isep Instituto Superior de Engenharia do Porto
DEPARTAMENTO DE ENGENHARIA INFORMÁTICA

Engenharia de Domínio

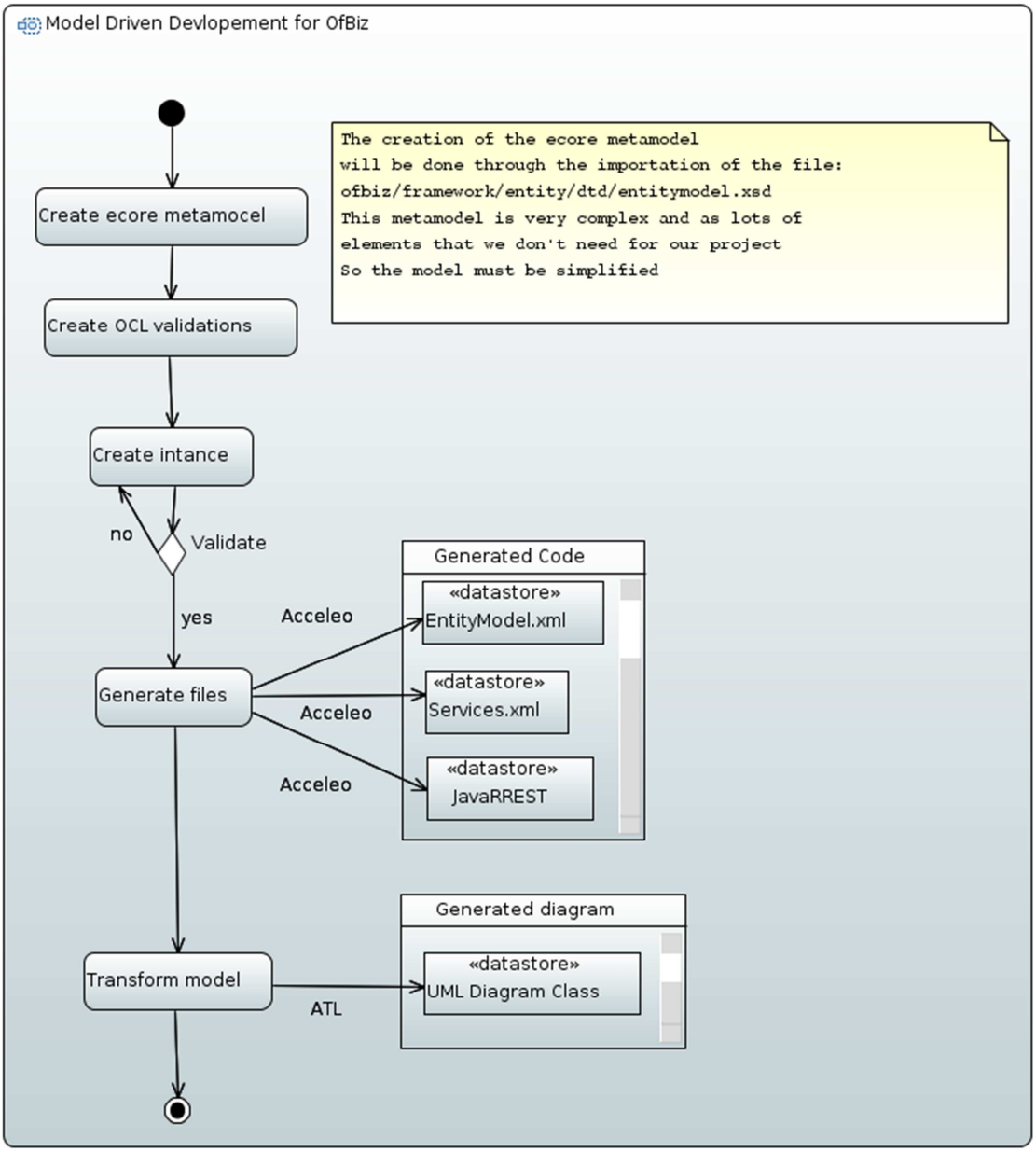
MYRECIPE – TECHNICAL REPORT

EVARISTO FIGUEIREDO – 1010836
NELSON LOPES – 1160098
GROUP NO. 101 - 2016-2017

CONTEÚDO

THEORETICAL GLOBAL PLAN

1. ACTIVITY DIAGRAM ILLUSTRATING THE ENVISIONED SOLUTION



1.1. DESCRIPTION

We will start by creating the ecore diagram according to the specifications described in the nest section. The meta-model will only contemplate the classes required for the concrete example MyRecepe and not for all the classes reported in the XSD file. Because it is not possible to infer all the information for the services from the information related with the entities the meta-model will also contain information about the services.

After the creation of the ecore meta-model, we will insert validation conditions in OCL, like restricting the value of the data types allowed for the table fields.

With the instance of the model validated, we will then use the option “create dynamic instance” to generate de XMI file that allows the creation of the model for the MyRecepe project.

To achieve the automatic generation of the xml files for the entities, for the services and for the REST service we will create an ACCELEO project and in the MTL file we will describe the structure for the files.

To finalize the project we will create an ATL project and use the meta-model we have created in the first step along with the meta-model for UML (that already exists), to transform the model for the MyRecipes into a class diagram in UML according to the rules that we will insert in the ATL file of this project.

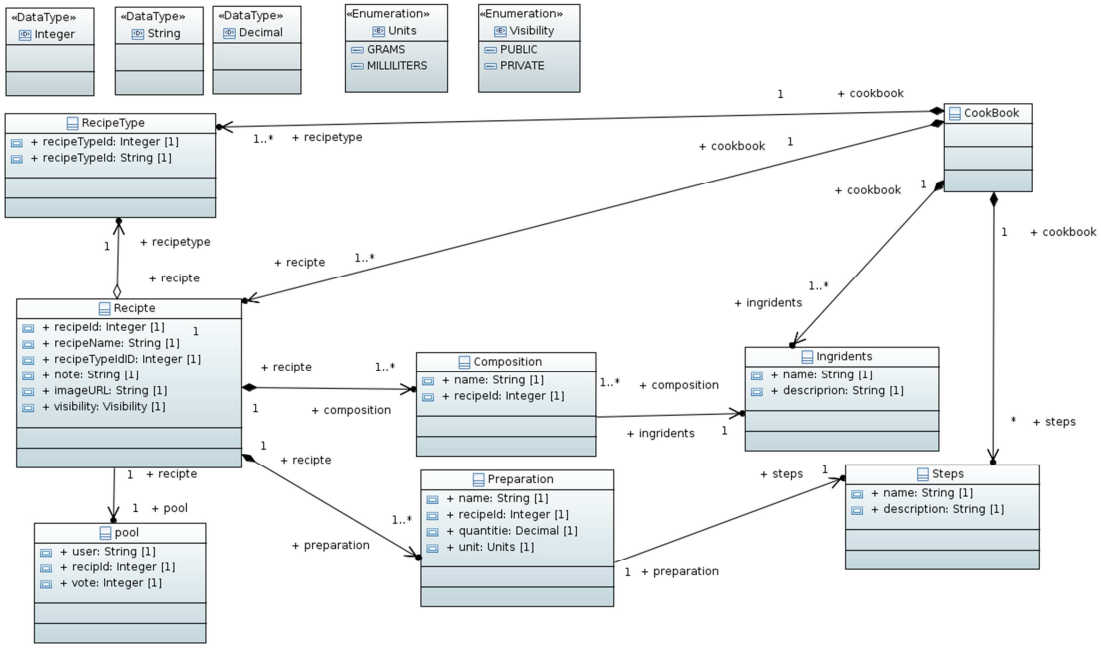
2. MAJOR CONCEPTS TO BE USED IN THE METAMODEL

Table 1 - Concepts in the meta-model

Elements			Attributes	Description
Level 1	**Level 2**	**Level 3**		
ofbiz				Root element to aggregate services and entities
services				Main element that aggregates all information about the services
			xmlns:xsi	Type of XSD
			xsi:noNamespaceSchemaLocation	Location of the XSD file
	description			String describing the service
service				
			name	String with the name of the method
			engine	Java
			location	String with the path to the Java file where the method can be found
			invoke	String with the name of the method
attribute				Parameters to be passed to the service
			name	String with the name of the parameter
			type	String with type of the parameter
			mode	IN/OUT
			optional	TRUE/FALSE
	description			Description of the service
entitymodel				Main element that aggregates all information about the entities
			xmlns:xsi	Type of XSD
			xsi:noNamespaceSchemaLocation	Location of the XSD file
	title			String identifying the entity model
	description			String describing the entity model
entity				Entity representing a table
			entity-name	String with the name of the entity

Elements			Attributes	Description
Level 1	**Level 2**	**Level 3**		
			package-name	String with the name of the java package where this entity is incorporated
			Title	Description of the entity
	field			Columns for the table
			Name	Name for the column
			Type	Type of the data that can be inserted in the column
	prim-key			The primary Key for the table
			Field	The name of the field that will be used as primary key
	relation			Foreign keys for the table
			Type	Cardinality of the relation
			fk-name	String with name of the relation
			rel-entity-name	String with the name of the table which this table relates to
		key-map		
			field-name	Foreign name of the primary key of the second table in the relation
			rel-field-name	?????

2.1. ENTITIES



All the information needed to generate the file **entitymodel.xml** can be obtained directly from the ecore meta-model as we can see in table 1.

2.2. SERVICES

To declare the services in the file **services.xml** it is also possible to get the information directly from the model, because the meta-model allows for the creation of a model with information about the entities and about the services-

2.3. REST COMPONET

OFBizRESTApplication.java,

In the first file it is only needed to add a line for the new component: classes .add(RecipeResource .class);, the variable parte in this line is the name of the component and we can get the name of the class Recipe to build it.

RecipeResource.java

We will have to create these methods: create_Recipe, getAll_RecipeTypes, create_RecipeType, getAll_Ingrediants, create_Ingridient, getAll_Steps, create_step, create_Vote.

The information we will need to generate this code is the name of the fields for this entity and the name of the method. The first piece of information can be found in the classes “attribute” of the model and the second one is registered in the attributes for the class “service”.

- **create_recipe**

This method will allow the register of the information about the recipe. To do this this method will need to provide information for the recipe. This information will be passed as parameters.

- **create_Vote**

Because we don't have to implement the alteration of recipes we only need this method for the pool.

- **getAll_RecipeTypes, create_RecipeType, getAll_Ingrediants, create_Ingridient, getAll_Steps, create_step**

Because we assumed that the ingredients, the steps and RecipeType are contained in different tables and can be reused in different recipes, when the user proceeds with introduction of the information for the recipe we will have to present to him, the list of elements that already exist as ingredients, steps and RecipeType. Then he can choose one of the existing or create a new one.

3. DRAFT FOR THE ECORE DIAGRAM