Domain-Specific Modelling Languages
Project Statistics

António Ferraz
Rafael Gameiro

November 2019

EVL Rules and ETL Transformations

0.1 EVL

For our project, we considered the following EVL rules:

- There can only be one package with the same name
- Each Package must have a name
- There can only a class with the same name
- There is no possibility of a class having a transition to itself
- Two classes cannot have 2 different transitions in between themselves
- If there is a aggregation or composition the class has to have a relationship in the opposite direction
- A class can only have 1 superclass
- Each class must have a name
- No duplicate variables in a Class
- No duplicate functions in a Class
- Each variable must have a name, a visibility and a type.
- Each function must have a name, a visibility and a type.
- No duplicate arguments in a function
- Each argument must have a name and a type

0.2 ETL

Our project has 2 types of transformations into 2 other models, into a Java Class Diagram model and into a Class Analysis Diagram. To the Java model there was no much difference other from now aggregations and compositions turn into collection variables on the source class. And for the Class Analysis Class Diagram it was only needed to transform into the correspondent elements, but with losing some specificity.

Project Assumptions

For our project, we considered the following assumptions:

- We tried to simplify the possible values of cardinality (Composition and Aggregation) we user could choose, and we gave the user 5 possible choices of values: 0, 1, * 1..*, 0..*, 0..1.
  This way, the user would have some margin to choose the correct cardinality, but at the same time those options would be limited.

- In the Model-to-Code transformations, we assumed that the Composition and Aggregation, being converted to a Java Model it would be represented as a collection of instances from the origin class.

Statistics

Statistics	**Values**
Number of classes	3
Number of associations	1
Mean of Depth of inheritance	0.33333334
Max of Depth of inheritance	1.0
Per class: Main	
The number of private methods	0
DAC[1]	0
DAC'[2]	0
$SIZE$[3]	2
Weighted Methods	0
Per class: Manager	
The number of private methods	0
DAC[1]	0
DAC'[2]	0
$SIZE$[3]	2
Weighted Methods	0
Per class: managerController	
The number of private methods	0
DAC[1]	0
DAC'[2]	0
$SIZE$[3]	1
Weighted Methods	0

[1]: The number of attributes in a class that have another class as their type
[2]: The number of different classes that are used as types of attributes in a class
[3]: The number of methods + The number of attributes