DESIGN AND MODELING OF SOFTWARE SYSTEMS
2018-2019

Workflow-DSL

A DSL for modeling workflows and generating process management applications

Author(s):

José Juan Peña Gómez

DNI: 21715188Z

Group ID:5

Table of Contents

1 Description of the application domain and the project goals

The domain of the project is about that business who need to implement quickly a web site for his business process. But it can be use too for other institutes that need web services like an hospital, or for some services that need to manage forms and data like matriculate in some sites.
So basically, provide a quick solution for implement basics web services.

2 Abstract syntax of Workflow-DSL

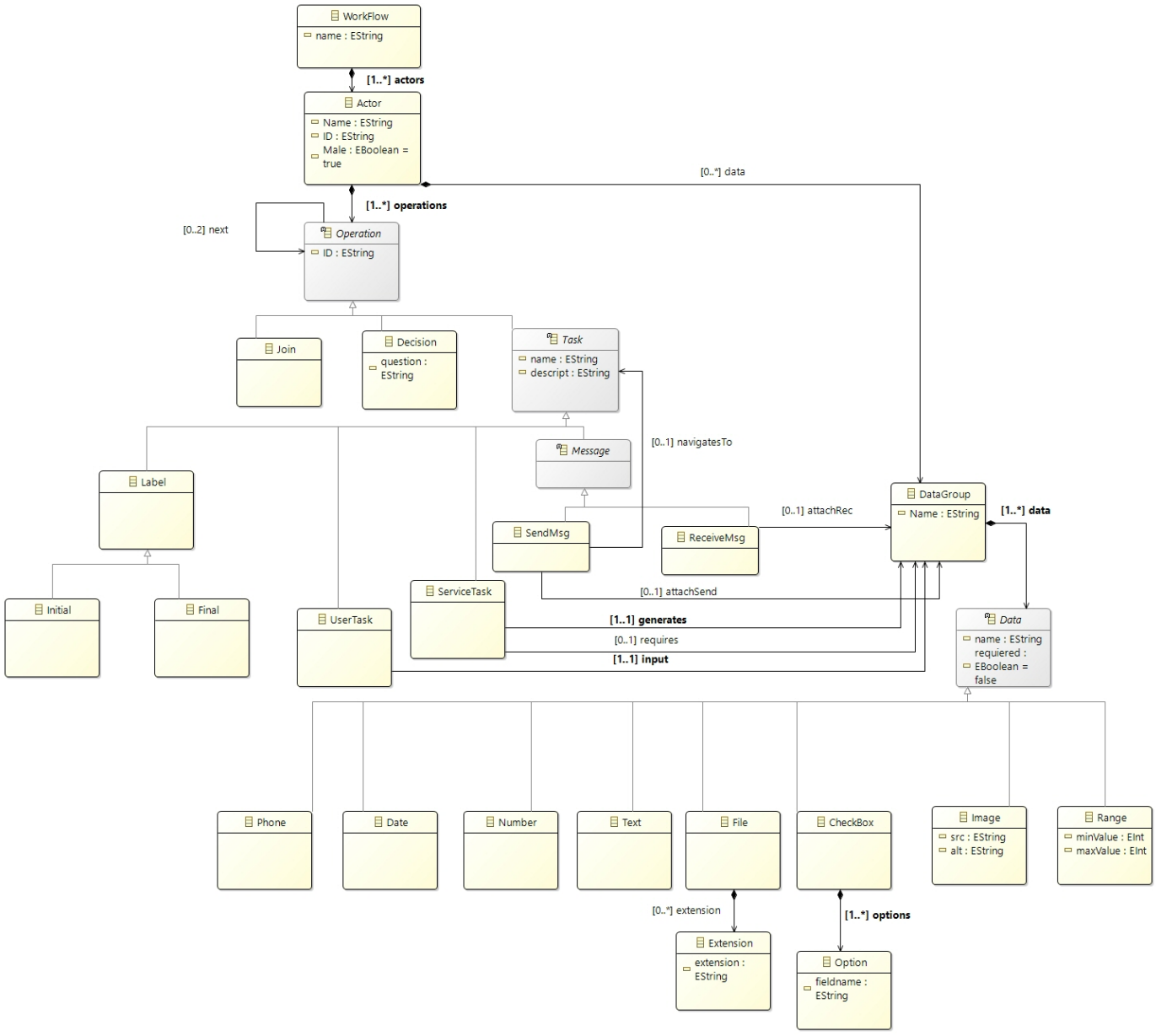

Figure 1. The DSL4DDW+SQ meta-model (diagram representing the abstract syntax of the DSL).

2.1 Description of the domain concepts

- EClass (Workflow) :
  Description: root fo the workflow
  EAttributes:
  - Name : EString

  EReferences :
  - (1..*) actors : Actor composition

  Resctriccions:

```
    invariant initialTaskMustBeOne:
                self.actors.operations->selectByType(Initial)->size() = 1;
    invariant finalTaskMustBeOne:
                self.actors.operations->selectByType(Final)->size() = 1;
    invariant TaskMustBePrecededByOneOperation:
          self.actors.operations->forAll( op1,op2 : Operation |
          op1.next->forAll(op3 : Operation |
          ((op3.oclIsKindOf(Task) or op3.oclIsKindOf(Decision))and op3 <> null and
op1 <> op2) implies op2.next->forAll(op4 : Operation |
          ((op4.oclIsKindOf(Task) or op4.oclIsKindOf(Decision)) and op4 <> null)
implies op3 <> op4
                )));
```

- EClass (Actor) :
  Description: represent the actor that carried the tasks of the workflow
  EAttributes:
  - Name : EString
  - ID : EString
  - Male : EBoolean

  EReferences :
  - (1..*) operations : Operation composition
  - (0..*) data : DataGroup composition

- EClass (Operation) **Abstract**:
  Description: Superclass of tasks, decision and join
  EAttributes:
  - ID : EString

  EReferences :
  - (0..2) next : Operation

    *As I explained in the video, I make this for a decision be able to have 2 next operation but I restrict that to be one in OCL for the tasks.

  Resctriccions:

```
    Invariant InitialTaskMustNotBePreceded:
                self.next->forAll(op : Operation |
                      not op.oclIsTypeOf(Initial)
                );
    Invariant NextOperationMustNotBeItself:
                self.next->forAll(op : Operation |
                      op <> self
                );
    Invariant NextOperationToAnotherActorMustNeedASendMsgTask:
                self.next->forAll(op : Operation |
```

```
                    (not self.oclIsTypeOf(SendMsg)) implies op.oclContainer =
self.oclContainer
                );
```

- EClass (Join) **extends** Operation:
  Description: make possible to join a flow that have 2 ways after a decision.

- EClass (Decision) **extends** Operation:
  Description:
  EAttributes:
  - question : EString

- EClass (Task) **Abstract extends** Operation :
  Description: supertype of all kind of tasks
  EAttributes:
  - Name : EString
  - Descript : EString

- EClass (Label) **extends** Task:
  Description: supertype of Initial and Final task.

- EClass (Initial) **extends** Label:
  Description: the task that starts the workflow
  Resctriccions:

```
    invariant TaskMustHasOneSuccessor:
                self.next->size() = 1;
```

- EClass (Final) **extends** Label:
  Description: task that ends the workflow
  Resctriccions:

```
     invariant FinalTaskMustNotHasAnySuccessor:
                self.next->size() = 0;
```

- EClass (UserTask) **extends** Task:
  Description: task where the user has to fill some data
  EReferences :
  - **(1..1)** input : DataGroup

  Resctriccions:

```
    invariant TaskMustHasOneSuccessor:
                self.next->size() = 1;
```

- EClass (ServiceTask) **extends** Task:
  Description: task that manage an automatic service done by the web
  EReferences :
  - (0..1) requires : DataGroup
  - **(1..1)** generates : DataGroup

  Resctriccions:

```
    invariant TaskMustHasOneSuccessor:
                self.next->size() = 1;
```

- EClass (Message) **Abstract extends** Task:
  Description: supertype of SendMsg and ReceiveMsg
  Resctriccions:

```
invariant TaskMustHasOneSuccessor:
                    self.next->size() = 1;
```

- EClass (SendMsg) **extends** Message:
  Description: task for send data or change the flow to another actor
  EReferences :
  - (0..1) navigatesTo : Task
  - (0..1) attachSend : DataGroup

  Resctriccions:

```
invariant SendMsgMustBeSendToADifferentActorThanItself:
      self.oclContainer <> self.next.oclContainer;
invariant
SendMsgMustBeConnectToAReceiveMsg:
      self.next->forAll(op : Operation |
            op <> null implies op.oclIsTypeOf(ReceiveMsg)
      );
```

- EClass (ReceiveMsg) **extends** Message:
  Description: task for receiving data or continue the workflow with the requests of the actor received
  EReferences :
  - (0..1) attachRec : DataGroup

- EClass (DataGroup) :
  Description: class that contains a set of data
  EAttributes:
  - Name : EString

  EReferences :
  - (1..*) data : Data composition

- EClass (Data) **extends** Task:
  Description: supertype of kinds of data
  EAttributes:
  - Name : EString
  - Required : EBoolean

- EClass (Phone) **extends** Data:
  Description: data for phone type

- EClass (Date) **extends** Data:
  Description: data for date type
- EClass (Number) **extends** Data:
  Description: data for numerical type

- EClass (Text) **extends** Data:
  Description: data for string type

- EClass (File) **extends** Data:
  Description: data for file type
  EReference:

- (0..*)extension : Extension composition

- EClass (Extension) :
  Description: name extension accepted for file type
  *if you don't put any extension by default accept all type of files.
  EAttributes:
  - Extension : EString

- EClass (CheckBox) **extends** Data:
  Description: data for checkbox type
  EReference:
  - (1..*) options : Option composition

- EClass (Option):
  Description: option name for data checkbox type
  EAttributes:
  - Fieldname : EString

- EClass (Image) **extends** Data:
  Description: data for image type
  EAttributes:
  - Src : EString
  - Alt : EString

- EClass (Range) **extends** Data:
  Description: data for range type
  EAttributes:
  - minValue : EInt
  - maxValue : EMax

3 Concrete syntax of Workflow-DSL

Domain concept	**Textual/graphical representation of concept**
`class WorkFlow`	`@gmf.diagram`
`class Actor`	`@gmf.node(label="Name", border.color="0,0,0", color="250,250,200")`
Actor : Ref operations	`@gmf.compartment`
Actor : Ref data	`@gmf.compartment(collapsible="true")`
Operation : Ref next	`@gmf.link(label.icon="false", width="2", color="0,0,0", target.decoration="arrow")`
Class Operation	`@gmf.node(label="name", label="ID")`
Class Decision	`@gmf.node(label="ID", border.color="0,0,0", color="250,250,200")`
ServiceTask : Ref generates	`@gmf.link(border.color="100,0,0", style="dash", width="1")`

ServiceTask : Ref requires	`@gmf.link(border.color="0,0,100", style="dot", width="2")`
Class DataGroup	`@gmf.node(label="Name")`
DataGroup : Ref data	`@gmf.compartment(collapsible="true", layout="list")`
UserTask : Ref input	`@gmf.link(border.color="0,100,0", style="dot", width="2")`
CheckBox : Ref options	`@gmf.compartment(collapsible="true", layout="list")`
Class Option	`@gmf.node(label="fieldname", border.color="0,0,0", color="100,100,100")`
Class SendMsg	`@gmf.node(label="name", label.icon="false", figure="svg",label.placement="external", svg.uri="platform:/plugin/WorkFlowDSL/src/generatedCode/images/send.svg")`
Class ReceiveMsg	`@gmf.node(label="name", label.icon="false", figure="svg",label.placement="external", svg.uri="platform:/plugin/WorkFlowDSL/src/generatedCode/images/receive.svg")`
SendMsg : Ref attachSend	`@gmf.link(border.color="100,0,100", style="dot", width="2")`
SendMsg : Ref NavigatesTo	`@gmf.link(label.icon="false", width="1", color="0,0,0", target.decoration="arrow")`
ReceiveMsg : Ref attachRec	`@gmf.link(border.color="0,100,100", style="dot", width="2")`
Class Data	`@gmf.node(label="name", border.color="0,0,0", color="150,100,100")`

3.1 Example Workflow-DSL models

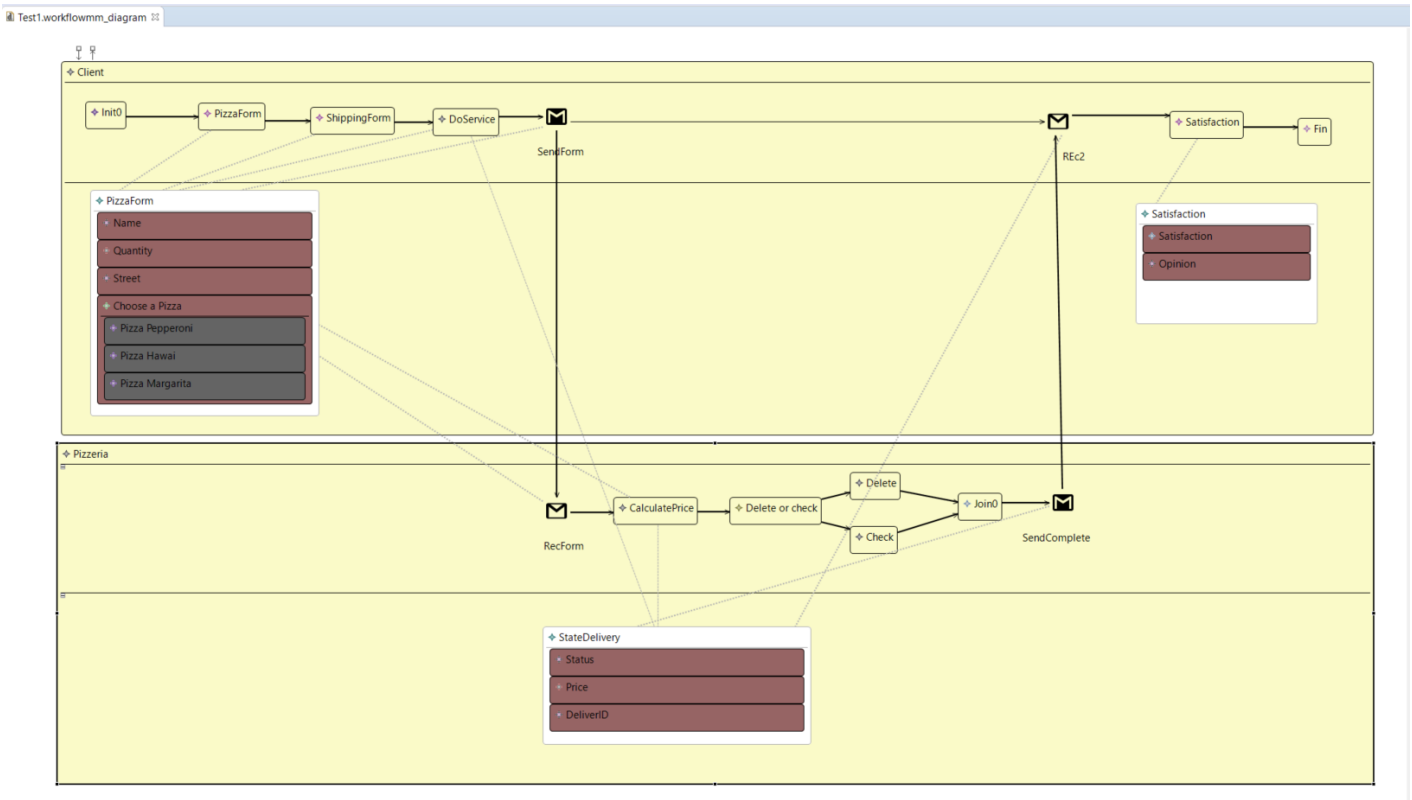

Figure 2. The Graphic Diagram for PizzeriaSimpleExample(diagram representing the concrete syntax of the DSL).

4 Code generation

The structure of the web page for the *figure 2* is:

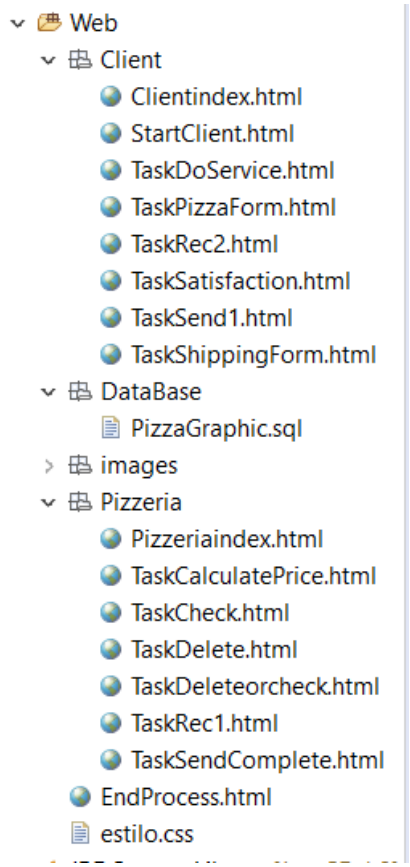

Figure 3. Folder Web Structure

5 Eclipse tools used to develop this project

Name	Version
EMF Model Comparison for EUnit (EMF Compare 3.x / Kepler and	1.4.0.201611012202
Epsilon Concordance	1.4.0.201611012202
Epsilon Core	1.4.0.201611012202
Epsilon Core Development Tools	1.4.0.201611012202
Epsilon Development Tools for EMF	1.4.0.201611012202
Epsilon Development Tools for UML	1.4.0.201611012202
Epsilon EMF Integration	1.4.0.201611012202
Epsilon GraphML Integration	1.4.0.201611012202
Epsilon UML Integration	1.4.0.201611012202
Epsilon Validation Language EMF Integration	1.4.0.201611012202
Epsilon Wizard Language EMF Integration	1.4.0.201611012202
Epsilon Wizard Language GMF Integration	1.4.0.201611012202
Eugenia	1.4.0.201611012202

6 Delivered contents

The .zip file, submitted using the project delivery task, available in the DMSS Virtual Classroom, contains the Eclipse projects and complementary files listed next:

- All the Eclipse projects developed as part of the assignment (abstract and concrete syntax, code generators, etc.):
  - WorkFlowDSL
  - WorkFlowDSL.edit
  - WorkFlowDSL.editor
  - WorkFlowDSL.test
  - WorkFlowDSL.diagram
  - org.eclipse.acceleo.module.generateApplication
- The example models developed with the WorkflowDSL graphical/textual model editor:
  - PizzeriaSimpleExample.src.Test1.workflowmm_diagram
- The code generated from the previous example models:
  - PizzeriaSimpleExample.Web.*
- Another example for generating code
  - PizzeriaWith2WorkersExample
  - Model in model. PizzeriaWith2WorkersExample
  - Code in src.*
- The project documentation (this document):
  - DMSS_Project Documentation_Group_5.docx
- An 5 minutes explanatory video:
  - DMSS_WorflowDSL_Group5_Parte1.mp4
  - DMSS_WorflowDSL_Group5_Parte2.mp4

7 Annex 1. Task planning

As a individual developer I didn't have a planning, I just carry out with the task when I have time, but I did it in an order, the following one:

1- Create the Meta-Model

2- Create OCL restrictions

3- Create a wrong model for test the OCL restrictions

4- Create a right model

5- Create Acceleo Code and test it with the right model

6- Create the EuGenia editor

And between all task, all time I have to came step behind to change something.

And about the video, I couldn't upload it because is was so heavy, so I sent you a email with the link, but I let you here too the link.

https://drive.google.com/open?id=109hgex9pKaM-Io53EphLgwucWG0SBIkv

8 Annotations for Extensions

For Extension 1, I make new operations called **Decision** and **Join**, for letting the user have more freedom of choice of the flow of the work, and add so **many data types** for have a better choice for what data you need, and then little things, like a attribute male in the EClass Actor for change the icon of the main actor page, and male people getting differents icon depending if their name is start lower than M in the alphabet, or a required attribute in the EClass Data for knows if a data if compulsory to be fill or not.