine. This deliverable is the result of T3.2 and T3.3. The software will be accompanied by a Technical Specification Report
Keyword List:	Simulation, classification, Eclipse IDE, plugin, java, Algorithm, Genetic
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	23/05/2019	Version ready for internal review	TECNALIA
V0.2	27/05/2019	Editorial changes from internal review	FHG
V0.3	28/05/2019	Modifications attending the internal review	TECNALIA
V1.0	30/05/2019	Ready for submission	TECNALIA

Table of Contents

List of Figures

List of Tables

Terms and abbreviations

ACSmI	Advanced Cloud Service meta-Intermediator
CS	Cloud Service
DB	Database
UI	User Interface
JSON	JavaScript Object Notation
NFR	Non-Functional Requirements
VH	Violation Handler

Executive Summary

This document contains the technical description of the DECIDE OPTIMUS tool, in its both versions plugin, web application as well as their server component, in its final version. The DECIDE OPTIMUS offers to the developer the five best possible deployment schemas for the multi-cloud application based on the classification of the microservices that compose the application, the defined Non-Functional Requirements (NFRs), and the cloud services handled by ACSmI (Discovery).

The new contribution to this document is the description of the DECIDE OPTIMUS UI, the web application integrated in the DECIDE DevOps framework that complements the Eclipse plug in and that has been newly developed for this final version. OPTIMUS UI connects to the OPTIMUS simulation service the same way the Eclipse plug in does and offers the best results to the developer for the multi-cloud schema selection.

During the last year the decision was taken to include this flavour of DECIDE OPTIMUS, through such an UI and integrated in the DECIDE DevOps framework, in order to increase the usability of DECIDE as it allows the DevOps teams to follow the whole DECIDE workflow without leaving the DevOps Framework.

Further to the description of the functional and technical aspects of OPTIMUS, this document includes sections on how to install and use DECIDE OPTIMUS, and the license under it is published.

The general architecture and design of DECIDE OPTIMUS tool is described in this document, appending, amending and extending the description already offered in previous deliverables D3.7 [1] and D3.8 [2]. While the main architecture and technical description from the Eclipse plug in version of OPTIMUS has not been changed, except for the new functionalities integrated, the OPTIMUS UI version has been completely developed from scratch in this iteration. This content is completely integrated in the text of the document. The general requirements and functionalities of OPTIMUS can be found in D3.4 [3].

This M30 version of the tool includes:

- New functionalities added to the previous prototype.
- Improvements to the existing functionalities in the M24 prototype
- DECIDE OPTIMUS UI, integrated in the DevOps Framework allowing the same functionality as the DECIDE OPTIMUS plugin tool.

1 Introduction

1.1 About this deliverable

This document presents the functionalities, design and development of the DECIDE OPTIMUS tool in its final version. The functionalities are as planned, and the only modifications that can be made in the future are those that arise from the integration and use cases testing processes.

1.2 Document structure

The global architecture and the functionalities covered by this tool, as well as all the instructions for installing the software and using it, are described in this deliverable.

The sections included in it cover both parts of the DECIDE OPTIMUS tool, the eclipse plugin and the DevOps Framework module.

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, including the classification and OPTIMUS editor (eclipse plugin and web application) and the REST service generated with Swagger [4]. The requirements that the tool covers, the architecture and the description of its main components. Also, the technical development aspects and how OPTIMUS fits in the global DECIDE architecture are included.
3. The delivery and usage section, about how to install it, how to use it and any licensing information to run the prototype.
4. Conclusions

2 Implementation

2.1 Functional description

DECIDE OPTIMUS is aimed at providing the best possible application deployment schemas, based on the non-functional requirements set by the developer and the requirements of the multi-cloud application, automating the provisioning and selection of cloud services offering for multi-cloud applications. This functionality of the tool has been set since the beginning of the project.

The DECIDE OPTIMUS tool consists of three separate parts, the first one covers the classification process through a local eclipse plugin installation, the second one has the same functionality as the plugin but integrated as a module of the DECIDE DevOps Framework, and the third one can be invoked by that plugin or by the DECIDE Framework as it is a REST [5] service by which the simulation process can be launched.

Functionalities:

The main functionalities of the DECIDE OPTIMUS are:

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

   For this purpose, the profiling of the microservices of the multi-cloud applications is a way to match the characteristics of those microservices with the group of Cloud Services features where they will be deployed.

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

   This final version presents the UI for introducing 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, if this resource access to a DB or not, and some characteristics of the microservice. The classification associated is presented in a combo with the value of it, allowing to the developer to change this value.

2. **Theoretical deployment generation.** When the classification is made, and the NFRs informed by the developer, OPTIMUS prepares a request to invoke ACSmI Discovery and obtains the cloud services that fulfil the requirements of the microservices for their deployment.

   This request is composed of generic Cloud Services and the list of resources that the microservices need. This functionality requires interacting with the ACSmI API to obtain the list of Cloud Services that meet the criteria requested.

   The request to ACSmI discovery is built and performed based on the classification of the microservice, the characteristics associated to it and the NFRs for the application level. The aggregate and disaggregate values needed for calculating the level of fulfilment of the NFRs are implemented as part of the simulation algorithm.

3. **Simulation**. The combination of the different optimized possibilities of deployment, considering the theoretical and individual deployment possibilities for each microservice and the list of cloud services (from ACSmI Discovery) that suit them, is ranked to select the five best of them and are presented to the developer to confirm the first of them or to select another one of that list of five. This process is implemented using a genetic algorithm for combining and optimizing problems, detailed in D3.6 [6].

The Schemas include information about the id of the cloud service in the ACSmI registry and the ids of the microservices that are set to deployed on it. This information is stored into the historical repository managed by the App Controller.

The Eclipse plugin and the OPTIMUS UI web launch the simulation process for the current application and the result with the five best deployments schemas is shown to the developer to select one of them. Then he or she can store it in the application description, and therefore in the schemas history, or launch again the simulation process, changing some data about the application.

It is the responsibility of the developer to upload the application description JSON file to the repository where it belongs, before and after launching the simulation, when using the eclipse plugin OPTIMUS tool. For that, the eclipse framework where the application description JSON file has been created or cloned, allows to commit the file once the information needed for the simulation is completed. If the developer needs to use another DECIDE tool inside the DevOps framework, he should upload again the JSON file. The OPTIMUS UI part allows to save directly the modifications in the repository associated to the application.

It has been implemented the interface for the ADAPT Violation Handler tool, described in D4.3 [7] section 3.1.3, which allows launching a simulation through the DevOps Framework when a violation occurs.

Requirements:

The requirements satisfied by this final version are described in the Table 1.

The five first rows are related to the classification process, performed by the Eclipse plugin and web part of OPTIMUS, the rest of them are about the simulation process.

Table 1. Requirements covered by DECIDE OPTIMUS tool.

Req. ID	Description	Status	Requirement coverage
WP3-PROFI-REQ1	Load/read information about the application (components).	Satisfied	The tool reads the information stored in the application description about the microservices and the NFRs.
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 compare it with the information about the (component) application.	Satisfied	The classification is made considering the information about the microservices that the tool knows, from the DECIDE.json file.
WP3-PROFI-REQ3	Request the developer to confirm the classification	Satisfied	The confirmation takes place when the developer saves the result of the classification into the application description JSON file.

Req. ID	Description	Status	Requirement coverage
WP3-PROFI-REQ4	Store the information about classification made.	Satisfied	The information is written in the DECIDE.json file.
WP3-PROFI-REQ5	Mechanisms for updating the "stereotypes of the components" information	Satisfied	Through the properties file, the tool allows to indicate the different classifications available and their correspondence to the Cloud Services classes
WP3-OPTI-REQ1	The OPTIMUS tool shall be capable of reading the non-functional characteristics of the app from NFR DB	Satisfied	The information in the DECIDE.json file is read.
WP3-OPTI-REQ2	The OPTIMUS tool shall be capable of reading the classification of the app (or its components)	Satisfied	The information in the DECIDE.json file isread.
WP3-OPTI-REQ3	OPTIMUS will analyse the application's NFRs 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.	Satisfied	Implemented the creation of a filter to ask ACSmI about the cloud services that fulfil the requirements, considering the NFRs, and the characteristics of the microservices.
WP3-OPTI-REQ4	For each component of the multi cloud application, OPTIMUS engine builds the theoretical composition of services needed to the best possible deployment topology	Satisfied	Obtaining the cloud service for a specific microservice is part of the optimization algorithm in the Simulation process.
WP3-OPTI-REQ5	Once OPTIMUS engine runs the simulations for each component of the multi cloud application, each of them will be ranked	Satisfied	The ranking is part of the Simulation process, and the algorithm implemented.
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	Implemented Testing integration	The NSGA-II Algorithm will search through different combinations of Cloud Services for the microservices and will find the perfect combination of them which could result in a list of five best solutions.

Req. ID	Description	Status	Requirement coverage
	and will combine the results of each of the possibilities.		
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. This will show in the UI and will require confirmation from the developer	Satisfied	The information about the selected schema is the schema itself, with the id of the cloud service where a group of microservices can be deployed, and the ids and names of that group of microservices.
WP3-OPTI-REQ8	OPTIMUS tool can 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	Implemented. Testing integration	Implemented the possibility for launching the simulation by the ADAPT Violation Handler module.
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	Implemented. Testing integration	The Simulation process optimizes the deployment schema taking into account the available characteristics of the CS.
WP3-OPTI-REQ10	DECIDE OPTIMUS [..] will provide [..] automation of the provisioning resources and deployment schemas for multi-cloud native applications	Rejected	Rejected because it is already included in WP3-OPTI-REQ4, WP3-OPTI-REQ5 AND WP3-OPTI-REQ7

2.1.1 Fitting into overall DECIDE Architecture

The global DECIDE architecture is shown in Figure 1 and it is described in the deliverable D2.5 [8]. The role of OPTIMUS in this global architecture has not changed compared to the previous version delivered in D3.8 [2].

Both OPTIMUS eclipse plugin and OPTIMUS UI web application have the same functionality and therefore are described in this section as a unique OPTIMUS.

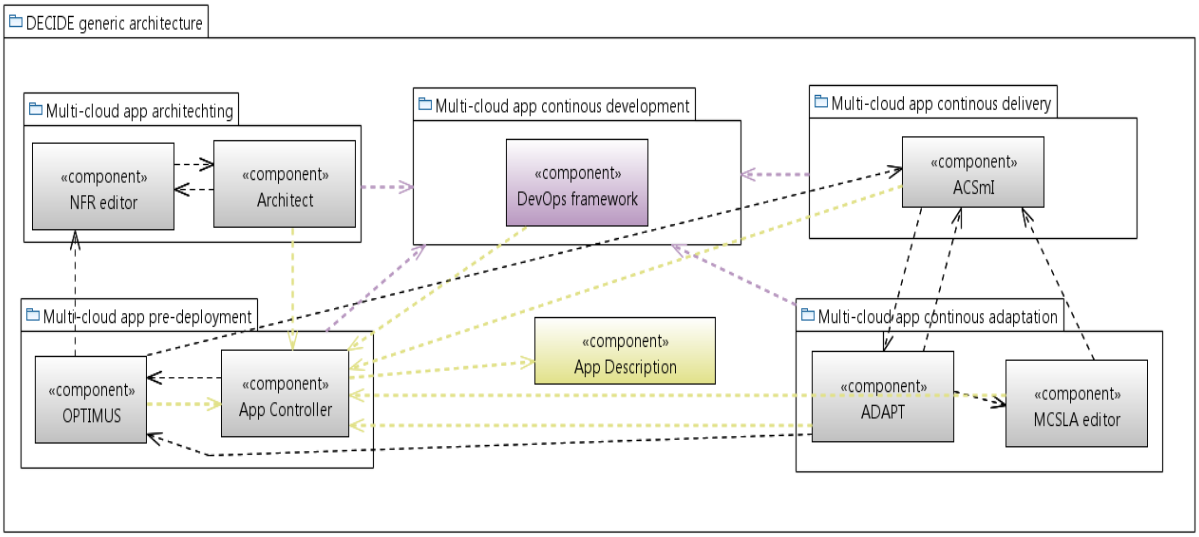

Figure 1. OPTIMUS in DECIDE architecture**.**

The interaction with OPTIMUS and the rest of the DECIDE tools remains as planned and was described in the previous deliverable [2].It is presented in the Figure 2.

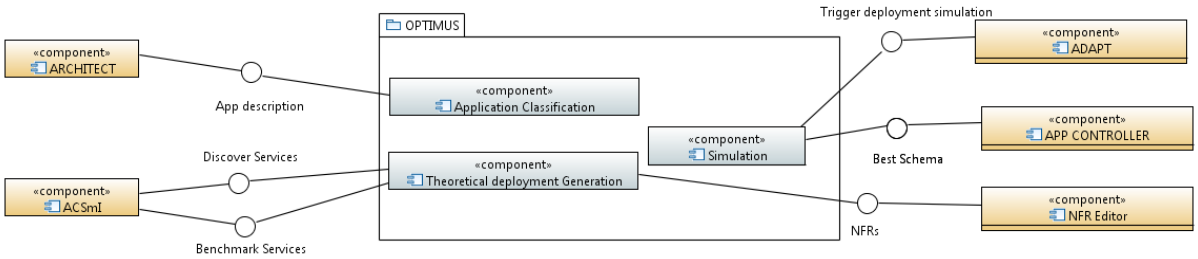

Figure 2. OPTIMUS interfaces within DECIDE framework

These interactions are as follows:

- ARCHITECT: This is the starting point to the DECIDE framework when the developer wants to use OPTIMUS tool. It provides a General Editor in eclipse with which the information of the application will be stored into the application description JSON file. Moreover, it is integrated in the DevOps framework with the same functionality.
- ACSmI: The information about the cloud services where the microservices can be deployed on, is provided by ACSmI Discovery.
- ADAPT: When a violation of the MCSLA is discovered, ADAPT Violation Handler triggers OPTIMUS to obtain a new best deployment schema.
- Application Controller: The structure of the application description JSON file is defined by the Application Controller component, as well as the operations that can be performed with it. Moreover, the deployment schema obtained by OPTIMUS Simulation is managed also through the Application Controller library when storing it into the historic repository and consulting it to avoid a deployment under the same schema.
- NFR Editor: The NFR Editor is part of the General Editor in ARCHITECT, and part of the DevOps framework. For each microservice the developer can specify Non-Functional Requirements associated to it. OPTIMUS tool uses the information about the NFRs to build the request to ACSMI Discovery and obtain the best schema for the deployment.

2.2 Technical description

In this section, the technical aspects about the development of this version of the OPTIMUS prototype are presented. It includes both versions, the eclipse plug in and the UI web app.

2.2.1 Prototype architecture

The general architecture for OPTIMUS tool is shown in Figure 3.

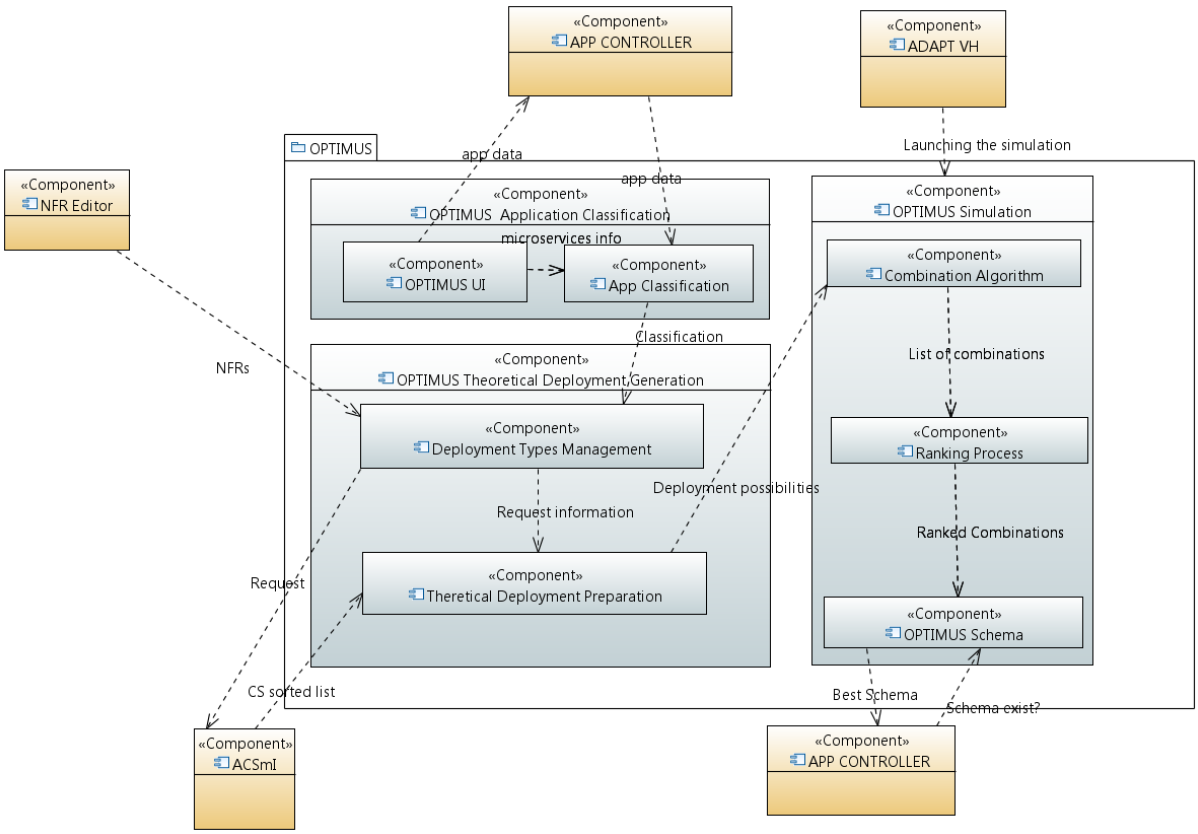

Figure 3. OPTIMUS High level architecture

OPTIMUS Application Classification component involves both parts of OPTIMUS, the eclipse plugin and the web part. Both have the same role in this architecture although they are implemented in a different way, as it can be seen in the following sections.

The rest of the components are referred in the Simulation process, being it a common part for the implementation. It has been developed as a swagger server and both classification modules, eclipse and web, call to the corresponding services from that server.

The following figure depicts the architecture of OPTIMUS UI web module. OPTIMUS UI web module is an Angular module (ngModule) that wraps different components, services and pages that are related to the OPTIMUS tab in DECIDE DevOps Framework.

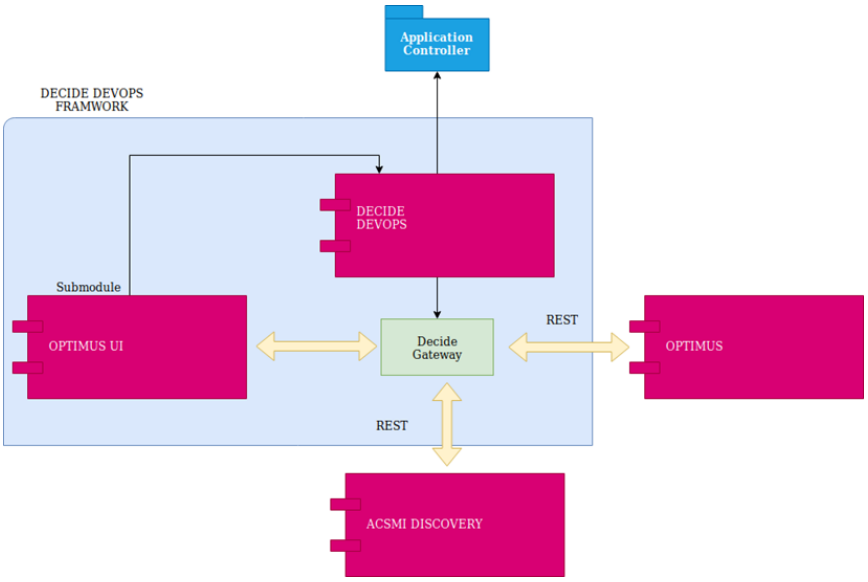

Figure 4. architecture of OPTIMUS web UI module

2.2.2 Components description

The main components detailed in the OPTIMUS general architecture are represented in Figure 5.

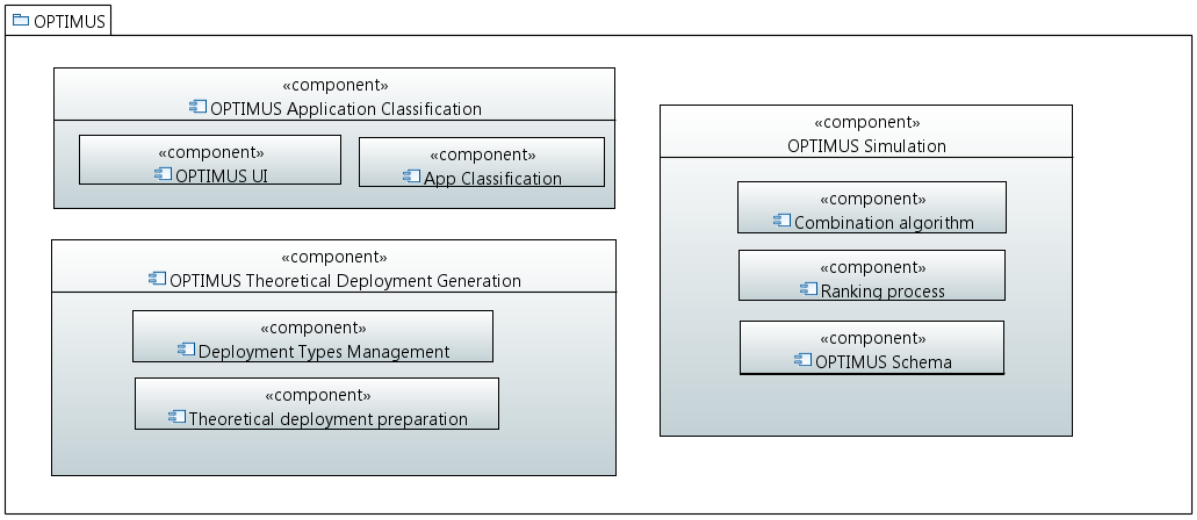

Figure 5. OPTIMUS component diagram

Application classification

The Application Classification component is presented to the developer as a tab of the OPTIMUS eclipse plugin and as a tab of the DevOps framework. Through this UI the developer provides the information about the application and the microservices which it is composed of, so as to classify each of those microservices.

The App Classification subcomponent will match the information stored about the different possible classifications and the characteristics associated to each of the multi-cloud application microservices, writing the corresponding value into the application description JSON file using the Application controller library.

The classification process is explained more in-depth in the DECIDE deliverable D3.5 Intermediate profiling and classification techniques [9]

Theoretical deployment generation

Considering the classification of each microservice, OPTIMUS can associate the cloud service class that can be candidate for its deployment. Knowing the cloud services objectives, together with the NFRs and some characteristics established by the developer, OPTIMUS builds a request to obtain from ACSmI the cloud services that fit the requirements set in that request.

Processing the ACSmI answer, storing each possibility as a structured object of information, the Theoretical Deployment preparation arranges all the input that the Simulation component needs.

Simulation

Once the request is made to ACSmI and the information about the cloud services that are suitable for the microservices, the tool analyses the level of fulfilment of this association with the requirements stablished by the developer for that microservice, including the NFRs.

The best combination of Cloud Services for the set of microservices that constitute the multi cloud application is deduced. For obtaining that optimized combination of elements, the NGSA-II Algorithm is used.

Once the five best ranked schemas have been obtained, they are presented to the developer to confirm it, and then sent the selected one to the App controller.

This Simulation phase can be also triggered by DECIDE ADAPT Violations Handler (VH) and in this case the schema is best one and it will be stored automatically in the application description JSON file.

More detailed information about the Simulation process and the algorithm implemented can be found in the D3.6 deliverable [6].

The OPTIMUS web UI module has been designed and implemented to work as an independent visual part in the web application, being modular and flexible to the data it visualizes and interacts.

OPTIMUS UI web has a strong dependency with Angular Material components, that are inside Material Module. These modules give the basic components for the construction of the UI page layout in OPTIMUS UI web part.

The created components and services are declared inside OPTIMUS Module (An Angular declaration type for modules), that allows the architecture to be better organized and clearer to understand.

The following diagram shows the components and services created in OPTIMUS web UI, and the dependency with Material Module:

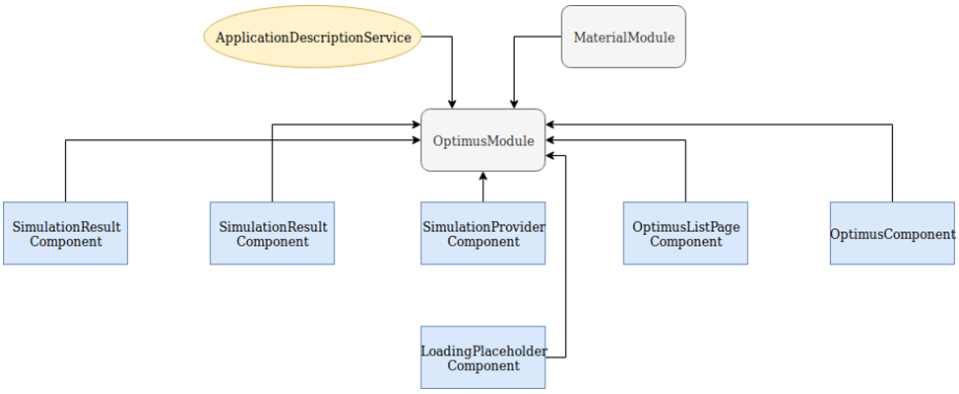

Figure 6. Components and services created in OPTIMUS UI web

There are 5 component classes and 1 injection service that are part of OPTIMUS web UI Module. The description of these modules is as follows:

1. OPTIMUSListPageComponent: this is the main page component for the OPTIMUS section in DECIDE DevOps Framework and wraps all the other components. The page is a composition of a microservices list sideview, and a main tabbed view: one tab for the microservice fields edition and another tab for the OPTIMUS simulation execution and the view of the schema results based on the criteria of OPTIMUS Simulation.

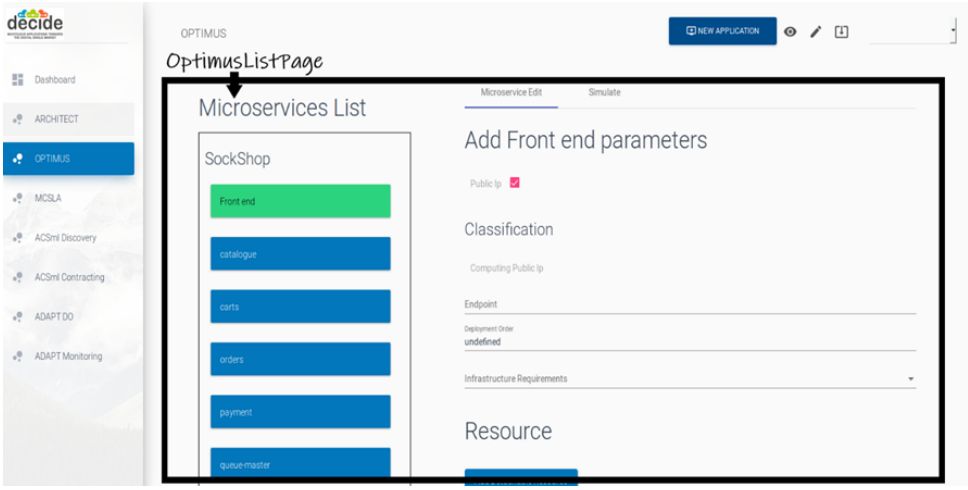

Figure 7. OPTIMUSListPageComponent

2. OPTIMUSComponent: the visible component from DECIDE DevOps Framework. Its main task is to serve as a container for the OPTIMUSListPageComponent and it subscribes to the selected application data.
3. SimulationResultComponent: the main responsibility of this component is to obtain the schema results from the OPTIMUS REST requests and to visualize it with the required fields as an animation with the JSON mapped data. This component will connect to OPTIMUS in the real scenario.
4. LoadingPlaceholderComponent: this is a non-functional and only aesthetic animated component. Its main purpose is to preview a size of square, when waiting for the microservices list request from the server.
5. SimulationProviderComponent: this component makes a request to ACSmI services, to obtain a list of the multi-cloud valid providers for the simulation. The selection of the providers is a criterion required for the OPTIMUS Simulation. This component is inserted in the simulation tab before simulation button.
6. ApplicationDescriptionService: this service has the responsibility to call the requests for the OPTIMUS simulation. There are three methods and there must be a future work testing and implementing the defined requirements
    a. getApplicationProviders: call to the providers list via REST to ACSmI Discovery.
    b. getApplicationDescription: call to the selected application and get the updated application description as JSON document. It internally calls to DecideGatewayService and connects to Application Controller.
    c. createSimulation: with the selected parameters it makes a new simulation request to OPTIMUS.
    d. getSimulationResultsSchema: makes an http request to OPTIMUS via REST and gets a JSON document with the schemas that OPTIMUS calculates (maximum 5 schemas).
    e. saveSelectedSchema: after checking the resultings schemas, the user can select the more suitable one and save it to Application Description.

2.2.3 Technical specifications

The final DECIDE OPTIMUS prototype has been developed as three separate parts:

1. DECIDE OPTIMUS Classification tool: in the form of an Eclipse plugin with the structure of a multipage editor. This part of the tool must be installed in an Eclipse framework for its use by the developer.
2. DECIDE OPTIMUS DevOps module: developed as an Angular Module. Integrated in the DECIDE DevOps Framework.
3. DECIDE OPTIMUS Simulation: swagger server that provides several REST services that allows to be invoked from different parts of other tools or software elements. This module can be triggered by the Violation Handler or launched by the DECIDE OPTIMUS DevOps module when the developer pushes the Simulation button.

2.2.3.1 DECIDE OPTIMUS Classification tool

Moreover, the plugin is executed together with the General Editor part of ARCHITECT. The implementation has been made using the extensions mechanism that allows adding tabs from a plugin to another previously launched.

The classification tool is part of the maven ARCHITECT project, as a maven module of it.

The multipage editor consists of three tabs. The first of them is part of the General Editor and contains the Application Description JSON file. It will reflect the information that the developer introduces using the other tabs. The second tab is for the classification, where a group of microservices are shown if they are in the application description JSON file. The third tab corresponds to the simulation process, from where the simulation can be launched, and its result can be seen.

For developing the graphical object related to the OPTIMUS UI (multipage editor), the WindowBuilder [10] Eclipse plugin has been used, which has been 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 five (5) basic elements and each of them is a java class element:

- *Classification.java*: it manages the appearance of the Classification tab and the processes assigned to each element on it.
- *MicroserviceClassification.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 with the corresponding information.
- *SimulateSchema.java*: it manages the Simulation tab. This OPTIMUS element will create this tab and will include the results of the simulation, that is, the best schemas for the deployment.
- *ClassificationPageBuilder*: This is the element that implements IPageBuilder and creates the additional tab for the classification. It follows the rules for working with extensions.
- *SimulationPageBuilder*: This is the element that implements IPageBuilder and creates the additional tab for the simulation. It follows the rules for working with extensions.

2.2.3.2 DECIDE OPTIMUS Web UI module

Programming Language: Typescript
Framework: Angular ^7.2.8
NPM Dependencies:

- "rxjs": "^6.4.0",
- "@angular/animations": "^7.2.8",

- "@angular/cdk": "^7.2.8",
- "@angular/common": "^7.2.8",
- "@angular/compiler": "^7.2.8",
- "@angular/core": "^7.2.8",
- "@angular/forms": "^7.2.8",
- "@angular/http": "^7.2.8",
- "@angular/material": "^7.2.8",
- "@angular/platform-browser": "^7.2.8",
- "@angular/platform-browser-dynamic": "^7.2.8",
- "@angular/platform-server": "^7.2.8",
- "@angular/router": "^7.2.8",

2.2.3.3 DECIDE OPTIMUS Simulation

On the other hand, the REST Server has been created using the Swagger specification [4] for defining the service and the format to manage it.

This specification is a JSON file from which Swagger generates the server with the REST service and the template for the implementation of the service, and the client, which will be the library for using the service from other modules or tools.

The template generated by Swagger for the server has been added to the Eclipse development framework as a Maven [11] project, so, once the lines of code needed for providing the planned functionality have been introduced, the server can be deployed in the integration framework for testing it.

The main elements developed in this project are three (3) class elements described as follows:

- *Bootstrap.java:* This is the starting point when the REST service generated by Swagger is called. It has been generated by Swagger and completed by TECNALIA.
- *SimulatorThread:* This class contains the optimization process to obtain the schema, performing the request to ACSmI discovery and returning the best schema to the Eclipse plugin by implementing an NGSA-II algorithm, or saving it into the application description JSON file in case the VH has been the service caller.
- *ApplicationApiServiceImpl:* This file gathers the different methods related to the available REST services that the server publishes. It has been generated by Swagger and completed by TECNALIA.
- *OPTIMUSProblem.*java: The class that create and define a OPTIMUS problem for being solved by the NSGA-II algorithm.

More java elements are generated by Swagger and are included in the Maven Eclipse project. They can be seen in Figure 8.

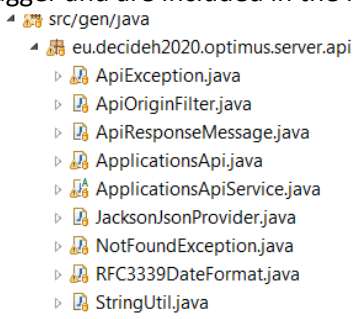

Figure 8. Generated java classes by Swagger**.**

3 Delivery and usage

3.1 Package information

The structure of the OPTIMUS plugin package in Eclipse is as follows:

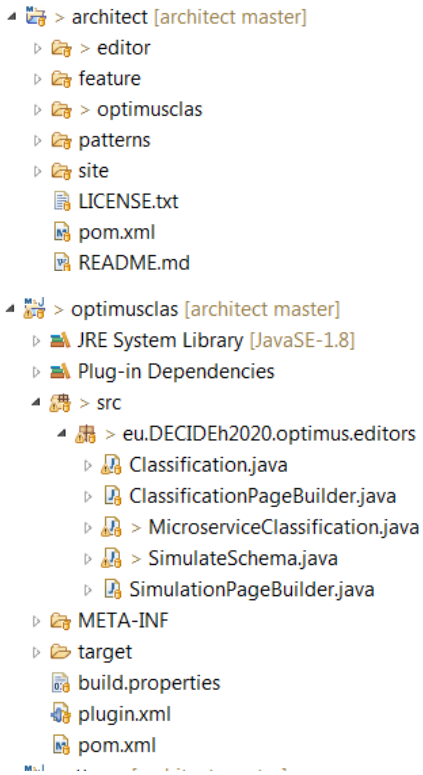

Figure 9. Eclipse Source folder structure of OPTIMUS plugin component**.**

The package *eu.DECIDEh2020.optimus.editors* contains the source code for the OPTIMUS plugin.

The server with the corresponding code for the REST services is a separate Maven project and has the following structure:

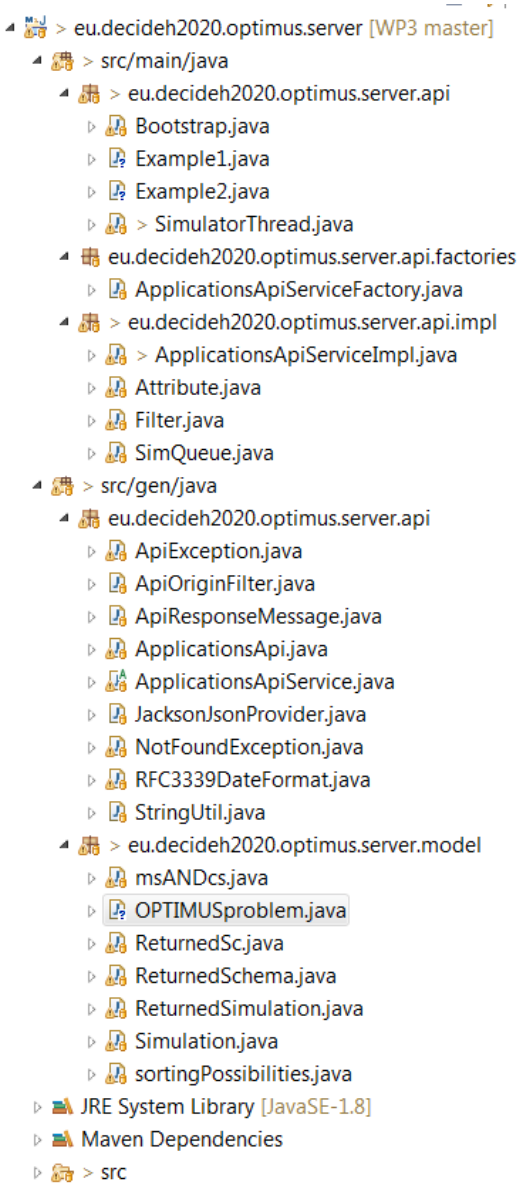

Figure 10. Eclipse Source folder structure of OPTIMUS server component**.**

The different packages depicted above contain the following elements:

- (main) *eu.decideh2020.optimus.server.api*: Where the main elements of the server implementation are placed, the starting point and the code for the simulation.
- (main) *eu.decideh2020.optimus.server.api.factories*: A generated file for starting and creating the service.
- (main) *eu.decideh2020.optimus.server.api.impl:* Elements needed for the implementation of the main java files placed in (main) *eu.decideh2020.optimus.server.api* package.
- (gen) *eu.decideh2020.optimus.server.api*: Elements to be internally used, generated by swagger.
- (gen) *eu.decideh2020.optimus.server.model*: Some data structures generated by swagger and completed by TECNALIA.

The following picture shows (partially) the file structure in the code from DECIDE OPTIMUS Web UI integrated in the DevOps framework.

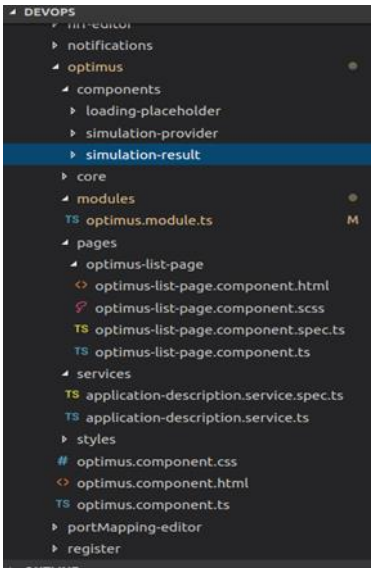

Figure 11. Structure of the DECIDE OPTIMUS Web UI code.

3.2 Installation instructions

The DECIDE OPTIMUS tool has different parts of code, each of them with a different method for installing it.

3.2.1 DECIDE OPTIMUS Classification tool

The installation of the eclipse plugin requires the eclipse IDE.

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

https://git.code.tecnalia.com/DECIDE_Public/DECIDE_Components.git

For running OPTIMUS Classification tool, via Eclipse java project:

a. Start Eclipse IDE (Eclipse Oxygen)
b. Clone the repository indicated above
c. Import the following projects:
    i. Architect and the rest of projects that depend of it. (as a maven project)
    ii. AppController (as a maven project)
    iii. Cloud-patterns (as a maven project)
d. Clean and install cloud-patterns and AppController projects.
e. Clean package the architect project.
f. Run configurations as an Eclipse Application.

It is pending for the integration to be done, so a proper update site method will be implemented at the end.

3.2.2 DECIDE OPTIMUS Simulation

For installing and running the server the following software is needed:

- Docker to create the container.

To get the server up and running it must to follow these steps:

1. Download the source code from the DECIDE repository.

```
git clone
https://git.code.tecnalia.com/DECIDE_Public/DECIDE_Components.git
```

2. Go to the folder where the docker file is located:

```
cd <OPTIMUS server folder>\
eu.decideh2020.int.optimus.server.src.dvp\src\main\docker
```

   i. Build the docker image with the following arguments:

```
docker build -f /docker.optimus.server -t
tecnalia/eu.decideh2020.int.optimus.server
```

   ii. Run the docker image with the following arguments:

```
docker run -d --restart=always -p 8090:8090 --name
eu.decideh2020.optimus.server
tecnalia/eu.decideh2020.optimus.server
```

3.2.3 DECIDE OPTIMUS DevOps module (OPTIMUS UI)

Docker and docker compose must be installed in your OS. (Tested version Docker-CE 18.09)

The steps for the correct installation and running are the following:

- "git clone git@git.code.tecnalia.com:decide/devops.git"
- "cd devops"
- "docker-compose build"
- "docker-compose up"
- Connect locally to http://localhost:4200 from the browser and go to OPTIMUS section.

Moreover, OPTIMUS Web UI part will be accessible through the DECIDE DevOps Framework, so it is not needed a specific installation.

3.3 User Manual

3.3.1 DECIDE OPTIMUS Classification tool

Once the developer runs the plugins, he must create a new DECIDE JSON file, containing the application description, licking right mouse button and selecting New → Other

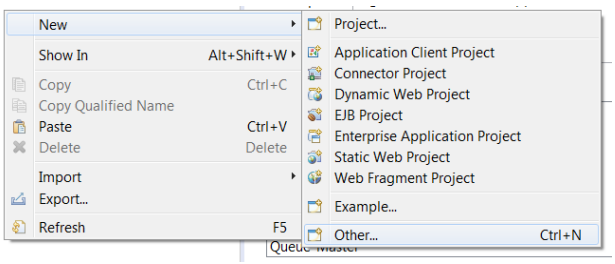

Figure 12. Creation of a new file.

In the next window, select "DECIDE project" from the DECIDE Wizards option

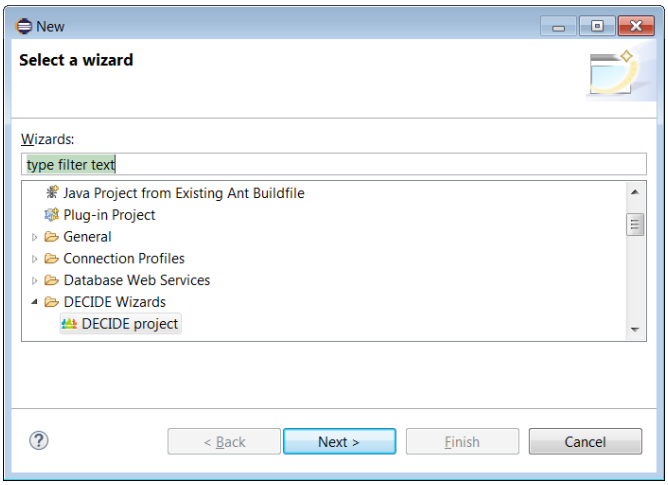

Figure 13. Selection of the DECIDE Editor file.

Then, indicate the Project name, the local folder, and the information about the git repository where the JSON file is stored for "Clone Remote Repository" option.

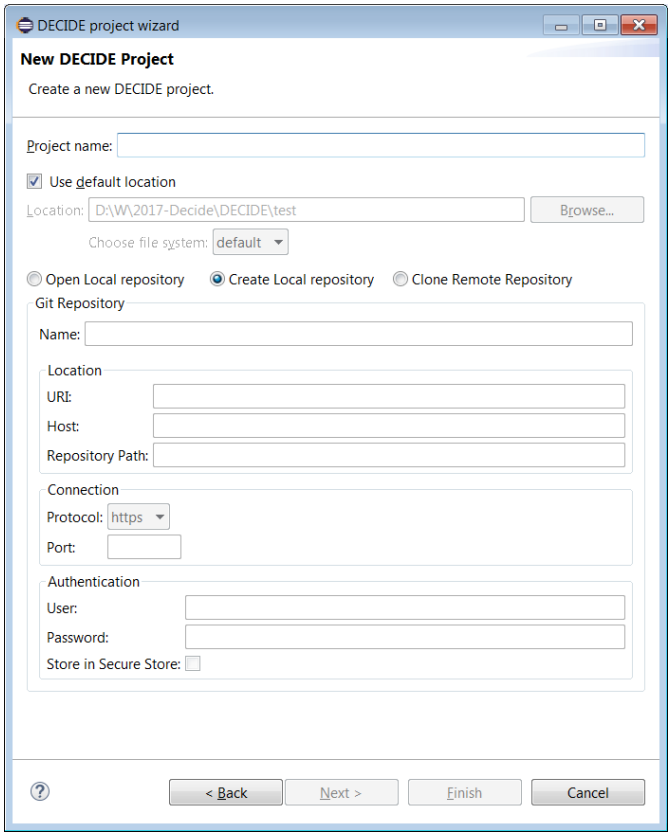

Figure 14. Selection of the DECIDE JSON file's information.

When selecting the "Finish" button, the initial content of the file is shown in the first tab of the editor.

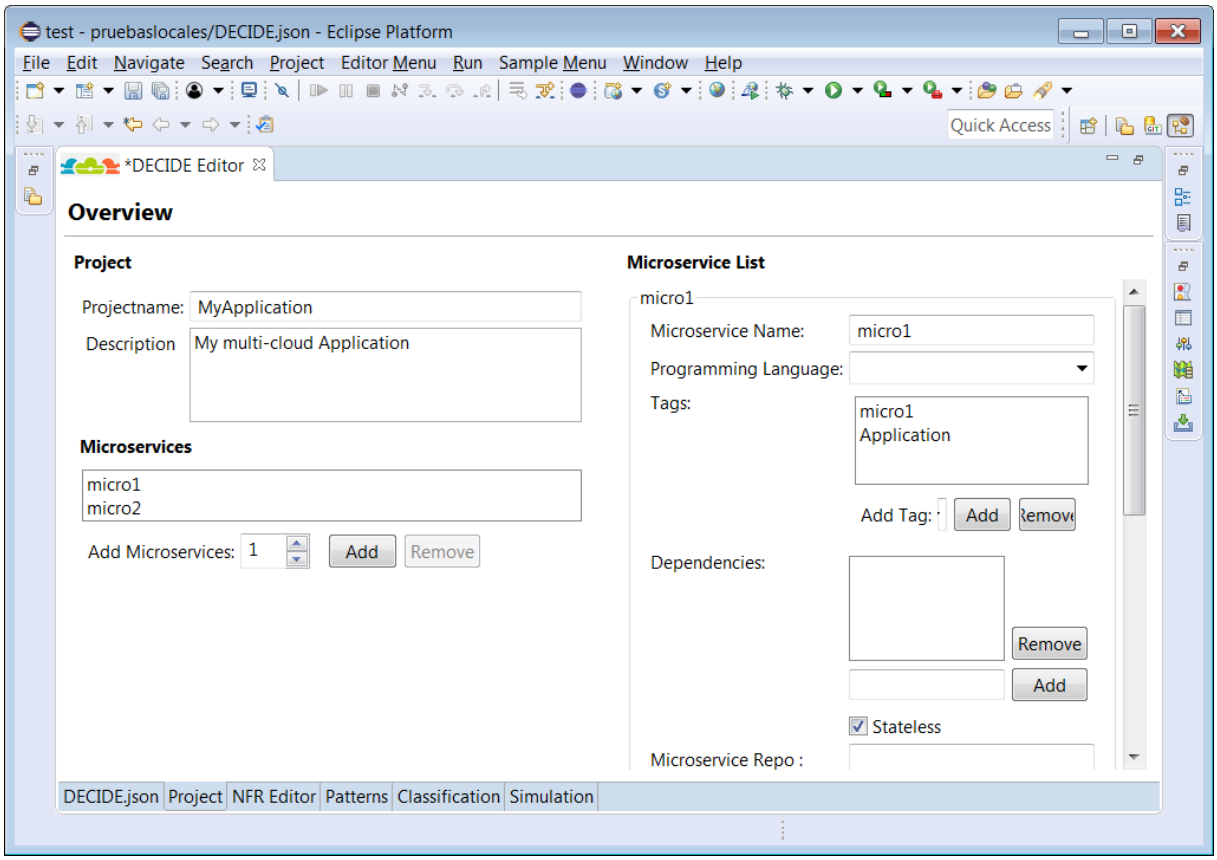

Figure 15. Initial new file.

The first three tabs correspond to the current version of the General Editor and show the JSON file in raw (DECIDE.json tab), the information as a UI (Project tab) and the information about the NFRs (NFR Editor tab). The forth tab is the corresponding to the Patterns editor. These tabs are not part of the OPTIMUS tool and they are implemented under the ARCHITECT tool.

OPTIMUS and ARCHITECT are two tools that can be executed in an Eclipse framework, so they need first a General Editor to allow the developer to introduce the general data about the Application, as well as the DevOps framework does.

The first OPTIMUS tab is the Classification tab. For example, consider that the new file has two microservices, and the classification tab reflects this situation:

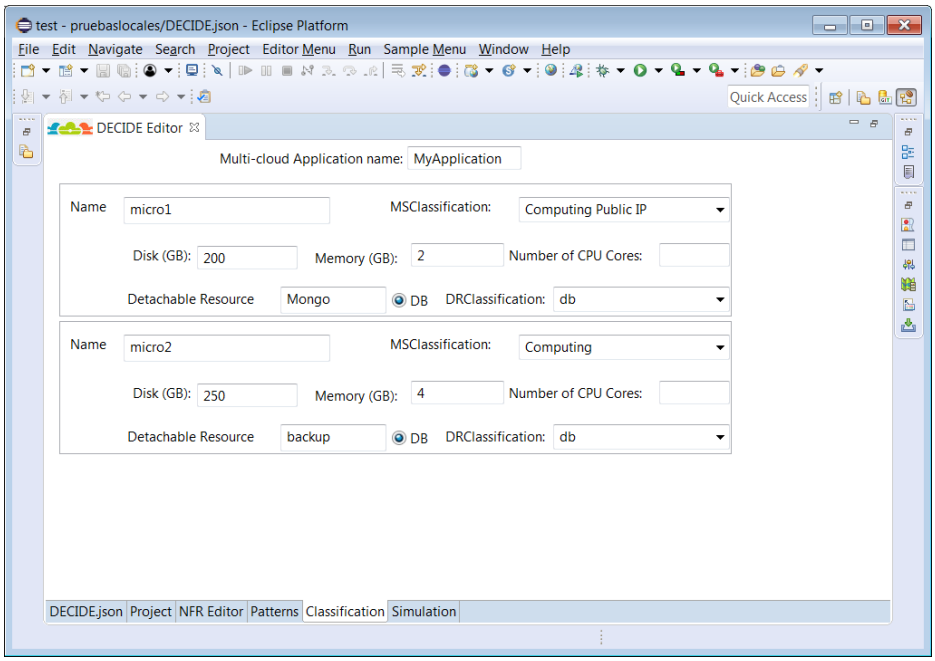

Figure 16. Classification tab.

The simulation tab is the area where the developer, once the information about the microservices has been fulfilled, can launch the simulation for that specific application. The developer also can assign a preferred provider to force DECIDE Simulation to obtain only Cloud Services from that provider.

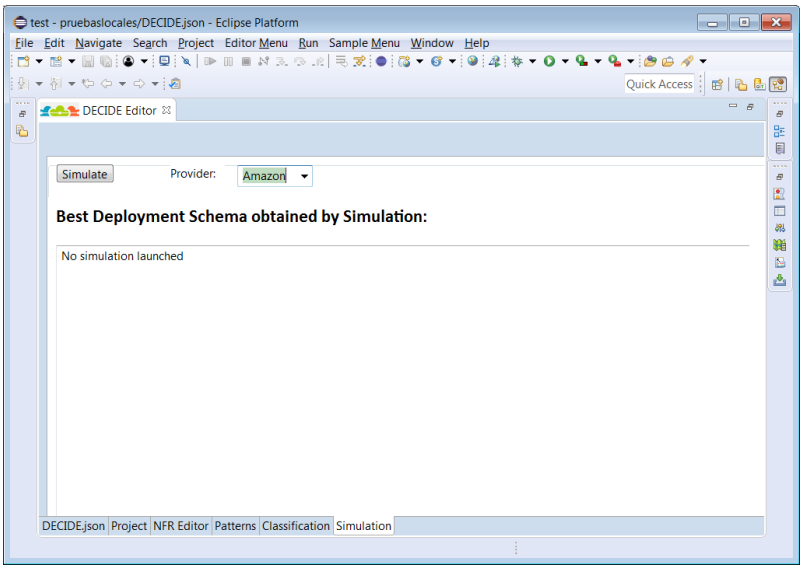

Figure 17. Simulation tab.

For classifying a microservice in the Classification tab, this prototype assigns the "Computer" value by default allowing the developer to change it to "Computing Public IP" if he considers it more appropriate.

Each microservice can have a detachable resource associated to it, of which the developer has to introduce the name and select the DB aspect when corresponds. In that moment, the value of its classification will change, and it will show the value "db", otherwise the value will be "storage".

More microservices can be added using the General Editor Tab, but the detachable resources can only be specified through the Classification tab. These elements are considered elements associated to a

principal microservice, and in the simulation they will be deployed in the same Cloud Service as its main microservice.

The result of the simulation appears in the tab, allowing to the developer for selecting one of the five Deployment Schemas presented.

The elements of each Deployment Schema are:

- index: internal index to identify a specific association of CSid and microservices Ids. It is useful for the further deployment process. Not appearing in this tab.
- List of elements composed by:
    - Group of microservices id and name: The ids of the microservices that should be deployed in the Cloud Service mentioned bellow and their names.
    - Cloud Service id: The id of a selected Cloud Service. The information about this CS can be found in the ACSmI Discovery registry.

3.3.2 DECIDE OPTIMUS web UI module

Once docker-compose is running with all the services up and healthy, OPTIMUS web UI can connect to OPTIMUS. From the main page in the left side view click “OPTIMUS” element.

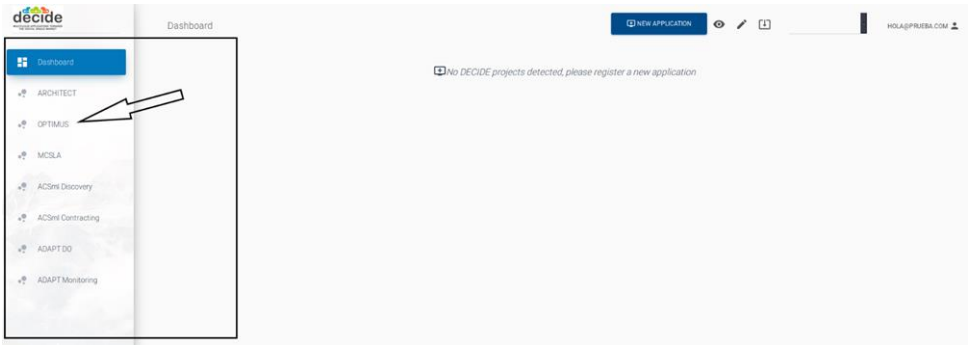

Figure 18. DECIDE OPTIMUS Web UI main page.

Once we click OPTIMUS, OPTIMUS web UI will appear with four main components:

- The microservices list.
- The central view with two tabs.
- The first tab is the editor of fields for the selected microservice.

The second tab is the simulation executor.

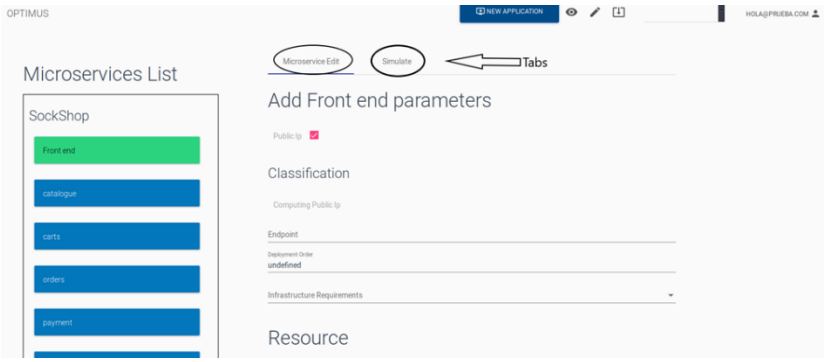

Figure 19. DECIDE OPTIMUS web UI classification page.

The form field for the selected microservice shows the required parameters. The user can check “Public IP” as true or false for the classification.

For the “Infrastructure Requirements” the user can include the properties Disk, Ram and numberCores.

Inside “Resource” area, a button with “Add detachable resource” permits to create new detachable resources, as many as needed.

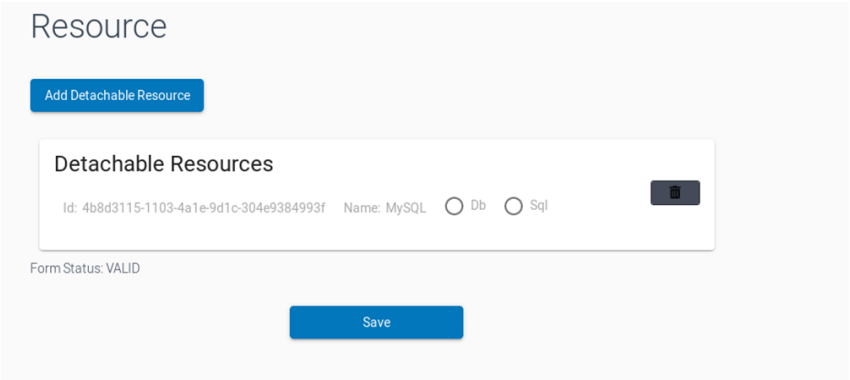

Figure 20. DECIDE OPTIMUS Web UI Detachable resources.

If the user wants to delete the created detachable resource, it can do that by using the trash icon button in each right side of the resource.

After each microservice field satisfies the user requirements and the resources are created, the user selects the simulation tab. It shows a selectable provider list with a simulate button:

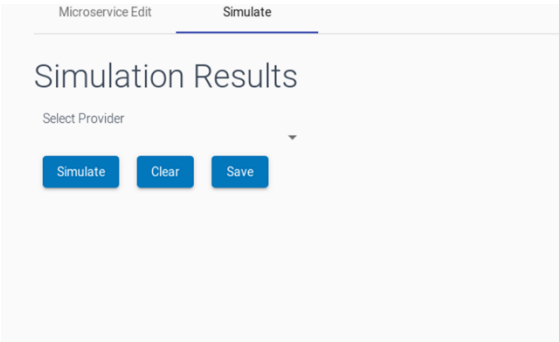

Figure 21. DECIDE OPTIMUS Web UI Simulation.

The provider selection field offers the options shown in the Figure 22

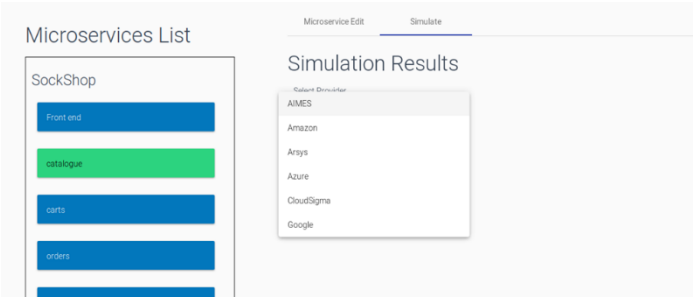

Figure 22. DECIDE OPTIMUS Web UI Preferred provider

After the selection of the provider the user can launch the simulation that will connect to OPTIMUS (backend), and get the optimized results after a successful operation:

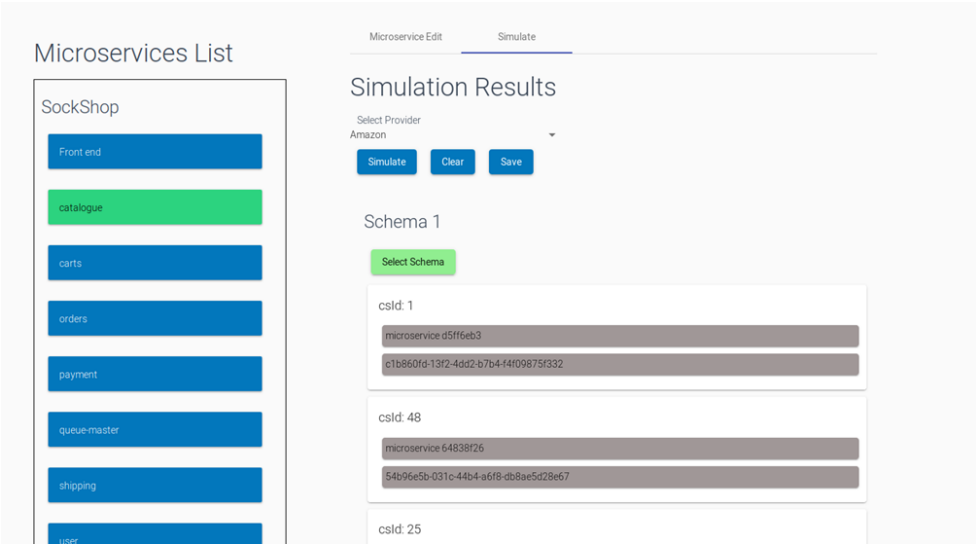

Figure 23. DECIDE OPTIMUS Web UI Simulation

The new results section is designed to show up to 5 schemas showing the CSIds and microservices names and their ids.

The "Select Schema" button serves as a selection operation, which will be finally added to the application description. After selecting the schema, it will be outlined and ready to save.

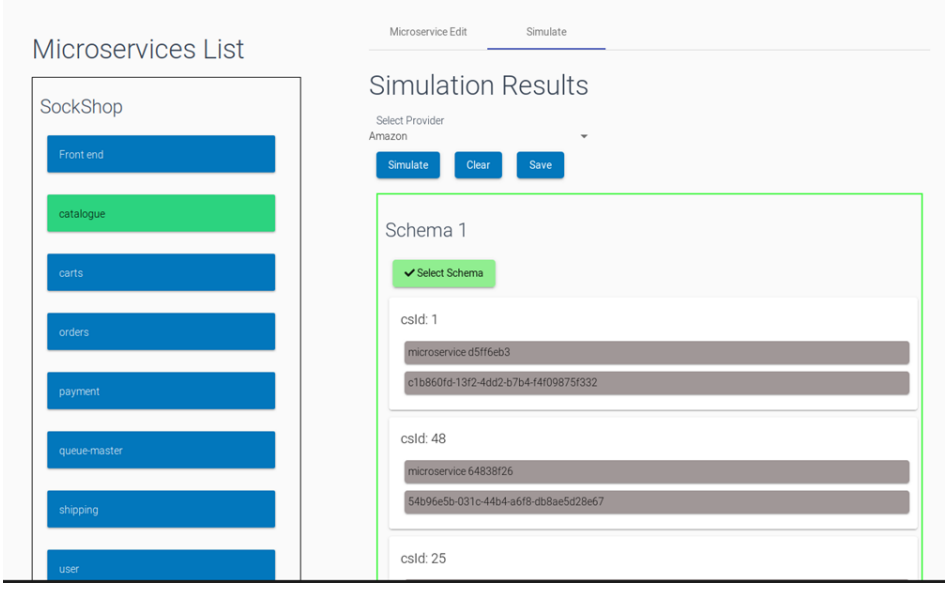

Figure 24. DECIDE OPTIMUS Web UI Deployment Schema Selection

3.4 Licensing information

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

These headers are composed of the following lines:

```
/*********************************************************************
* Copyright (c) 2019 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):
* Alberto Molinuevo                        Tecnalia
* 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 (eclipse plugin and server) is available in the DECIDE open git repository, accessing this URL:

https://git.code.tecnalia.com/DECIDE_Public/DECIDE_Components.git

OPTIMUS can be found under this URL: https://git.code.tecnalia.com/DECIDE_Public/DECIDE_Components/tree/master/OPTIMUS. The current version has the tag M30. This code is the project with the source code corresponding to the DECIDE OPTIMUS final tool.

To download DECIDE OPTIMUS DevOps module code, please also refer to https://git.code.tecnalia.com/DECIDE_Public/DECIDE_Components/tree/master/OPTIMUS.

4 Conclusions

This deliverable presents the evolution of the DECIDE OPTIMUS tool and its status as final prototype.

There are no more deliverables describing more versions of the tool, however, when the integration is finished, and the use cases approved, some changes can be made. The same situation can occur with the application description and consequently OPTIMUS will have to adapt to the new schema.

The most important achievements in OPTIMUS in this version are on one hand, the implementation of the algorithm, as described in D3.6 [6], for establishing the different possibilities of combinations of the available cloud services, to present the five best schemas, considering the NFRs and the characteristics of the microservices involved, and on the other, the implementation of the web application version mimicking the functionality of OPTIMUS eclipse, which allows for the provision of two possibilities of the same tool.

5 References

[1] DECIDE Consortium, “D3.7 Initial DECIDE optimus,” Bilbao, 2017.

[2] DECIDE Consortium, “D3.8 Intermediate DECIDE OPTIMUS,” Bilbao.

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

[4] S. Software, “SWAGGER,” 2018. [Online]. Available: https://swagger.io/. [Accessed November 2018].

[5] W3C, “Web Services Architecture,” 2004. [Online]. Available: https://www.w3.org/TR/2004/NOTE-ws-arch-20040211/#relwwwrest. [Accessed November 2018].

[6] DECIDE Consortium, “D3.6 Final profiling and classification techniques,” Bilbao, 2019.

[7] DECIDE Consortium, “Final DECIDE ADAPT Architecture,” 2019.

[8] DECIDE Consortium, “D2.5 Detailed Architecture v2,” 2018.

[9] DECIDE Consortium, “D3.5 – Intermediate profiling and classification techniques,” 2018.

[10] I. Eclipse Foundation, “WindowBuilder,” [Online]. Available: https://www.eclipse.org/windowbuilder/. [Accessed 2018].

[11] T. A. S. Foundation, “Apache Maven project,” 2018. [Online]. Available: https://maven.apache.org/. [Accessed 2018].

[12] I. Weaveworks, “Sock Shop: A Microservices Demo Application,” 2017. [Online]. Available: https://github.com/microservices-demo/microservices-demo/blob/master/internal-docs/design.md. [Accessed November 2018].

[13] DECIDE Consortium, “D3.7 Initial DECIDE OPTIMUS,” 2017.