# fUML Refactoring with EMF

## Business Informatic Group

Kristof Meixner
Sebastian Geiger

6 Mai 2014

# Overview

- Refactoring Overview
- UML Models and Refactoring
- Semantic Preservation
- Insurance Company Example
- fUML Introduction
- fUML Refactoring
- Refactoring Constraints with OCL
- Toolchain
- EMF Refactor

# Refactoring Overview

- What is refactoring?
  - "defines a set of program restructuring operations" that "preserve the behavior of a program" [6]
- Why do we need it?
  - Increases software and/or model quality
  - Ensures reusability of components
  - Supports change management in software lifecycle
- Examples: rename class, extract superclass, encapsulate field.
- Detailed catalogues with refactorings exist (e.g. [1])

# UML Repetition

- Unified Modeling Language (v2.4.1) standardized by Object Management Group [4]
- General-purpose modeling language in the field of software engineering (Wikipedia)
- Includes different diagram types for architecture structure & behavior
- Allows contraint definition via OCL [3]

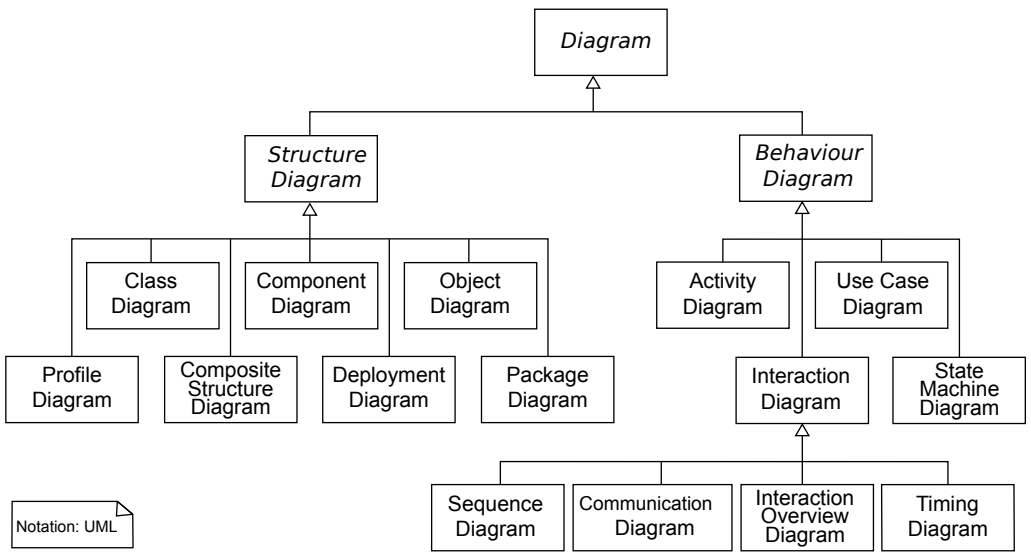



Figure : *UML* diagram type hierarchy (Derfel73, PMerson)

# UML Models and Refactoring

- Whats the difference between source code and model refactoring?
  - Consider all interconnected views/diagrams
  - Consider model constraints
  - Consider different abstraction levels
    - Not all aspects fully modeled
- Example:
  - In Java fields and methods are directly in one class
  - In UML Activities are modeled separate from Classes

# Semantic Preservation

- How to preserve semantics and verify models?
  - Static analysis: Specify pre- and postconditions with OCL constraints
  - Validate refactored models.
  - Dynamics analysis: Execute models and analyse behavior and execution properties (trace).
- What means semantic preservation?
  - Same execution trace?
  - Same output?
  - Same state?
- Depends on refactoring!

# Insurance Company Example 1/3

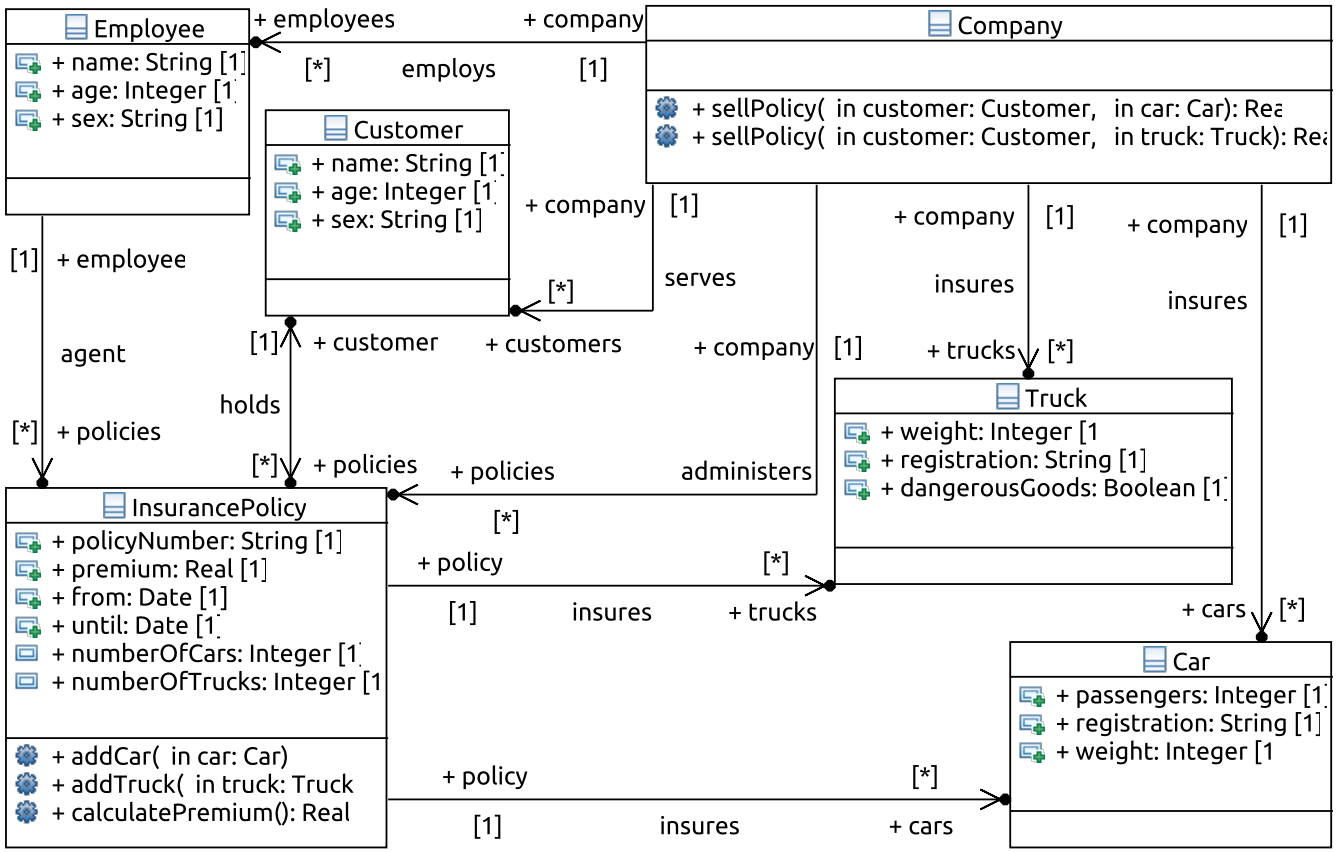



Figure : Insurance class diagram

# Insurance Company Example 2/3

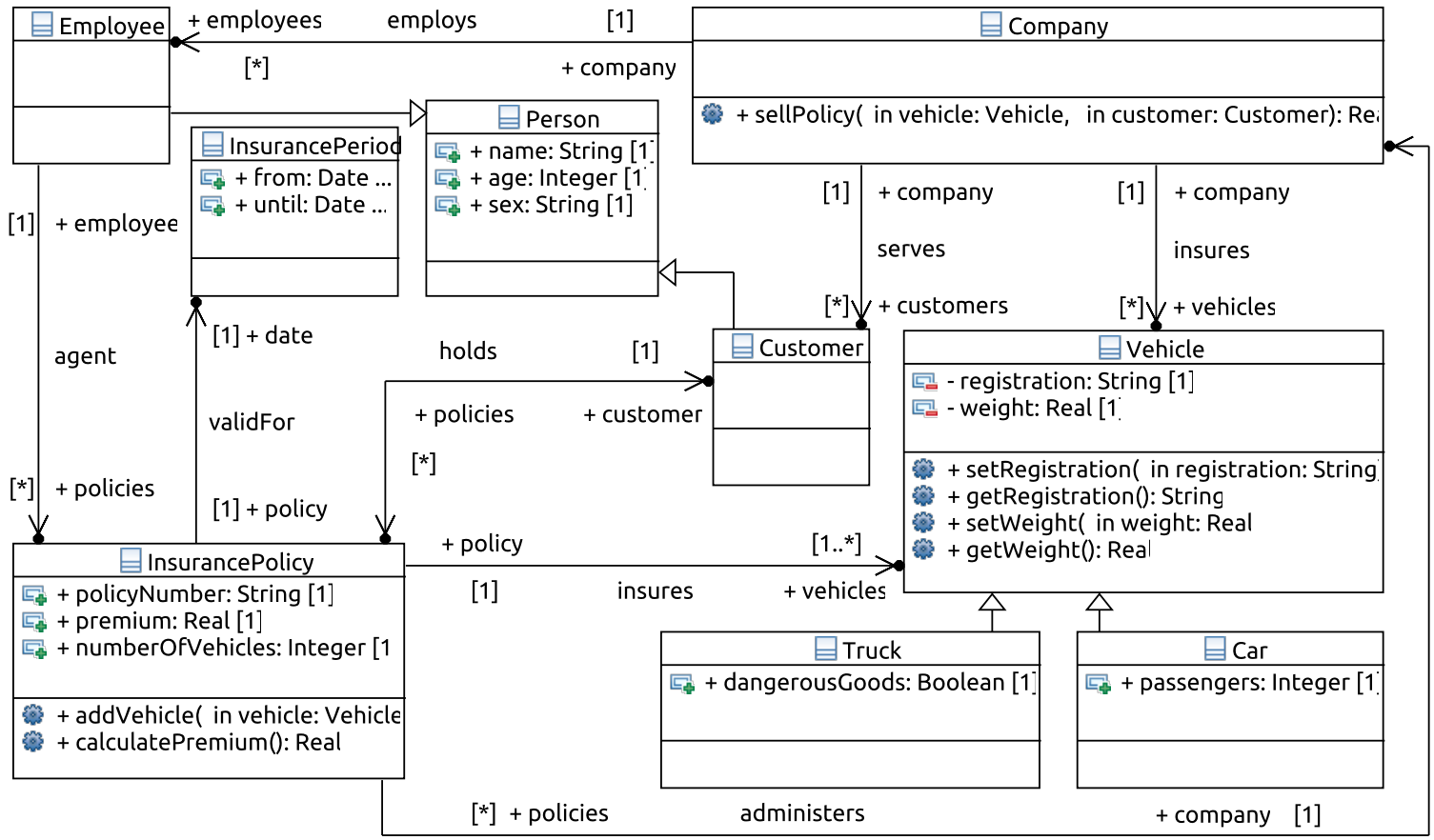



Figure : Insurance class diagram with refactorings

# Insurance Company Example 3/3

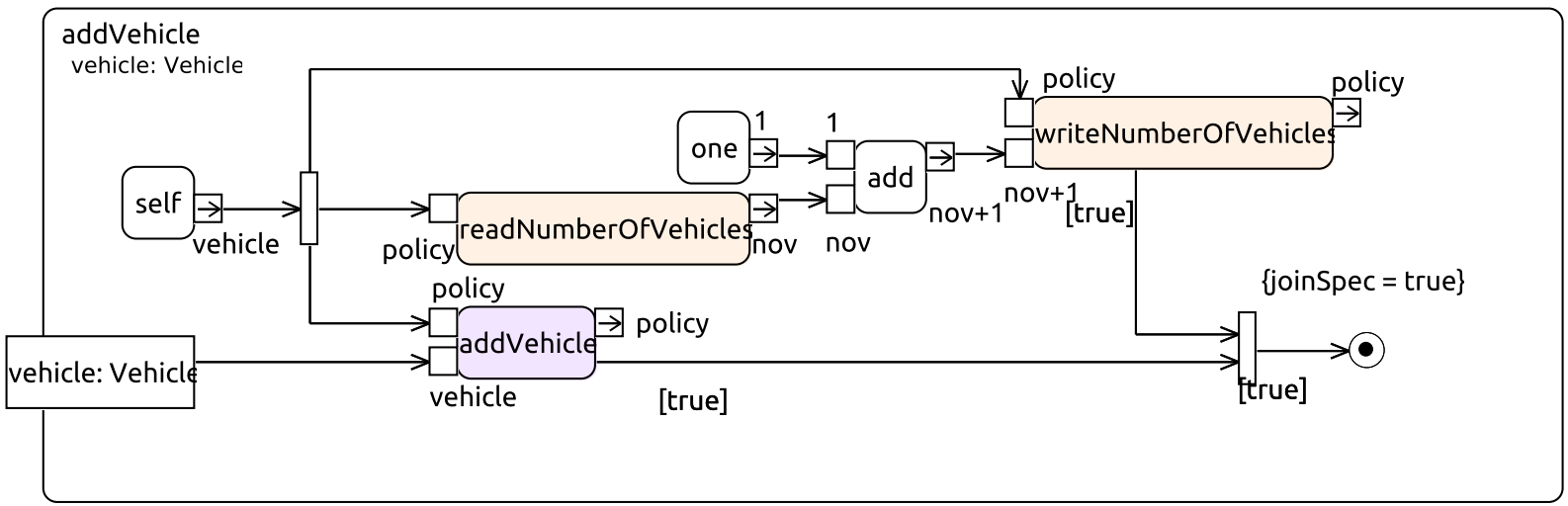



Figure : Add vehicle activity

# fUML Introduction

- fUML = foundational UML [5]
- fUML 1.1 is based on UML 2.4.1
- Subset of UML (Class and Activity diagrams)
- Enhanced with consise semantics
- Turing complete and allows execution or interpretation
- Existing VM to execute models
- Extended VM for testing and debugging (Moliz) [2]

# fUML Abstract Syntax for Classifiers

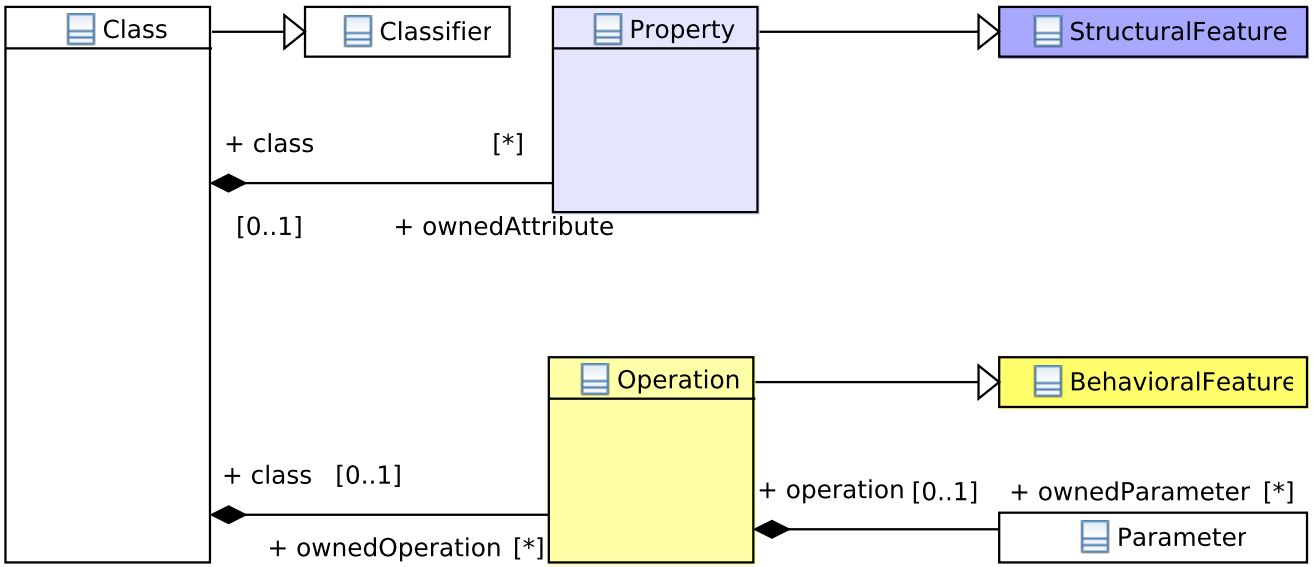
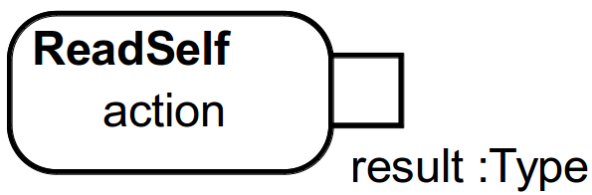



Figure : Classifiers in fUML

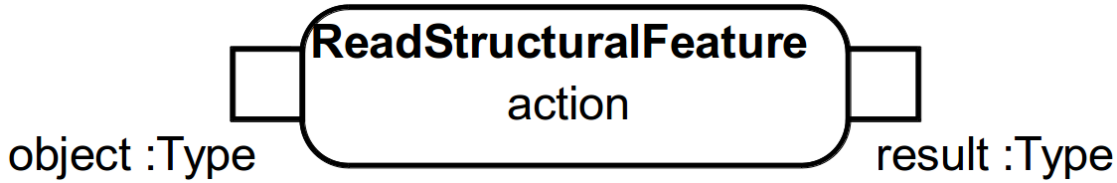
# fUML Actions

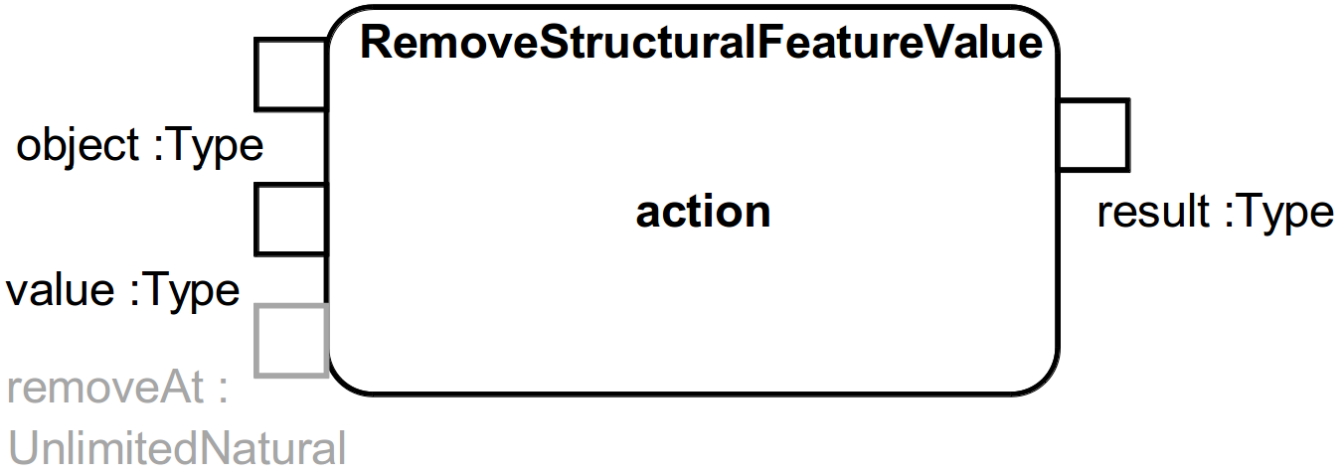
Actions provide the functionality of activity diagrams. Every behavior is based on an action.

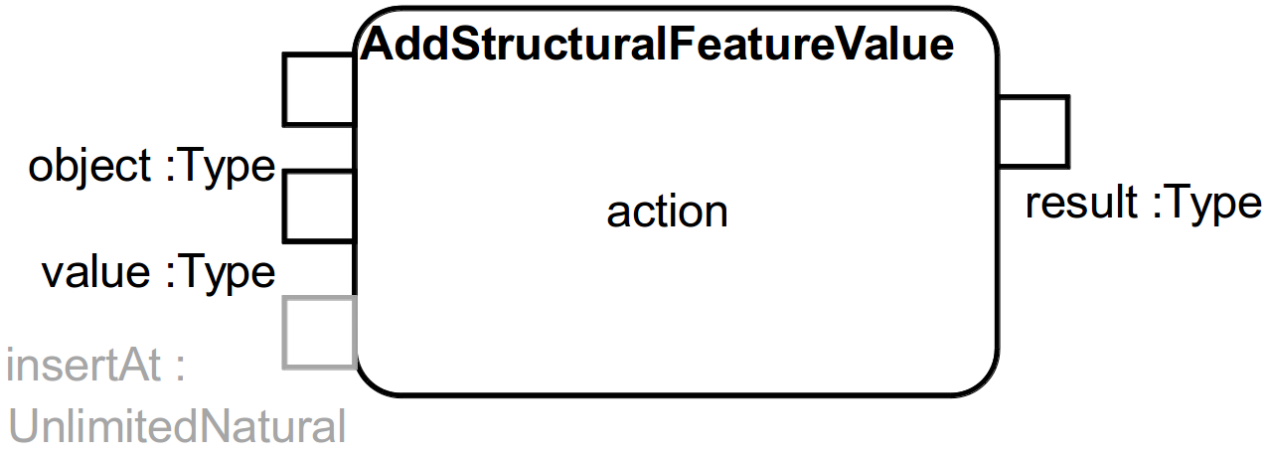
- ReadSelfAction
- AddStructuralFeatureValueAction
- RemoveStructuralFeatureValueAction
- ReadStructuralFeatureAction
- WriteStructuralFeatureAction
- ValueSpecification



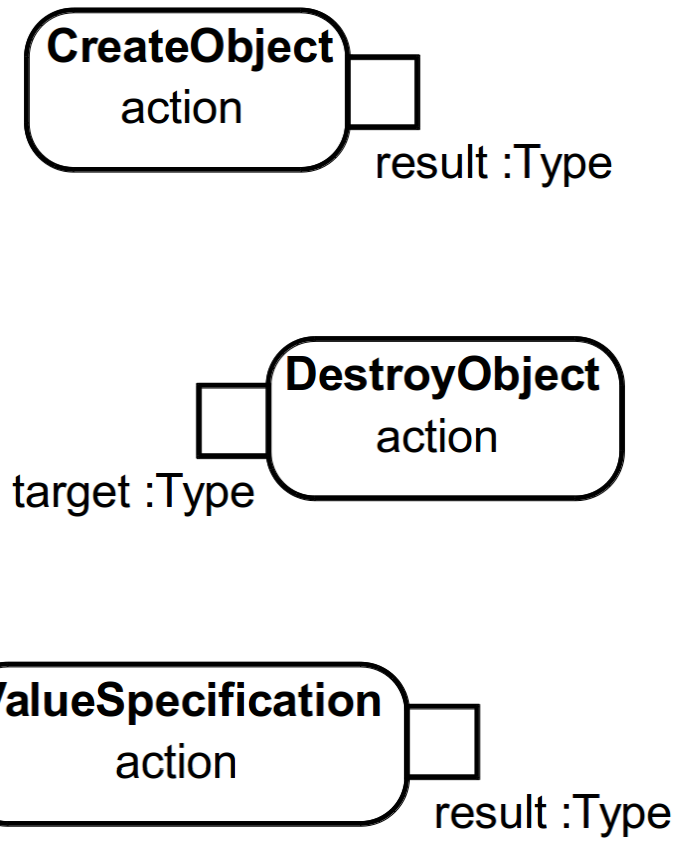
# fUML Actions (2)



# fUML Actions (3)



# fUML Abstract Syntax for Actions

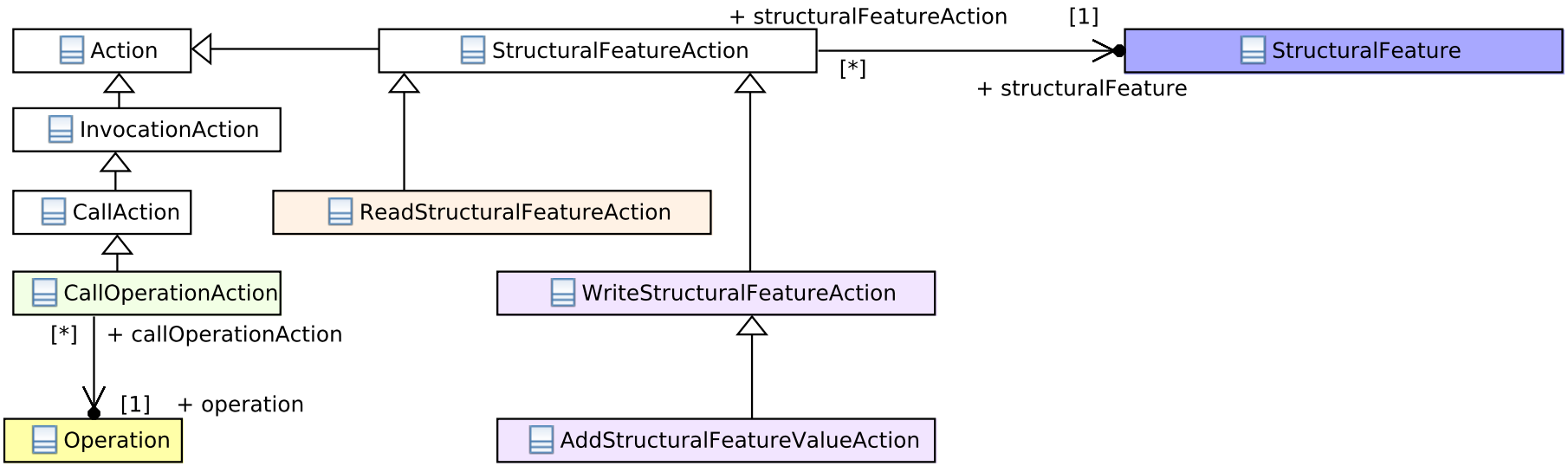
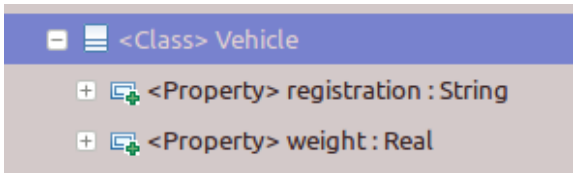



Figure : Actions in fUML

# Encapsulate Field Prerefactoring

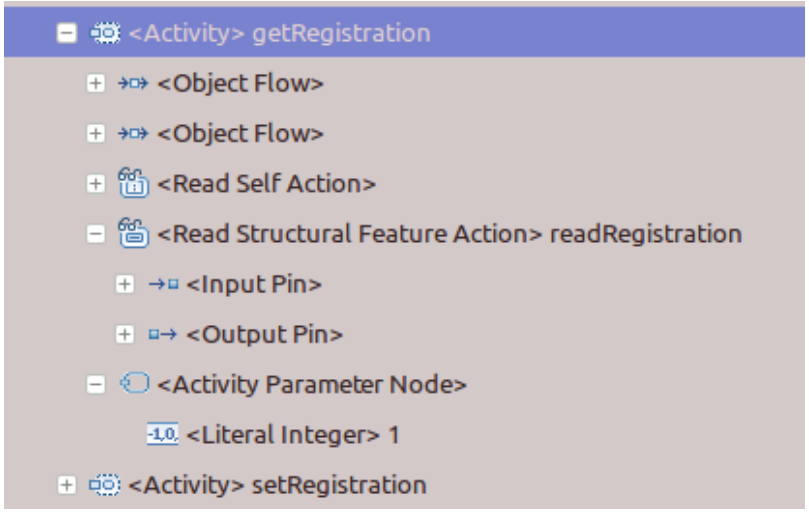
Public field (property) **policyNumber**:



# Encapsulate Field Postrefactoring

Property is private, operations and activities have been added:


The activity for the getter:



# Refactoring Constraints with OCL

Constraint ensure that the model is in a good state before and after the refactoring.
Example:

```
1 context Property:
2 pre : self.visibility <>
3             uml::VisibilityKind::private and
4       self.class.ownedOperation
5       ->forAll(o | o.isDistinguishableFrom(
6             setOperation,
7             self.namespace) and
8       o.isDistinguishableFrom(getOperation,
9                                     self.namespace))
```

# Refactoring Constraints with OCL

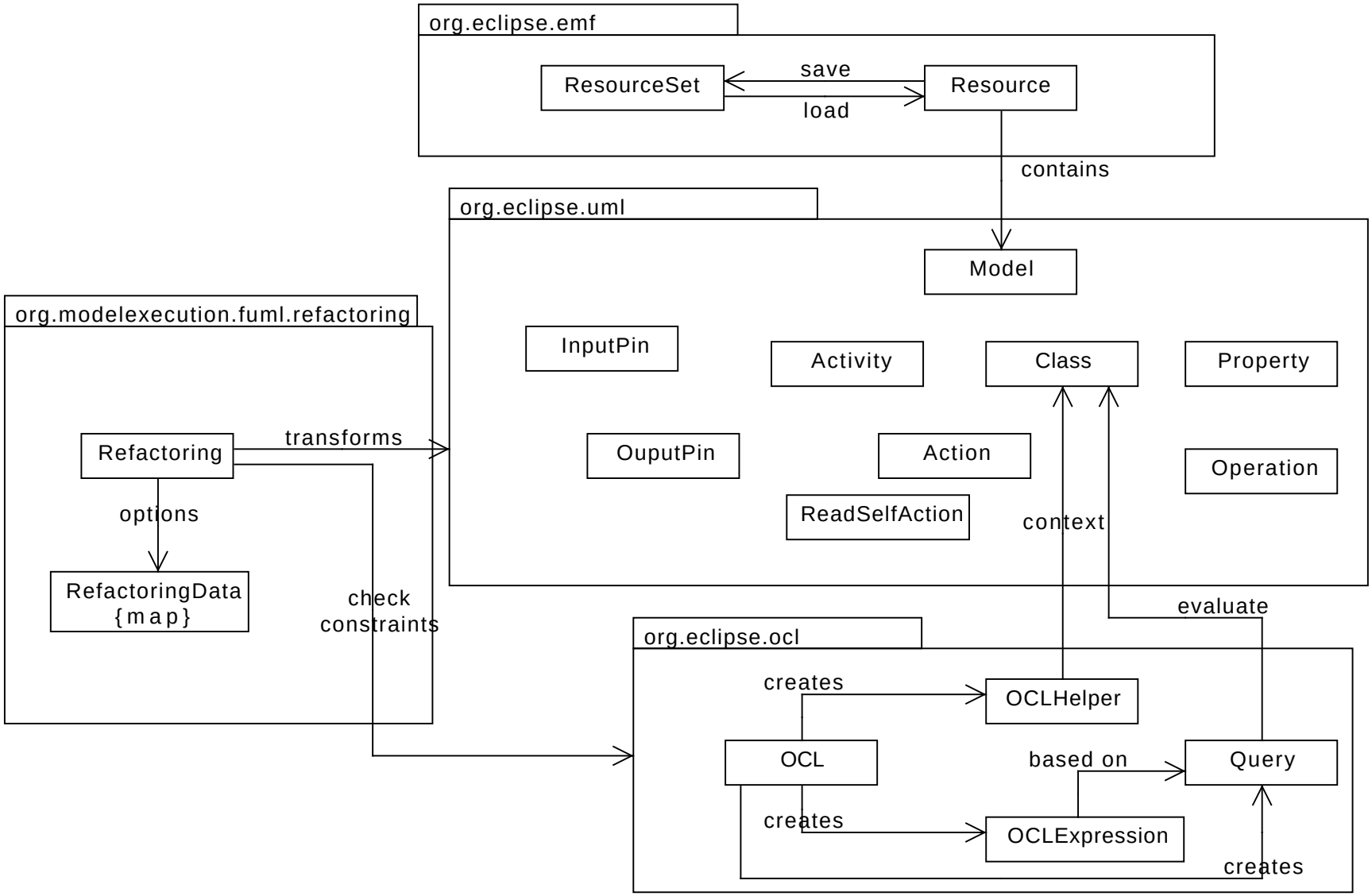
Constraint for searching parts of the model that need to be adapted.
Example:

```
1 context Package:
2 self.member->select(c|c.oclIsTypeOf(Class)).
3     oclAsType(Class).member->
4     select(a|a.oclIsTypeOf(Activity)).
5     oclAsType(Activity).node->
6     select(n|n.oclIsTypeOf(
7     ReadStructuralFeatureAction)).
8     oclAsType(ReadStructuralFeatureAction).
9     structuralFeature
```

# Toolchain

- Used Eclipse Modeling Framework and Ecore
- Java implementation of UML 2.4.1 (org.eclipse.uml2.uml)
- Created constraints with Eclipse OCL Console
- Evaluate constraints with OCL Java API (org.eclipse.ocl)
- Model transformation is performed through UML's abstract syntax.

# Toolchain


org.eclipse.emf
ResourceSet
save
load
Resource
contains
org.eclipse.uml
Model
org.modelexecution.fuml.refactoring
InputPin
Activity
Class
Property
Refactoring
transforms
OuputPin
Action
Operation
options
ReadSelfAction
context
RefactoringData
{map}
check
constraints
evaluate
org.eclipse.ocl
creates
OCLHelper
OCL
based on
Query
creates
OCLExpression
creates

# EMF Refactor

- Framework for Eclipse to
  - check for “model smells” and show metrics (e.g. complexity)
  - refactor models and their properties
  - define your own refactorings/metrics
- Suggests refactorings based on metrics
- Generates new stubs for Java, OCL, Henshin and ComRel
- Is Gui/Wizard based and works for
  - Papyrus
  - EMF Treeeditor

# Questions?

# References



Fowler, M.
*Refactoring - Improving the Design of Existing Code*.
AddisonWesley, July 1999.


Mayerhofer, T., Langer, P., and Kappel, G.
A runtime model for fuml.
In *Models@run.time* (2012), pp. 53–58.


OMG.
*OMG Object Constraint Language*, 2.3.1 ed.
OMG, http://www.omg.org/spec/OCL/2.3.1, 01 2011.


OMG.
*OMG Unified Modeling Language*, 2.4.1 ed.
OMG, http://www.omg.org/spec/UML/2.4.1/, 05 2011.


OMG.
*Semantics of a Foundational Subset for Executable UML Models*, 1.1 ed.
OMG, http://www.omg.org/spec/FUML/1.1, 08 2013.