Kermeta tutorial

How to create a metamodel

François Tanguy, Didier Vojtisek

Abstract

This tutorial is part of a serie of tutorials that explain step by step one of the aspect of Kermeta. This one will explain you how to a metamodel to be used by kermeta. However, there are many ways to create them, this tutorial will present some of them.

Published Build date: 22-October-2009
2006-10-27T16:24:23
19/11/2006

List of Figures

CHAPTER

Preface

Kermeta is a Domain Specific Language dedicated to metamodel engineering. It fills the gap let by MOF which defines only the structure of meta-models, by adding a way to specify static semantic (similar to OCL) and dynamic semantic (using operational semantic in the operation of the metamodel). Kermeta uses the object-oriented paradigm like Java or Eiffel. This document presents various aspects of the language, including the textual syntax, the metamodel (which can be viewed as the abstract syntax) and some more advanced features typically included in its framework.

Important

Kermeta is an evolving software and d espite that we put a lot of attention to this document, it may contain errors (more likely in the code samples). If you find any error or have some information that improves this document, please send it to us using the bug tracker in the forge: http://gforge.inria.fr/tracker/?group_id=32 or using the kermeta user mailing list (kermeta-users@lists.gforge.inria.fr) Last check: v1.3.2

Tip

The most update version of this document is available on line from http://www.kermeta.org .

CHAPTER 1

Prerequisites

No particular background is necessary to read this tutorial.

> **Important**
>
> KerMeta must be installed. If not, please refer to "How to install KerMeta" tutorial.

CHAPTER 2

Introduction

There are two ways to create a the structure of a meta model. The first one is to use Kermeta textual syntax and creating a ".kmt" file. using this approach you will have the sensation of "programming" your metamodel. The second one is to create an Ecore file using the tools proposed by EMF. (for example the EMF reflexive editor, or the ecore diagram editor from ecorre tools project).

You'll then be able to transform the ecore files to kmt files and vice versa as needed.

This tutorial should help you in manipulating the Ecore and KerMeta files. It gives an overview on how to create meta models and how to transform them.

Start by creating a new general empty project in Eclipse.

CHAPTER 3

A meta model using Kermeta textual syntax

Select in the main menu of Eclipse

"File" > "New" >> "Other..."

This action opens the following window.

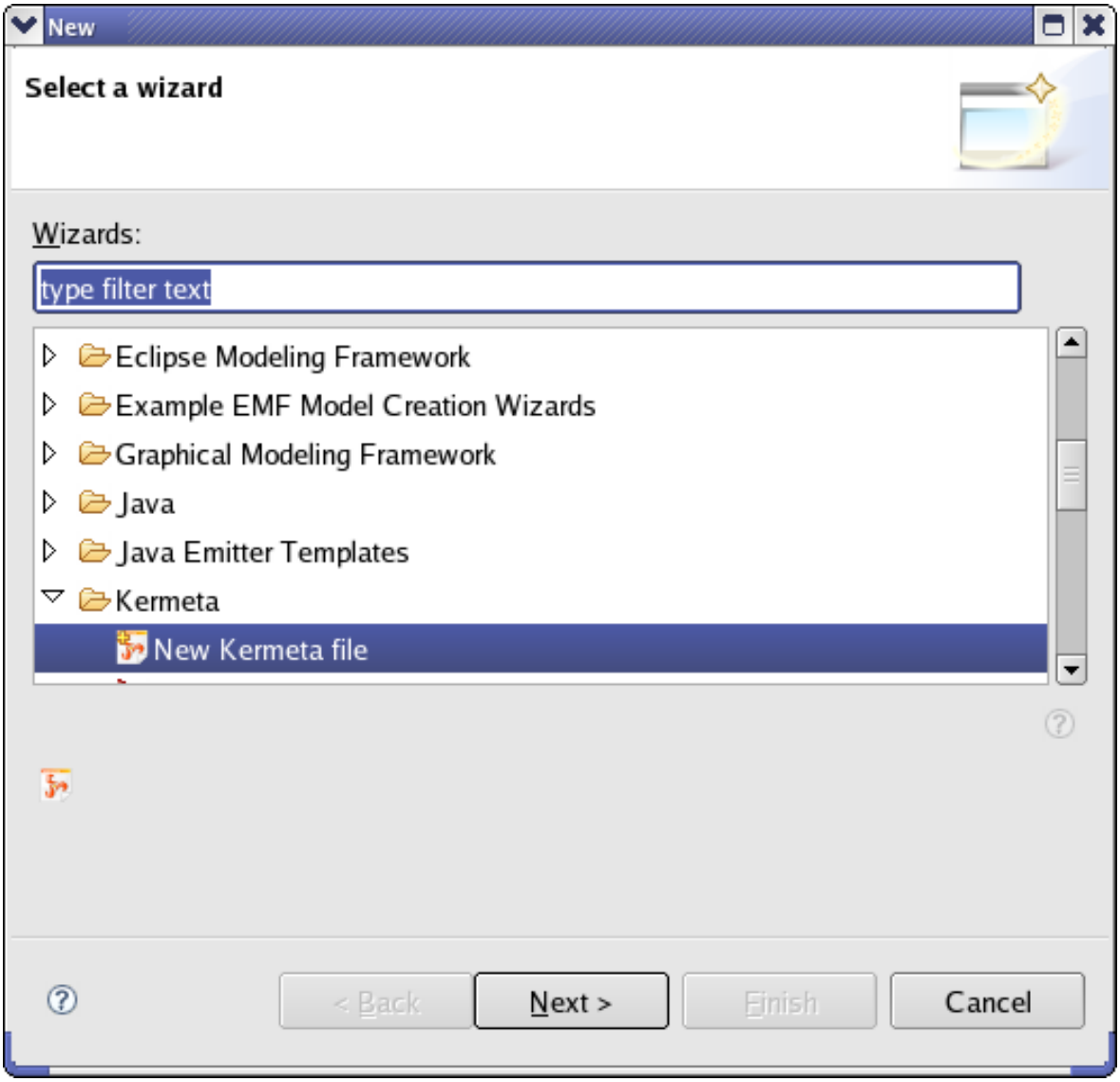

Figure 3.1.

Select "New Kermeta file" and press "Next >".

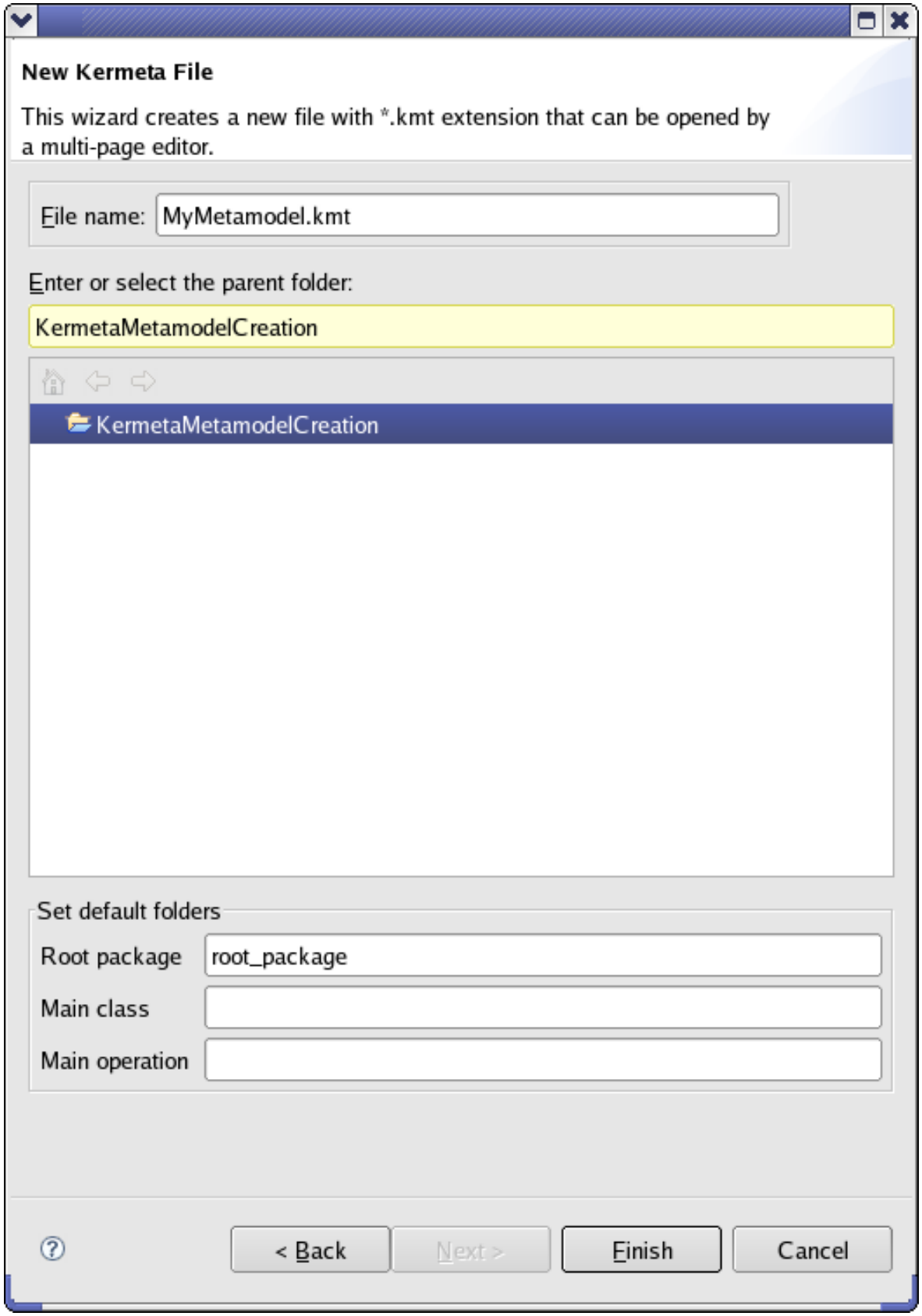
New Kermeta File
This wizard creates a new file with *.kmt extension that can be opened by a multi-page editor.
File name: MyMetamodel.kmt
Enter or select the parent folder:
KermetaMetamodelCreation
KermetaMetamodelCreation
Set default folders
Root package
root_package
Main class
Main operation
< Back
Next >
Finish
Cancel

Figure 3.2.

Give a name (MyMetamodel.kmt for instance) to the new file, select a location (The default one is good for us) and leave Main class and Main operation blank for the moment. It will be used later to run models. Click on "Finish".

Now your main window should look like the one above. The file is written in KerMeta language. So you can edit this file to add some classes, attributes and so on using the KerMeta language (read http://www.kermeta.org/documents/manual/ for more details on KerMeta language).

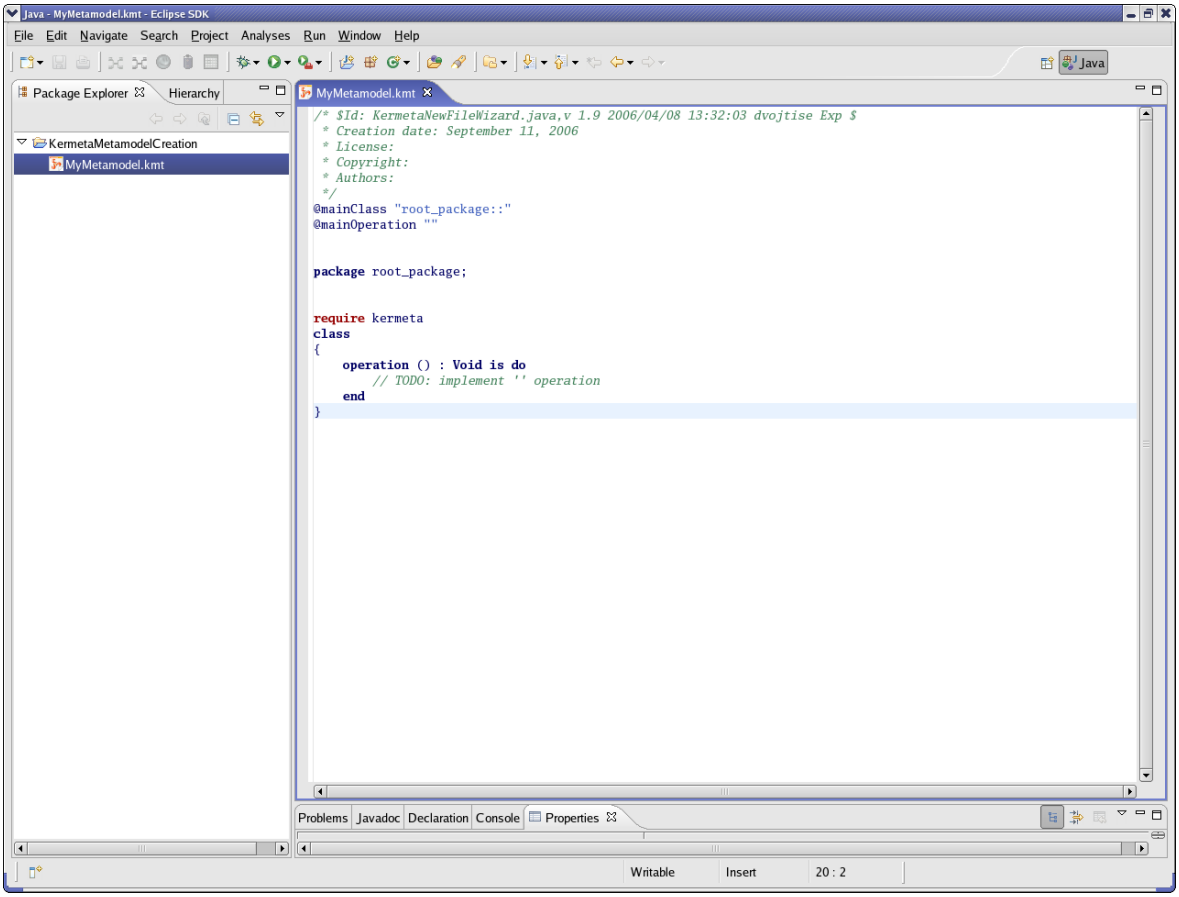

Figure 3.3.

CHAPTER 4

A meta model from an Ecore file

To create an Ecore file, go in the main menu of Eclipse select :

"File" > "New" > "Other..."

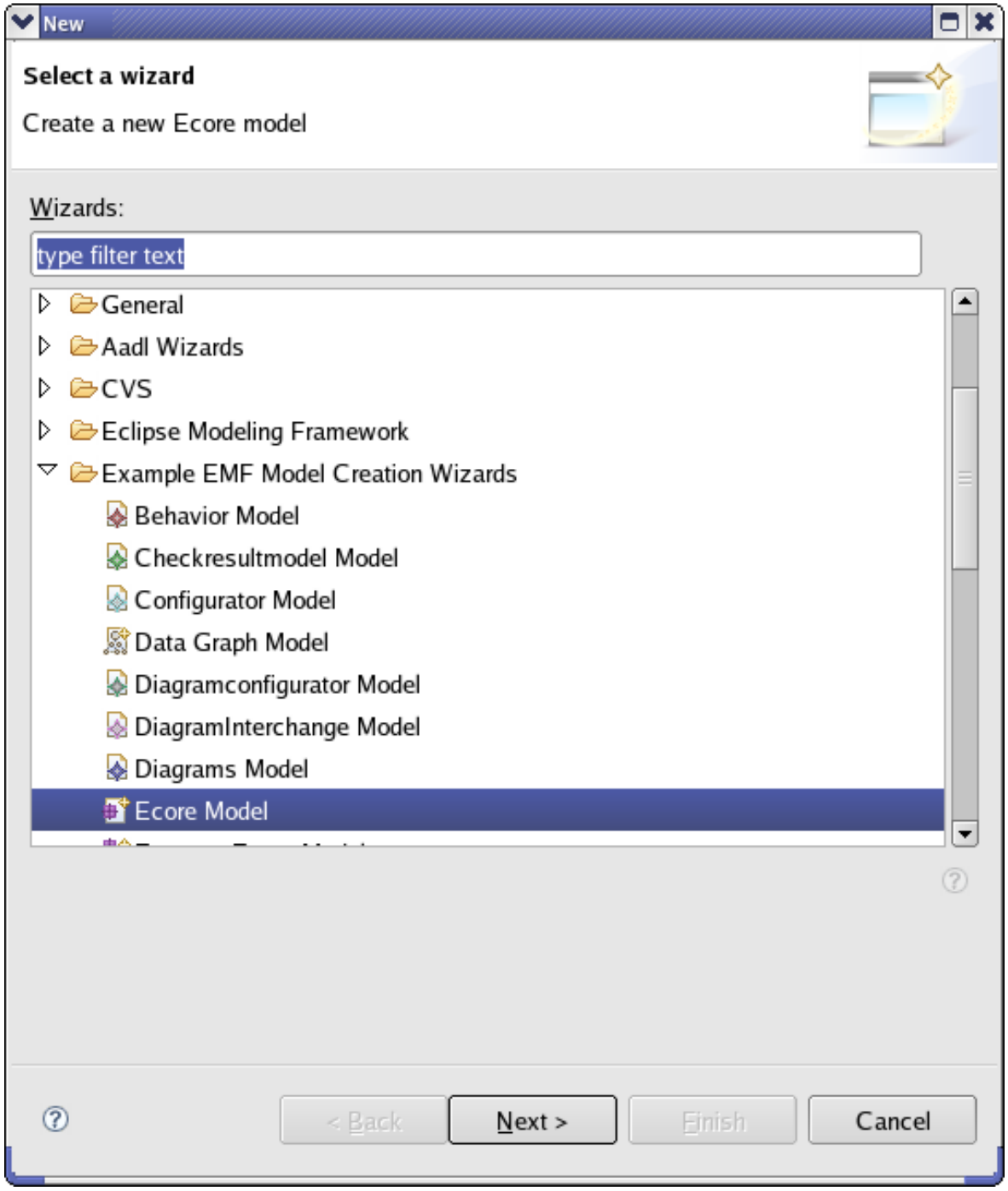

Figure 4.1.

Select "Ecore model" wizard and click "Next >". Select a location and a name for the file. Call it "AnOther-Metamodel.ecore" for example. Click on "Finish".

Your main window now looks like this :

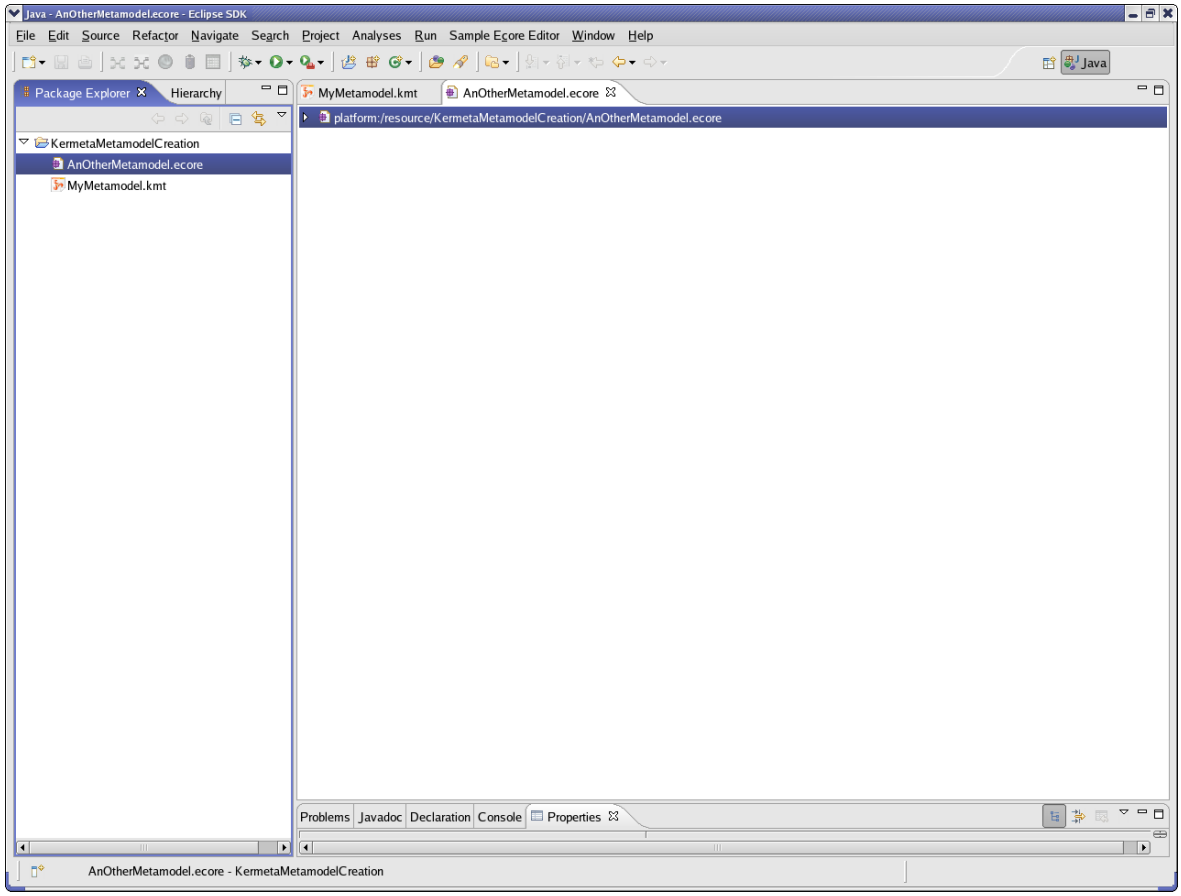

Figure 4.2.

You can edit this meta model. If you do not know how to do that, please have a look in section about the Ecore editor (page 26) in the following link :

http://www.eclipsecon.org/2005/presentations/EclipseCon2005_Tutorial28.pdf

> **Important**
>
> You must at least fill in the name, ns prefix and ns URI of the package.

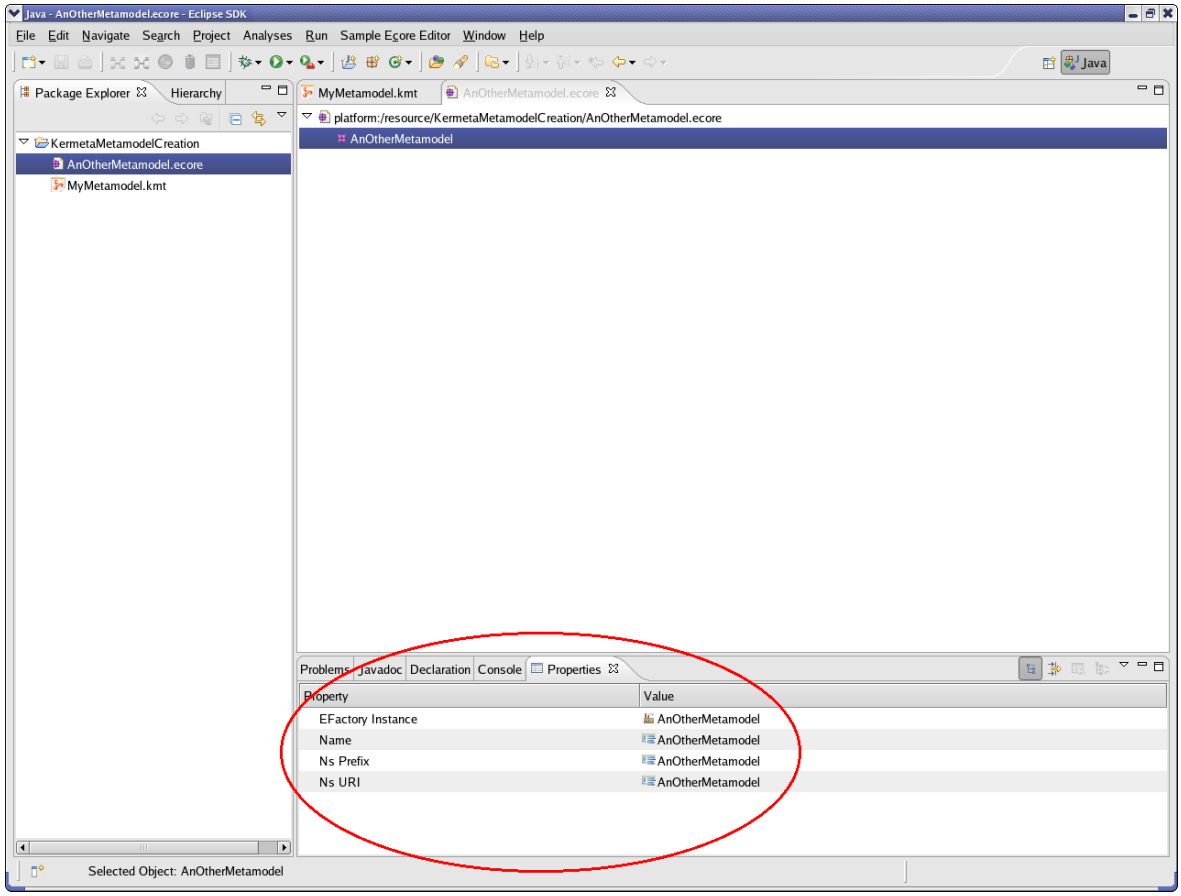

Figure 4.3.

You can also edit these ecore file using a diagram editor. For example, the ecore tools project provides a class diagram editor that works on top of ecore files. The diagram informations (ie. shape and position of the classes on the sheet) are stored in a separated file : `*.ecorediag`

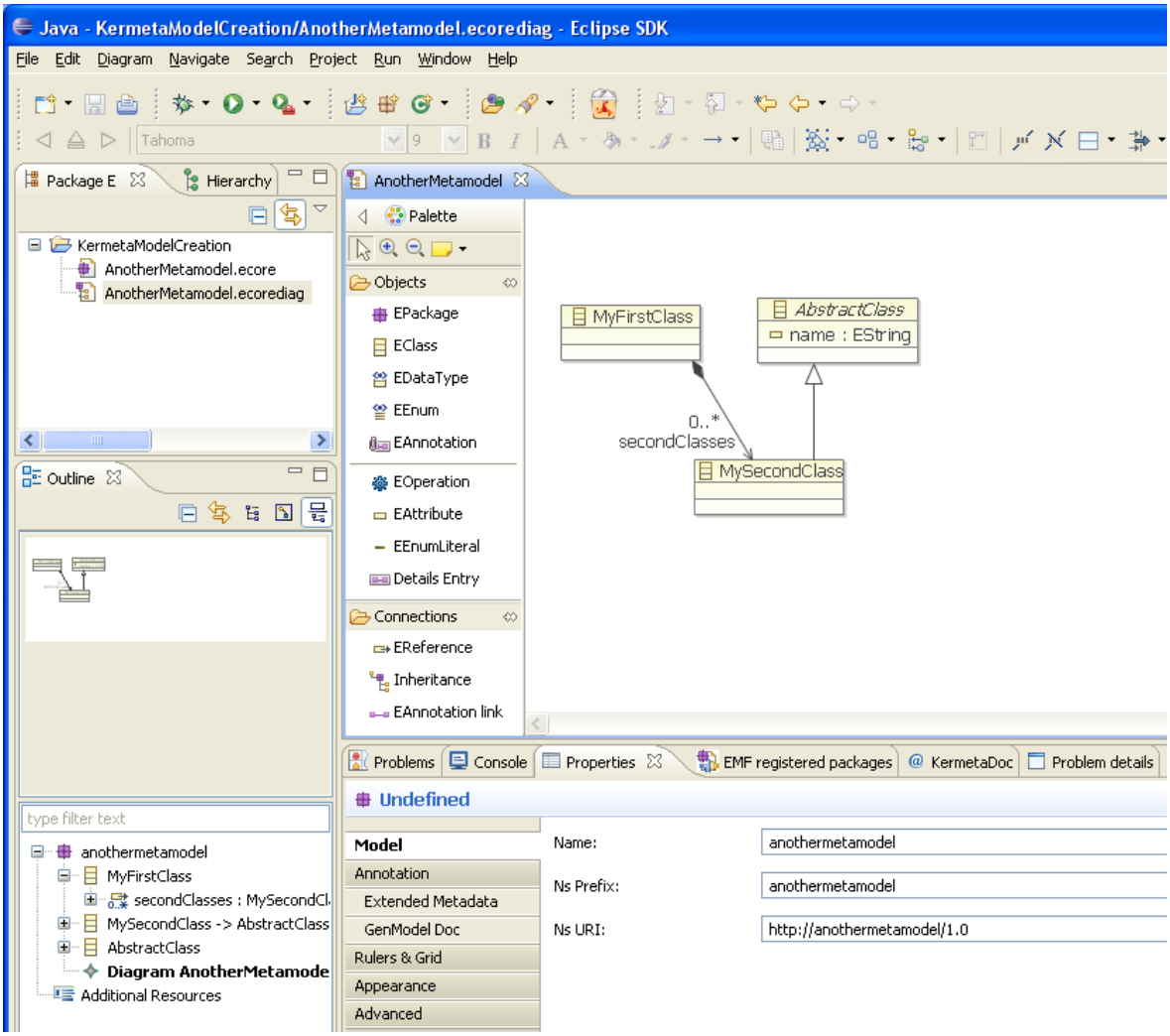

Figure 4.4. Sample use of Ecore tool diagram editor on top of an ecore file

Let say that you edited this meta model and now you want to add some behaviors. You have now two possibilities :

- **Use kermeta aspects**

  You can directly require the ecore file and then using the `aspect` keyword you can create a class that will extend the definition imported from the ecore file. Since the addition of aspect in kermeta (v1.0.0), this is usually the recommended way to add behavior.

- **Transform your ecore file into a KMT text file** and then directly add the expected behavior in it.

  This is the old way to achieve that, you simply have to transform your ecore file to and from the textual syntax (ie. kmt file) so yopu'll be able to add new operations, atributes, ... and define their behavior using the textual syntax.

You'll find more details and tip and tricks about adding behavior to a metamodel in the "How to add behavior

to a metamodel" tutorial : http://www.kermeta.org/documents/tutorials/tut-add_behavior/

CHAPTER 5

Transformations of KerMeta meta models

Several transformations are available. This will allow you to choose the representation of your meta model from one format to another. For example, if you prefer coding, then you will probably prefer the kmt format (KerMeta text). Otherwise you will design with the reflexive editor and the km files (serialized files).

So transformations give you the ability to adapt the meta model to your preferred point of view : coding view or graphical view.

List of formats related to metamodels

***.ecore**	This is the format from EMF metamodel: it contains only the structural part of the metamodel. It is used as a reference for the serialisation of models. Other information may be stored as EAnnotations (but in this case they are specific to a tool.
***.kmt**	This is the textual form of kermeta metamodels.
***.km**	This is the model form (ie. XMI) of kermeta metamodels. Ie.
***.ecorediag**	This is the class diagram information on top of a ecore model. (ie. position , shape, ... information in the diagram)
***.kmdi**	This is the class diagram information on top of a km model. (ie. position , shape, ... information in the diagram)
***.uml**	For uml models. There are some ways to transform part of it into ecore or kermeta (class diagram part and stereotype definiton part)
***.ocl**	For OCL constraints in a textual format. OCL invariants can be transformed into Kermeta invariants.

Yopu can usually relate/transform these formats using a right click on the input file. More information about the options provided by these transformation is provided in the UserInteface guide. http://www.kermeta.org/documents/ui_user_guide/

List of available transformations (not exhaustive)

ecore to km	Transform an ecore model into a km model.
ecore to kmt	Transform an ecore model into a kmt text file.

km to ecore	Transform a km model into an ecore model. Kermeta specific information are stored into EAnnotations (see Kermeta reference manual). Roundtrip is supported
kmt to ecore	Transform a kmt text file into an ecore model. Kermeta specific information are stored into EAnnotations (see Kermeta reference manual). Roundtrip is supported
uml class to ecore	Transform the uml classes into ecore classes.
uml profile to ecore	Transform the profile definitions into ecore classes to be used for loading uml model using such stereotypes.
ocl to kermeta	Transform the ocl invariants into kermeta invariants.