Advanced SW Engineering: MDD / DSL Technologies

Uwe Zdun
Software Architecture Research Group
Faculty of Computer Science
University of Vienna
http://cs.univie.ac.at/swa

EMF

EMF

- EMF is a technology that can be used to model your domain model
- EMF distinguishes between the meta-model and the actual model
- EMF allows to create the meta-model via different means, e.g. the EMF tools, XMI, Java annotations, UML or a XML Schema

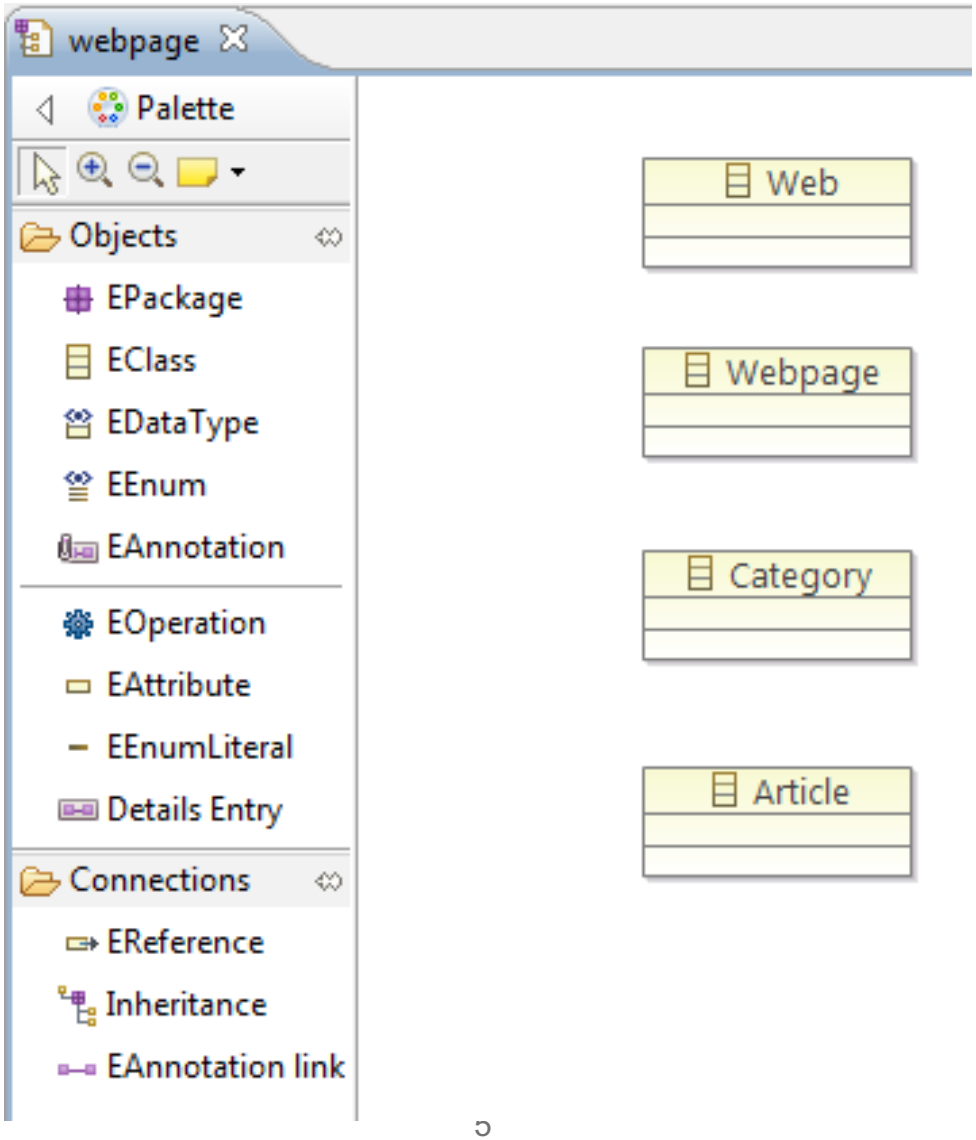
Ecore Metamodel and Genmodel

EMF is based on two meta-models; the Ecore and the Genmodel model.

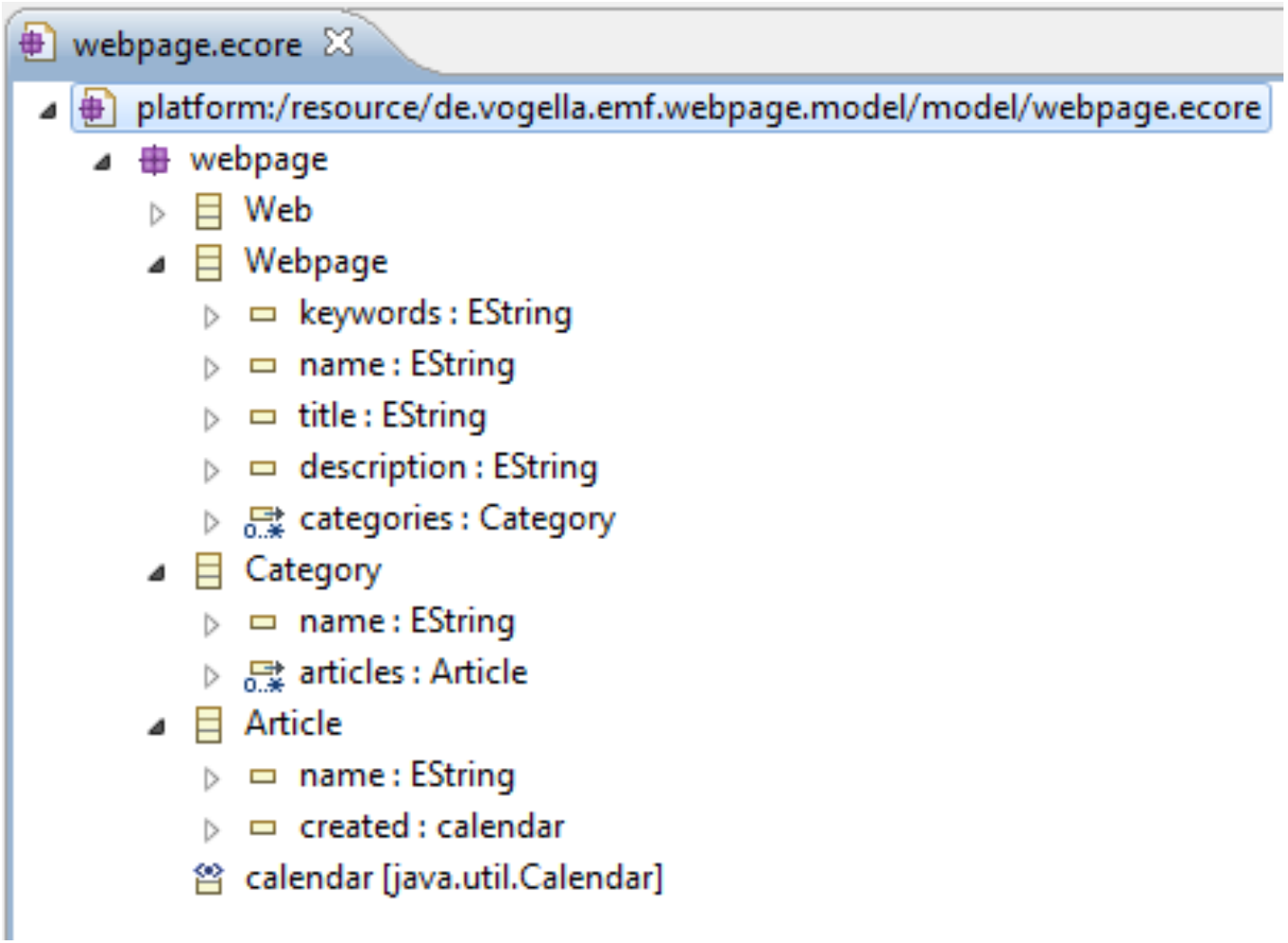
The Ecore metamodel contains the information about the defined classes.

The Genmodel contains additional information for the code generation, e.g. the path and file information.

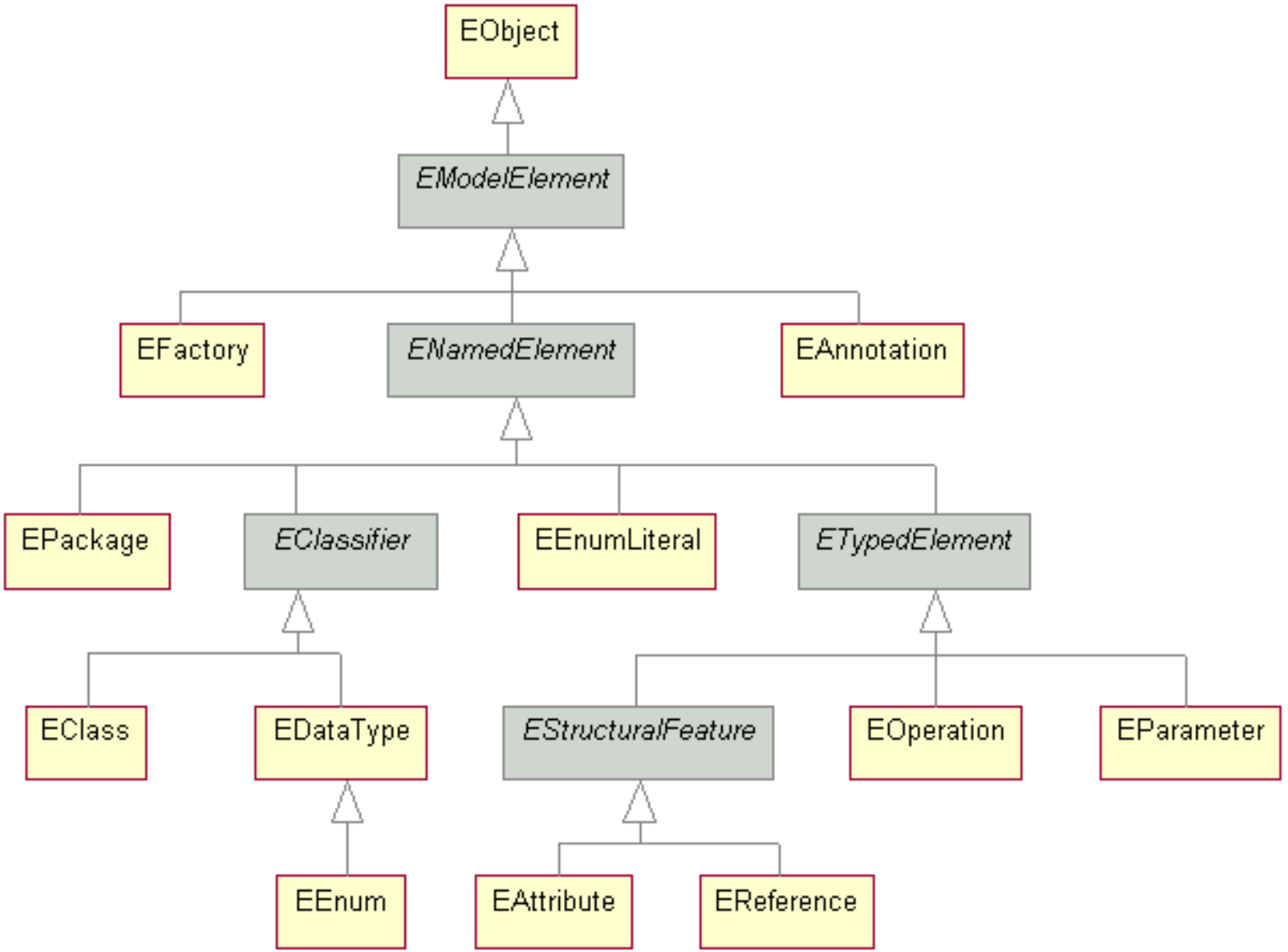
Graphical Ecore Editor



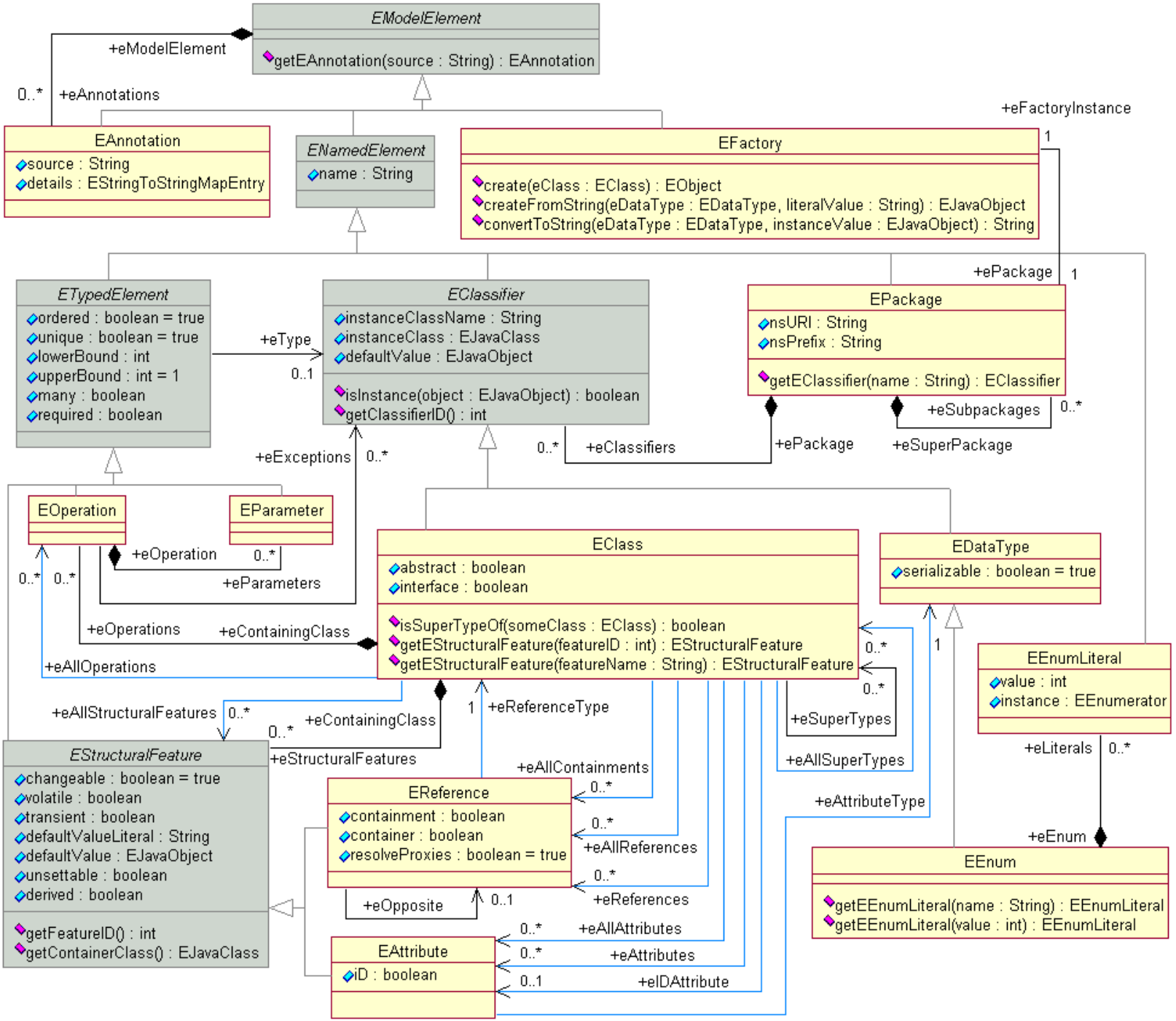
Tree-Based Ecore Editor



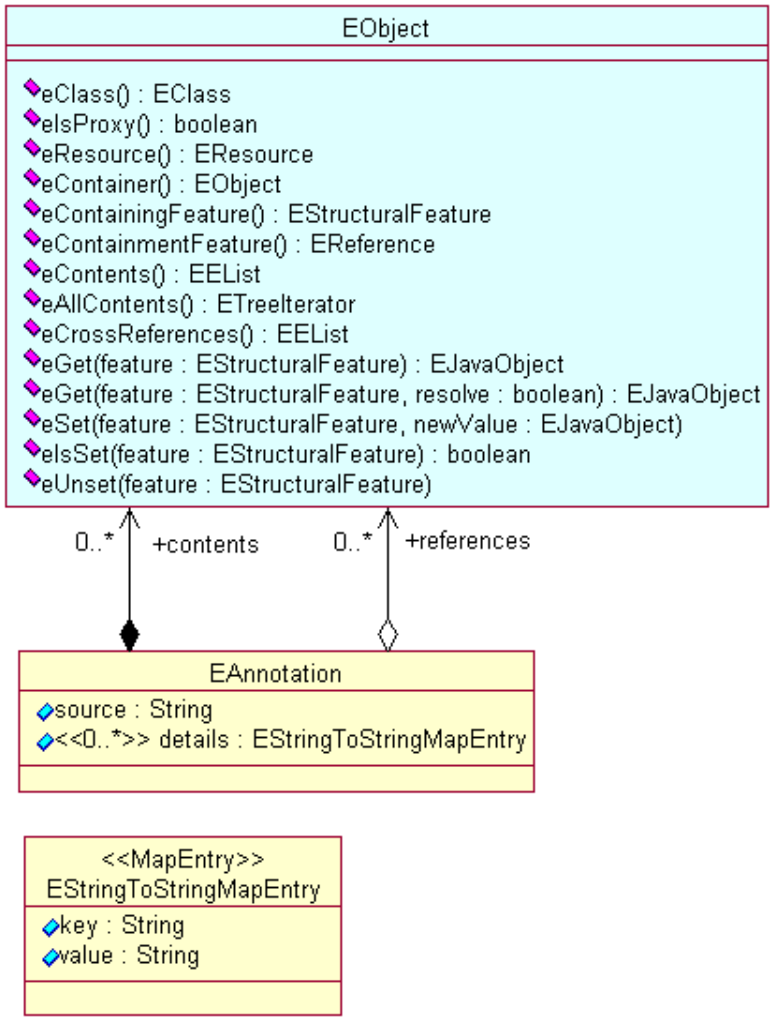
Ecore Metamodel



Ecore Metamodel



EObject Interface



XTEXT AND XTEND

Xtext

Xtext is a framework for development of programming languages and domain specific languages

Example: Develop a small DSL with Xtext

```
datatype String

entity Blog {
    title: String
    many posts: Post
}

entity HasAuthor {
    author: String
}

entity Post extends HasAuthor {
    title: String
    content: String
    many comments: Comment
}

entity Comment extends HasAuthor {
    content: String
}
```

Grammar of the Example Language

```
grammar org.example.domainmodel.DomainModel with
       org.eclipse.xtext.common.Terminals
generate domainmodel
       "http://www.example.org/domainmodel/Domainmodel"
```

Each Xtext grammar starts with a header that defines some properties of the grammar

```
Domainmodel :
  elements += Type*;
Type:
  DataType | Entity;
DataType:
  'datatype' name = ID;
Entity:
  'entity' name = ID ('extends' superType = [Entity])? '{'
     features += Feature*
  '}';
Feature:
  many?='many'? name = ID ':' type = [Type];
```

Grammar of the Example Language

```
grammar org.example.domainmodel.DomainModel with
        org.eclipse.xtext.common.Terminals
generate domainmodel
        "http://www.example.org/domainmodel/Domainmodel"
```

```
Domainmodel :
  elements += Type*;
Type:
  DataType | Entity;
DataType:
  'datatype' name = ID;
Entity:
  'entity' name = ID ('extends' superType = [Entity])? '{'
     features += Feature*
  '}';
Feature:
  many?='many'? name = ID ':' type = [Type];
```

Parser Rules

Grammar of the Example Language

```
grammar org.example.domainmodel.DomainModel with
       org.eclipse.xtext.common.Terminals
generate domainmodel
       "http://www.example.org/domainmodel/Domainmodel"

Domainmodel :
  elements += Type*;
Type:
  DataType | Entity;
DataType:
  'datatype' name = ID;
Entity:
  'entity' name = ID ('extends' superType = [Entity])? '{'
     features += Feature*
  '}';
Feature:
  many?='many'? name = ID ':' type = [Type];
```

Alternative

Sequence (Group)

Grammar of the Example Language

```
grammar org.example.domainmodel.DomainModel with
        org.eclipse.xtext.common.Terminals
generate domainmodel
        "http://www.example.org/domainmodel/Domainmodel"

Domainmodel :
  elements += Type*;
Type:
  DataType | Entity;
DataType:
  'datatype' name = ID;
Entity:
  'entity' name = ID ('extends' superType = [Entity])? '{'
     features += Feature*
  '}';
Feature:
  many?='many'? name = ID ':' type = [Type];
```

Assignments
= : straightforward assignment
+= : the add operator
?= : boolean assignment operator

Grammar of the Example Language

```
grammar org.example.domainmodel.DomainModel with
       org.eclipse.xtext.common.Terminals
generate domainmodel
       "http://www.example.org/domainmodel/Domainmodel"

Domainmodel :
  elements += Type*;
Type:
  DataType | Entity;
DataType:
  'datatype' name = ID;
Entity:
  'entity' name = ID ('extends' superType = [Entity])? '{'
     features += Feature*
  '}';
Feature:
  many?='many'? name = ID ':' type = [Type];
```

References to other rules

Keywords

Cross-references to EClasses

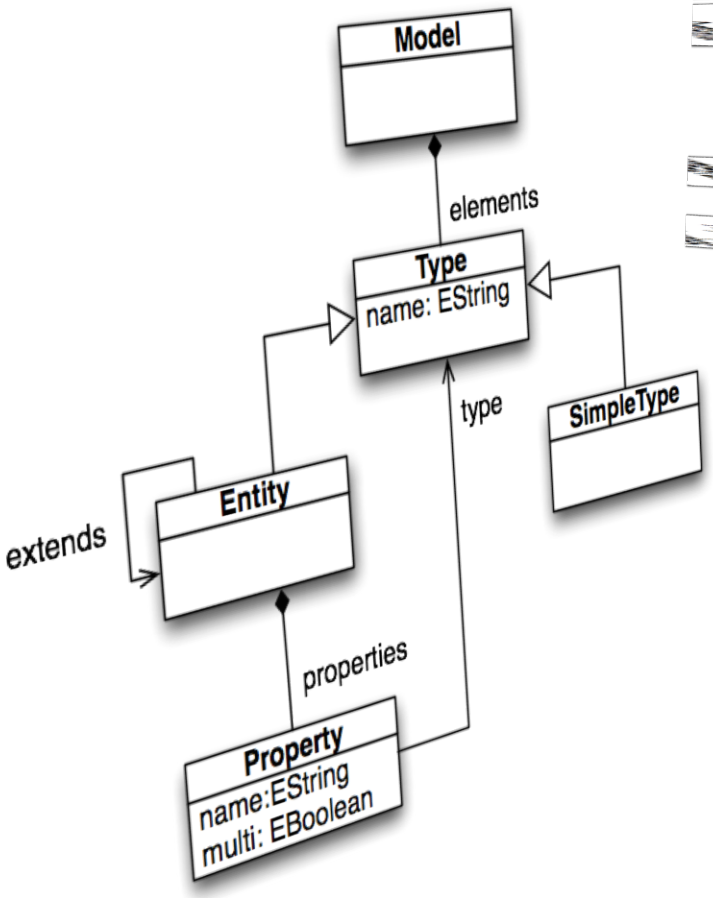
Xtext Grammar Language

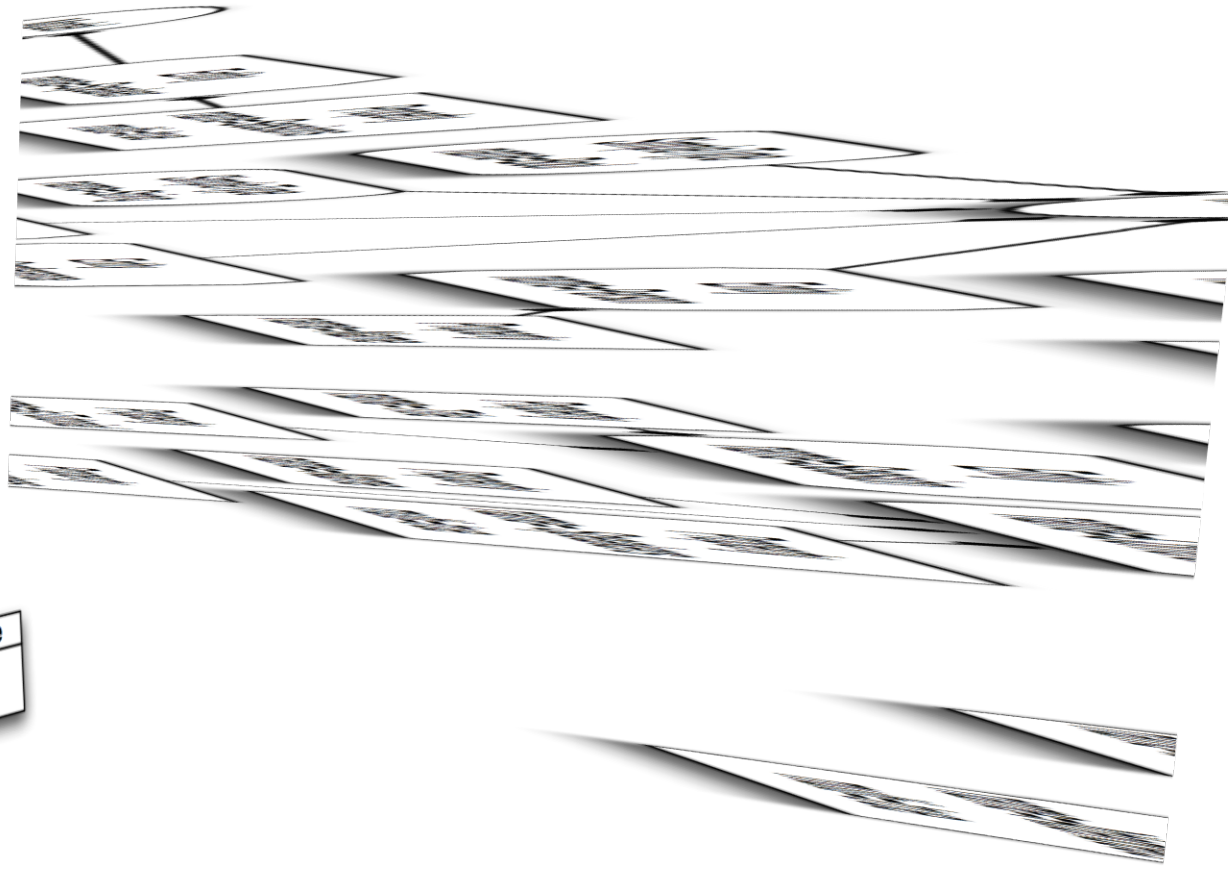
- Domain-specific language, carefully designed for the description of textual languages
- Cornerstone of Xtext
- Supports EBNF-like grammar syntax
- Describes the concrete syntax and how it is mapped to an in-memory representation

Relation to EMF

- Xtext uses EMF models as the in-memory representation of any parsed text files
- This in-memory object graph is called the **Abstract Syntax Tree** (AST)

Example: An Ecore Model and a Corresponding AST





From http://www.eclipse.org/Xtext/documentation.html

Xtend2

- Xtend2: A DSL for Generating Code
- Motivation: Why do we need a new language for code generation? Why not just use Java?
  - no multiline string literals
  - instanceof / if cascades
  - noisy string concatenation

Xtend2: The generator specifies the files to be generated

```
override void doGenerate(Resource resource, IFileSystemAccess fsa) {
  for(e: resource.allContents.toIterable.filter(Entity)) {
    fsa.generateFile(
      e.fullyQualifiedName.toString("/") + ".java", e.compile)
  }
}
```

Here we generate Java code

Xtend2: Methods for code generation contain templates

```
def compile(Entity e) '''
  «IF e.eContainer.fullyQualifiedName != null»
    package «e.eContainer.fullyQualifiedName»;
  «ENDIF»

  public class «e.name» «IF e.superType != null
      »extends «e.superType.fullyQualifiedName» «ENDIF»{
  «FOR f:e.features»
    «f.compile»
  «ENDFOR»
}
'''

def compile(Feature f) '''
  private «f.type.fullyQualifiedName» «f.name»;
  public «f.type.fullyQualifiedName» get«f.name.toFirstUpper»() {
    return «f.name»;
  }
  public void set«f.name.toFirstUpper»(«f.type.fullyQualifiedName» «f.name») {
    this.«f.name» = «f.name»;
  }
'''
```

Rich strings contain code generation templates

XPAND syntax in guillemets

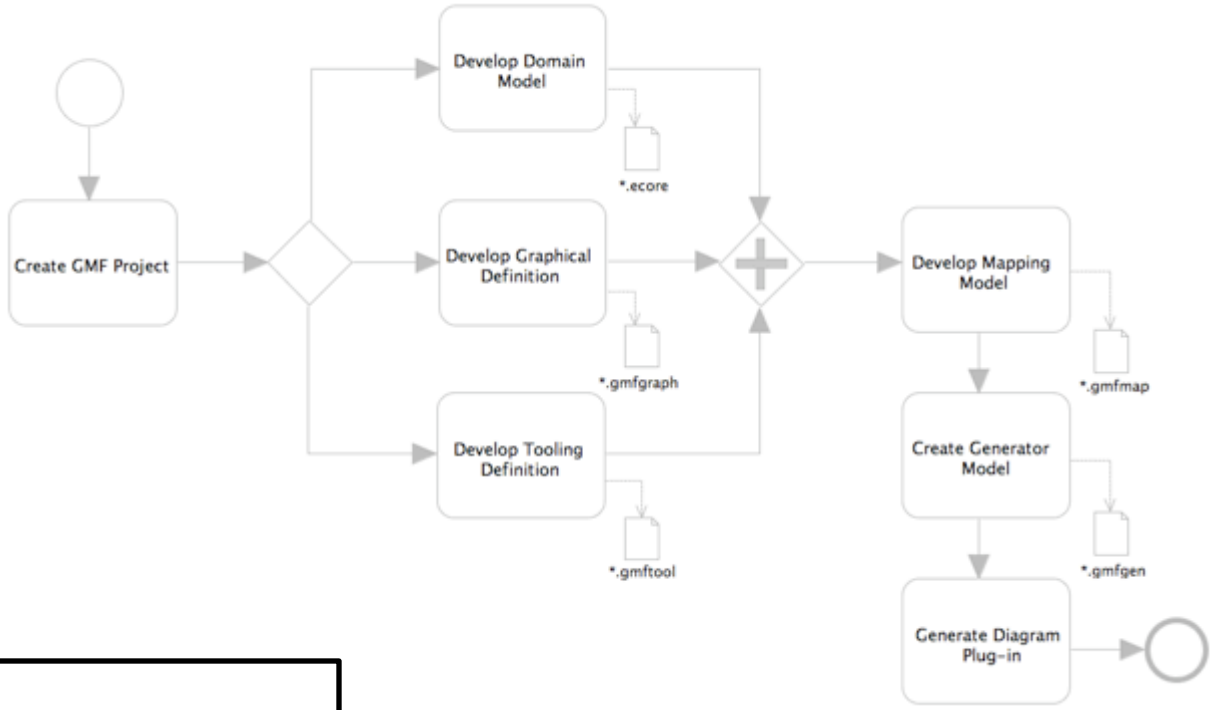
Accessing the EMF model elements

Navigating the EMF model

GMF

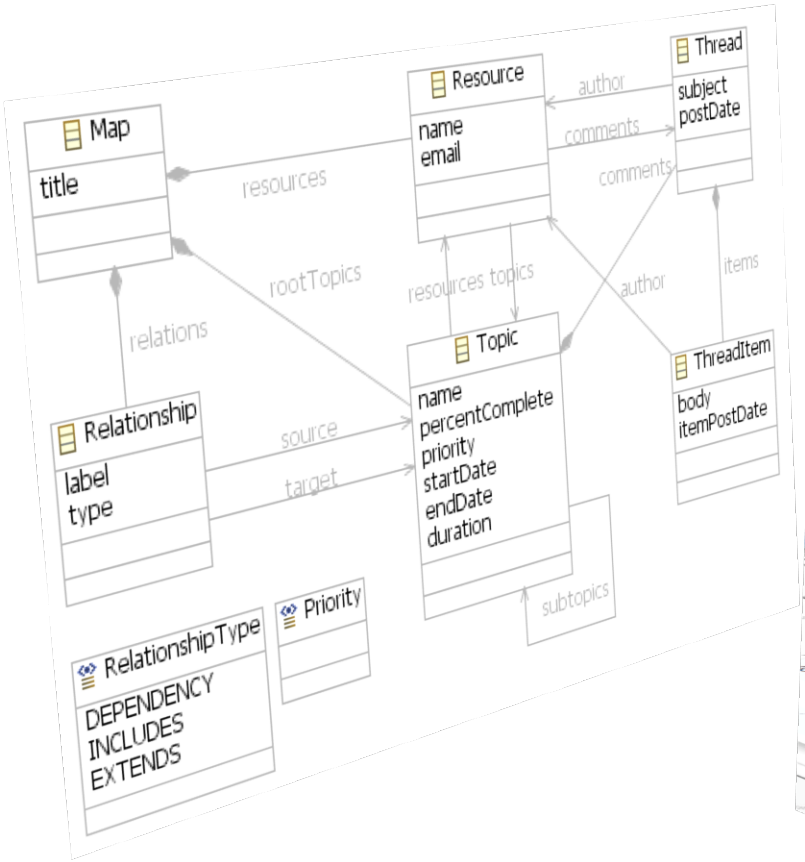
Graphical Modeling Framework (GMF)

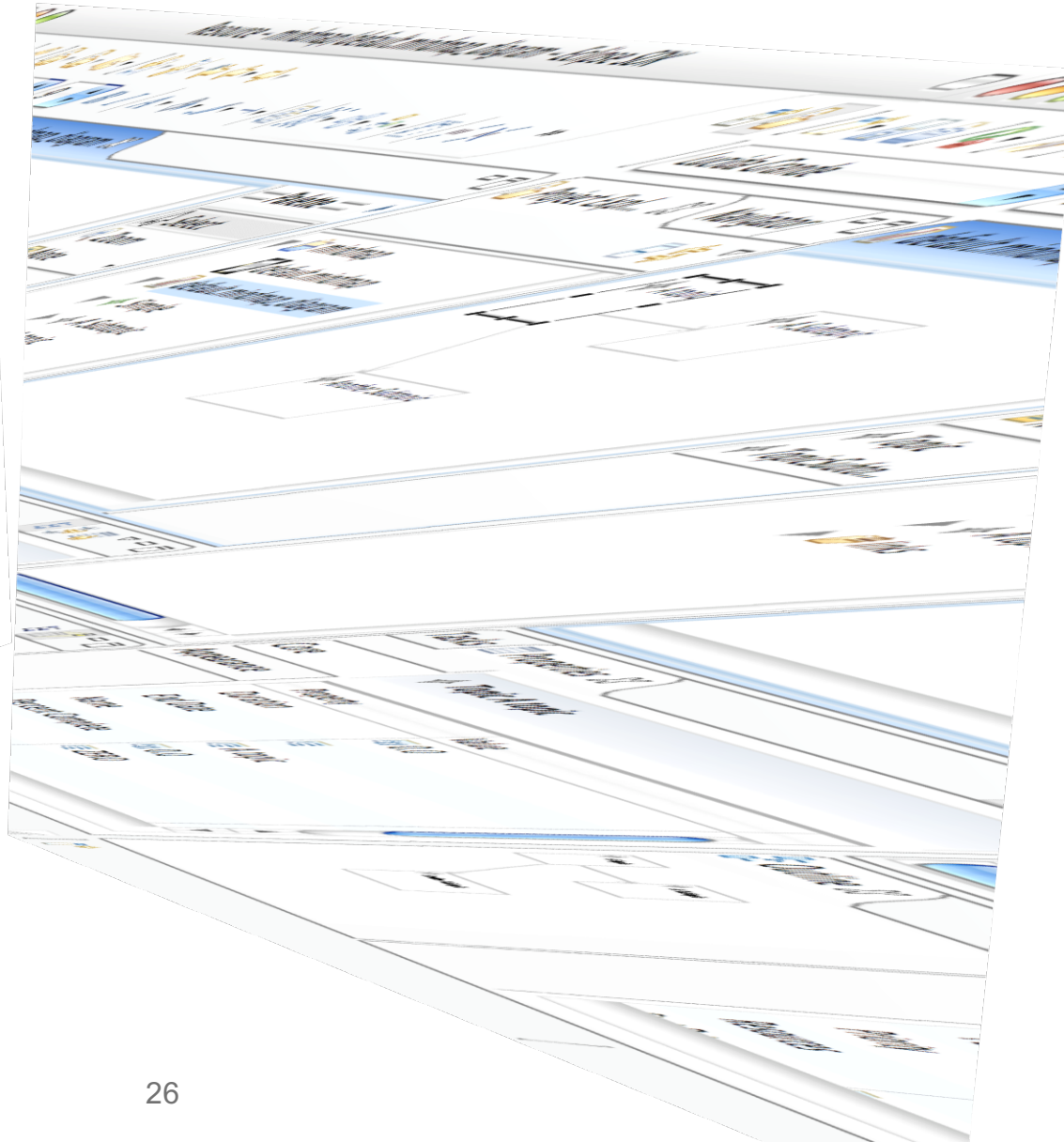
The Eclipse Graphical Modeling Framework (GMF) provides a generative component and runtime infrastructure for developing graphical editors based on EMF



From http://wiki.eclipse.org/Graphical_Modeling_Framework/Tutorial/Part_1

Example: GMF Mindmap Editor and its Ecore Model





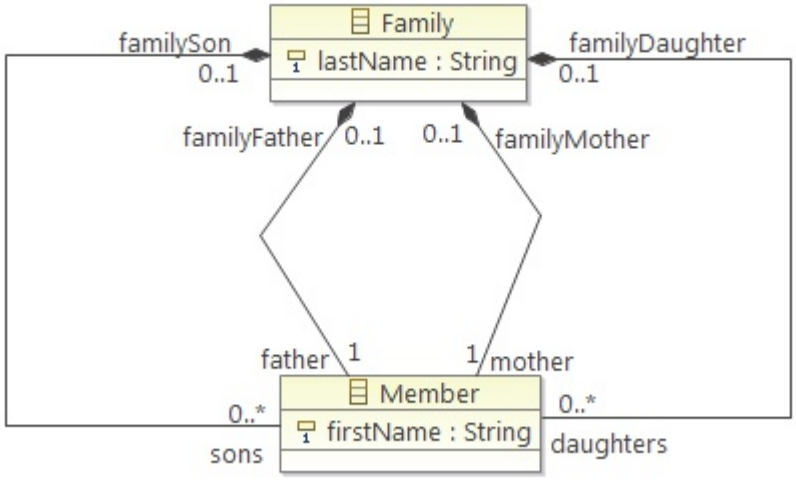
From
http://wiki.eclipse.org/Graphical_Modeling_Framework/Tutorial/Part_1

ATL

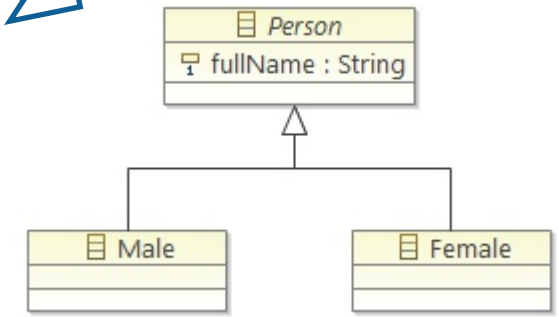
Model-to-model transformation with the Atlas Transformation Language (ATL)

- ATL aims at providing a set of model-to-model transformation tools
- Model-to-model transformation: Create models with a different perspective from other models
  - Map a high-level model to a low-level model
  - Refinement of models

Example: Map a list of families to a list of persons



model-to-model transformation



From http://wiki.eclipse.org/ATL/Tutorials_-_Create_a_simple_ATL_transformation

ATL transformation code (1)

```
module Families2Persons;

-- @path Families=/Families2Persons/Families.ecore
-- @path Persons=/Families2Persons/Persons.ecore

create OUT: Persons from IN: Families;

helper context Families!Member def: isFemale(): Boolean =
            if not self.familyMother.oclIsUndefined() then
                        true
            else
                        if not self.familyDaughter.oclIsUndefined() then
                                    true
                        else
                                    false
                        endif
            endif;


helper context Families!Member def: familyName: String =
            if not self.familyFather.oclIsUndefined() then
                        self.familyFather.lastName
            else
                        if not self.familyMother.oclIsUndefined() then
                                    self.familyMother.lastName
                        else
                                    if not self.familySon.oclIsUndefined() then
                                                self.familySon.lastName
                                    else
                                                self.familyDaughter.lastName
                                    endif
                        endif
            endif;
```

ATL transformation code (2)

```
rule Member2Male {
        from
                s: Families!Member (not s.isFemale())
        to
                t: Persons!Male (
                        fullName <- s.firstName + ' ' + s.familyName
                )
}
rule Member2Female {
        from
                s: Families!Member (s.isFemale())
        to
                t: Persons!Female (
                        fullName <- s.firstName + ' ' + s.familyName
                )
}
```

MPS

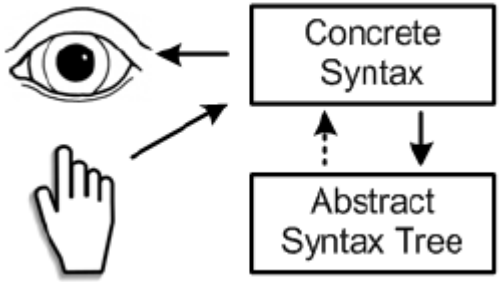
MPS

- JetBrains' Meta Programming System is a language workbench
- It uses a projectional editor that renders the abstract syntax tree in a notation that looks and feels textual while the user directly edits the tree

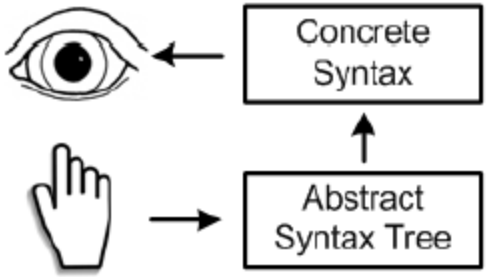
Projectional Editing

- In projectional editing, as a user edits the program, the AST is modified directly
- A projection engine then creates some representation of the AST with which the user interacts, and which reflects the changes

Parser-Based Systems vs. Projectional Editing



In parser-based systems, the user only interacts with the concrete syntax, and the AST is constructed from the information in the text

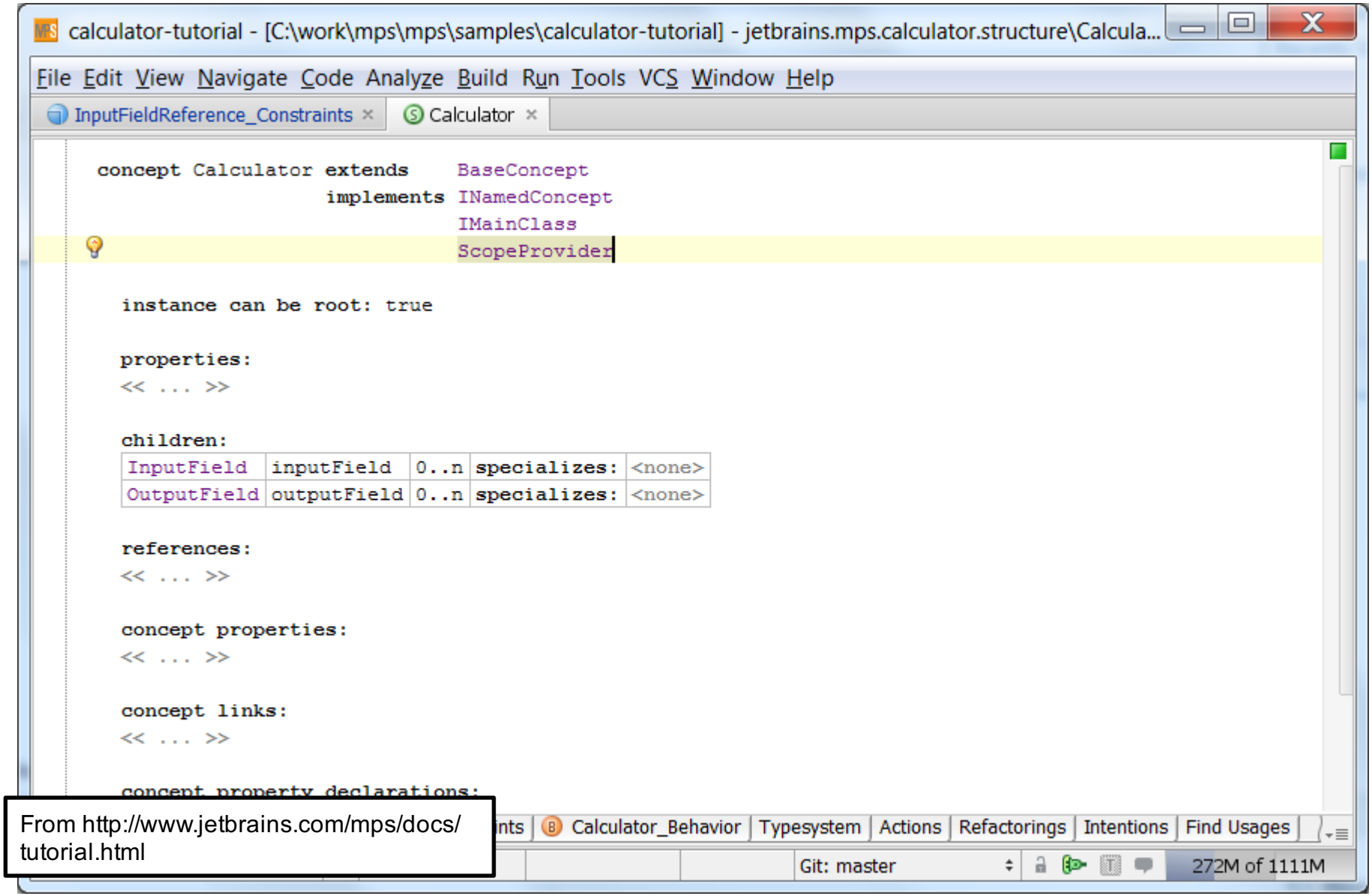


In projectional systems, the user sees the concrete syntax, but all editing gestures directly influence the AST

The AST is not extracted from the concrete syntax, which means the concrete syntax does not have to be parsable

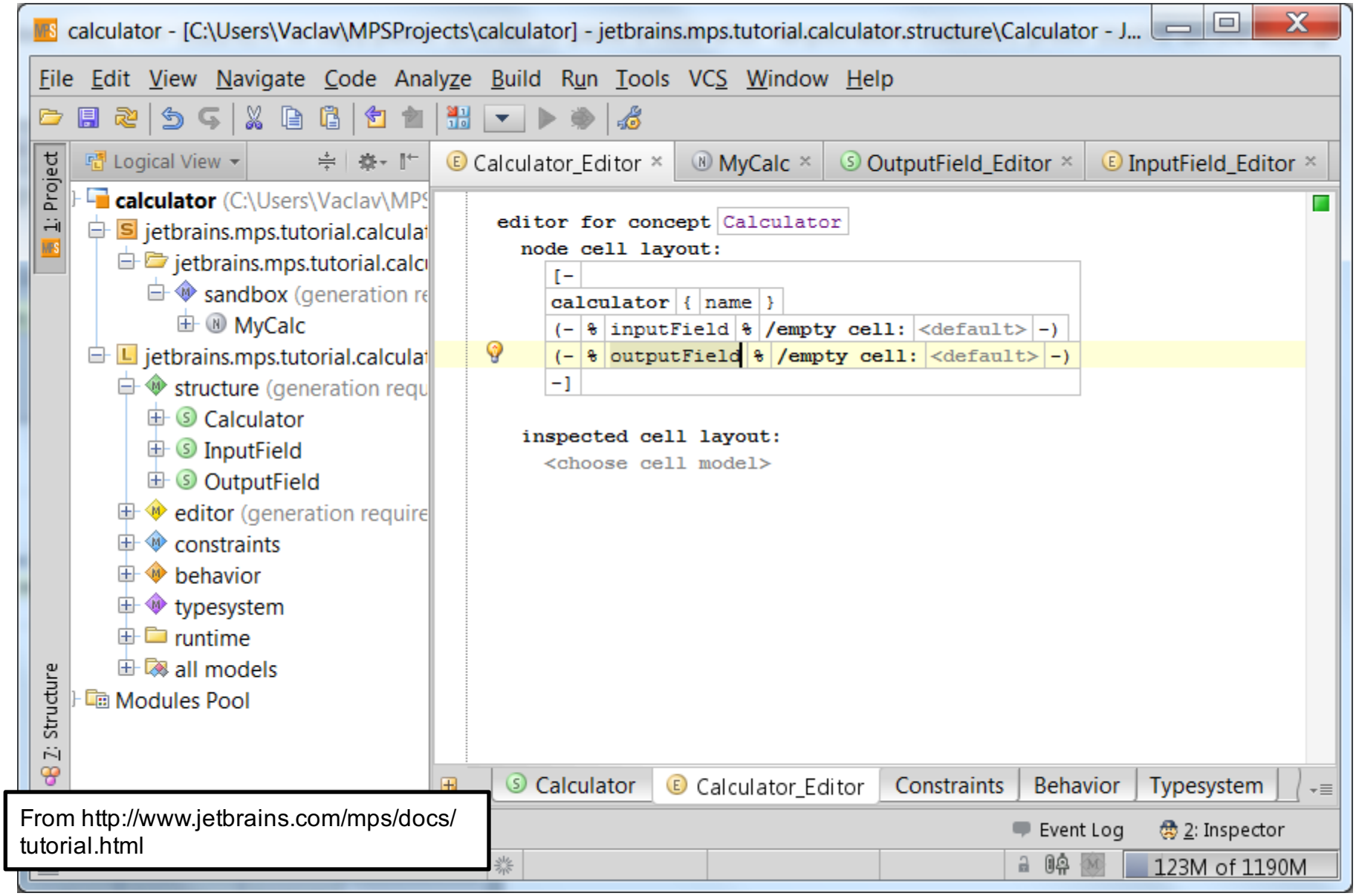
From M. Völter. DSL Engineering - Designing, Implementing and Using Domain-Specific Languages. http://dslbook.org/, 2013.

Example: Simple Calculator Concept Definition



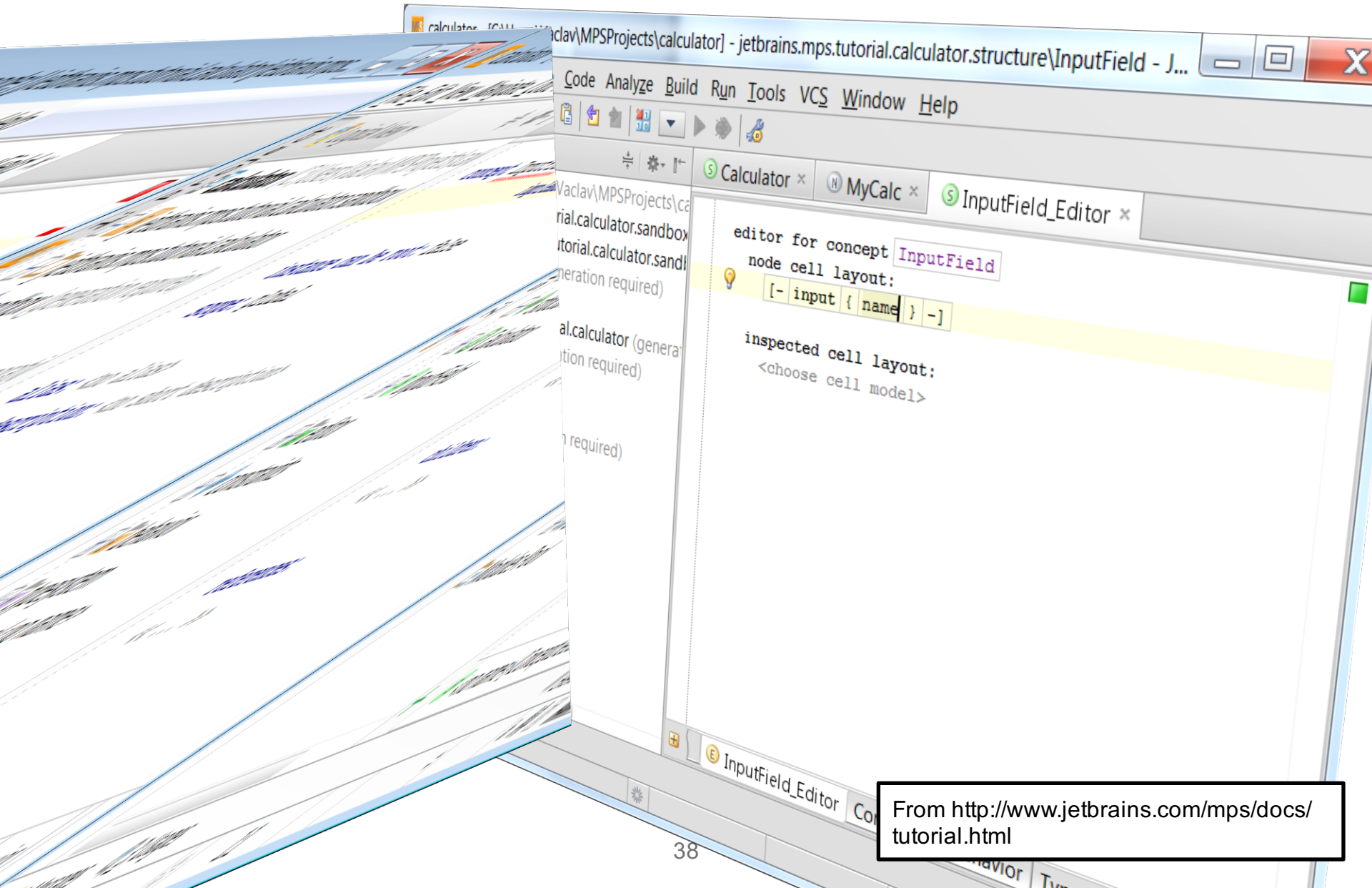
From http://www.jetbrains.com/mps/docs/tutorial.html

Example: Simple Calculator Editor Definition



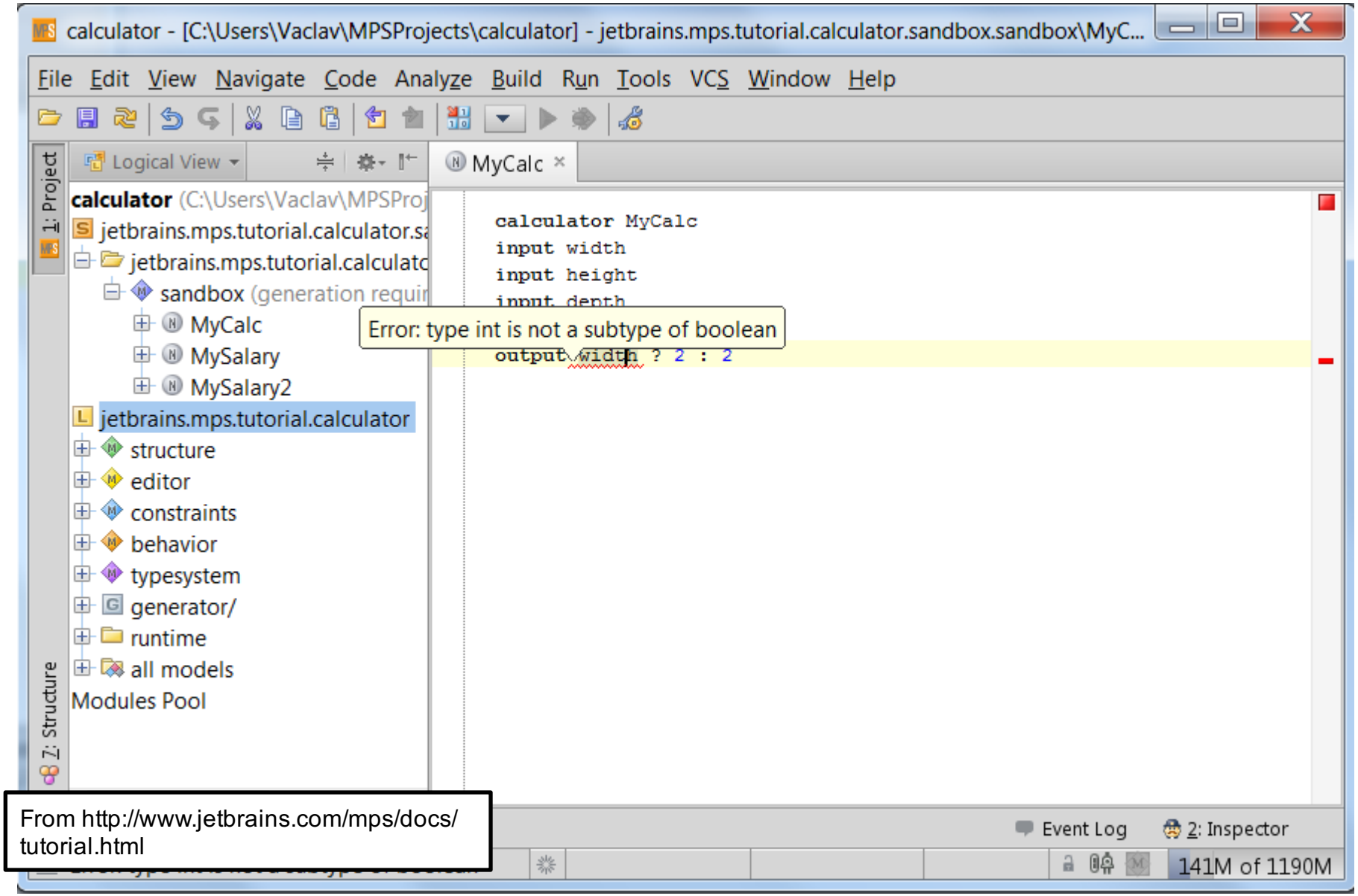
From http://www.jetbrains.com/mps/docs/tutorial.html

Example: Concept and Editor Definition for Input Field



From http://www.jetbrains.com/mps/docs/tutorial.html

Example: Using the Calculator



From http://www.jetbrains.com/mps/docs/tutorial.html

Example: An extension to C that supports working with physical units (such as kg and lb)

```
module Units from cdesignpaper.units imports nothing {

  unit kg for int8_t
  unit lb for int8_t
  kg/int8_t addWeights(kg/int8_t w1, kg/int8_t w2) {
    return w1 + w2;
  } addWeights (function)
  void s
    addWeights(10kg, 20lb);
    addWeights(10kg, 20kg);
  } someClientCode (function)
  exported test case simpleUnits {
    kg/int8_t m1 = 10kg;
    lb/int8_t m2 = 10lb;
    assert(0) m1 + 10kg == 20kg;
  } simpleUnits(test case)
}
```

Error: type lb is not a subtype of kg

From M. Völter. DSL Engineering - Designing, Implementing and Using Domain-Specific Languages. http://dslbook.org/, 2013.

Example: A C module with an embedded decision table

```
module DecisionTableExample from cdesignpaper.gswitch imports nothing {

  enum mode { MANUAL; AUTO; FAIL; }

  mode nextMode(mode mode, int8_t speed) {
    return mode, FAIL
```

	mode == MANUAL	mode == AUTO
speed < 30	MANUAL	AUTO
speed > 30	MANUAL	MANUAL

```
;
  } nextMode (function)
}
```

From M. Völter. DSL Engineering - Designing, Implementing and Using Domain-Specific Languages. http://dslbook.org/, 2013.

Example: A language in which users can specify insurance rules

```
simple product SimpleHomeInsurance

  attributes
    squareMeters : int16
    numberOfRooms : int16

  calculate
    int16, 0
```

	squareMeters < 150	squareMeters >= 150
numberOfRooms < 3	10	20
numberOfRooms == 3	30	40
numberOfRooms > 3	40	50

tests

squareMeters	numberOfRooms		
100	2	10	10.0
150	3	40	40.0
200	5	35	50.0

evaluate

Add Test Case

From M. Völter. DSL Engineering - Designing, Implementing and Using Domain-Specific Languages. http://dslbook.org/, 2013.

Relevant Literature and Sources

- http://www.eclipse.org/modeling/emf/
- http://www.eclipse.org/Xtext/
- http://www.eclipse.org/xtend/
- http://www.eclipse.org/modeling/gmp/
- http://www.eclipse.org/atl/
- http://www.jetbrains.com/mps/
- M. Völter. DSL Engineering - Designing, Implementing and Using Domain-Specific Languages. http://dslbook.org/, 2013.

Many thanks for your attention!

Uwe Zdun

Software Architecture Research Group
Faculty of Computer Science
University of Vienna
http://cs.univie.ac.at/swa