Core - The Contextual Role Editor

Henri Mühle

May 19, 2011

1 About Core

Core is a small tool that is thought to assist the process of developing role-oriented software models[1] in a context-based fashion. It consists of two parts: a) a role model diagram editor and b) a command line tool.

The diagram editor is a GMF-based Eclipse plugin. Therewith you can create UML style class diagrams (*.rd-files), describing role models. Even more, if you omit the role hierarchy, you can design standard object-oriented class hierarchies as well.

The command line tool is little bit more than a Java-based file converter. It is able to create the respective formal contexts from given *.rd-files as well as the appropriate *.rd-file from given formal contexts (representing the base and role hierarchies). Even more, it lists the possible bonds between the given base and role context, since these bonds are precisely the possible role-play relations. The user can then interactively determine the desired role-play functionality. In extension to this, one can determine the mergings between both hierarchies as well, which has, however, not been included into the role models yet.

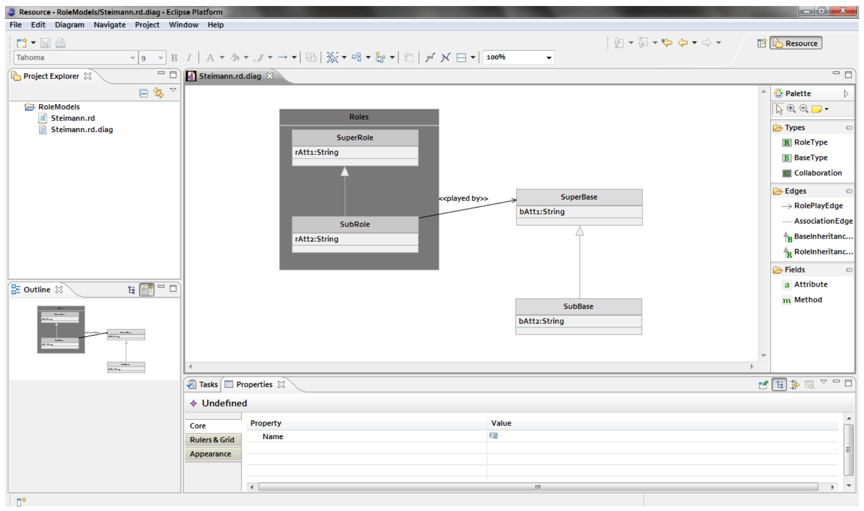
Note, that you can create standard object-oriented software models as well using this tool. Simply leave the hierarchy of role types empty. A possible extension for this editor can be a code generator, that generates the Java classes from the diagram represented in the *.rd-file.

If you have any ideas to improve Core, any questions or found some bugs, do not hesitate to contact me via `henri.muehle@univie.ac.at`.

2 Using the Diagram Editor

After you have successfully installed or copied (which is the same) the archive files for the diagram editor into your Eclipse directory, you can run Eclipse and should note a new entry in *File* → *New* → *Example...* called *Role Model Diagram*. You should create a new folder (*File* → *New* → *Folder*) in your workspace, in which to store the role model diagrams. After the you created the *.rd- and *.rd.diag-files with the above mentioned wizard (*File* → *New* → *Example...* → *Role Model Diagram*), the Eclipse perspective should change and look something like this:

[1]For a general introduction to the topic, we refer to [3] and [2].



Right-clicking somewhere on the canvas and selecting *Show Properties View* allows you to edit the properties of the canvas elements. The palette on the right-hand side contains all the possible elements that can be placed on the canvas. Note that you can only place BaseTypes and Collaborations directly on the canvas. RoleTypes have to be placed inside collaborations. (This needs some patience sometimes. Try it close beneath the horizontal grey name separator.) Attributes and Methods can be added to BaseTypes and RoleTypes, edges can only be drawn between the appropriate elements. (RolePlayEdges go from a RoleType towards a BaseType.)

If the editor does not open, when double-clicking an *.rd.diag-file, open it via right-clicking the *.rd.diag-file, select *Open With* → *Other...* and choose *Role Model Editing*.

If you already have a generated *.rd-file, initialize the diagram by right-clicking the *.rd-file and choose *Initialize rd.diag diagram file*. Unfortunately you have to resize and relocate the generated elements by hand.

3 Using the Command Line Tool

After you have unzipped the archive there is a file called core.jar. Navigate to it using a command line window and type

```
java -jar core.jar
```

You should now see a welcome screen. Table 1 shows the possible commands that can be entered and Table 2 explains the arguments.

If the installation process for any reason does not succeed (e. g. if you don't have the appropriate user rights), you should integrate the folders *features* and *plugins* into the according folders of your Eclipse installation by hand.

command	description	arguments
`help`	lists the available commands	
`path`	shows the path to the local Eclipse distribution	
`setPath`	sets the path to the local Eclipse distribution	⟨path⟩
`eclipse`	starts an Eclipse instance from the given path	
`install`	installs role diagram editor plugin to eclipse	⟨'all'⟩
`exit`	exits CORE	
`clear`	deletes all cached contexts and generators	
`load`	loads an input file	⟨'bc'|'rc'|'cc'⟩ ⟨path⟩
`print`	prints a context	⟨'all'|'bc'|'rc'|'cc'|'mc'⟩
`store`	stores a context as *.csv or the diagram file as *.rd	⟨'all'|'bc'|'rc'|'cc'|'mc'|'rd'⟩ [⟨path⟩]
`nextBond`	shows the next available bond between the given role and base context	
`mergings`	prints the number of (proper) mergings between the given contexts and optionally creates the context of mergings	[⟨'mc'⟩][⟨'p'⟩]

Table 1: The available CORE-commands

argument	description
all	in case of `install` it installs the GMF framework, otherwise it `store`s resp. `print`s all active contexts
bc	refers to the base type context
rc	refers to the role type context
cc	refers to the composition context
mc	refers to the merging context
p	refers to the proper mergings

Table 2: Description of the arguments

The first step will probably be to load the a context for the base hierarchy and a context for the role hierarchy. To do so, type `load bc` ⟨`path`⟩ resp. `load rc` ⟨`path`⟩ for entering the base resp. role context.

`nextBond` returns the next possible bond (resp. the first one, if you type it for the first time) between the contraordinal scales of base and role type contexts. You can proceed to the next bond (until infinity if you like, since there is a loop between the largest and the smallest bond) or accept the provided bond.

With typing `store rd` ⟨`path`⟩ Core creates the *.rd-file from the given contexts at the given path. Have a look at the generated diagram using the Diagram Editor, if you like. If you load a role diagram file via `load rd` ⟨`path`⟩ the tool automatically creates the base and role context from this file. It additionally creates the composition context as described in [2]. To have a look at the generated contexts, use the command `print` with the proper argument; to save the contexts to your disk, use `store`. If you want to ensure that there do not remain any objects from a previous load (and you do not want to restart Core), you can clear the cache with `clear`.

There are two ways to use the command `mergings`. On the one hand, use it without arguments or with the argument `p`. This returns the number of (proper) mergings between the base and role type context. On the other hand use `mergings mc` without further arguments or with the argument `p` to create the context of (proper) mergings between the base and the role type context. The notion of mergings is hereby understood as introduced in [1].

4 Developing Core

If you have downloaded the net.core.editor.*-projects in order to extend or customize the editor, there are a few things to take care of.

4.1 Accesss Restriction Error

Sometimes, Eclipse will report several errors caused by an access restriction. This is an annoying, but irrelevant error, that can be solved as follows: Select *Window* → *Preferences* → *Java* → *Compiler* → *Errors/Warnings*. Locate *Forbidden reference (access rules)* under *Deprecated and restricted API*. Change the selection from “Error” to “Warning” and you will not be bugged again[2].

4.2 Exporting the Diagram Editor

If you wan to export the customized diagram editor to make it usable without the source files in the net.core.editor.*-projects, you need to create two more projects:

- create a Plugin-Feature-Project
  File → *New* → *Other...* → *Plug-in Development* → *Feature Project*

[2]Found at `http://lkamal.blogspot.com/2008/09/eclipse-access-restriction-on-library.html`.

- create a Plugin-Site-Project
  File → *New* → *Other...* → *Plug-in Development* → *Update Site Project*

In the feature project, you have to list all the net.core.editor.*-projects. You can then use the site projet to create jars from these projects that you can eventually distribute[3].

References

[1] Bernhard Ganter, Christian Meschke, and Henri Mühle. Merging Ordered Sets. In Robert Jäschke and Petko Valtchev, editors, *Proceedings of the 9th International Conference Formal Concept Analysis*, volume 6628 of *Lecture Notes in Artificial Intelligence*. Springer, 2011.

[2] Henri Mühle and Christian Wende. Describing Role Models in Terms of Formal Concept Analysis. In Léonard Kwuida and Baris Sertkaya, editors, *Proceedings of the 8th International Conference Formal Concept Analysis*, volume 5986 of *Lecture Notes in Artificial Intelligence*. Springer, 2010.

[3] Friedrich Steimann. On the Representation of Roles in Object-Oriented and Conceptual Modelling. *Data Knowledge Engineering*, 35, 2000.

[3]Found at `http://dev.eclipse.org/newslists/news.eclipse.modeling.gmf/msg16304.html`.