Actifsource

Language Workbench Challenge 2011

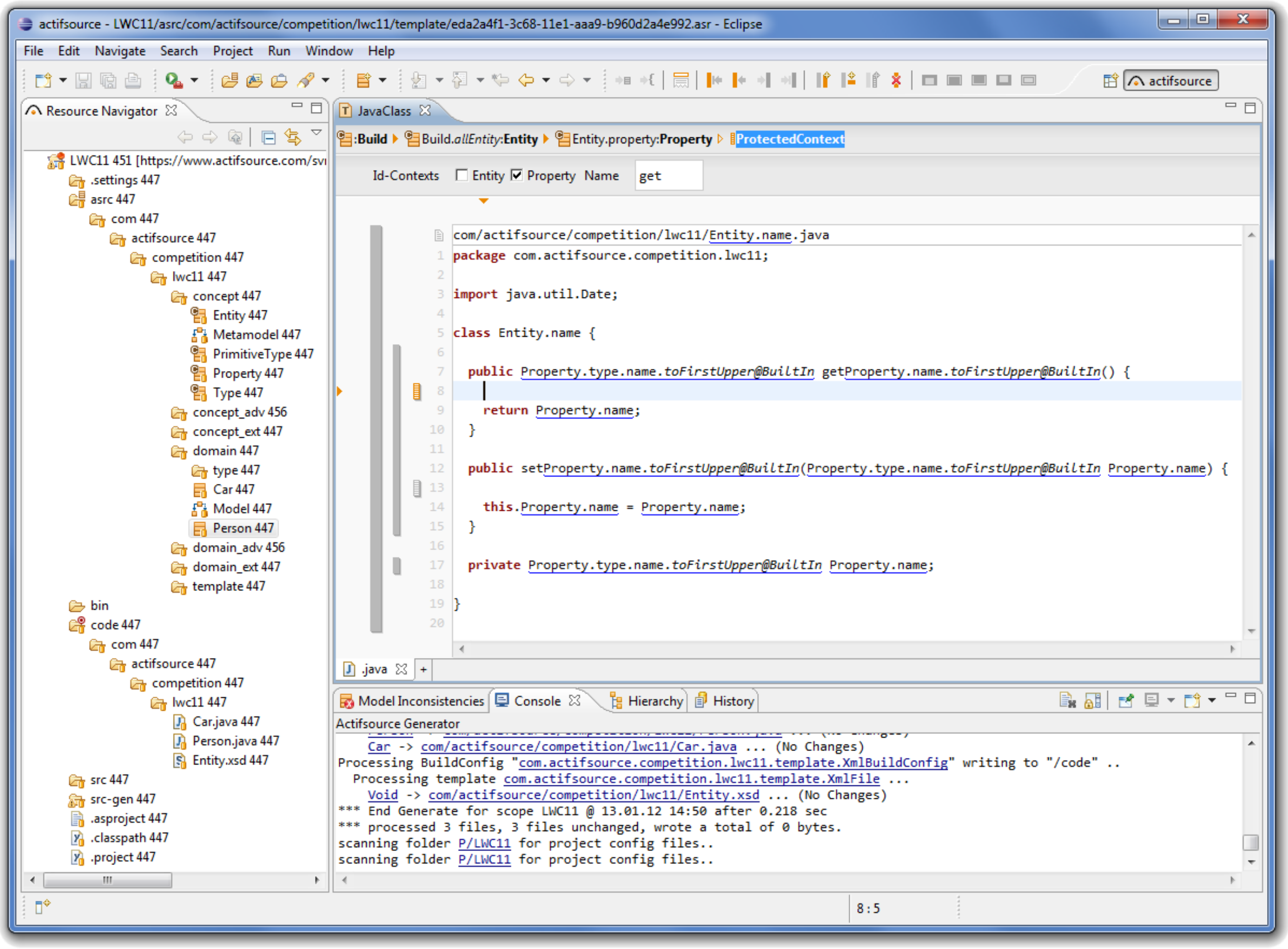
This paper shows how the problems of the Language Workbench Challenge 2011 are solved with Actifsource. The tasks of the LWC11 are described in the assignments at http://www.languageworkbenches.net. Further information about Actifsource can be found at http://www.actifsource.com.

LWC11

Phase 0 - Basics

This phase is intended to demonstrate basic language design, including IDE support (code completion, syntax coloring, outlines, etc).

Actifsource is implemented as Plugin to the Eclipse IDE and fully integrated into this environment.

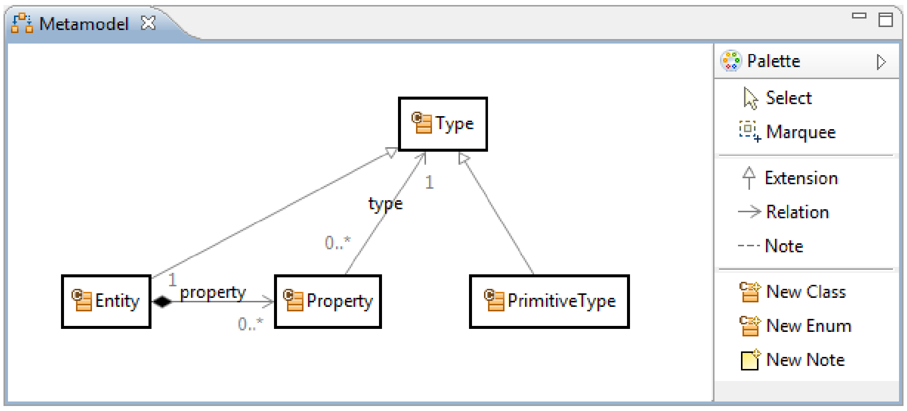


```
com/actifsource/competition/lwc11/Entity.name.java
package com.actifsource.competition.lwc11;

import java.util.Date;

class Entity.name {

  public Property.type.name.toFirstUpper@BuiltIn getProperty.name.toFirstUpper@BuiltIn() {
    
    return Property.name;
  }

  public setProperty.name.toFirstUpper@BuiltIn(Property.type.name.toFirstUpper@BuiltIn Property.name) {

    this.Property.name = Property.name;
  }

  private Property.type.name.toFirstUpper@BuiltIn Property.name;

}
```

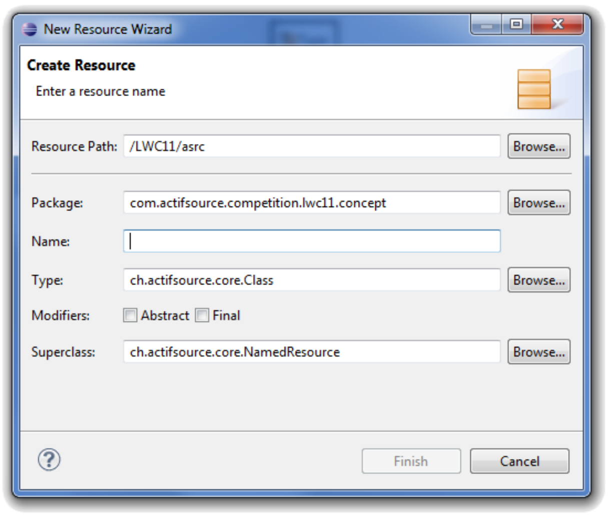
```
    Car -> com/actifsource/competition/lwc11/Car.java ... (No Changes)
Processing BuildConfig "com.actifsource.competition.lwc11.template.XmlBuildConfig" writing to "/code" ..
  Processing template com.actifsource.competition.lwc11.template.XmlFile ...
    Void -> com/actifsource/competition/lwc11/Entity.xsd ... (No Changes)
*** End Generate for scope LWC11 @ 13.01.12 14:50 after 0.218 sec
*** processed 3 files, 3 files unchanged, wrote a total of 0 bytes.
scanning folder P/LWC11 for project config files..
scanning folder P/LWC11 for project config files..
```

0.1 Simple (structural) DSL without any fancy expression language or such.

The meta-model is created straight-forward in the graphical Class Diagram editor:



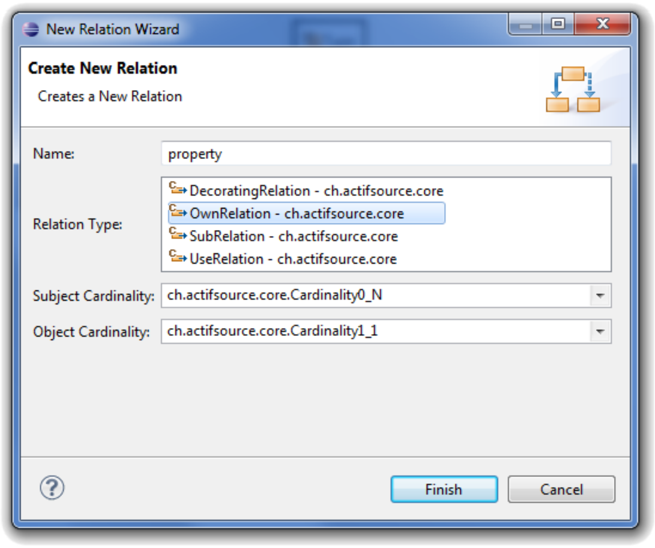
Using the *New Class* and *Relation* tool, objects are created by just clicking into the diagram area and choosing the desired type in a dialog.


New Resource Wizard

Create Resource

Enter a resource name

Resource Path:	/LWC11/asrc
Package:	com.actifsource.competition.lwc11.concept
Name:	
Type:	ch.actifsource.core.Class
Modifiers:	☐ Abstract ☐ Final
Superclass:	ch.actifsource.core.NamedResource


New Relation Wizard

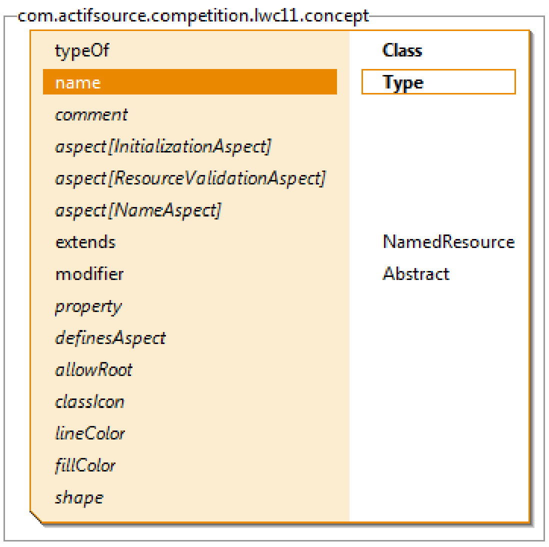
Create New Relation

Creates a New Relation

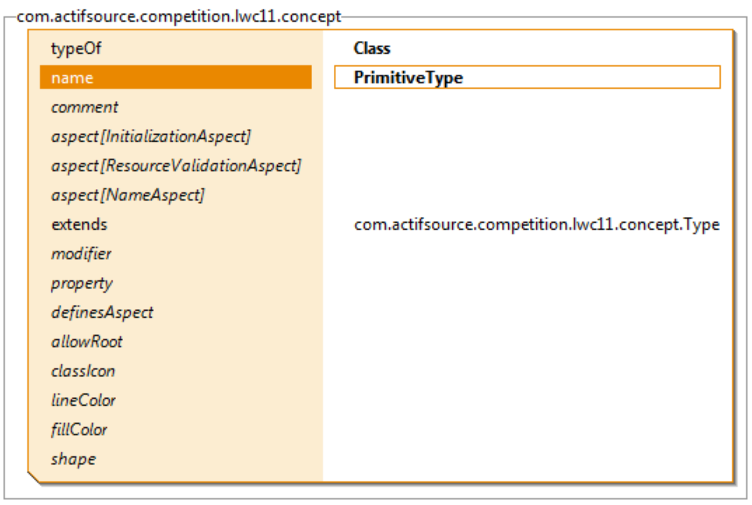
Name:	property
Relation Type:	DecoratingRelation - ch.actifsource.core OwnRelation - ch.actifsource.core SubRelation - ch.actifsource.core UseRelation - ch.actifsource.core
Subject Cardinality:	ch.actifsource.core.Cardinality0_N
Object Cardinality:	ch.actifsource.core.Cardinality1_1

Ctrl+Click on a diagram element shows the respective model element in the Resource editor.

The following elements are created:

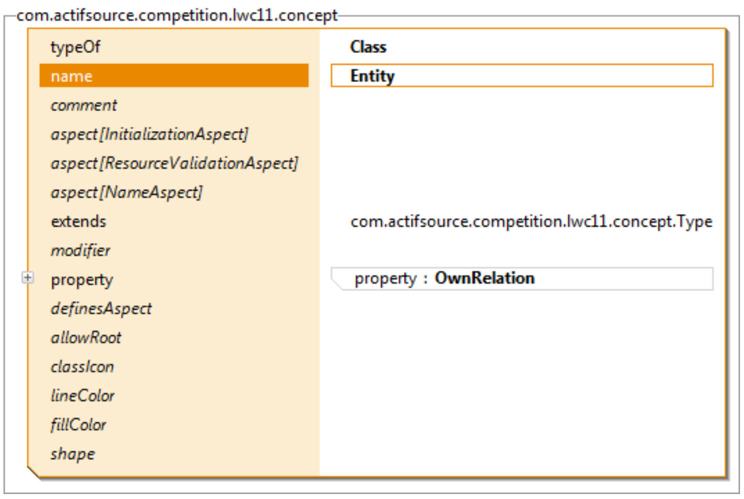

com.actifsource.competition.lwc11.concept

typeOf	Class
name	**Type**
comment	
aspect[InitializationAspect]	
aspect[ResourceValidationAspect]	
aspect[NameAspect]	
extends	NamedResource
modifier	Abstract
property	
definesAspect	
allowRoot	
classIcon	
lineColor	
fillColor	
shape	

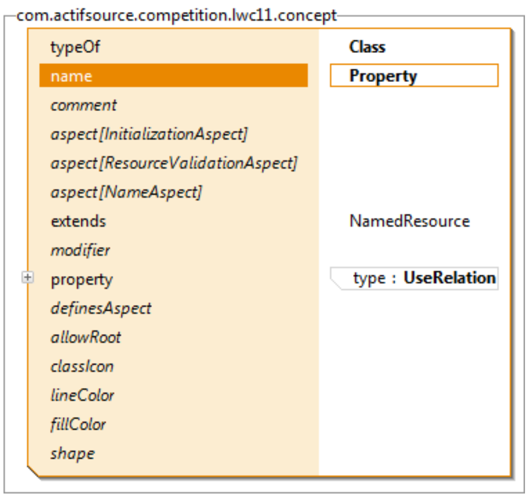

com.actifsource.competition.lwc11.concept

typeOf	Class
name	**PrimitiveType**
comment	
aspect[InitializationAspect]	
aspect[ResourceValidationAspect]	
aspect[NameAspect]	
extends	com.actifsource.competition.lwc11.concept.Type
modifier	
property	
definesAspect	
allowRoot	
classIcon	
lineColor	
fillColor	
shape	

Type has a (unique) *name*, and no other properties. It cannot be instantiated.
PrimitiveType is a concrete subtype of *Type* with no other properties than the inherited *name*.


com.actifsource.competition.lwc11.concept

typeOf	Class
name	**Entity**
comment	
aspect[InitializationAspect]	
aspect[ResourceValidationAspect]	
aspect[NameAspect]	
extends	com.actifsource.competition.lwc11.concept.Type
modifier	
property	property : **OwnRelation**
definesAspect	
allowRoot	
classIcon	
lineColor	
fillColor	
shape	

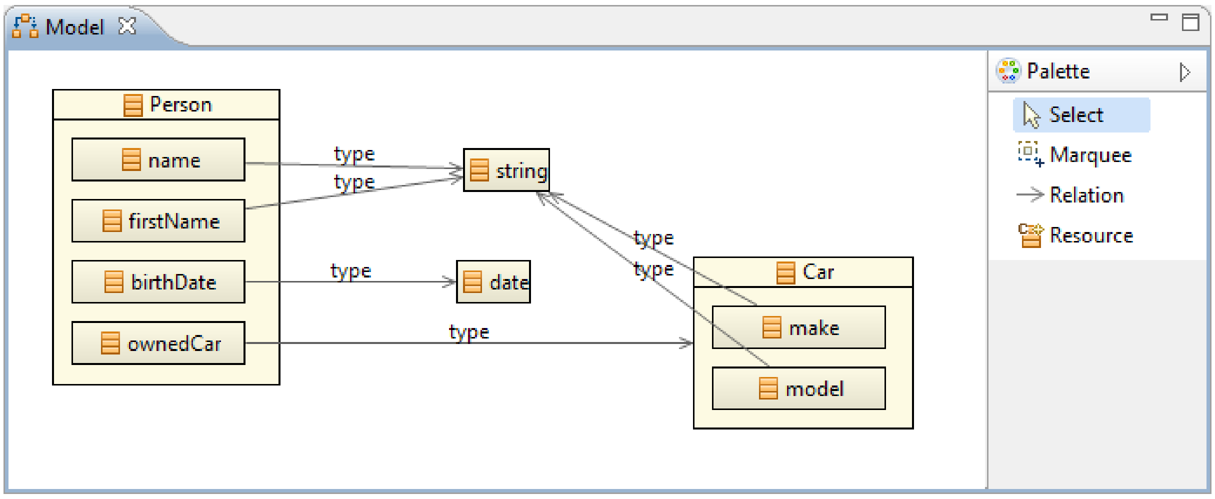

com.actifsource.competition.lwc11.concept

typeOf	Class
name	**Property**
comment	
aspect[InitializationAspect]	
aspect[ResourceValidationAspect]	
aspect[NameAspect]	
extends	NamedResource
modifier	
property	type : **UseRelation**
definesAspect	
allowRoot	
classIcon	
lineColor	
fillColor	
shape	

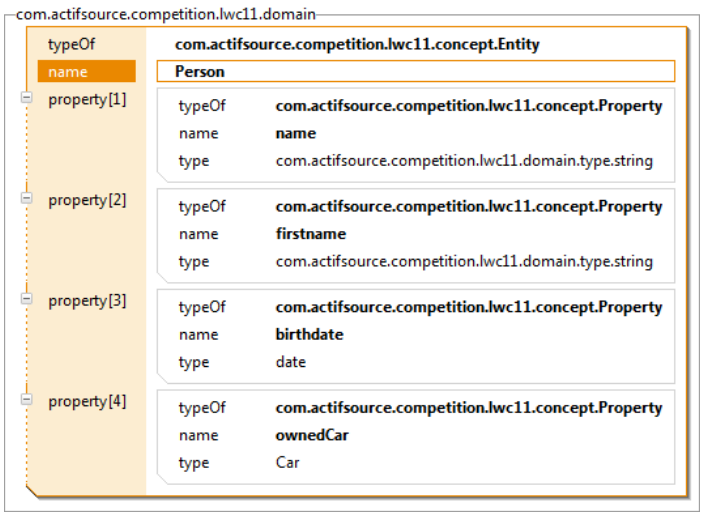
Entity is also a concrete subtype of *Type*. It has a property *property* which points to the type Property, and the *name* it inherits from *Type*.

Property has a property *type*, which points to *Type*. The *Type* is not owned but shared among all properties, so *type* is only a *UseRelation*, not an *OwnRelation*.

Users of the Enterprise Edition can create the model itself in a graphical Domain Diagram editor:

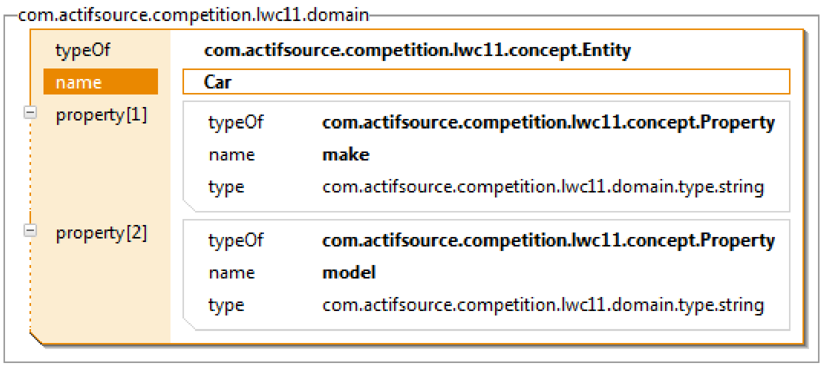


The following elements are created – either by the diagram editor or by hand:



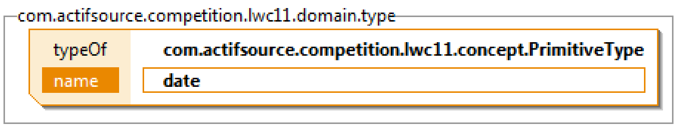
com.actifsource.competition.lwc11.domain

typeOf	com.actifsource.competition.lwc11.concept.Entity
name	Person
property[1]	typeOf com.actifsource.competition.lwc11.concept.Property name name type com.actifsource.competition.lwc11.domain.type.string
property[2]	typeOf com.actifsource.competition.lwc11.concept.Property name firstname type com.actifsource.competition.lwc11.domain.type.string
property[3]	typeOf com.actifsource.competition.lwc11.concept.Property name birthdate type date
property[4]	typeOf com.actifsource.competition.lwc11.concept.Property name ownedCar type Car



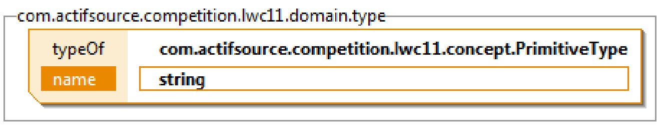
com.actifsource.competition.lwc11.domain

typeOf	com.actifsource.competition.lwc11.concept.Entity
name	Car
property[1]	typeOf com.actifsource.competition.lwc11.concept.Property name make type com.actifsource.competition.lwc11.domain.type.string
property[2]	typeOf com.actifsource.competition.lwc11.concept.Property name model type com.actifsource.competition.lwc11.domain.type.string



com.actifsource.competition.lwc11.domain.type

typeOf	com.actifsource.competition.lwc11.concept.PrimitiveType
name	date

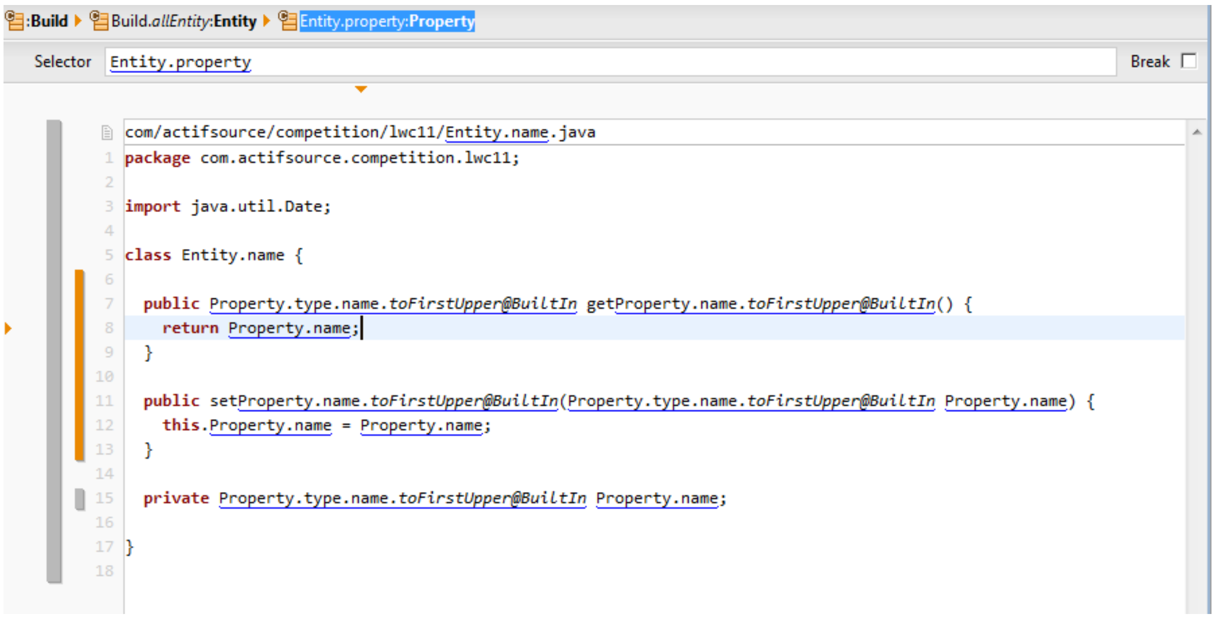


com.actifsource.competition.lwc11.domain.type

typeOf	com.actifsource.competition.lwc11.concept.PrimitiveType
name	string

0.2 Code generation to GPL such as Java, C#, C++ or XML

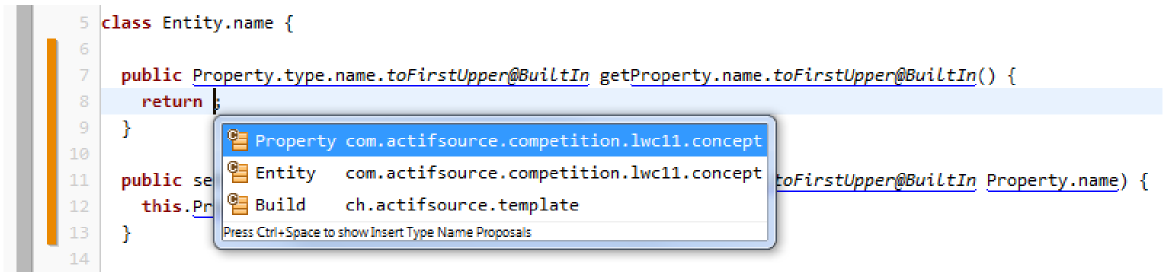
Code is generated using templates.



Selector: Entity.property

```
com/actifsource/competition/lwc11/Entity.name.java
package com.actifsource.competition.lwc11;

import java.util.Date;

class Entity.name {

  public Property.type.name.toFirstUpper@BuiltIn getProperty.name.toFirstUpper@BuiltIn() {
    return Property.name;
  }

  public setProperty.name.toFirstUpper@BuiltIn(Property.type.name.toFirstUpper@BuiltIn Property.name) {
    this.Property.name = Property.name;
  }

  private Property.type.name.toFirstUpper@BuiltIn Property.name;

}
```

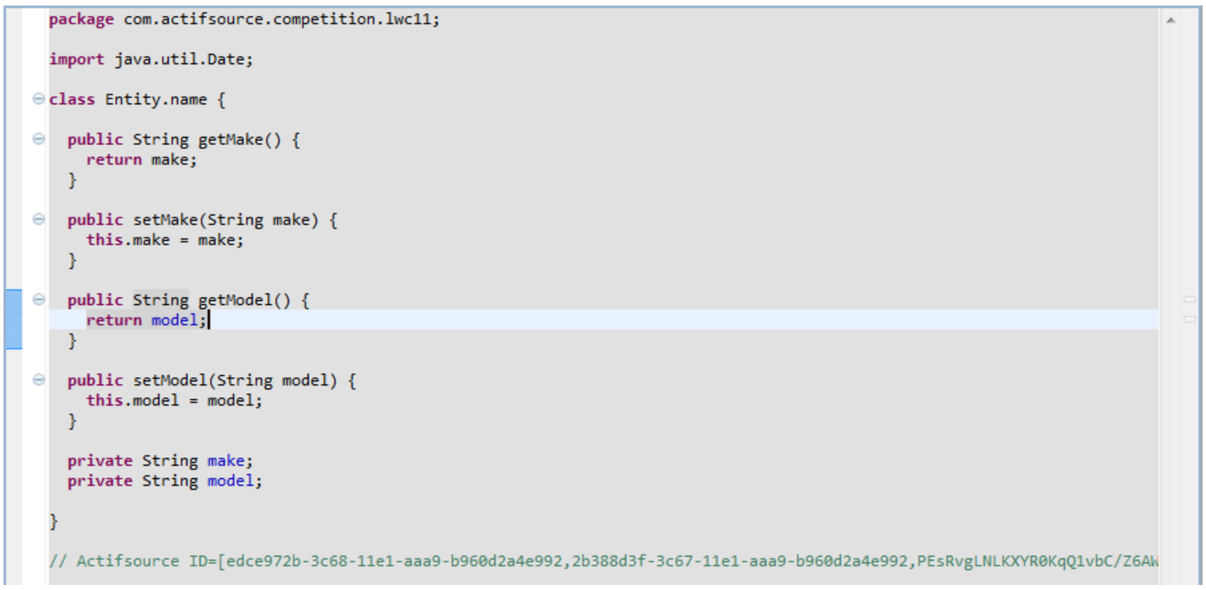
There is no syntax needed to access the model elements – model elements can be selected using the Eclipse QuickAssist feature (Ctrl+Space) as shown below:



```
class Entity.name {

  public Property.type.name.toFirstUpper@BuiltIn getProperty.name.toFirstUpper@BuiltIn() {
    return ;
  }

  public se                                                        toFirstUpper@BuiltIn Property.name) {
    this.Pr
  }
```

```
Property   com.actifsource.competition.lwc11.concept
Entity     com.actifsource.competition.lwc11.concept
Build      ch.actifsource.template
Press Ctrl+Space to show Insert Type Name Proposals
```

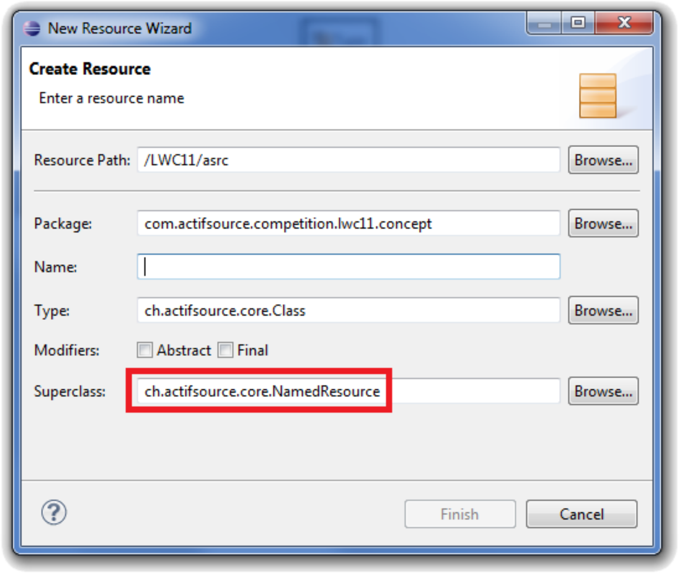
The generated Java file is shown in the common Eclipse Java editor (or the editor, that is registered for the generated file type).



```
package com.actifsource.competition.lwc11;

import java.util.Date;

class Entity.name {

  public String getMake() {
    return make;
  }

  public setMake(String make) {
    this.make = make;
  }

  public String getModel() {
    return model;
  }

  public setModel(String model) {
    this.model = model;
  }

  private String make;
  private String model;

}

// Actifsource ID=[edce972b-3c68-11e1-aaa9-b960d2a4e992,2b388d3f-3c67-11e1-aaa9-b960d2a4e992,PEsRvgLNLKXYR0KqQ1vbC/Z6AW
```

0.3 Simple constraint checks such as name-uniqueness

Name-uniqueness is already checked, if your class extends the built-in class *NamedResource*, which is the regular case.


New Resource Wizard

Create Resource

Enter a resource name

Resource Path: /LWC11/asrc

Package: com.actifsource.competition.lwc11.concept

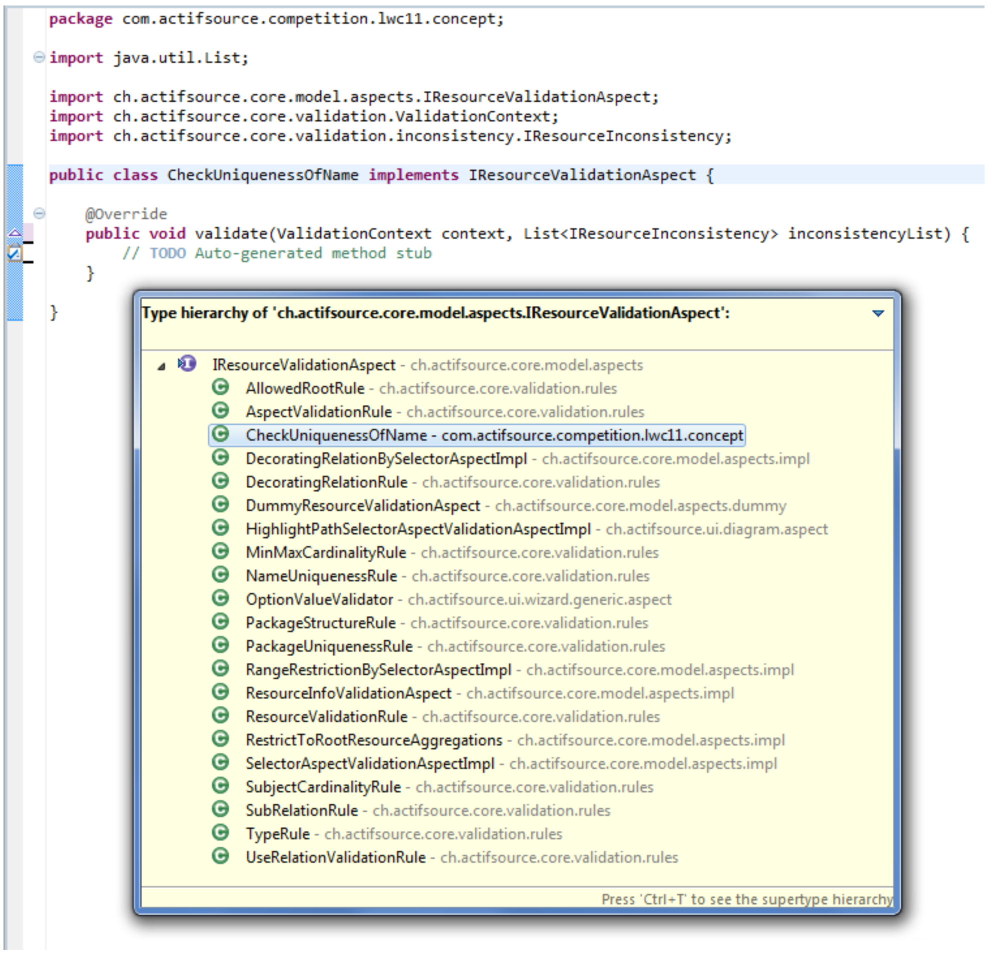
Name:

Type: ch.actifsource.core.Class

Modifiers: Abstract Final

Superclass: ch.actifsource.core.NamedResource

If you are using anonymous classes, that only extend Resource, it is necessary to write a *ResourceValidationAspect* in Java, which checks, whether an object is conflicting and gives some error messages.

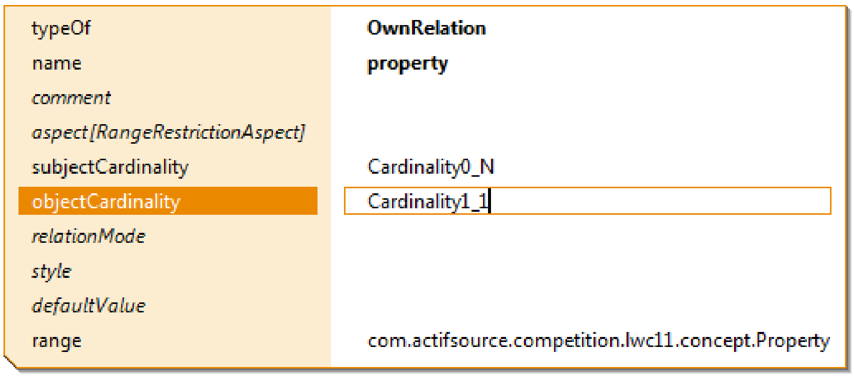

```
package com.actifsource.competition.lwc11.concept;

import java.util.List;

import ch.actifsource.core.model.aspects.IResourceValidationAspect;
import ch.actifsource.core.validation.ValidationContext;
import ch.actifsource.core.validation.inconsistency.IResourceInconsistency;

public class CheckUniquenessOfName implements IResourceValidationAspect {

    @Override
    public void validate(ValidationContext context, List<IResourceInconsistency> inconsistencyList) {
        // TODO Auto-generated method stub
    }

}
```

Type hierarchy of 'ch.actifsource.core.model.aspects.IResourceValidationAspect':

```
IResourceValidationAspect - ch.actifsource.core.model.aspects
    AllowedRootRule - ch.actifsource.core.validation.rules
    AspectValidationRule - ch.actifsource.core.validation.rules
    CheckUniquenessOfName - com.actifsource.competition.lwc11.concept
    DecoratingRelationBySelectorAspectImpl - ch.actifsource.core.model.aspects.impl
    DecoratingRelationRule - ch.actifsource.core.validation.rules
    DummyResourceValidationAspect - ch.actifsource.core.model.aspects.dummy
    HighlightPathSelectorAspectValidationAspectImpl - ch.actifsource.ui.diagram.aspect
    MinMaxCardinalityRule - ch.actifsource.core.validation.rules
    NameUniquenessRule - ch.actifsource.core.validation.rules
    OptionValueValidator - ch.actifsource.ui.wizard.generic.aspect
    PackageStructureRule - ch.actifsource.core.validation.rules
    PackageUniquenessRule - ch.actifsource.core.validation.rules
    RangeRestrictionBySelectorAspectImpl - ch.actifsource.core.model.aspects.impl
    ResourceInfoValidationAspect - ch.actifsource.core.model.aspects.impl
    ResourceValidationRule - ch.actifsource.core.validation.rules
    RestrictToRootResourceAggregations - ch.actifsource.core.model.aspects.impl
    SelectorAspectValidationAspectImpl - ch.actifsource.core.model.aspects.impl
    SubjectCardinalityRule - ch.actifsource.core.validation.rules
    SubRelationRule - ch.actifsource.core.validation.rules
    TypeRule - ch.actifsource.core.validation.rules
    UseRelationValidationRule - ch.actifsource.core.validation.rules
```

Press 'Ctrl+T' to see the supertype hierarchy

For the case, that the referenced object is not only a Literal but a Resource, the uniqueness can by specified by setting the cardinality of an association to 1 resp. 0..1.



typeOf	**OwnRelation**
name	**property**
comment	
aspect [RangeRestrictionAspect]	
subjectCardinality	Cardinality0_N
objectCardinality	Cardinality1_1
relationMode	
style	
defaultValue	
range	com.actifsource.competition.lwc11.concept.Property

0.4 Show how to break down a (large) model into several parts, while still cross-referencing between the parts

In Actifsource, every aggregate structure is saved in its own file. References are handled with *globally unique identifiers* (GUIDs). It is even possible to save the structure in different projects, while still referencing resources from another project.

There can be different visualizations of the model referring to the same model elements. Each of them can contain a subset of elements which it is supposed to visualize.

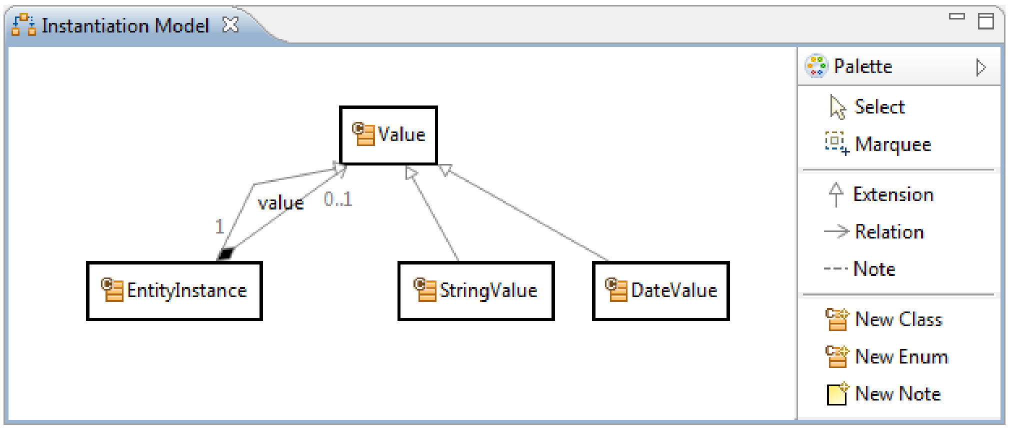
Phase 1 - Advanced

This phase demonstrates advanced features not necessarily available to the same extent in every LWB.

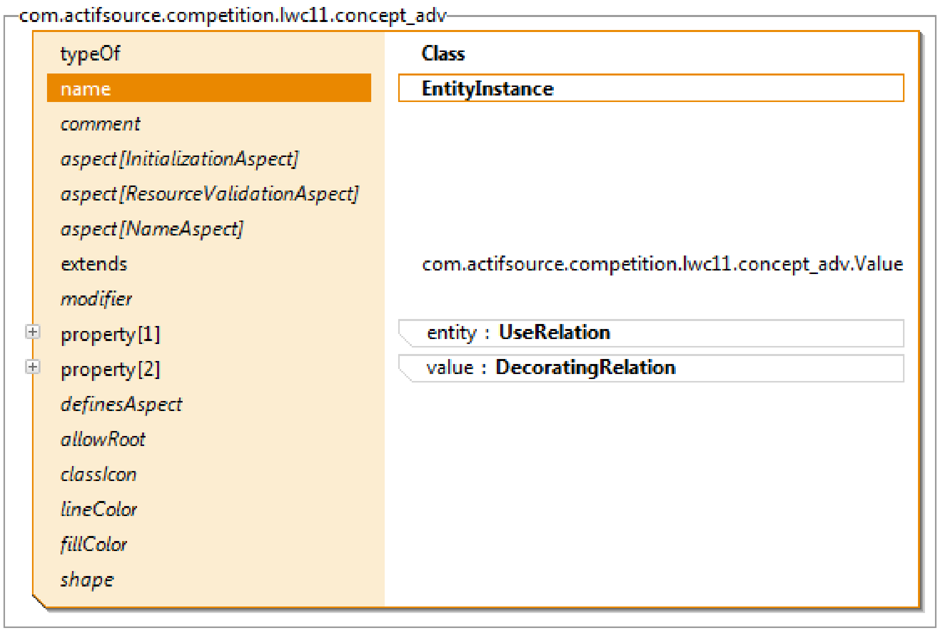
1.1 Show the integration of several languages

There is no instatiation of domain objects provided. As a workaround for instantiation, the properties of the model can be decorated with values.

The following meta-model shows how it is done.



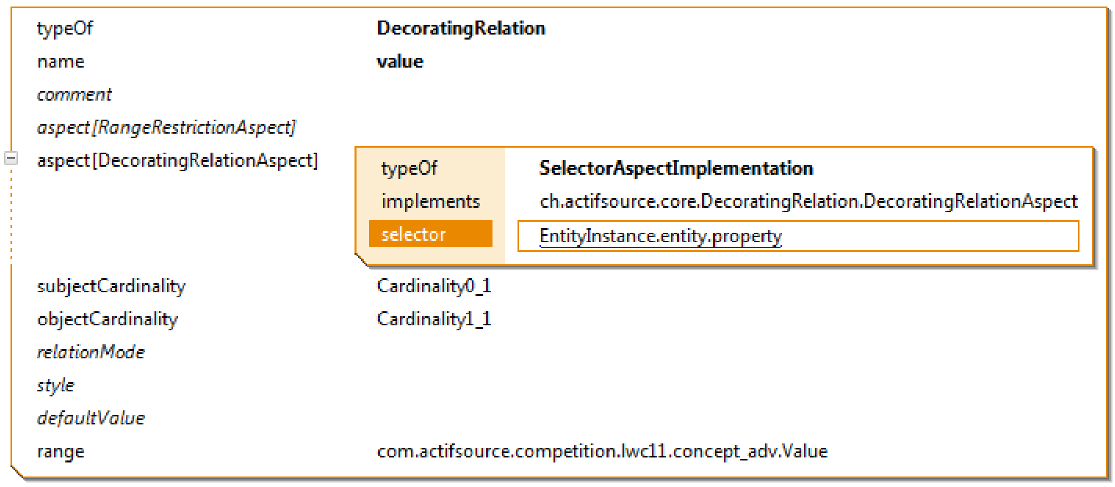
There is a new type *EntityInstance* which refers via the property *entity* to the *Entity* object.



com.actifsource.competition.lwc11.concept_adv

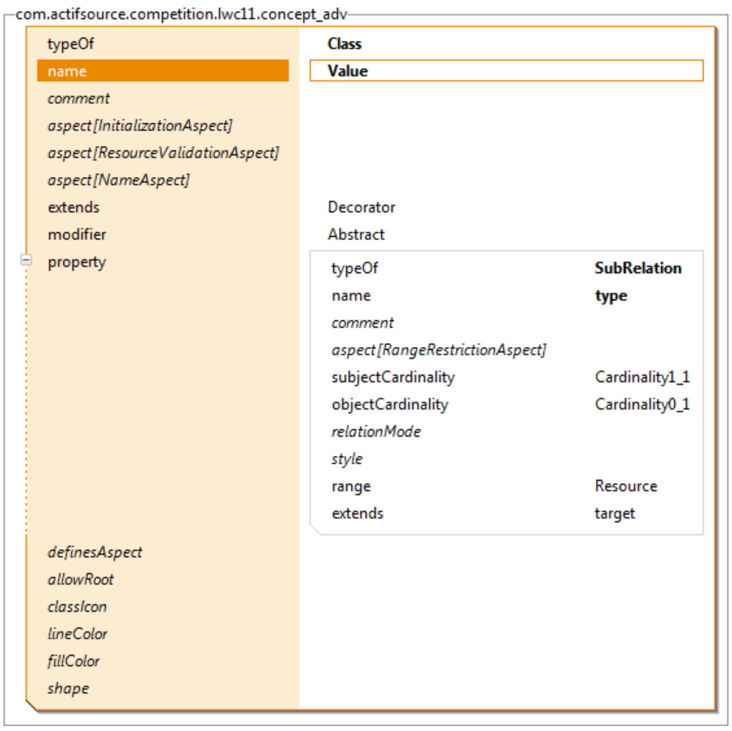
typeOf	**Class**
name	**EntityInstance**
comment	
aspect[InitializationAspect]	
aspect[ResourceValidationAspect]	
aspect[NameAspect]	
extends	com.actifsource.competition.lwc11.concept_adv.Value
modifier	
property[1]	entity : **UseRelation**
property[2]	value : **DecoratingRelation**
definesAspect	
allowRoot	
classIcon	
lineColor	
fillColor	
shape	

The *value* property of *EntityInstance* is a decorates every *property* of the *Entity* with a *Value*.



typeOf	**DecoratingRelation**
name	**value**
comment	
aspect[RangeRestrictionAspect]	
aspect[DecoratingRelationAspect]	typeOf: **SelectorAspectImplementation**; implements: ch.actifsource.core.DecoratingRelation.DecoratingRelationAspect; selector: EntityInstance.entity.property
subjectCardinality	Cardinality0_1
objectCardinality	Cardinality1_1
relationMode	
style	
defaultValue	
range	com.actifsource.competition.lwc11.concept_adv.Value

Value is the abstract type for *EntityInstance*, *StringValue* and *DateValue*.

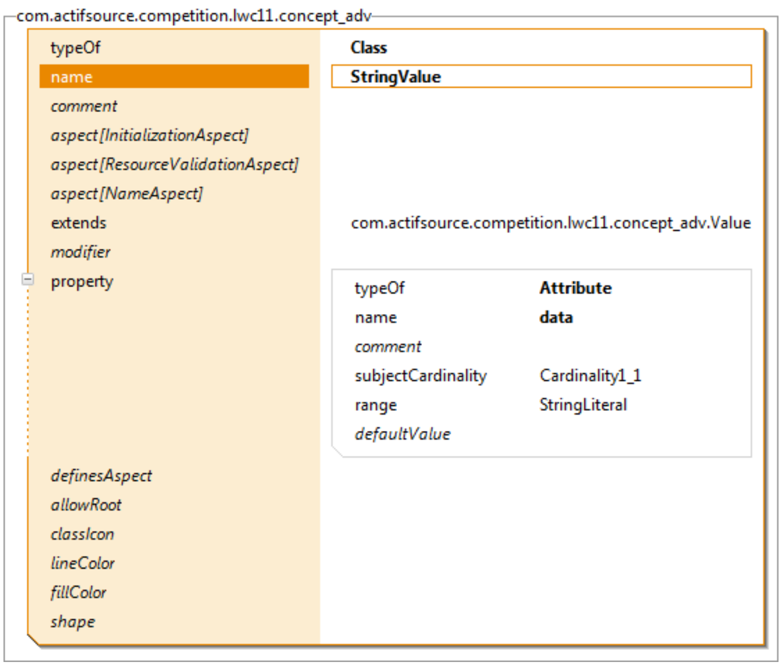


com.actifsource.competition.lwc11.concept_adv

typeOf	**Class**
name	**Value**
comment	
aspect[InitializationAspect]	
aspect[ResourceValidationAspect]	
aspect[NameAspect]	
extends	Decorator
modifier	Abstract
property	typeOf: **SubRelation**; name: **type**; *comment*; *aspect[RangeRestrictionAspect]*; subjectCardinality: Cardinality1_1; objectCardinality: Cardinality0_1; *relationMode*; *style*; range: Resource; extends: target
definesAspect	
allowRoot	
classIcon	
lineColor	
fillColor	
shape	

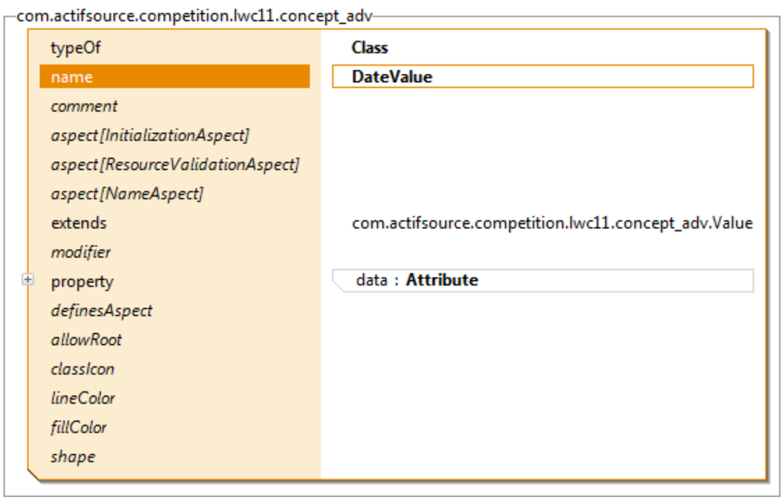
The property *type* will be an alias for the relation to the decorated object.

StingValue and DateValue are implemented as Resources encapsulating the string literals and date literals.



com.actifsource.competition.lwc11.concept_adv

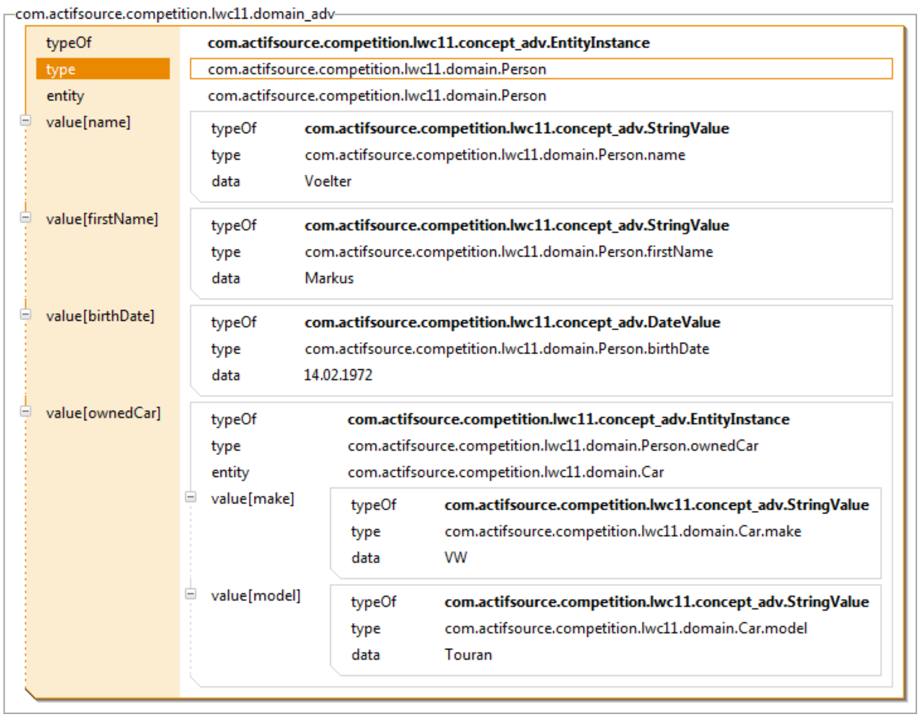
typeOf	**Class**
name	**StringValue**
comment	
aspect [InitializationAspect]	
aspect [ResourceValidationAspect]	
aspect [NameAspect]	
extends	com.actifsource.competition.lwc11.concept_adv.Value
modifier	
property	typeOf **Attribute**; name **data**; *comment*; subjectCardinality Cardinality1_1; range StringLiteral; *defaultValue*
definesAspect	
allowRoot	
classIcon	
lineColor	
fillColor	
shape	



com.actifsource.competition.lwc11.concept_adv

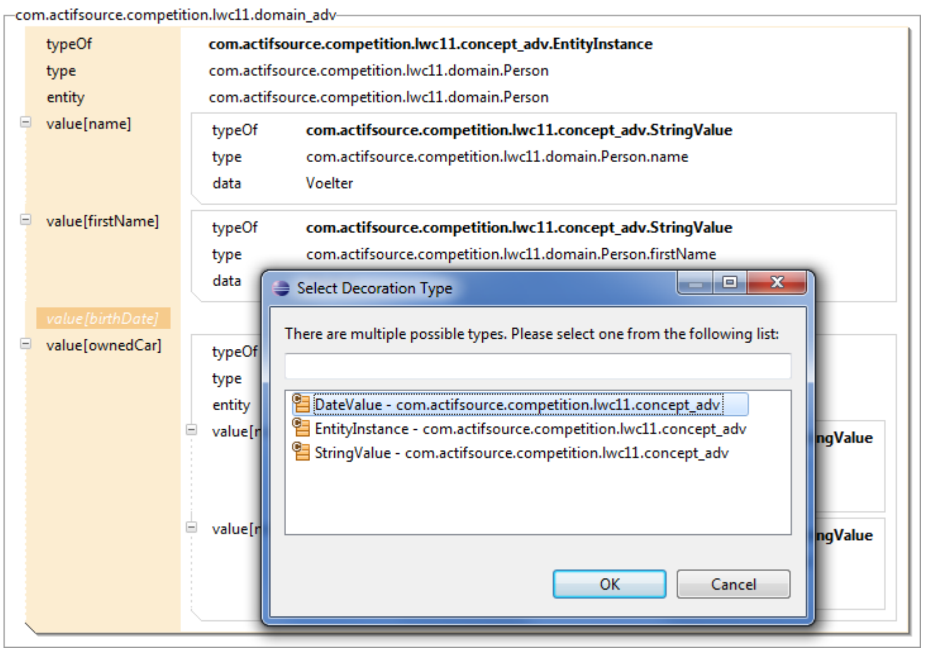
typeOf	**Class**
name	**DateValue**
comment	
aspect [InitializationAspect]	
aspect [ResourceValidationAspect]	
aspect [NameAspect]	
extends	com.actifsource.competition.lwc11.concept_adv.Value
modifier	
property	data : **Attribute**
definesAspect	
allowRoot	
classIcon	
lineColor	
fillColor	
shape	

The instantiated object looks as follows:



```
com.actifsource.competition.lwc11.domain_adv
typeOf            com.actifsource.competition.lwc11.concept_adv.EntityInstance
type              com.actifsource.competition.lwc11.domain.Person
entity            com.actifsource.competition.lwc11.domain.Person
value[name]       typeOf   com.actifsource.competition.lwc11.concept_adv.StringValue
                  type     com.actifsource.competition.lwc11.domain.Person.name
                  data     Voelter
value[firstName]  typeOf   com.actifsource.competition.lwc11.concept_adv.StringValue
                  type     com.actifsource.competition.lwc11.domain.Person.firstName
                  data     Markus
value[birthDate]  typeOf   com.actifsource.competition.lwc11.concept_adv.DateValue
                  type     com.actifsource.competition.lwc11.domain.Person.birthDate
                  data     14.02.1972
value[ownedCar]   typeOf         com.actifsource.competition.lwc11.concept_adv.EntityInstance
                  type           com.actifsource.competition.lwc11.domain.Person.ownedCar
                  entity         com.actifsource.competition.lwc11.domain.Car
                  value[make]    typeOf   com.actifsource.competition.lwc11.concept_adv.StringValue
                                 type     com.actifsource.competition.lwc11.domain.Car.make
                                 data     VW
                  value[model]   typeOf   com.actifsource.competition.lwc11.concept_adv.StringValue
                                 type     com.actifsource.competition.lwc11.domain.Car.model
                                 data     Touran
```

The types of the instances are not checked – that would require the writing of a *ResourceValidationAspect*.



```
com.actifsource.competition.lwc11.domain_adv
typeOf            com.actifsource.competition.lwc11.concept_adv.EntityInstance
type              com.actifsource.competition.lwc11.domain.Person
entity            com.actifsource.competition.lwc11.domain.Person
value[name]       typeOf   com.actifsource.competition.lwc11.concept_adv.StringValue
                  type     com.actifsource.competition.lwc11.domain.Person.name
                  data     Voelter
value[firstName]  typeOf   com.actifsource.competition.lwc11.concept_adv.StringValue
                  type     com.actifsource.competition.lwc11.domain.Person.firstName
                  data
value[birthDate]
value[ownedCar]   typeOf
                  type
                  entity
                  value[r                                     ngValue
                  value[r                                     ngValue

Select Decoration Type
There are multiple possible types. Please select one from the following list:
DateValue - com.actifsource.competition.lwc11.concept_adv
EntityInstance - com.actifsource.competition.lwc11.concept_adv
StringValue - com.actifsource.competition.lwc11.concept_adv
OK    Cancel
```

1.2 Demonstrate how to implement runtime type systems

See: 1.1 – Actifsource does not support runtime type systems.

1.3 Show how to do a model-to-model transformation

There is no model-to-model transformation provided in Actifsource. There are only transformations resulting in a text and are defined in a template.

However, it is possible to integrate Java functions (as described in 1.5) as an abstraction layer that allow to navigate and iterate over the model as if it was a transformed model.

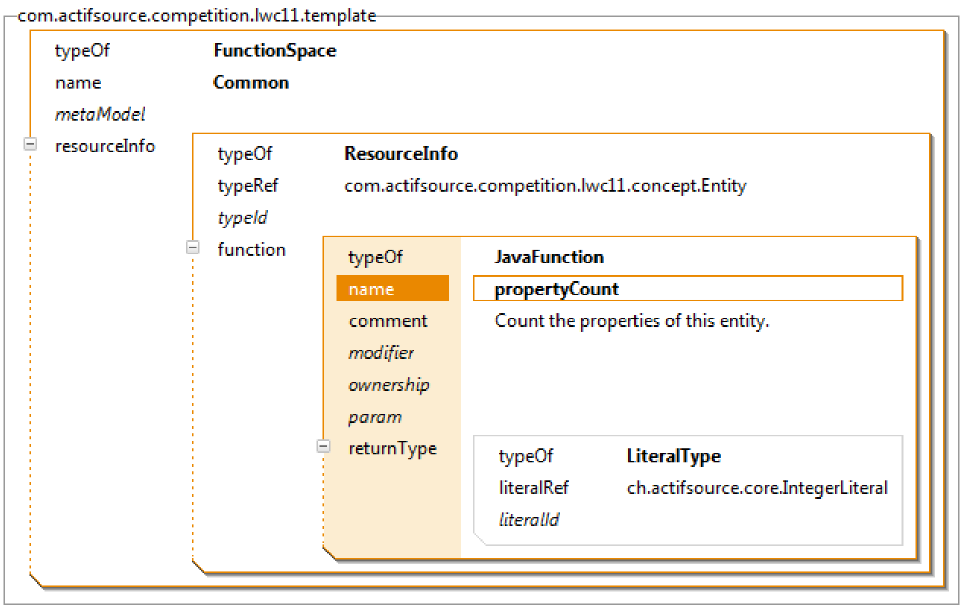
1.4 Some kind of visibility/namespaces/scoping for references

In Actifsource, the package name is the path to the folder which the resource is saved in. It is arbitrary for non-aggregated resources. Aggregated resources have the same package as their owner.

There is no concept of namespace - resources are referenced by their GUID.

1.5 Integrating manually written code (again in Java, C# or C++)

User-defined functions are written in Java and can be attached to any model element.



com.actifsource.competition.lwc11.template

typeOf	**FunctionSpace**
name	**Common**
metaModel	
resourceInfo	

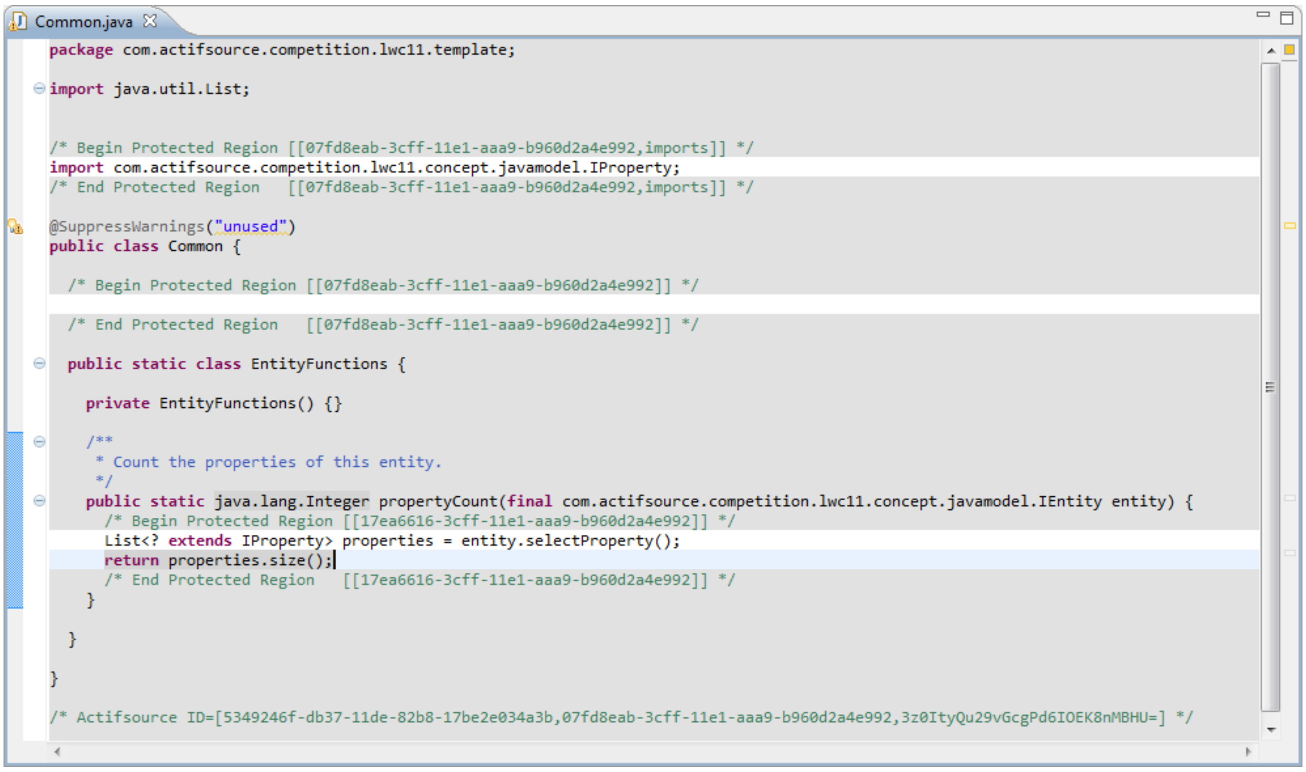
typeOf	**ResourceInfo**
typeRef	com.actifsource.competition.lwc11.concept.Entity
typeId	
function	

typeOf	**JavaFunction**
name	**propertyCount**
comment	Count the properties of this entity.
modifier	
ownership	
param	
returnType	

typeOf	**LiteralType**
literalRef	ch.actifsource.core.IntegerLiteral
literalId	

The functions are declared in a *FunctionSpace* resource, so a Java class is generated out of the declarations.

The generated Java file contains so-called *Protected Regions* in which we can fill in the user code and which will be preserved upon generation.



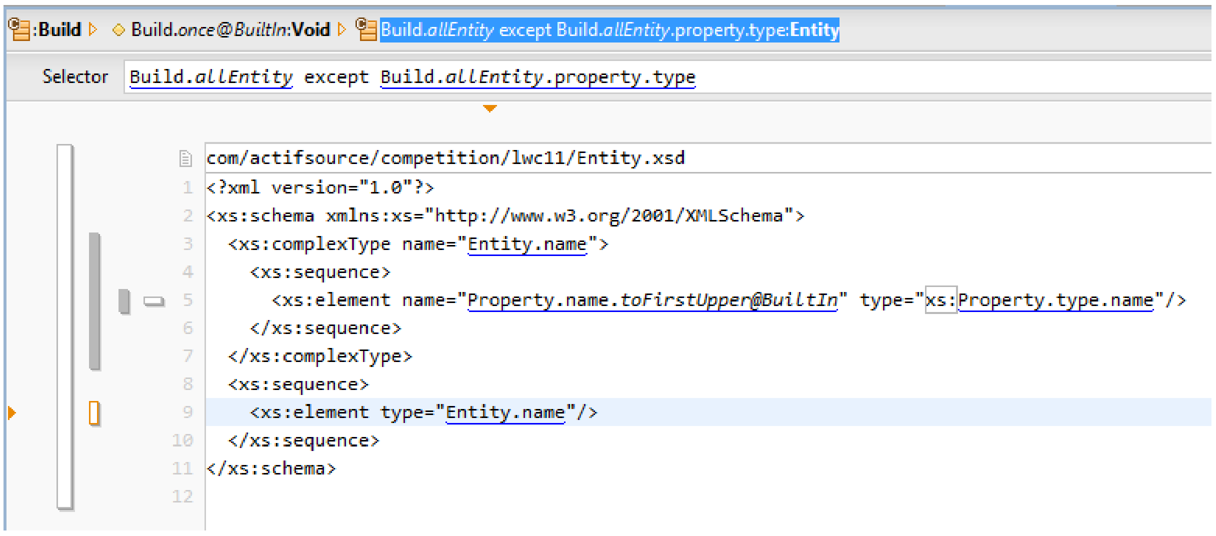
Common.java

```java
package com.actifsource.competition.lwc11.template;

import java.util.List;


/* Begin Protected Region [[07fd8eab-3cff-11e1-aaa9-b960d2a4e992,imports]] */
import com.actifsource.competition.lwc11.concept.javamodel.IProperty;
/* End Protected Region   [[07fd8eab-3cff-11e1-aaa9-b960d2a4e992,imports]] */

@SuppressWarnings("unused")
public class Common {

  /* Begin Protected Region [[07fd8eab-3cff-11e1-aaa9-b960d2a4e992]] */

  /* End Protected Region   [[07fd8eab-3cff-11e1-aaa9-b960d2a4e992]] */

  public static class EntityFunctions {

    private EntityFunctions() {}

    /**
     * Count the properties of this entity.
     */
    public static java.lang.Integer propertyCount(final com.actifsource.competition.lwc11.concept.javamodel.IEntity entity) {
      /* Begin Protected Region [[17ea6616-3cff-11e1-aaa9-b960d2a4e992]] */
      List<? extends IProperty> properties = entity.selectProperty();
      return properties.size();
      /* End Protected Region   [[17ea6616-3cff-11e1-aaa9-b960d2a4e992]] */
    }

  }

}

/* Actifsource ID=[5349246f-db37-11de-82b8-17be2e034a3b,07fd8eab-3cff-11e1-aaa9-b960d2a4e992,3z0ItyQu29vGcgPd6IOEK8nMBHU=] */
```

1.6 Multiple generators

The Actifsource Template editor is not restricted to any language. It can generate any textual language desired.

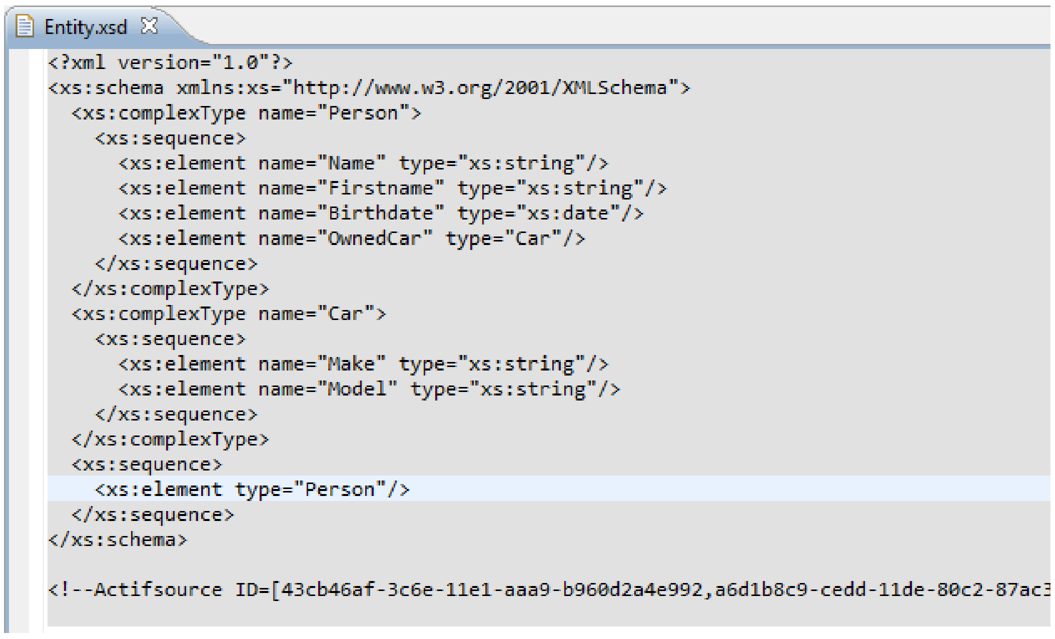
Instead of Java code, we can also generate e.g. an XML schema definition as seen below:


:Build ▷ Build.once@BuiltIn:Void ▷ Build.allEntity except Build.allEntity.property.type:Entity

Selector `Build.allEntity except Build.allEntity.property.type`

```
com/actifsource/competition/lwc11/Entity.xsd
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:complexType name="Entity.name">
    <xs:sequence>
      <xs:element name="Property.name.toFirstUpper@BuiltIn" type="xs:Property.type.name"/>
    </xs:sequence>
  </xs:complexType>
  <xs:sequence>
    <xs:element type="Entity.name"/>
  </xs:sequence>
</xs:schema>
```

This will be the generated file:



```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:complexType name="Person">
    <xs:sequence>
      <xs:element name="Name" type="xs:string"/>
      <xs:element name="Firstname" type="xs:string"/>
      <xs:element name="Birthdate" type="xs:date"/>
      <xs:element name="OwnedCar" type="Car"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="Car">
    <xs:sequence>
      <xs:element name="Make" type="xs:string"/>
      <xs:element name="Model" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
  <xs:sequence>
    <xs:element type="Person"/>
  </xs:sequence>
</xs:schema>

<!--Actifsource ID=[43cb46af-3c6e-11e1-aaa9-b960d2a4e992,a6d1b8c9-cedd-11de-80c2-87ac3
```

Phase 2 - Non-Functional

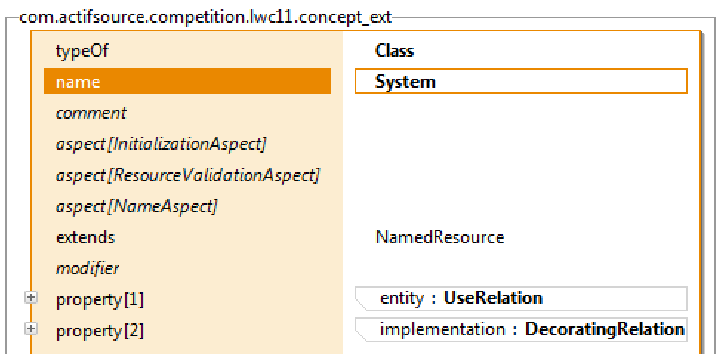
Phase 2 is intended to show a couple of non-functional properties of the LWB. The task outlined below does not elaborate on how to do this.

2.1 How to evolve the DSL without breaking existing models

If a extension to the model is needed, without touching the model itself, then decorations are the means of choice.

As an example we will add a property persistent to Entity, without touching the model we have created.

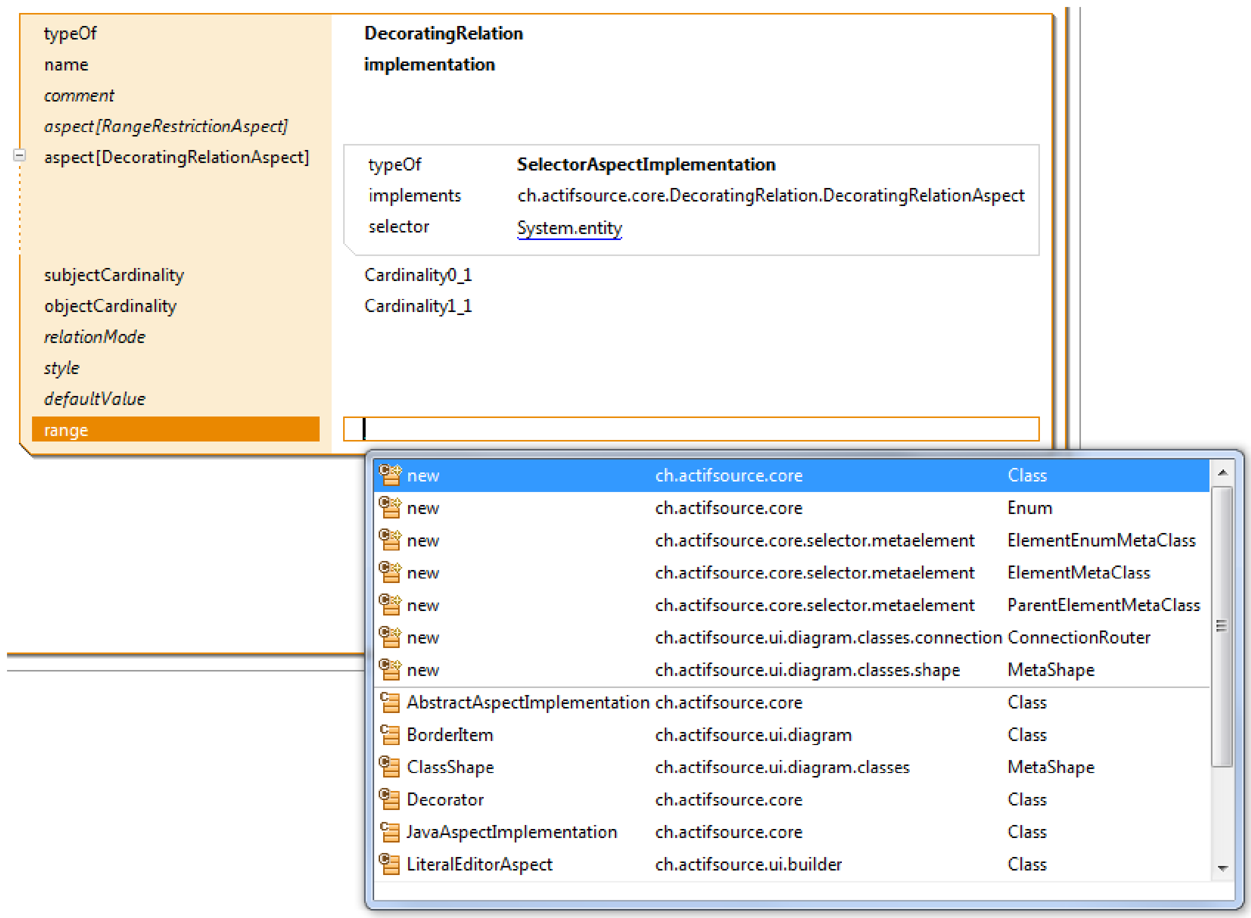
First, there is a path to the resources needed, we want to decorate. In our case, we have to add a resource which has a *UseRelation* to the entities, e.g. a type *System*.



com.actifsource.competition.lwc11.concept_ext

typeOf	**Class**
name	**System**
comment	
aspect[InitializationAspect]	
aspect[ResourceValidationAspect]	
aspect[NameAspect]	
extends	NamedResource
modifier	
property[1]	entity : **UseRelation**
property[2]	implementation : **DecoratingRelation**

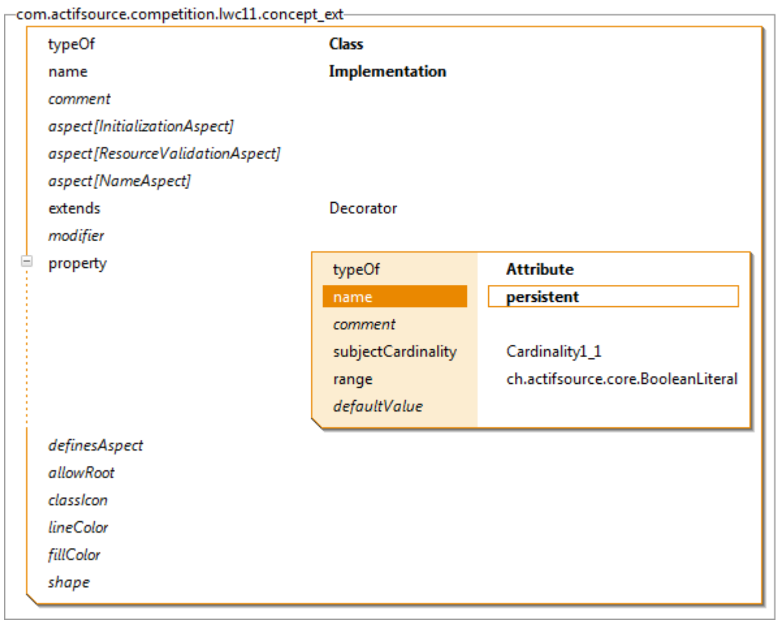
Second, we add a *DecoratingRelation* which decorates the given path to the entities.



typeOf	**DecoratingRelation**
name	**implementation**
comment	
aspect[RangeRestrictionAspect]	
aspect[DecoratingRelationAspect]	typeOf **SelectorAspectImplementation** implements ch.actifsource.core.DecoratingRelation.DecoratingRelationAspect selector System.entity
subjectCardinality	Cardinality0_1
objectCardinality	Cardinality1_1
relationMode	
style	
defaultValue	
range	

new	ch.actifsource.core	Class
new	ch.actifsource.core	Enum
new	ch.actifsource.core.selector.metaelement	ElementEnumMetaClass
new	ch.actifsource.core.selector.metaelement	ElementMetaClass
new	ch.actifsource.core.selector.metaelement	ParentElementMetaClass
new	ch.actifsource.ui.diagram.classes.connection	ConnectionRouter
new	ch.actifsource.ui.diagram.classes.shape	MetaShape
AbstractAspectImplementation	ch.actifsource.core	Class
BorderItem	ch.actifsource.ui.diagram	Class
ClassShape	ch.actifsource.ui.diagram.classes	MetaShape
Decorator	ch.actifsource.core	Class
JavaAspectImplementation	ch.actifsource.core	Class
LiteralEditorAspect	ch.actifsource.ui.builder	Class

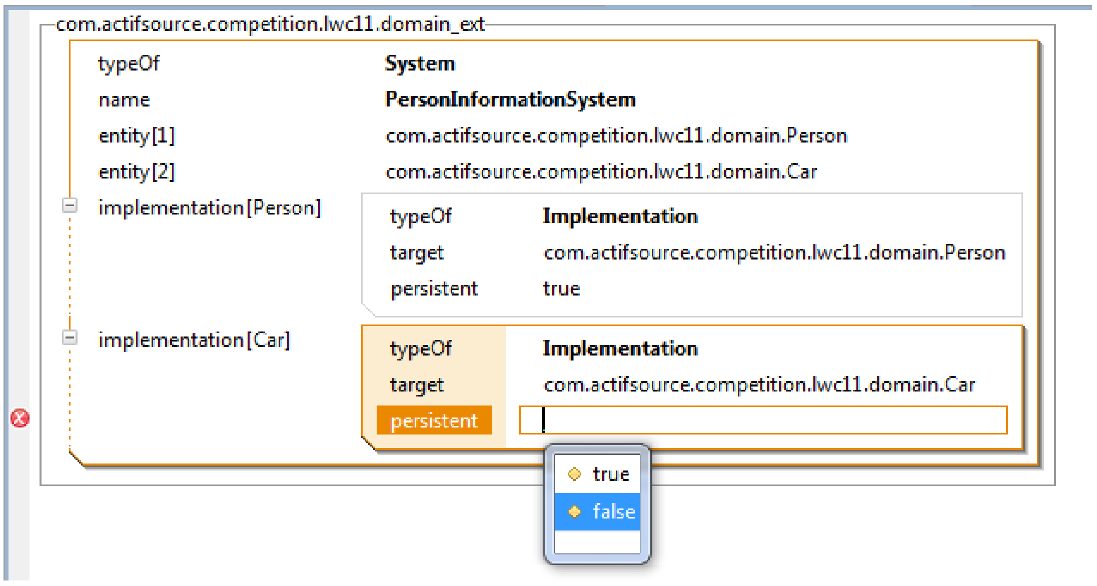
Finally, we create a new type for the new property, and set it as *range* of the *DecoratingRelation*.



com.actifsource.competition.lwc11.concept_ext

typeOf	**Class**
name	**Implementation**
comment	
aspect[InitializationAspect]	
aspect[ResourceValidationAspect]	
aspect[NameAspect]	
extends	Decorator
modifier	
property	typeOf: **Attribute**; name: **persistent**; *comment*; subjectCardinality: Cardinality1_1; range: ch.actifsource.core.BooleanLiteral; *defaultValue*
definesAspect	
allowRoot	
classIcon	
lineColor	
fillColor	
shape	

In an instance of the new *System* type, there is an item for every entity, so we can set the new property.

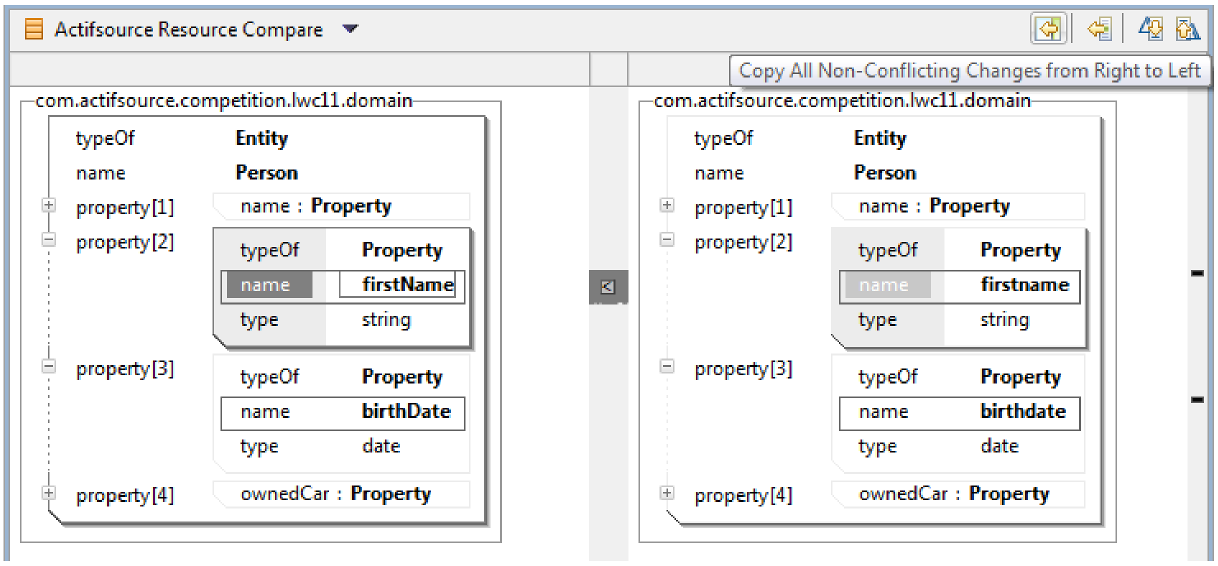


com.actifsource.competition.lwc11.domain_ext

typeOf	**System**
name	**PersonInformationSystem**
entity[1]	com.actifsource.competition.lwc11.domain.Person
entity[2]	com.actifsource.competition.lwc11.domain.Car
implementation[Person]	typeOf: **Implementation**; target: com.actifsource.competition.lwc11.domain.Person; persistent: true
implementation[Car]	typeOf: **Implementation**; target: com.actifsource.competition.lwc11.domain.Car; persistent:

2.2 How to work with the models efficiently in the team

All Actifsource resources are saved as XML files together with the generated artifacts in the source control system of the user. Changes can be visualized in the editor and merged as desired.


Actifsource Resource Compare
Copy All Non-Conflicting Changes from Right to Left
com.actifsource.competition.lwc11.domain
typeOf Entity
name Person
property[1] name : Property
property[2] typeOf Property
name firstName
type string
property[3] typeOf Property
name birthDate
type date
property[4] ownedCar : Property
com.actifsource.competition.lwc11.domain
typeOf Entity
name Person
property[1] name : Property
property[2] typeOf Property
name firstname
type string
property[3] typeOf Property
name birthdate
type date
property[4] ownedCar : Property

2.3 Demonstrate Scalability of the tools

Phase 3 - Freestyle

Every LWB has its own special "cool features". In phase three we want the participants to show off these features. Please make sure, though, that the features are built on top of the task described below, if possible.

Actifsource does not require the knowledge of a new syntax: The model elements are selected using QuickAssist and are displayed graphically.

The format of the generated code can be any textual format. Since there are no template keywords, the template already looks very similar to the generated code.