

APPENDIX for the paper on: *UML Model Querying with OCL and Prolog: A Performance Study.*

Joanna Chimiak–Opoka, Chris Lenz, Michael Felderer
Institute of Computer Science,
University of Innsbruck
Technikerstrasse 21a, A–6020 Innsbruck

Christian Lange
Dep. of Math. and Computer Science,
Eindhoven University of Technology
Den Dolech 2, NL–5600 MB Eindhoven

1 Methods in of the model generator

Below all methods of the model generation are described. The generator can be downloaded from <http://squam.info/source/umlgenerator/>.

createMyModel: The top level of our UML models is a Model. All other elements are packaged directly into this model without other structuring.

createPrimitiveTypes: For the experiment only three primitive types have been used, `int`, `String` and `boolean`.

createClasses: Create the number of classes (NoC) which is one input parameter, and decide if the class should be abstract or not. For that we have the Abstract Class Factor (AC), if the following statement returns true the created class would be abstract (`random.nextInt(AC)==0`).

createGeneralization: For each generated class there is determined if it should have a generalization or not (`random.nextInt(Gen)==0`). If their should be a generalization it is randomly determined which of the other classes should act as super class. Because of that it is also possible to get generalization cycles.

createAttributes: For each class 0 up to NoAt attributes are created. The attributes types are only primitive type, where the type is randomly determined. The randomization generates 2/5 `int`, 2/5 `String` and 1/5 `boolean`.

createAssociations: The created number of associations for each class is 0 to NoAs. The associated class is determined randomly out of all generated classes. The cardinality of all associations is always 1 to 1.

createInstances: Creates 0 to NoI Instances per class.

createSlots: Creates for NoS% attributes a slot. We decided to create in 1/3 of all cases a false type for the slot, which means the type of the slot differs to the type of the attribute.

createLinks: Creates for NoL% association a link.

saveModel: Saves the model to file.

2 Format of log files

Both evaluators generate for each execution one entry into the log file and reports the result of the expression into an extra result file. Each log entry is saved in a separate line and consists of the following parameters, separated by `||`:

```
|| timestamp || framework name || system  
|| expression name || model name || modelsize  
|| evaluation netto time || mem init  
|| mem max || name of the results file ||
```

Results and log files have fixed structures and were automatically analysed for results consistency and performance statistics, respectively.

3 Configuration of OCL evaluation suite

The input files have to be in a directory where the application is started.

Table 1. Input Parameters and Files

name	description
NoC	number of classes for selecting the model
MSN	model serial number for selecting the model
repetitions	how often the test is executed, each execution returns a result
size of a cycle	how often a query is executed, the needed time is divided by repeat, 0 is handled like 1
expressions.ocl	includes all executed expressions. These file is a text file where each expression is contained in the following statement block: <pre>expression eexpressionname -- expression call endexpression</pre>
defines.ocl	includes all needed defines which can be referenced either by the expressions or by the defines themselves: <pre>package packagename -- expression definition endpackage</pre>