EMF Refactor Creation Review

EMF Refactor Development Team

Thorsten Arendt[1], Florian Mantz[1], Lars Schneider[2]

[1] Philipps-Universität Marburg, Germany, [2] Capgemini sd&m, Offenbach, Germany

October 7, 2009

Communication channel:
`‘eclipse.technology.emft’` newsgroup

Outline

Introduction

Scope

Code Contributions

Participants

Roadmap

Introduction

Motivation

- Model driven development and model based development are promising trends in software development.
  - Models become primary artifacts.
  - Consequence: Models must be of high quality.
- Code refactoring is a well-known and investigated technique to improve software structures without changing the observable behavior.
  - Consequence: Model refactoring is an essential technique for model quality assurance.
- Tool support for model refactoring is limited, particularly for models using the Eclipse Modeling Framework (EMF).

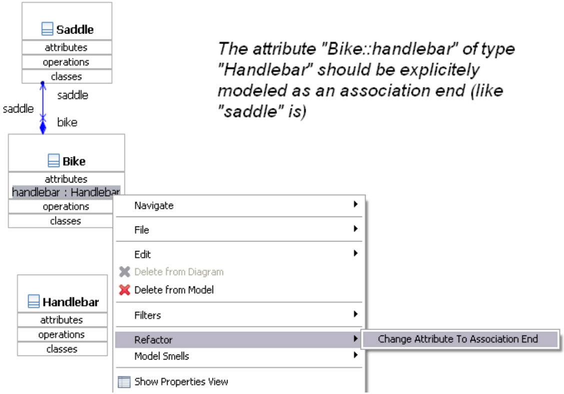
Introduction

Goals

Extensible tool support for the refactoring of EMF based models.

- Structured suite of predefined model refactorings.
- Definition and test of individual model refactorings.
- Uniform and user-friendly refactoring application.

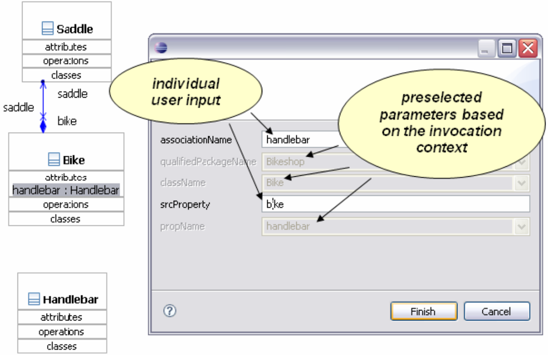
Refactoring *Change Attribute to Association End*



The attribute "Bike::handlebar" of type "Handlebar" should be explicitely modeled as an association end (like "saddle" is)

Figure: Refactoring invocation from within UML2Tools.

Refactoring *Change Attribute to Association End*



associationName: handlebar

qualifiedPackageName: Bikeshop

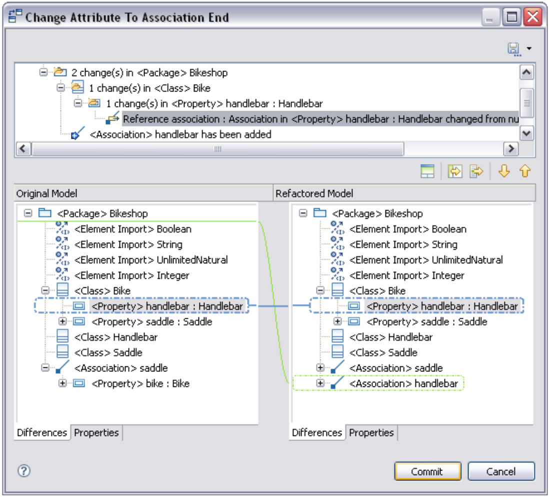
className: Bike

srcProperty: bike

propName: handlebar

Figure: Parameter definition wizard.

Refactoring *Change Attribute to Association End*



2 change(s) in <Package> Bikeshop
1 change(s) in <Class> Bike
1 change(s) in <Property> handlebar : Handlebar
Reference association : Association in <Property> handlebar : Handlebar changed from nu
<Association> handlebar has been added

Original Model	Refactored Model
<Package> Bikeshop	<Package> Bikeshop
<Element Import> Boolean	<Element Import> Boolean
<Element Import> String	<Element Import> String
<Element Import> UnlimitedNatural	<Element Import> UnlimitedNatural
<Element Import> Integer	<Element Import> Integer
<Class> Bike	<Class> Bike
<Property> handlebar : Handlebar	<Property> handlebar : Handlebar
<Property> saddle : Saddle	<Property> saddle : Saddle
<Class> Handlebar	<Class> Handlebar
<Class> Saddle	<Class> Saddle
<Association> saddle	<Association> saddle
<Property> bike : Bike	<Association> handlebar

Figure: Result preview using EMF Compare.

Refactoring *Change Attribute to Association End*

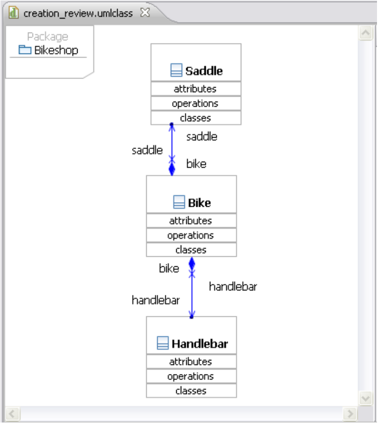


Figure: Refactored model.

Architecture

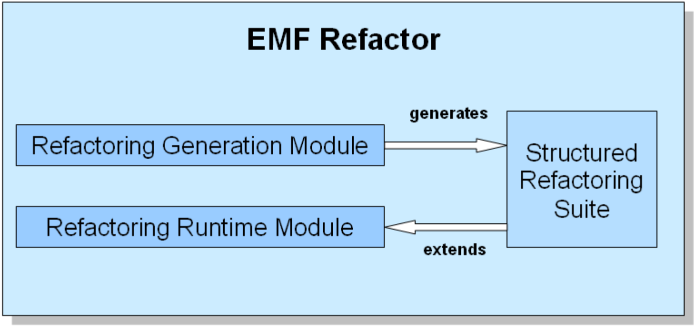

EMF Refactor
Refactoring Generation Module
generates
Structured Refactoring Suite
Refactoring Runtime Module
extends

Figure: Architecture of EMF Refactor.

Refactoring Generation Module

Refactoring Specification

- Specifying model refactorings by several model transformation approaches.
  - EMF Tiger,
  - EWL (Epsilon Wizard Language),
  - Java code, ...
- Combining existing model refactorings to more complex ones (composite refactorings).
  - Using a user-friendly refactoring composition editor.
- Each generated refactoring extends the refactoring suite of EMF Refactor.

Refactoring Runtime Module

Refactoring Application

- Invoking model refactorings from within several editors.
    - Standard EMF instance editor.
    - UML2Tools graphical diagram editors.
- Selection of the model element(s) of interest (context).
- Choosing the appropriate model refactoring.
- Internal use of LTK (Language Toolkit) for refactoring application.
    - Parameter definition by a user-friendly wizard.
    - Support for result preview, undo, redo, and significant error reporting.

Scope

Additional Features

- Invoking model refactorings from within generated editors.
  - Editors generated by GMF (Graphical Modeling Framework).
  - Editors generated by Xtext.
- Combining refactorings of different modeling languages.
  - Especially model and code refactorings.
- Integration with a model smell tool.
  - Application of refactorings in a quick-fix manner.

Used Eclipse Components

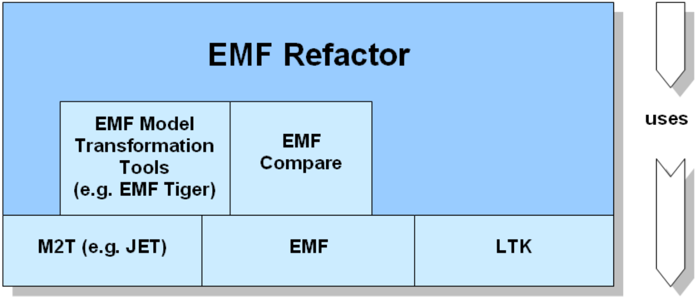


Figure: Eclipse components used by EMF Refactor.

Code Contributions

Initial contribution

- Refactoring Generation Module that generates code for each defined model refactoring and extends the refactoring suite.
  - Uses EMF Tiger for refactoring specification.
- A suite of more than 23 predefined refactorings for Ecore and UML2 models.
- Refactoring Application Module that allows to select a model refactoring, to set all parameters needed, to preview the result of a refactoring, and to actually perform a refactoring.
  - Refactoring invocation from within EMF instance editor and UML2Tools.
  - EMF Compare for result preview and EMF Tiger interpreter for refactoring application.

Code Contributions

Later contributions

1. Model refactoring application using LTK.
2. Model refactoring specification by EWL, Java code and other model transformation approaches.
3. Editor for specifying composite refactorings.
4. Model refactoring invocation from within generated GMF and Xtext editors.
5. Integration into a model smell tool.
6. Combined refactorings of several models or model and code.

Ongoing contributions will extend the suite of predefined refactorings for Ecore and UML2 models.

Mentors and Committers

Mentors

- Ed Merks – Eclipse Modeling, EMF, Macro Modeling, Itemis
- Bernd Kolb – SAP AG

Initial committers

- Thorsten Arendt (project lead) – Philipps-Universität Marburg, Germany
- Florian Mantz (committer) – Philipps-Universität Marburg, Germany
- Lars Schneider (committer) – Capgemini sd&m, Offenbach, Germany

Initial committers

Thorsten Arendt (proposed project lead):

- PhD candidate and scientific staff member at Dept. of Mathematics and Computer Science, working group Software Engineering at the Philipps-Universität Marburg.
- Research focus on model quality, formal definition of model quality, and model quality assurance techniques.
- Diploma thesis on standard conform development of software in medical devices.
- 7 years experience in software development with Eclipse and Java.
- Supervises diploma and bachelor projects on model quality assurance.

Initial committers

Florian Mantz:

- Scientific staff member at Dept. of Mathematics and Computer Science, working group Software Engineering at the Philipps-Universität Marburg.
- Research interests: model based software development, model quality, and model quality assurance techniques (model metrics, model smells, and model refactorings).
- Diploma thesis on syntactic quality assurance techniques for software models.
- 9 years experience in software development with Java.
- 8 years experience in software development with Eclipse.

Initial committers

Lars Schneider:

- Working for Capgemini sd&m as a software engineer.
- Diploma thesis on the development of a refactoring plug-in for the Eclipse Modeling Framework.
- 7 years experience in software development with Java.
- 5 years experience in software development with Eclipse.
- 2 years experience in Eclipse plug-in development.

Roadmap

Tentative plan

- First community technical preview: spring 2010.
- First release: autumn 2010.