

Model Management in Xtend

(first part)

Mathieu Acher

Maître de Conférences

mathieu.acher@irisa.fr

Material

<http://mathieuacher.com/teaching/MDE/>

Plan

- Model Management in a nutshell
 - Loading, serializing, transforming models
- Xtend
 - Java 10, cheatsheet
 - Advanced features: extension methods, active annotations, template expressions
 - Xtend: behind the magic (Xtext+MDE)
- Model Management + Xtend
 - Model transformations
 - @Aspect annotation
 - Xtend + Xtext (breathing life into DSLs)

Contract

- **Practical foundations of model management**
- Learning and understanding Java 10 (aka Xtend)
 - advanced features of a general GPL, implementation of a sophisticated language using MDE
- Model transformations
 - Model-to-Text
 - Model-to-Model
- Metaprogramming
 - Revisit annotations (e.g., as in JPA or many frameworks)
- DSLs and model management: all together (Xtext + Xtend)

Model Management

Scenarios

Motivating Scenario

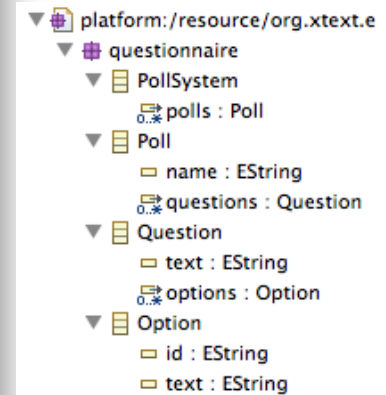
```
PollSystem {  
  Poll Quality {  
    Question q1 {  
      "Value the user experience"  
      options {  
        A : "Bad"  
        B : "Fair"  
        C : "Good"  
      }  
    }  
    Question q2 {  
      "Value the layout"  
      options {  
        A : "It was not easy to locate elements"  
        B : "I didn't realize"  
        C : "It was easy to locate elements"  
      }  
    }  
  }  
  Poll Performance {  
    Question q1 {  
      "Value the time response"  
      options {  
        A : "Bad"  
        B : "Fair"  
        C : "Good"  
      }  
    }  
  }  
}
```



Challenge 1:

web forms so that admins can specify and edit « polls »

```
PolSystem {
  Poll Quality {
    Question q1 {
      "Value the user experience"
      options {
        A : "Bad"
        B : "Fair"
        C : "Good"
      }
    }
    Question q2 {
      "Value the layout"
      options {
        A : "It was not easy to locate elements"
        B : "I didn't realize"
        C : "It was easy to locate elements"
      }
    }
  }
  Poll Performance {
    Question q1 {
      "Value the time response"
      options {
        A : "Bad"
        B : "Fair"
        C : "Good"
      }
    }
  }
}
```



JSON ?

Interoperability with the Xtext MM ?

Challenge 2:

enable the customization of « widgets » (checkbox, images)

```
PollSystem {  
  Poll Quality {  
    Question q1 {  
      "Value the user experience"  
      options {  
        A : "Bad"  
        B : "Fair"  
        C : "Good"  
      }  
    }  
    Question q2 {  
      "Value the layout"  
      options {  
        A : "It was not easy to locate elements"  
        B : "I didn't realize"  
        C : "It was easy to locate elements"  
      }  
    }  
  }  
  Poll Performance {  
    Question q1 {  
      "Value the time response"  
      options {  
        A : "Bad"  
        B : "Fair"  
        C : "Good"  
      }  
    }  
  }  
}
```



poll1

What is A ?

- B
- C
- D

poll2

What is D ?

- E
- F



designer

User Interface Designer with a passion for designing beautiful and functional user experiences. Minimalist who believes that less is more.



<coder>

Front End Developer who focuses on writing clean, elegant and efficient code. Love HTML5, CSS3, WordPress and a touch of jQuery.

```
<html>  
  <head>  
    <meta charset="utf-8">  
    <title>Jedi</title>  
  </head>  
  <body>  
    <div class="jedi">  
      <h1>Jedi</h1>  
    </div>  
  </body>  
</html>
```

Challenge 2:

enable the customization of « widgets » (checkbox, images)

```
PollSystem {  
  Poll Quality {  
    Question q1 {  
      "Value the user experience"  
      options {  
        A : "Bad"  
        B : "Fair"  
        C : "Good"  
      }  
    }  
    Question q2 {  
      "Value the layout"  
      options {  
        A : "It was not easy to locate elements"  
        B : "I didn't realize"  
        C : "It was easy to locate elements"  
      }  
    }  
  }  
  Poll Performance {  
    Question q1 {  
      "Value the time response"  
      options {  
        A : "Bad"  
        B : "Fair"  
        C : "Good"  
      }  
    }  
  }  
}
```



~~poll1~~

~~What is A ?~~

- ~~• B~~
- ~~• C~~
- ~~• D~~

~~poll2~~

~~What is D ?~~

- ~~• E~~
- ~~• F~~



designer

User Interface Designer with a passion for designing beautiful and functional user experiences. Minimalist who believes that less is more.

<coder>

Front End Developer who focuses on writing clean, elegant and efficient code. Love HTML5, CSS3, WordPress and a touch of jQuery.



#1 A dedicated DSL for specifying the customization (separated or augment the initial DSL)

Challenge 2:

enable the customization of « widgets » (checkbox, images)

```
PollSystem {  
  Poll Quality {  
    Question q1 {  
      "Value the user experience"  
      options {  
        A : "Bad"  
        B : "Fair"  
        C : "Good"  
      }  
    }  
    Question q2 {  
      "Value the layout"  
      options {  
        A : "It was not easy to locate elements"  
        B : "I didn't realize"  
        C : "It was easy to locate elements"  
      }  
    }  
  }  
  Poll Performance {  
    Question q1 {  
      "Value the time response"  
      options {  
        A : "Bad"  
        B : "Fair"  
        C : "Good"  
      }  
    }  
  }  
}
```



~~poll1~~

~~What is A ?~~

- ~~• B~~
- ~~• C~~
- ~~• D~~

~~poll2~~

~~What is D ?~~

- ~~• E~~
- ~~• F~~



designer

User Interface Designer with a passion for designing beautiful and functional user experiences. Minimalist who believes that less is more.

<coder>

Front End Developer who focuses on writing clean, elegant and efficient code. Love HTML5, CSS3, WordPress and a touch of jQuery.



#2 A « pivot » metamodel for the generators
identify common abstractions

MDE chain

Pivot
MM

Model-
to-Model

UI
model

Model-
to-Text

Generator



```
pollSystem {
  pollQuality {
    question q1 {
      "Value the user experience"
      options {
        A : "Bad"
        B : "Fair"
        C : "Good"
      }
    }
    question q2 {
      "Value the layout"
      options {
        A : "It was not easy to locate elements"
        B : "I didn't realize"
        C : "It was easy to locate elements"
      }
    }
  }
  pollPerformance {
    question q1 {
      "Value the time response"
      options {
        A : "Bad"
        B : "Fair"
        C : "Good"
      }
    }
  }
}
```

Another DSL

Text-to-
Model



(web forms)

Model Transformations

Taxonomy + Examples

Abstraction Gap

Transformation is the key

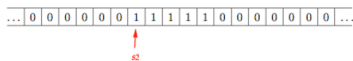
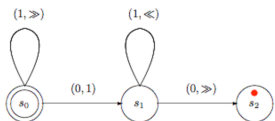
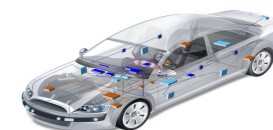
Problem space
domain-specific
language

Transformation

Solution space
implementation
language



ANDROID

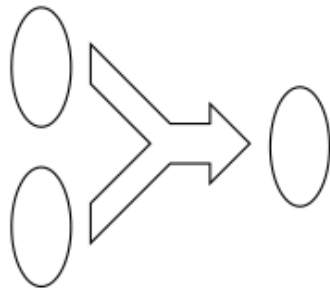


Overview of Generative Software Development

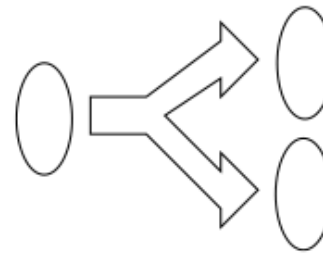
Krzysztof Czarnecki



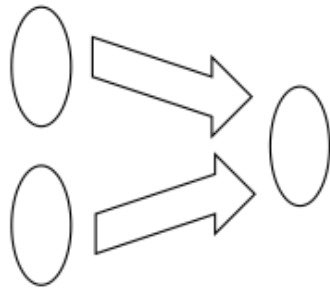
a. Chaining of mappings



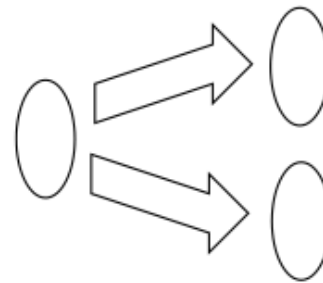
b. Multiple problem spaces



c. Multiple solution spaces



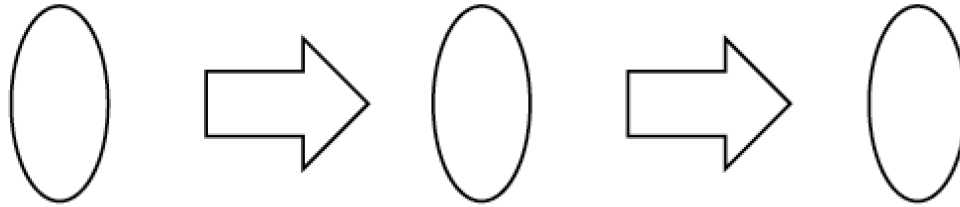
d. Alternative problem spaces



e. Alternative solution spaces

One step/stage transformation

hardly the case



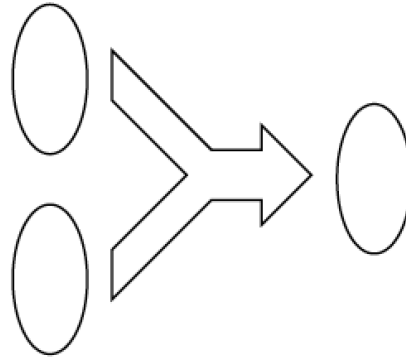
a. Chaining of mappings

```
PollSystem {  
  Poll Quality {  
    Question q1 {  
      "Value the user experience"  
      options {  
        A : "Bad"  
        B : "Fair"  
        C : "Good"  
      }  
    }  
    Question q2 {  
      "Value the layout"  
      options {  
        A : "It was not easy to locate elements"  
        B : "I didn't realize"  
        C : "It was easy to locate elements"  
      }  
    }  
  }  
  Poll Performance {  
    Question q1 {  
      "Value the time response"  
      options {  
        A : "Bad"  
        B : "Fair"  
        C : "Good"  
      }  
    }  
  }  
}
```



Problem space

Combination of expertises/aspects/DSLs



b. Multiple problem spaces

Another DSL
(graphical aspect)

```
PolSystem {  
  Poll Quality {  
    Question q1 {  
      "Value the user experience"  
      options {  
        A : "Bad"  
        B : "Fair"  
        C : "Good"  
      }  
    }  
    Question q2 {  
      "Value the layout"  
      options {  
        A : "It was not easy to locate elements"  
        B : "I didn't realize there were so many elements"  
        C : "It was easy to locate elements"  
      }  
    }  
  }  
  Poll Performance {  
    Question q1 {  
      "Value the time response"  
      options {  
        A : "Bad"  
        B : "Fair"  
        C : "Good"  
      }  
    }  
  }  
}
```

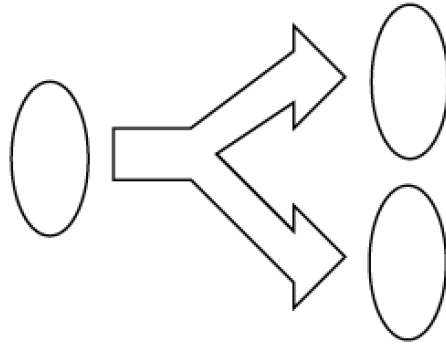
(structural/data aspect)

```
PolSystem {  
  Poll Quality {  
    Question q1 {  
      "Value the user experience"  
      options {  
        A : "Bad"  
        B : "Fair"  
        C : "Good"  
      }  
    }  
    Question q2 {  
      "Value the layout"  
      options {  
        A : "It was not easy to locate elements"  
        B : "I didn't realize there were so many elements"  
        C : "It was easy to locate elements"  
      }  
    }  
  }  
  Poll Performance {  
    Question q1 {  
      "Value the time response"  
      options {  
        A : "Bad"  
        B : "Fair"  
        C : "Good"  
      }  
    }  
  }  
}
```

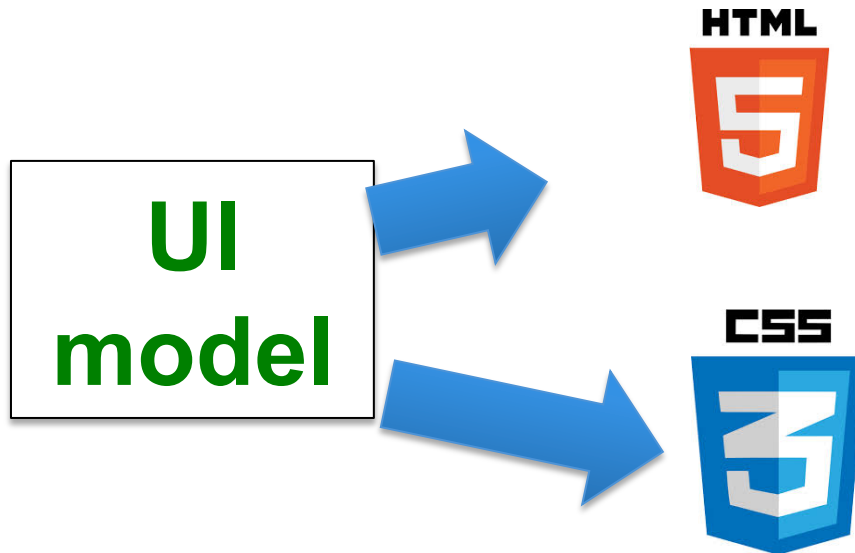
**UI
model**

Solution space

Different targets (e.g., technological platforms)

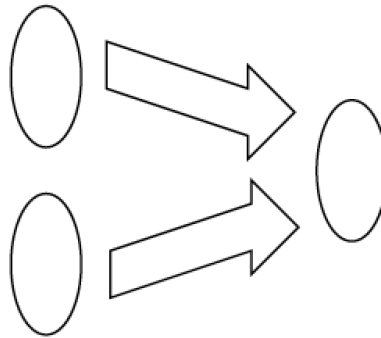


c. Multiple solution spaces



Problem space

e.g., different concrete syntaxes



d. Alternative problem spaces

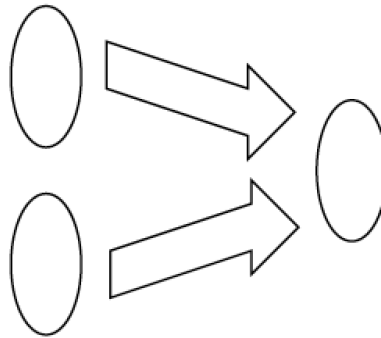
```
PollSystem {  
  Poll Quality {  
    Question q1 {  
      "Value the user experience"  
      options {  
        A : "Bad"  
        B : "Fair"  
        C : "Good"  
      }  
    }  
    Question q2 {  
      "Value the layout"  
      options {  
        A : "It was not easy to locate elements"  
        B : "I didn't realize"  
        C : "It was easy to locate elements"  
      }  
    }  
  }  
  Poll Performance {  
    Question q1 {  
      "Value the time response"  
      options {  
        A : "Bad"  
        B : "Fair"  
        C : "Good"  
      }  
    }  
  }  
}
```

**UI
model**

JSON

Problem space

e.g., different concrete syntaxes



d. Alternative problem spaces

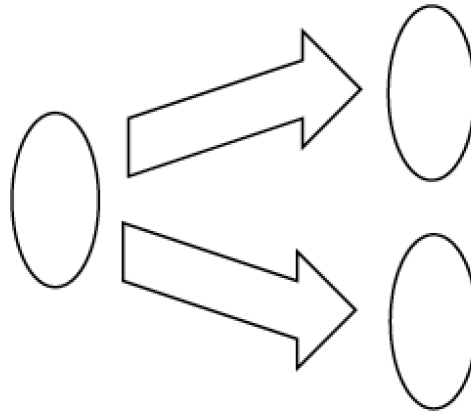
```
PollSystem {  
  Poll Quality {  
    Question q1 {  
      "Value the user experience"  
      options {  
        A : "Bad"  
        B : "Fair"  
        C : "Good"  
      }  
    }  
    Question q2 {  
      "Value the layout"  
      options {  
        A : "It was not easy to locate elements"  
        B : "I didn't realize"  
        C : "It was easy to locate elements"  
      }  
    }  
  }  
  Poll Performance {  
    Question q1 {  
      "Value the time response"  
      options {  
        A : "Bad"  
        B : "Fair"  
        C : "Good"  
      }  
    }  
  }  
}
```

**UI
model**

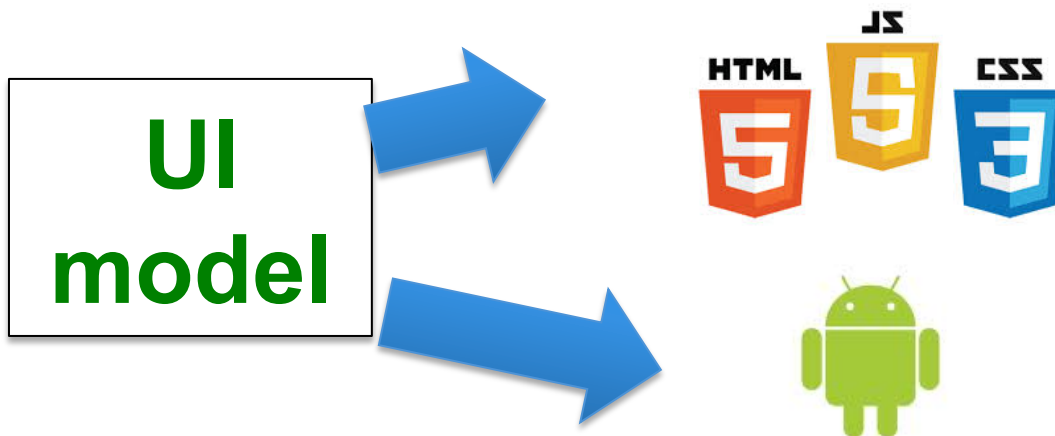
Feature Model
(see next courses and TP6)

Solution space

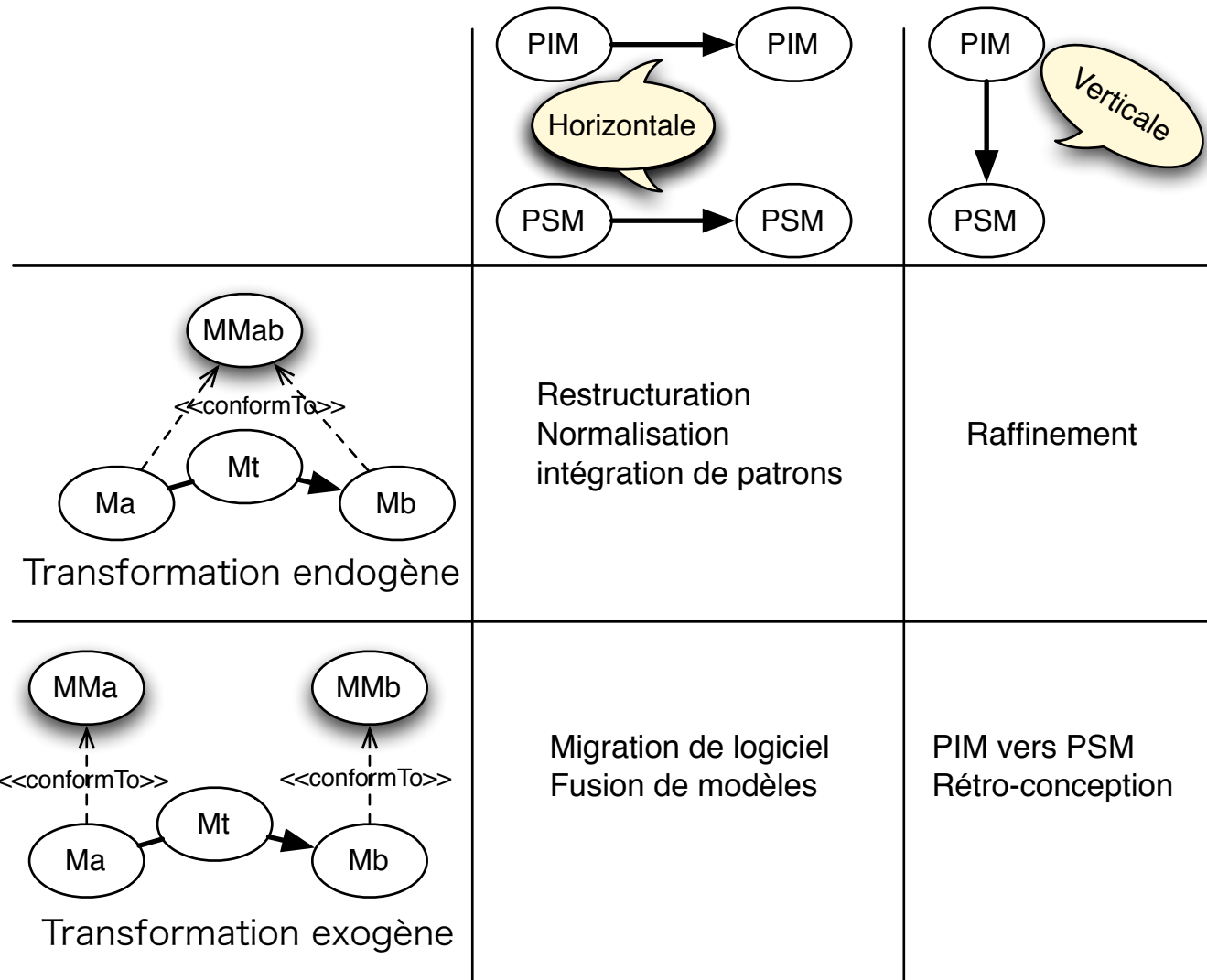
Different targets (e.g., technological platforms)



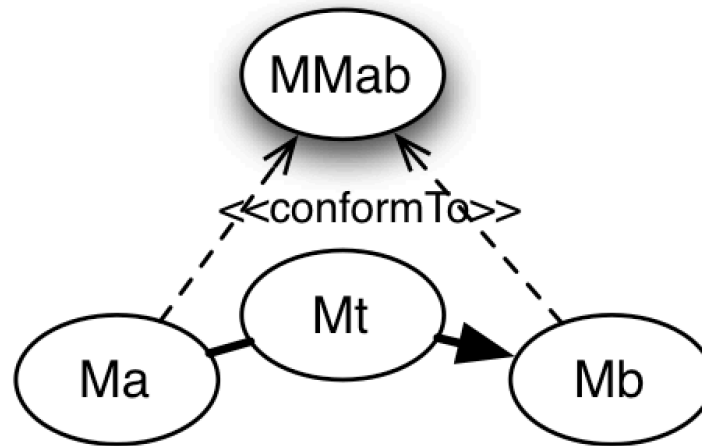
e. Alternative solution spaces



Model Transformation: Taxonomy



Endogeneous Transformation



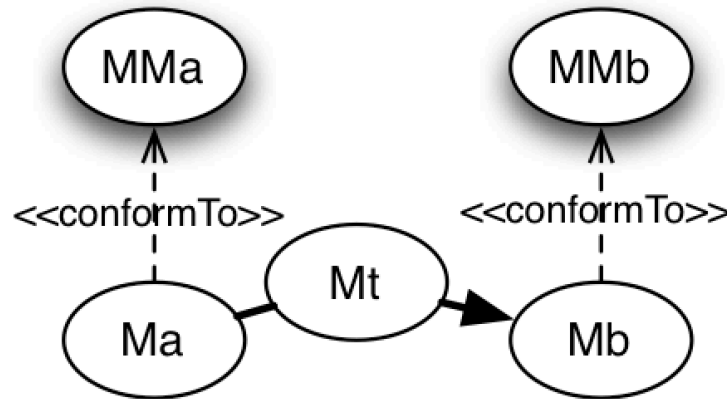
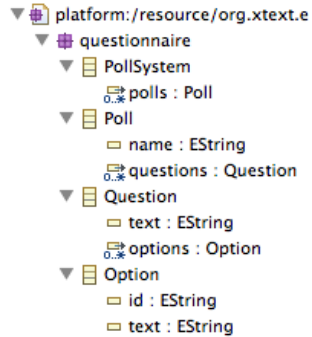
```
fool.q
PollSystem {
    Poll poll1 {
        Question A {
            "What is A ?"
            options
                b : "B"
                c : "C"
                d : "D"
        }
    }
    Poll poll2 {
        Question D {
            "What is D ?"
            options
                e : "E"
                f : "F"
        }
    }
}
```

```
platform:/resource/org.xtext.e
questionnaire
  PollSystem
    polls : Poll
    Poll
      name : EString
      questions : Question
    Question
      text : EString
      options : Option
    Option
      id : EString
      text : EString
```

```
fool.q  foo2.q
PollSystem {
    Poll poll1_poll {
        Question {
            "What is A ?"
            options
                b : "B"
                c : "C"
                d : "D"
        }
    }
    Poll poll2_poll {
        Question {
            "What is D ?"
            options
                e : "E"
                f : "F"
        }
    }
}
```

Exogeneous Transformation

(metamodel)



poll1

What is A ?

- B
- C
- D

poll2

What is D ?

- E
- F

```
fool.q
PollSystem {
    Poll poll1 {
        Question A {
            "What is A ?"
            options
            b : "B"
            c : "C"
            d : "D"
        }
    }
    Poll poll2 {
        Question D {
            "What is D ?"
            options
            e : "E"
            f : "F"
        }
    }
}
```

Vertical Transformation

source and target models reside at the same abstraction level
(e.g., refactoring)

```
fool.q
PollSystem {
  Poll poll1 {
    Question A {
      "What is A ?"
      options
        b : "B"
        c : "C"
        d : "D"
    }
  }
  Poll poll2 {
    Question D {
      "What is D ?"
      options
        e : "E"
        f : "F"
    }
  }
}
```

```
platform:/resource/org.xtext.e
questionnaire
  PollSystem
    polls : Poll
  Poll
    name : EString
    questions : Question
  Question
    text : EString
    options : Option
  Option
    id : EString
    text : EString
```

```
fool.q  foo2.q
PollSystem {
  Poll poll1_poll {
    Question {
      "What is A ?"
      options
        b : "B"
        c : "C"
        d : "D"
    }
  }
  Poll poll2_poll {
    Question {
      "What is D ?"
      options
        e : "E"
        f : "F"
    }
  }
}
```

Horizontal Transformation

the source and target models reside at different abstraction levels
(e.g., refinement)

```
fool.q
PollSystem {
  Poll poll1 {
    Question A {
      "What is A ?"
      options
        b : "B"
        c : "C"
        d : "D"
    }
  }
  Poll poll2 {
    Question D {
      "What is D ?"
      options
        e : "E"
        f : "F"
    }
  }
}
```



poll1

What is A ?

- B
- C
- D

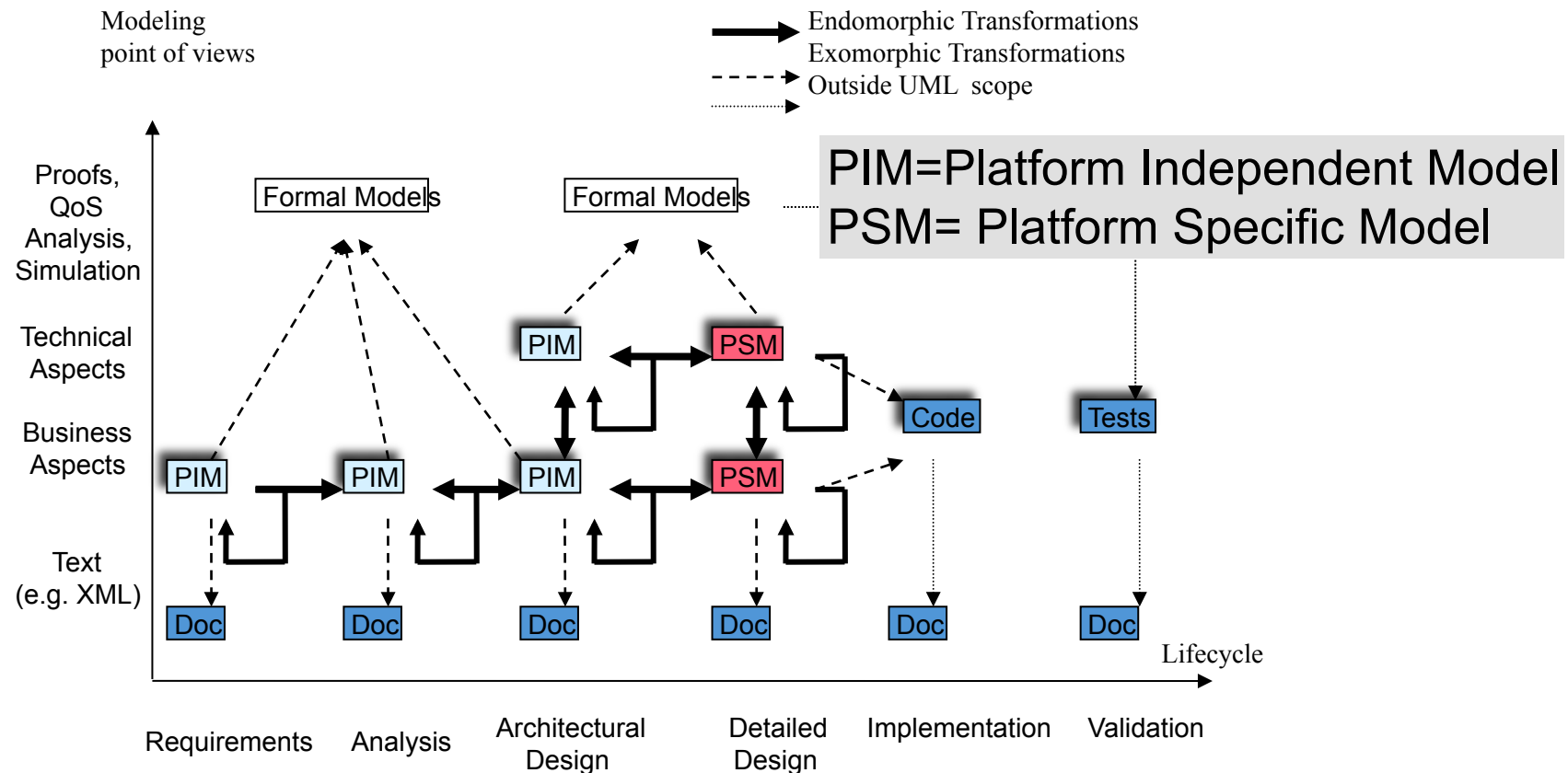
poll2

What is D ?

- E
- F

Chaining of Transformations

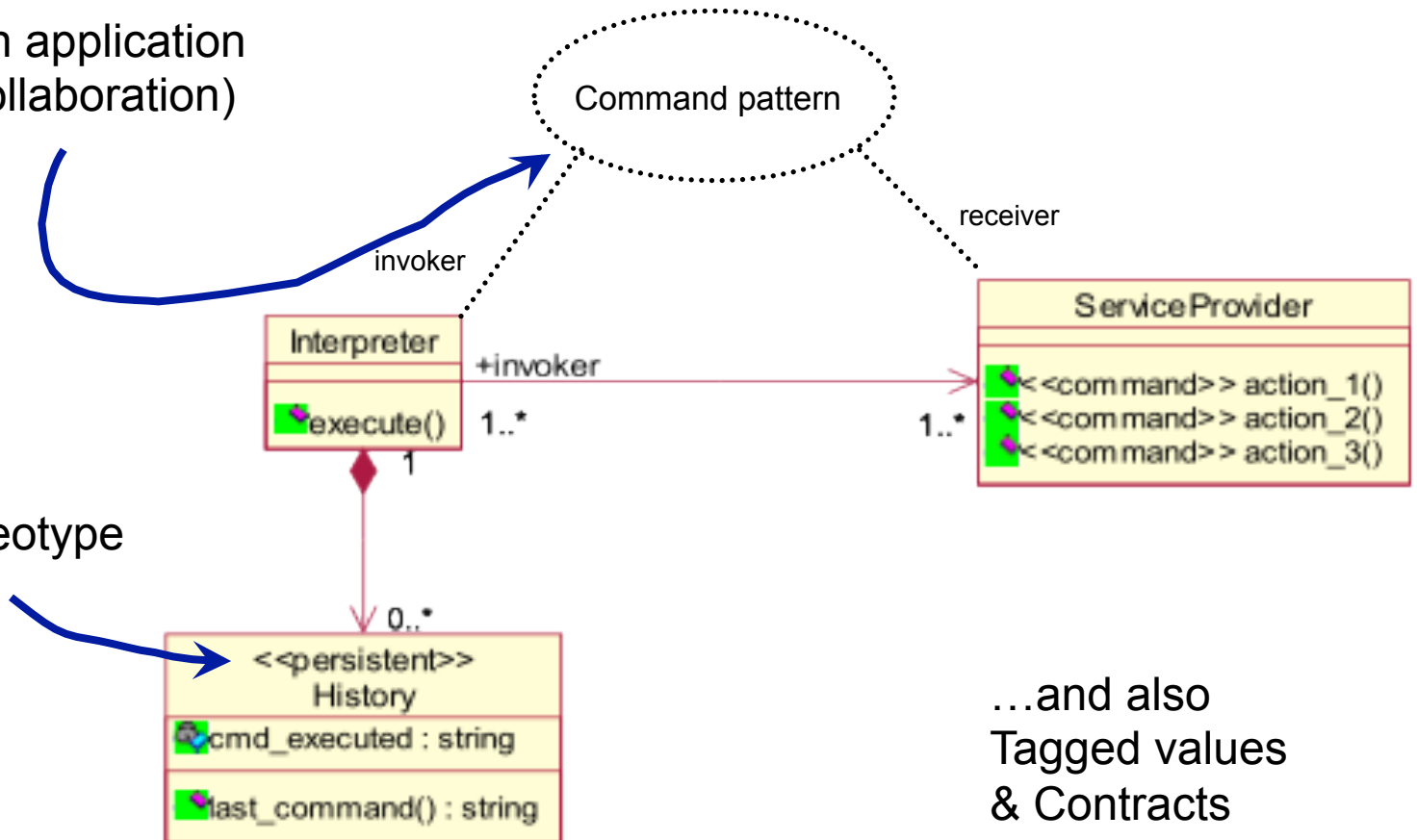
Back to the first courses



Embedding implicit semantics into a model

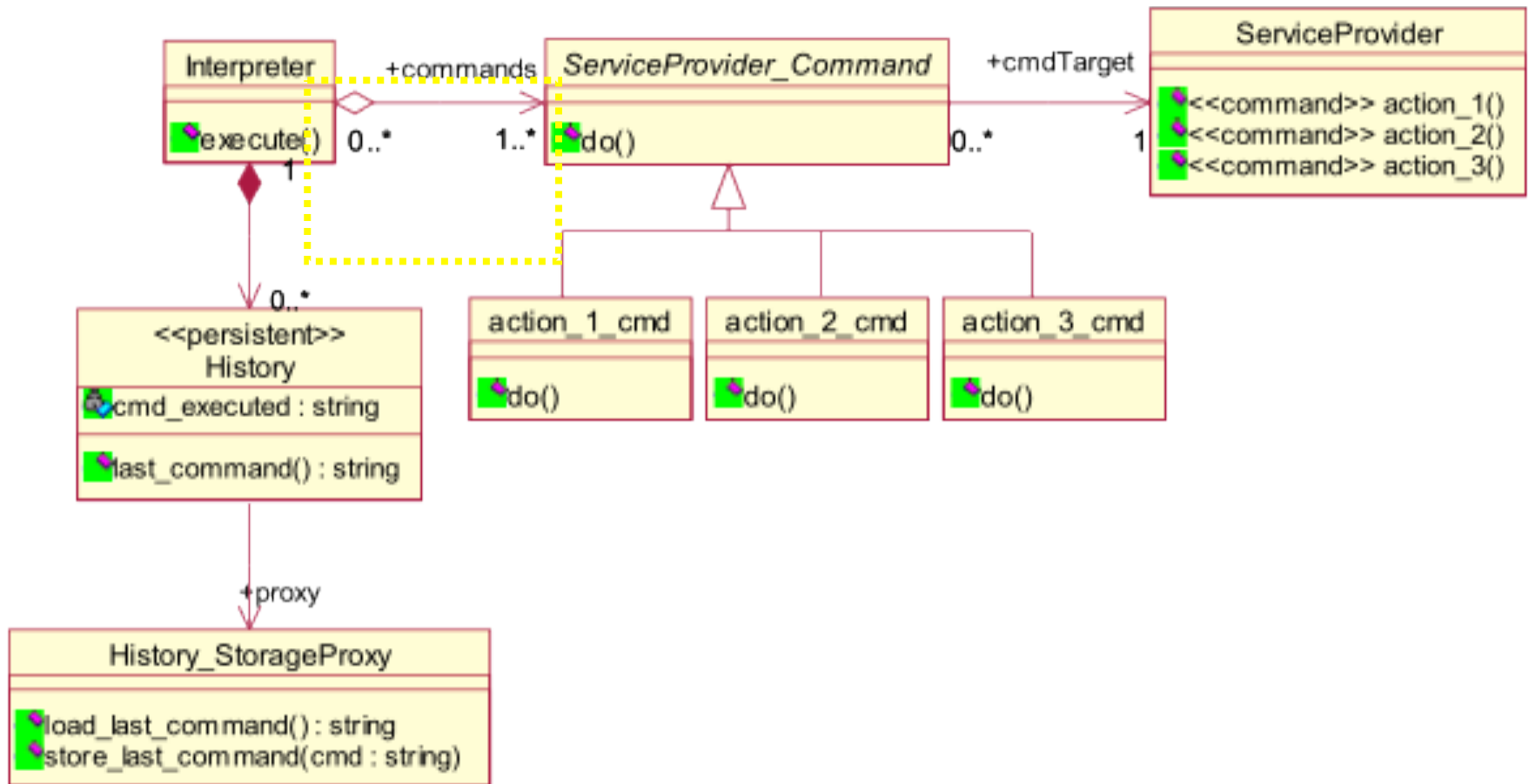
Design pattern application
(parametric collaboration)

Element stereotype

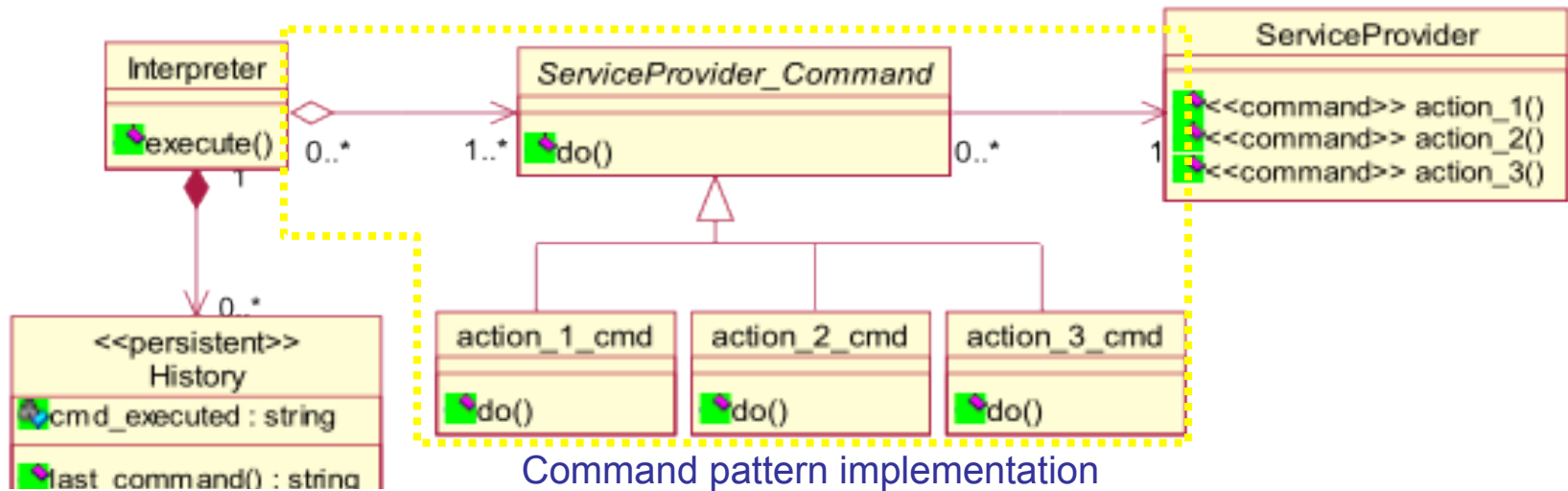


...and also
Tagged values
& Contracts

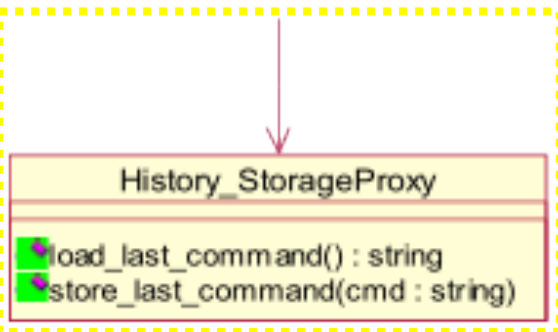
...and the result we want...



How To: Automatic Model Transformations



Command pattern implementation



Persitence implementation

In some domains (e.g.; RT systems) transformations can get more complex than initial model!
=> must be managed with sound SE principles

Xtend

A possible solution
for
model management

Effective Model Management

- How to load/serialize a model?
- How to visit, analyze and transform models?
- You can do it in Java (EMF API)

- We arbitrarily choose  
 - Java 10, interesting « features »
 - Integration within Eclipse ecosystem (incl. Xtext) and facilities to manage models
 - An example of a sophisticated language

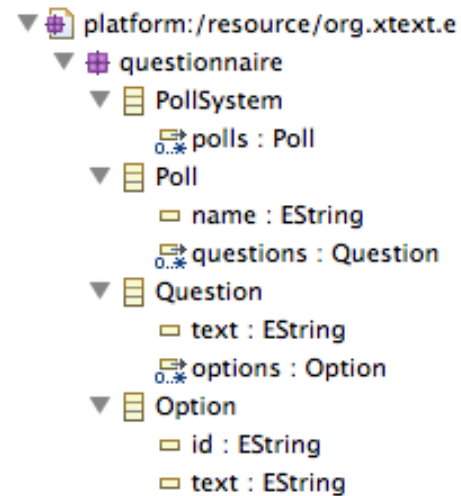
Before going into details of Xtend...

- Recap of the scenarios
 - Text-to-Model
 - Model(s)-to-Model transformation
 - Metamodels as a « bridge » between technologies
 - Model-to-Text
- The solution of some of the « scenarios »
 - Just to give an overview of Xtend capabilities
 - To give a more practical/concrete view of some of the previous scenarios

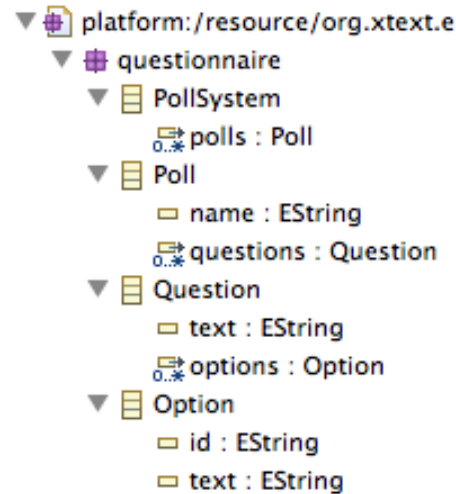
```
def loadPollSystem(URI uri) {  
    new QuestionnaireStandaloneSetupGenerated().createInjectorAndDoEMFRegistration()  
    var res = new ResourceSetImpl().getResource(uri, true);  
    res.contents.get(0) as PollSystem  
}
```

```
def savePollSystem(URI uri, PollSystem pollS) {  
    var Resource rs = new ResourceSetImpl().createResource(uri);  
    rs.getContents().add(pollS);  
    rs.save(new HashMap());  
}
```

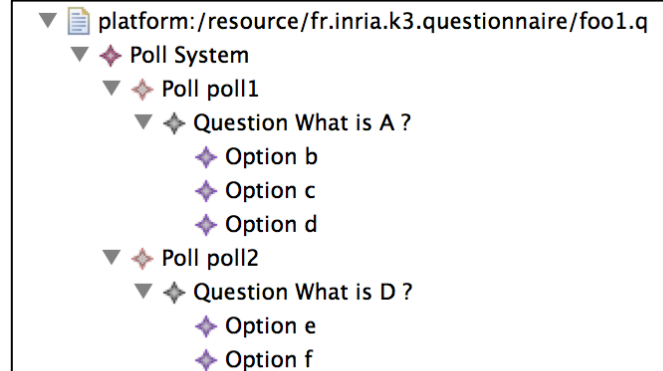
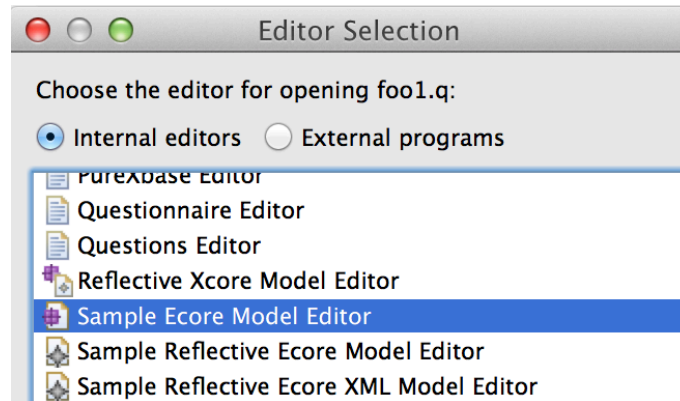
```
@Test  
def test1() {  
  
    // loading  
    var pollS = loadPollSystem(URI.createURI("foo1.q"))  
    assertNotNull(pollS)  
    assertEquals(2, pollS.polls.size)  
  
    // MODEL MANAGEMENT (ANALYSIS, TRANSFORMATION)  
    pollS.polls.forEach[p | p.name = p.name + "_poll"]  
  
    // serializing  
    savePollSystem(URI.createURI("foo2.q"), pollS)  
}
```



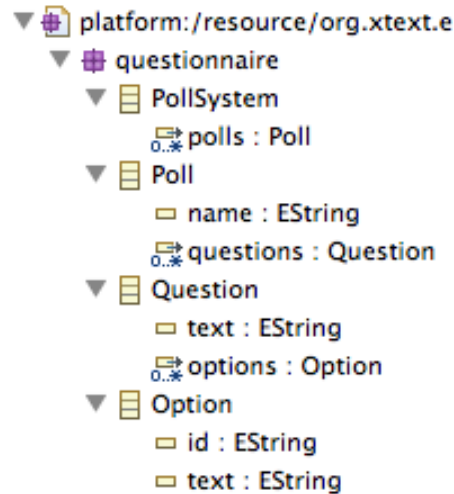
Loading Models (1)



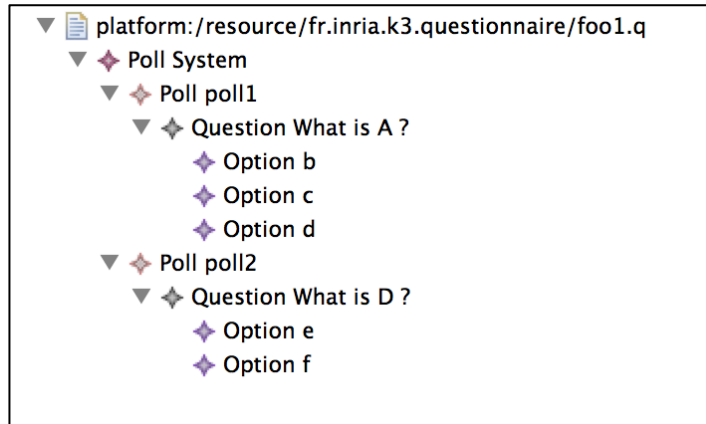
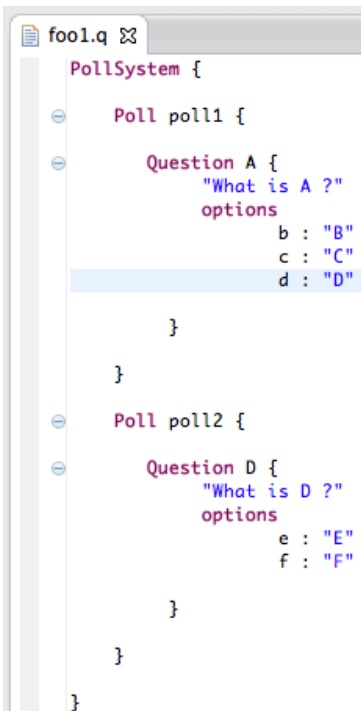
```
foo1.q
PollSystem {
  Poll poll1 {
    Question A {
      "What is A ?"
      options
        b : "B"
        c : "C"
        d : "D"
    }
  }
  Poll poll2 {
    Question D {
      "What is D ?"
      options
        e : "E"
        f : "F"
    }
  }
}
```



Loading Models (2)

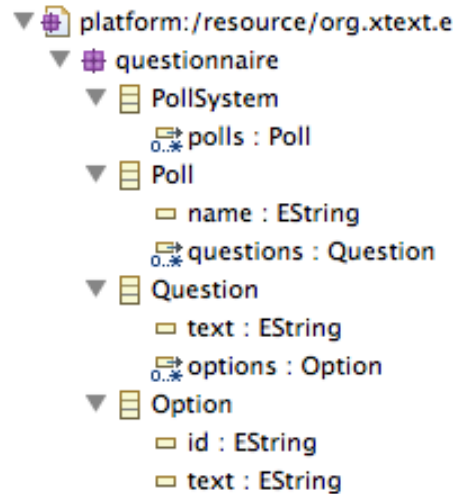


XMI

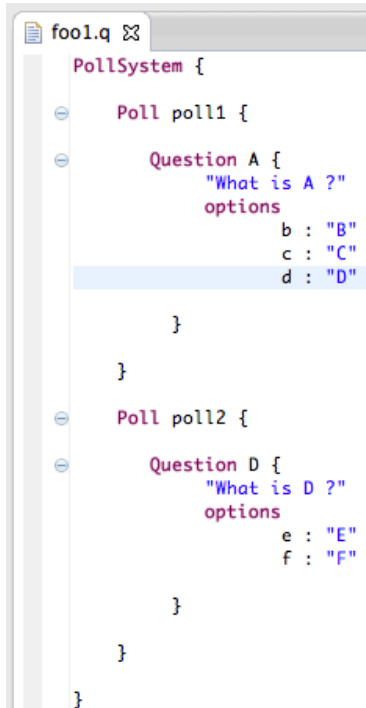


```
<?xml version="1.0" encoding="ASCII"?>
<questionnaire:PollSystem xmi:version="2.0"
xmlns:xmi="http://www.omg.org/XMI"
xmlns:questionnaire="http://www.xtext.org/example/mya">
  <polls name="poll1">
    <questions text="What is A ?">
      <options id="b" text="B"/>
      <options id="c" text="C"/>
      <options id="d" text="D"/>
    </questions>
  </polls>
  <polls name="poll2">
    <questions text="What is D ?">
      <options id="e" text="E"/>
      <options id="f" text="F"/>
    </questions>
  </polls>
</questionnaire:PollSystem>
```

Loading Models (3)

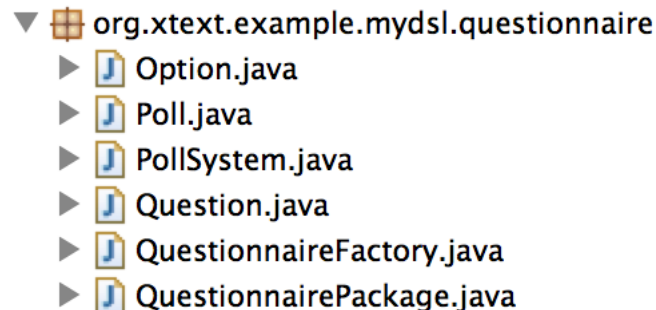
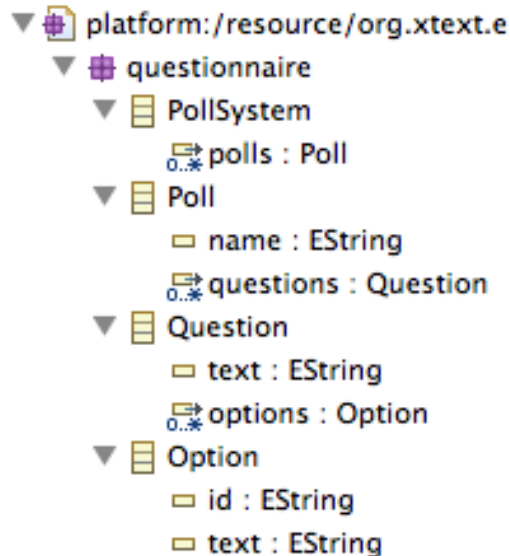


Persistence of Models in XMI (XML Metadata Interchange)



```
<?xml version="1.0" encoding="ASCII"?>
<questionnaire:PollSystem xmi:version="2.0"
xmlns:xmi="http://www.omg.org/XMI"
xmlns:questionnaire="http://www.xtext.org/example/mydsl/Questionnaire">
  <polls name="poll1">
    <questions text="What is A ?">
      <options id="b" text="B"/>
      <options id="c" text="C"/>
      <options id="d" text="D"/>
    </questions>
  </polls>
  <polls name="poll2">
    <questions text="What is D ?">
      <options id="e" text="E"/>
      <options id="f" text="F"/>
    </questions>
  </polls>
</questionnaire:PollSystem>
```


Meta(models) and Java



```
public interface Poll extends EObject
{
    /**
     * Returns the value of the '<em><b>Name</b></em>' attribute.
     * <!-- begin-user-doc -->
     * <p>
     * If the meaning of the '<em>Name</em>' attribute isn't clear,
     * there really should be more of a description here...
     * </p>
     * <!-- end-user-doc -->
     * @return the value of the '<em>Name</em>' attribute.
     * @see #setName(String)
     * @see org.xtext.example.mydsl.questionnaire.QuestionnairePackage#getPoll_Name()
     * @model
     * @generated
     */
    String getName();

    /**
     * Sets the value of the '{@link org.xtext.example.mydsl.questionnaire.Poll#getName <em>Name</em>}' attribute.
     * <!-- begin-user-doc -->
     * <!-- end-user-doc -->
     * @param value the new value of the '<em>Name</em>' attribute.
     * @see #getName()
     * @generated
     */
    void setName(String value);

    /**
     * Returns the value of the '<em><b>Questions</b></em>' containment reference list.
     * The list contents are of type {@link org.xtext.example.mydsl.questionnaire.Question}.
     * <!-- begin-user-doc -->
     * <p>
     * If the meaning of the '<em>Questions</em>' containment reference list isn't clear,
     * there really should be more of a description here...
     * </p>
     * <!-- end-user-doc -->
     * @return the value of the '<em>Questions</em>' containment reference list.
     * @see org.xtext.example.mydsl.questionnaire.QuestionnairePackage#getPoll_Questions()
     * @model containment="true"
     * @generated
     */
    EList<Question> getQuestions();
} // Poll
```

Meta(models) and Java

platform:/resource/org.xtext.e
questionnaire
PollSystem
polls : Poll
Poll
name : EString
questions : Question
Question
text : EString
options : Option
Option
id : EString
text : EString

org.xtext.example.mydsl.questionnaire
Option.java
Poll.java
PollSystem.java
Question.java
QuestionnaireFactory.java
QuestionnairePackage.java

```
public interface Poll extendsEObject {
    /**
     * Returns the value of the '<em>doName</em>' attribute.
     * <!-- begin-user-doc -->
     * @em
     * If the meaning of the '<em>doName</em>' attribute isn't clear,
     * there really should be more of a description here...
     * </em>
     * <!-- end-user-doc -->
     * Returns the value of the '<em>doName</em>' attribute.
     * @see #setName(String)
     * @see org.xtext.example.mydsl.questionnaire.QuestionnairePackage#getPoll_Name()
     * @model
     * @generated
     */
    String getName();

    /**
     * Sets the value of the '{@link org.xtext.example.mydsl.questionnaire.Poll#getName <em>doName</em>}' attribute.
     * <!-- begin-user-doc -->
     * <!-- end-user-doc -->
     * Returns the value of the '<em>doName</em>' attribute.
     * @param value the new value of the '<em>doName</em>' attribute.
     * @see #getName()
     * @generated
     */
    void setName(String value);

    /**
     * Returns the value of the '<em>doQuestions</em>' containment reference list.
     * The list contents are of type {@link org.xtext.example.mydsl.questionnaire.Question}.
     * <!-- begin-user-doc -->
     * <!-- end-user-doc -->
     * If the meaning of the '<em>doQuestions</em>' containment reference list isn't clear,
     * there really should be more of a description here...
     * </em>
     * <!-- end-user-doc -->
     * Returns the value of the '<em>doQuestions</em>' containment reference list.
     * @see org.xtext.example.mydsl.questionnaire.QuestionnairePackage#getPoll_Questions()
     * @model containment="true"
     * @generated
     */
    EList<Question> getQuestions();

    /**
     * Returns the value of the '<em>doOptions</em>' containment reference list.
     * The list contents are of type {@link org.xtext.example.mydsl.questionnaire.Option}.
     * <!-- begin-user-doc -->
     * <!-- end-user-doc -->
     * If the meaning of the '<em>doOptions</em>' containment reference list isn't clear,
     * there really should be more of a description here...
     * </em>
     * <!-- end-user-doc -->
     * Returns the value of the '<em>doOptions</em>' containment reference list.
     * @see org.xtext.example.mydsl.questionnaire.QuestionnairePackage#getPoll_Options()
     * @model containment="true"
     * @generated
     */
    EList<Option> getOptions();
}
```

« Eclipse Modeling Framework (EMF) runtime support to produce a set of Java classes for the model »

```
foo1.q
PollSystem {
    Poll poll1 {
        Question A {
            "What is A ?"
            options
            b : "B"
            c : "C"
            d : "D"
        }
    }
    Poll poll2 {
        Question D {
            "What is D ?"
            options
            e : "E"
            f : "F"
        }
    }
}
```



<http://eclipsesource.com/blogs/tutorials/emf-tutorial/>



Ecore Model

EPackage
EClass
EAttribute
EReference

Code generation



Java Code

Package
Class
Attribute
Reference

```
fool1.q ✕  
  
PollSystem {  
  Poll poll1 {  
    Question A {  
      "What is A ?"  
      options  
        b : "B"  
        c : "C"  
        d : "D"  
    }  
  }  
  
  Poll poll2 {  
    Question D {  
      "What is D ?"  
      options  
        e : "E"  
        f : "F"  
    }  
  }  
}
```



```
def loadPollSystem(uri: URI) {  
  new QuestionnaireStandaloneSetupGenerated().createInjectorAndDoEMFRegistration()  
  var res = new ResourceSetImpl().getResource(uri, true);  
  res.contents.get(0) as PollSystem  
}  
  
def savePollSystem(uri: URI, PollSystem pollS) {  
  var Resource rs = new ResourceSetImpl().createResource(uri);  
  rs.getContents().add(pollS);  
  rs.save(new HashMap());  
}  
  
@Test  
def test1() {  
  // loading  
  var pollS = loadPollSystem(URI.createURI("fool1.q"))  
  assertEquals(2, pollS.polls.size)  
  
  // MODEL MANAGEMENT (ANALYSIS, TRANSFORMATION)  
  pollS.polls.foreach(p => p.name = p.name + "_poll")  
  
  // serializing  
  savePollSystem(URI.createURI("foo2.q"), pollS)  
}
```

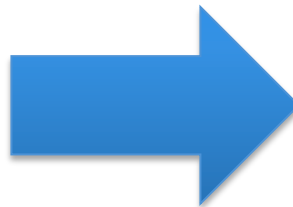
```
fool1.q  foo2.q ✕  
  
PollSystem {  
  Poll poll1_poll {  
    Question {  
      "What is A ?"  
      options  
        b : "B"  
        c : "C"  
        d : "D"  
    }  
  }  
  
  Poll poll2_poll {  
    Question {  
      "What is D ?"  
      options  
        e : "E"  
        f : "F"  
    }  
  }  
}
```

**Questionnaire
MM (ecore)**

**Questionnaire
MM (ecore)**

**Questionnaire
Model 1 (xmi)**

**Questionnaire
Model 2 (xmi)**



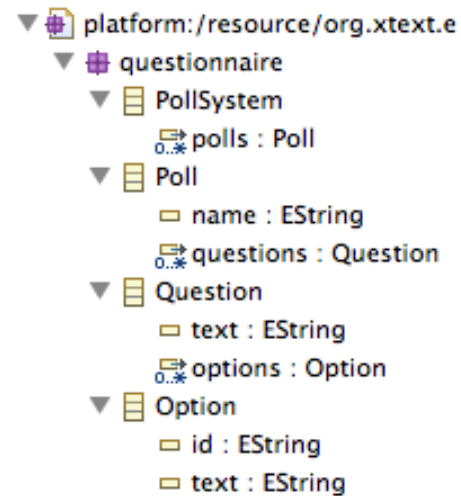
```
PollSystem {
  Poll Quality {
    Question q1 {
      "Value the user experience"
      options {
        A : "Bad"
        B : "Fair"
        C : "Good"
      }
    }
    Question q2 {
      "Value the layout"
      options {
        A : "It was not easy to locate elements"
        B : "I didn't realize"
        C : "It was easy to locate elements"
      }
    }
  }
  Poll Performance {
    Question q1 {
      "Value the time response"
      options {
        A : "Bad"
        B : "Fair"
        C : "Good"
      }
    }
  }
}
```

```
PollSystem {
  Poll Quality {
    Question q1 {
      "Value the user experience"
      options {
        A : "Bad"
        B : "Fair"
        C : "Good"
      }
    }
    Question q2 {
      "Value the layout"
      options {
        A : "It was not easy to locate elements"
        B : "I didn't realize"
        C : "It was easy to locate elements"
      }
    }
  }
  Poll Performance {
    Question q1 {
      "Value the time response"
      options {
        A : "Bad"
        B : "Fair"
        C : "Good"
      }
    }
  }
}
```

```
def loadPollSystem(URI uri) {  
    new QuestionnaireStandaloneSetupGenerated().createInjectorAndDoEMFRegistration()  
    var res = new ResourceSetImpl().getResource(uri, true);  
    res.contents.get(0) as PollSystem  
}
```

```
def savePollSystem(URI uri, PollSystem pollS) {  
    var Resource rs = new ResourceSetImpl().createResource(uri);  
    rs.getContents().add(pollS);  
    rs.save(new HashMap());  
}
```

```
@Test  
def test1() {  
  
    // loading  
    var pollS = loadPollSystem(URI.createURI("foo1.q"))  
    assertNotNull(pollS)  
    assertEquals(2, pollS.polls.size)  
  
    // MODEL MANAGEMENT (ANALYSIS, TRANSFORMATION)  
    pollS.polls.forEach[p | p.name = p.name + "_poll"]  
  
    // serializing  
    savePollSystem(URI.createURI("foo2.q"), pollS)  
}
```



```

@Test
def test2() {

// loading
var pollS = loadPollSystem(URI.createURI("foo1.q"))

// MODEL MANAGEMENT (ANALYSIS, TRANSFORMATION)
var html = toPolls(pollS.polls)
assertNotNull(html)

// serializing (note: we could type check the HTML
// with Xtext by specifying the grammar for instance)
val fw = new FileWriter("foo1.html")
fw.write(html.toString)
fw.close

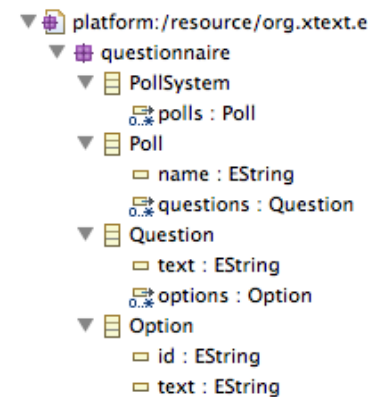
}

```

```

def toPolls(List<Poll> polls) '''
<html>
<body>
  «FOR p : polls»
    «IF p.name != null»
      <h1>«p.name»</h1>
    «ENDIF»
    «FOR q : p.questions»
      <p>
        <h2>«q.text»</h2>
        <ul>
          «FOR o : q.options»
            <li>«o.text»</li>
          «ENDFOR»
        </ul>
      </p>
    «ENDFOR»
  «ENDFOR»
</body>
</html>
'''

```



poll1

What is A ?

- B
- C
- D

poll2

What is D ?

- E
- F

```

foo1.q
PollSystem {
  Poll poll1 {
    Question A {
      "What is A ?"
      options
        b : "B"
        c : "C"
        d : "D"
    }
  }
  Poll poll2 {
    Question D {
      "What is D ?"
      options
        e : "E"
        f : "F"
    }
  }
}

```

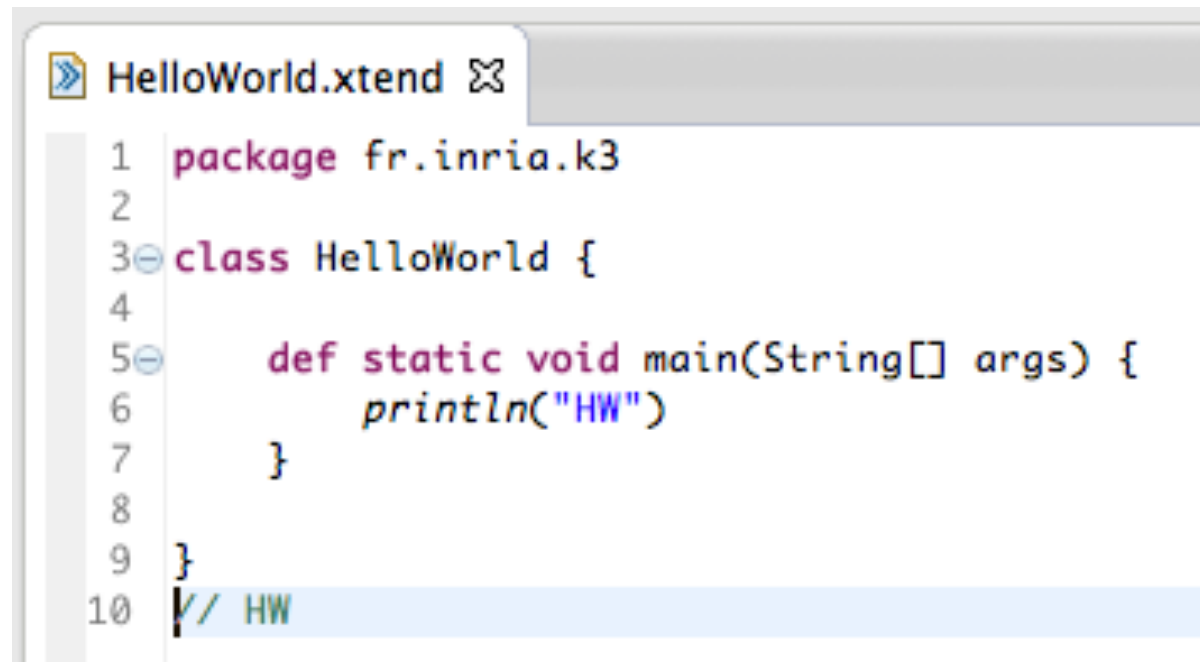
Contract

- Practical foundations of model management
- **Learning and understanding Java 10 (aka Xtend)**
 - advanced features of a general GPL, implementation of a sophisticated language using MDE
- Model transformations
 - Model-to-Text
 - Model-to-Model
- Metaprogramming
 - Revisit annotations (e.g., as in JPA or many frameworks)
- DSLs and model management: all together (Xtext + Xtend)

Xtend

The Basics
(~ Java cheatsheet)

Hello World



The image shows a code editor window with a tab titled 'HelloWorld.xtend'. The code is written in Xtend, a language that combines Java-like syntax with Ruby-like readability. It defines a package 'fr.inria.k3', a class 'HelloWorld', and a static method 'main' that prints 'HW' to the console. The code is as follows:

```
1 package fr.inria.k3
2
3 class HelloWorld {
4
5     def static void main(String[] args) {
6         println("HW")
7     }
8
9 }
10 // HW
```

Semi-colon is optional (within class, methods, etc.)

```
1 package fr.inria.k3.fields
2
3 class MyFielder {
4
5     int count = 1
6     static boolean debug = false
7     var name = 'Foo'           // type String is inferred
8     val UNIVERSAL_ANSWER = 42 // final field with inferred type int
9     // ...
10    public int count2 = 2 ;
11
12 }
13
```

Package Declaration

Semi-colon ';' is optional

HelloWorld.xtend

```
1 package fr.inria.k3
2
3 class HelloWorld {
4
5     def static void main(String[] args) {
6         println("HW")
7     }
8
9 }
10 // HW
```

HelloWorld.xtend

HelloWorld.java

PackageExploder.xtend

```
1 package fr.inria.k3.^def
2
3 class PackageExploder {
4
5 }
```

'^' for avoiding keyword conflicts

Methods

By default:
visibility
conditions set
to public

```
1 package fr.inria.k3.methods
2
3 class FooMethod {
4
5     def foo1() {
6         "A"
7     }
8
9     def foo2() {
10         6 + 3
11     }
12
13     def private foo3() {
14         6 + 3
15     }
16
17     def public foo4() {
18         foo3() * 8
19     }
20 }
21
22 class FooMethodUses {
23
24     def fooUse() {
25         new FooMethod().foo1
26         new FooMethod().foo3()
27
28         new FooMethod().f
29
30         }
31
32         }
33
34     }
35 }
```

foo1 : String - FooMethod.foo1()
foo2 : int - FooMethod.foo2()
foo4 : int - FooMethod.foo4()

```
HelloWorld.xtend
1 package fr.inria.k3
2
3 class HelloWorld {
4
5     def static void main(String[] args) {
6         println("HW")
7     }
8
9 }
10 // HW
```

Methods

Type inference (return type)

```
1 package fr.inria.k3.methods
2
3 class FooMethod {
4
5     def foo1() {
6         "A"
7     }
8
9     def foo2() {
10         6 + 3
11     }
12
13     def private foo3() {
14         6 + 3
15     }
16
17     def public foo4() {
18         foo3() * 8
19     }
20 }
21
22 class FooMethodUses {
23
24     def fooUse() {
25         new FooMethod().foo1
26         new FooMethod().foo3()
27
28
29
30         new FooMethod().f
31
32
33     }
34
35 }
```

foo1 : String – FooMethod.foo1()
foo2 : int – FooMethod.foo2()
foo4 : int – FooMethod.foo4()

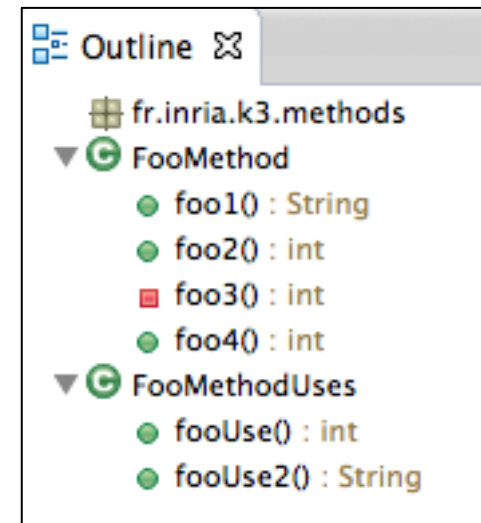
Outline

- fr.inria.k3.methods
 - FooMethod
 - foo1() : String
 - foo2() : int
 - foo3() : int
 - foo4() : int
 - FooMethodUses
 - fooUse() : Object

Method Calling

You can omit parentheses

```
33 def fooUse2() {  
34     3.toString  
35     "4".length  
36     new FooMethod().foo1  
37     new FooMethod().foo1()  
38     new FooMethod().foo1 + 3.toString  
39  
40 }
```



Fields

By default:
visibility
conditions set
to private

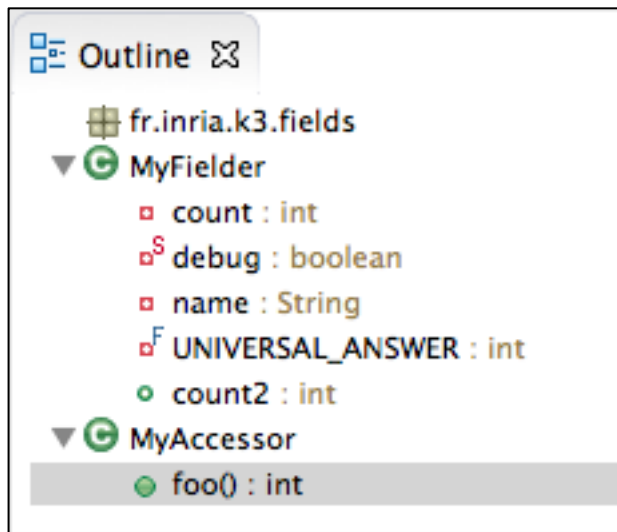
```
1 package fr.inria.k3.fields
2
3 class MyFielder {
4
5     int count = 1
6     static boolean debug = false
7     var name = 'Foo' // type String is inferred
8     val UNIVERSAL_ANSWER = 42 // final field with inferred type int
9     // ...
10    public int count2 = 2 ;
11
12 }
13
14 class MyAccessor {
15
16     def foo() {
17         new MyFielder().
18     }
19 }
20
```

- count2 : int – MyFielder
- class : Class<?> – Object.getClass()
- equals(Object obj) : boolean – Object
- hashCode : int – Object.hashCode()
- identityEquals(Object b) : boolean – Object
- notify : void – Object.notify()
- notifyAll : void – Object.notifyAll()
- toString : String – Object.toString()
- wait : void – Object.wait()
- wait(long timeout) : void – Object

^Space to show shortest proposals

Fields

```
1 package fr.inria.k3.fields
2
3 class MyFielder {
4
5     int count = 1
6     static boolean debug = false
7     var name = 'Foo' // type String is inferred
8     val UNIVERSAL_ANSWER = 42 // final field with inferred type int
9     // ...
10    public int count2 = 2 ;
11
12 }
13
```



primitive types of Java (int, boolean, etc) with autoboxing

var: type inference

val: constant, « final » in Java

Static Methods (::)

```
1 package fr.inria.k3.stat
2
3 import java.util.Collections
4
5 class FooStati {
6
7
8     static var colors = newArrayList(46, 76, 89, 53)
9
10
11     def static void main(String... args) {
12         println("B " + colors)
13         Collections::sort(colors)
14         println("A " + colors)
15
16         colors.add(45)
17         println("A " + colors)
18
19     }
20 }
```

```
B [46, 76, 89, 53]
A [46, 53, 76, 89]
A [46, 53, 76, 89, 45]
```

Pairs

```
1 package fr.inria.k3.pairs
2
3 class FooSugar {
4
5     // syntactic sugar
6     val pair = "spain" -> "italy"
7     var k = pair.key
8     var v = pair.value
9
10    def foo() {
11
12        println("key=" + k + " value=" + v)
13
14    }
15 }
```

fr.inria.k3.pairs

▼ FooSugar

- pair : Pair<String, String>
- k : String
- v : String
- foo() : String

Pairs

```
1 package fr.inria.k3.pairs
2
3 class FooSugar {
4
5
6     static val greetings = newHashMap(
7         "german" -> "Hallo",
8         "english" -> "Hello",
9         "french" -> "Bonjour"
10    )
11
12    def static main(String... args) {
13        greetings.forEach[key, value | println("HW in " + key + " : " + value)]
14    }
15 }
16 }
```

Outline

fr.inria.k3.pairs

FooSugar

greetings : HashMap<String, String>

main(String[]) : void

Problems @ Javadoc Declaration Console

```
<terminated> FooSugar [Java Application] /Library/Java/JavaVirtualMa
HW in english : Hello
HW in french : Bonjour
HW in german : Hallo
```

Immutable data structure

```
1 package fr.inria.k3.stat
2
3 import java.util.Collections
4
5 class FooStati {
6
7
8     static var colors = #[46, 76, 89, 53] // newList(46, 76, 89, 53)
9
10
11     def static void main(String... args) {
12         println("B " + colors)
13         Collections::sort(colors)
14         println("A " + colors)
15
16         colors.add(45)
17         println("A " + colors)
18     }
19 }
20 }
```

```
<terminated> FooStati [Java Application] /Library/Java/JavaVirtualMachines/jdk1.7.0_13.jdk/Contents/
B [46, 76, 89, 53]
Exception in thread "main" java.lang.UnsupportedOperationException
    at java.util.Collections$UnmodifiableList$1.set(Collections.java:1244)
    at java.util.Collections.sort(Collections.java:159)
    at fr.inria.k3.stat.FooStati.main(FooStati.java:15)
```

Constructor

Default visibility: public

```
1 package fr.inria.k3.classes
2
3 class FooConstructor {
4     var String l
5
6     new() {
7         this("F00")
8     }
9
10
11 new (String v) {
12     l = v
13 }
14
15
16 override toString() {
17     l
18 }
19
20 def static void main (String... args) {
21     println("1 " + new FooConstructor())
22     println("2 " + new FooConstructor("F00 2"))
23 }
24 }
```

Problems @ Javadoc Declaration Console

<terminated> FooConstructor [Java Application] /Library/Java/JavaVirt

```
1 F00
2 F00 2
```

override keyword:
mandatory

Cast and Type

```
1 package fr.inria.k3.types
2
3 class FooTypes {
4
5     static val Object obj = "a string"
6     // static val String s = obj
7     static val String s = obj as String // cast
8
9     def static void main(String... args) {
10         println(typeof(String) + " ") // String.class
11         println("\t" + s + "\n")
12     }
13 }
14 }
```

Extension Methods...

« ... allow to add new methods to existing types without modifying them. »

```
def removeVowels (String s){  
    s.replaceAll("[aeiouAEIOU]", "")  
}
```

We can call this method either like in Java:

```
removeVowels("Hello")
```

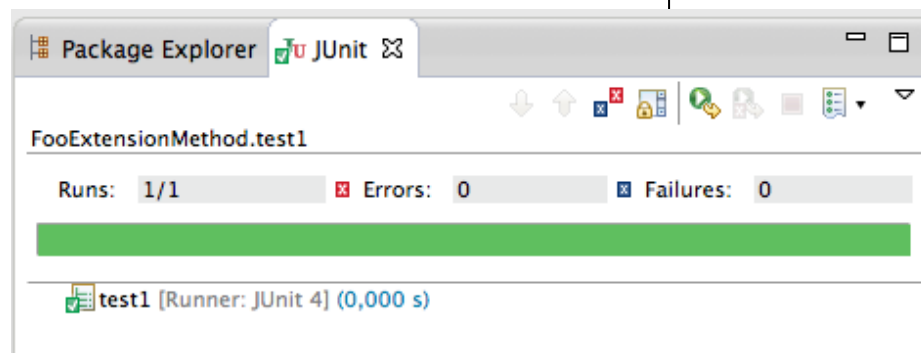
or as an extension method of String:

```
"Hello".removeVowels
```

The first parameter of a method can either be passed in after opening the parentheses or before the method call

Extension Method (local)

```
1 package fr.inria.k3.methods
2
3 import org.junit.Test
4 import static org.junit.Assert.*
5
6 class FooExtensionMethod {
7
8
9     def removeVowels (String s){
10         s.replaceAll("[aeiouAEIOU]", "")
11     }
12
13     def foo() {
14         "HelloWorld".removeVowels
15     }
16
17     @Test
18     def test1() {
19         assertEquals("HllWrld", new FooExtensionMethod().foo)
20     }
21 }
```



Extension Method (library)

```
1 package fr.inria.k3.methods
2
3 import org.junit.Test
4
5 import static org.junit.Assert.*
6
7 class FooExtensionLibrary {
8
9     var listOfStrings = #["A", "b", "c", "D", "e"]
10
11     def foo() {
12         var String h = "hello".toFirstUpper // calls StringExtensions.toFirstUpper(String)
13         listOfStrings.map[ toUpperCase ] // calls ListExtensions.<T, R>map(List<T> list, Function<? super T, ? extends R> mapFunction)
14     }
15
16
17     @Test
18     def test1() {
19         assertEquals("C", new FooExtensionLibrary().foo.get(2))
20     }
21 }
```

```
/**
 * Returns a list that performs the given {@code transformation} for each element of {@code original} when
 * requested. The mapping is done lazily. That is, subsequent iterations of the elements in the list will
 * repeatedly apply the transformation. The returned list is a transformed view of {@code original}; changes to
 * {@code original} will be reflected in the returned list and vice versa (e.g. invocations of {@link List#remove(int)}).
 *
 *
 * @param original
 *     the original list. May not be null.
 * @param transformation
 *     the transformation. May not be null.
 * @return a list that effectively contains the results of the transformation. Never null.
 */
@Pure
public static <T, R> List<R> map(List<T> original, Function1<? super T, ? extends R> transformation) {
    return Lists.transform(original, new FunctionDelegate<T, R>(transformation));
}
```

```
public class ListExtensions {
```

Extension Method (library)

```
1 package fr.inria.k3.methods
2
3 import org.junit.Test
4
5 import static org.junit.Assert.*
6
7 class FooExtensionLibrary {
8
9     var listOfStrings = #["A", "b", "c", "D", "e"]
10
11     def foo() {
12         var String h = "hello".toFirstUpper // calls StringExtensions.toFirstUpper(String)
13         listOfStrings.map[ toUpperCase ] // calls ListExtensions.<T, R>map(List<T> list, Function<? super T, ? extends R> mapFunction)
14     }
15
16
17     @Test
18     def test1() {
19         assertEquals("C", new FooExtensionLibrary().foo.get(2))
20     }
21 }
```

```
/**
 * Returns the {@link String} {@code s} with an {@link Character#isUpperCase(char) upper case} first character. This
 * function is null-safe.
 *
 * @param s
 *         the string that should get an upper case first character. May be null.
 * @return the {@link String} {@code s} with an upper case first character or null if the input
 *         {@link String} {@code s} was null.
 */
@Pure
public static String toFirstUpper(String s) {
    if (s == null || s.length() == 0)
        return s;
    if (Character.isUpperCase(s.charAt(0)))
        return s;
    if (s.length() == 1)
        return s.toUpperCase();
    return s.substring(0, 1).toUpperCase() + s.substring(1);
}
```

```
public class StringExtensions {
```

Lambda Expression

(Java 8 will support it)

Anonymous classes can be found everywhere in Java code...

```
1. // Java Code!
2. final JTextField textField = new JTextField();
3. textField.addActionListener(new ActionListener() {
4.     @Override
5.     public void actionPerformed(ActionEvent e) {
6.         textField.setText("Something happened!");
7.     }
8. });
```

... And have always been the poor-man's replacement for lambda expressions in Java.

Lambda Expression

(Xtend answer)

Anonymous classes can be found everywhere in Java code...

```
1. // Java Code!  
2. final JTextField textField = new JTextField();  
3. textField.addActionListener(new ActionListener() {  
4.     @Override  
5.     public void actionPerformed(ActionEvent e) {  
6.         textField.setText("Something happened!");  
7.     }  
8. });
```

No need to
specify the
type for e

```
1. textField.addActionListener([ e |  
2.     textField.setText("Something happened!")  
3. ])
```

You can even
ommit e

```
1. textField.addActionListener([  
2.     textField.setText("Something happened!")  
3. ])
```

Lambda Expression

(Xtend answer, more impressive examples)

```
1. Collections.sort(someStrings) [ a, b |  
2.   a.length - b.length  
3. ]
```

Java 8: `shapes.forEach(s -> { s.setColor(RED); });`

Xtend: `shapes.forEach[color = RED]`

Java 8: `shapes.stream()
 .filter(s -> s.getColor() == BLUE)
 .forEach(s -> { s.setColor(RED); });`

Xtend: `shapes.stream
 .filter[color == BLUE]
 .forEach[color = RED]`

Lambda Expression

(Xtend answer, more impressive examples)

```
class FooLambda {  
  
    val l = [String s | s.length]  
    var (String)=>int l2 = [it.length] // it is a keyword for refering to the first parameter  
    var (String)=>int l3 = [length] // we can even ommit it or the first parameter  
  
    @Test  
    def test1() {  
        assertEquals(l.apply("RRRR"), l2.apply("PPPP"))  
        assertEquals(l2.apply("RRRR"), l3.apply("PPPP"))  
    }  
}
```

▼ FooLambda

- ▣ l : Function1<String, Integer>
- ▣ l2 : (String)=>int
- ▣ l3 : (String)=>int
- test1() : void

Templates

```
1 package fr.inria.k3.templates
2
3 import org.junit.Test
4 import static org.junit.Assert.*
5
6 class FooTempl {
7
8     def someHTML(String content) '''<html><body>«content»</body></html>'''
9
10
11
12 @Test
13 def test1() {
14     assertEquals("<html><body>HW</body></html>", someHTML('HW').toString)
15 }
16
17 }
```


Templates (2)

```
@Test
def test2() {

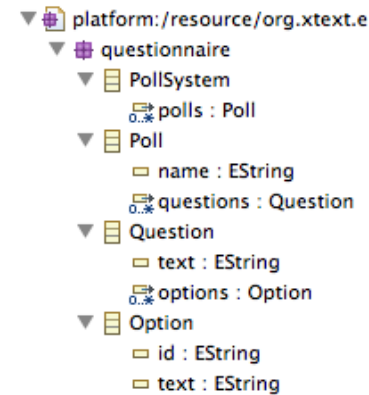
    // loading
    var pollS = loadPollSystem(URI.createURI("foo1.q"))

    // MODEL MANAGEMENT (ANALYSIS, TRANSFORMATION)
    var html = toPolls(pollS.polls)
    assertNotNull(html)

    // serializing (note: we could type check the HTML
    // with Xtext by specifying the grammar for instance)
    val fw = new FileWriter("foo1.html")
    fw.write(html.toString)
    fw.close

}
```

```
def toPolls(List<Poll> polls) '''
    <html>
    <body>
        «FOR p : polls»
        «IF p.name != null»
            <h1>«p.name»</h1>
        «ENDIF»
        «FOR q : p.questions»
            <p>
                <h2>«q.text»</h2>
                <ul>
                    «FOR o : q.options»
                        <li>«o.text»</li>
                    «ENDFOR»
                </ul>
            </p>
        «ENDFOR»
    «ENDFOR»
    </body>
</html>
'''
```



poll1

What is A ?

- B
- C
- D

poll2

What is D ?

- E
- F

```
foo1.q
PollSystem {
    Poll poll1 {
        Question A {
            "What is A ?"
            options
            b : "B"
            c : "C"
            d : "D"
        }
    }
    Poll poll2 {
        Question D {
            "What is D ?"
            options
            e : "E"
            f : "F"
        }
    }
}
```

Templates (3)

- You already experiment with web templating engines (JSP, Scala templates in Play!, Symfony templates, etc.)

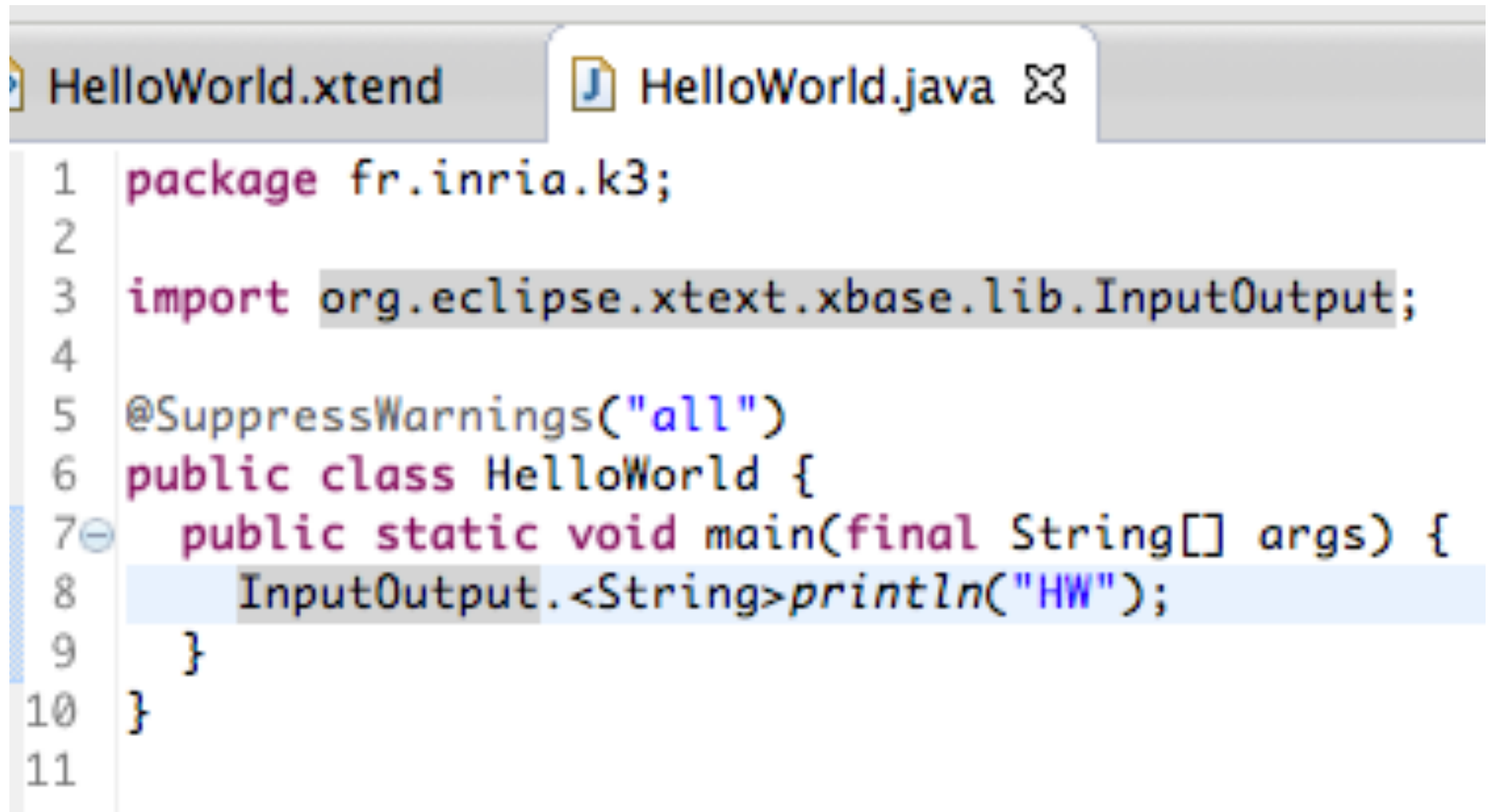
```
<h1>Exemple de page JSP</h1>
<!-- Impression de variables -->
<p>Au moment de l'exécution de ce script, nous sommes le <%= date %>.</p>
<p>Cette page a été affichée <%= nombreVisites %> fois !</p>
</body>
</html>
```

- Alternatives exist in the modeling world
 - Multiple pre-defined and customizable generators



- Xtend: seamless integration into a general purpose language

Xtend to Java



```
1 package fr.inria.k3;
2
3 import org.eclipse.xtext.xbase.lib.InputOutput;
4
5 @SuppressWarnings("all")
6 public class HelloWorld {
7     public static void main(final String[] args) {
8         InputOutput.<String>println("HW");
9     }
10 }
11
```

Xtend to Java (2)

more after

```
HelloWorld.xtend  HelloWorld.java ✕  
1 package fr.inria.k3;  
2  
3 import org.eclipse.xtext.xbase.lib.InputOutput;  
4  
5 @SuppressWarnings("all")  
6 public class HelloWorld {  
7     public static void main(final String[] args) {  
8         InputOutput.<String>println("HW");  
9     }  
10 }  
11
```



```
package org.eclipse.xtext.xbase.lib;  
  
import com.google.common.annotations.GwtCompatible;  
  
/**  
 * Utilities to print information to the console.  
 *  
 * @author Sven Efftinge - Initial contribution and API  
 */  
@GwtCompatible public class InputOutput {  
  
    /**  
     * Prints a newline to standard out, by delegating directly to System.out.println()  
     * @since 2.3  
     */  
    public static void println() {  
        System.out.println();  
    }  
  
    /**  
     * Prints the given {@code object} to {@link System#out System.out} and terminate the line. Useful to log partial  
     * expressions to trap errors, e.g. the following is possible: println(1 + println(2)) + 3  
     *  
     * @param object  
     *         the to-be-printed object  
     * @return the printed object.  
     */  
    public static <T> T println(T object) {  
        System.out.println(object);  
        return object;  
    }  
}
```

Xtend/Xtext

Back to our scenarios

MDE chain

Pivot
MM

Model-
to-Model

UI
model

Model-
to-Text

Generator



(web forms)

```
pollSystem {
  pollQuality {
    question q1 {
      "Value the user experience"
      options {
        A: "Bad"
        B: "Fair"
        C: "Good"
      }
    }
    question q2 {
      "Value the layout"
      options {
        A: "It was not easy to locate elements"
        B: "I didn't realize"
        C: "It was easy to locate elements"
      }
    }
  }
  pollPerformance {
    question q1 {
      "Value the time response"
      options {
        A: "Bad"
        B: "Fair"
        C: "Good"
      }
    }
  }
}
```

Another DSL

Text-to-
Model

Refactoring of Polls

```
foo1.q ✕  
  
PollSystem {  
  
  Poll poll1 {  
  
    Question A {  
      "What is A ?"  
      options  
        b : "B"  
        c : "C"  
        d : "D"  
    }  
  
  }  
  
  Poll poll2 {  
  
    Question D {  
      "What is D ?"  
      options  
        e : "E"  
        f : "F"  
    }  
  
  }  
  
}
```



```
def loadPollSystem(uri: URI) {  
  new QuestionnaireSetupGenerated().createInjectorAndDoEMFRegistration()  
  var res = new ResourceSetImpl().getResource(uri, true);  
  res.contents.get(0) as PollSystem  
}  
  
def savePollSystem(uri: URI, PollSystem pollS) {  
  var Resource rs = new ResourceSetImpl().createResource(uri);  
  rs.getContents().add(pollS);  
  rs.save(new HashMap());  
}  
  
@Test  
def test1() {  
  // loading  
  var pollS = loadPollSystem(URI.createURI("foo1.q"))  
  assertEquals(1, pollS)  
  assertEquals(2, pollS.polls.size)  
  
  // MODEL MANAGEMENT (ANALYSIS, TRANSFORMATION)  
  pollS.polls.foreach(p | p.name = p.name + "_poll")  
  
  // serializing  
  savePollSystem(URI.createURI("foo2.q"), pollS)  
}
```

```
foo1.q  foo2.q ✕  
  
PollSystem {  
  
  Poll poll1_poll {  
  
    Question {  
      "What is A ?"  
      options  
        b : "B"  
        c : "C"  
        d : "D"  
    }  
  
  }  
  
  Poll poll2_poll {  
  
    Question {  
      "What is D ?"  
      options  
        e : "E"  
        f : "F"  
    }  
  
  }  
  
}
```

**Questionnaire
MM (ecore)**

**Questionnaire
MM (ecore)**

**Questionnaire
Model 1 (xmi)**

**Questionnaire
Model 2 (xmi)**

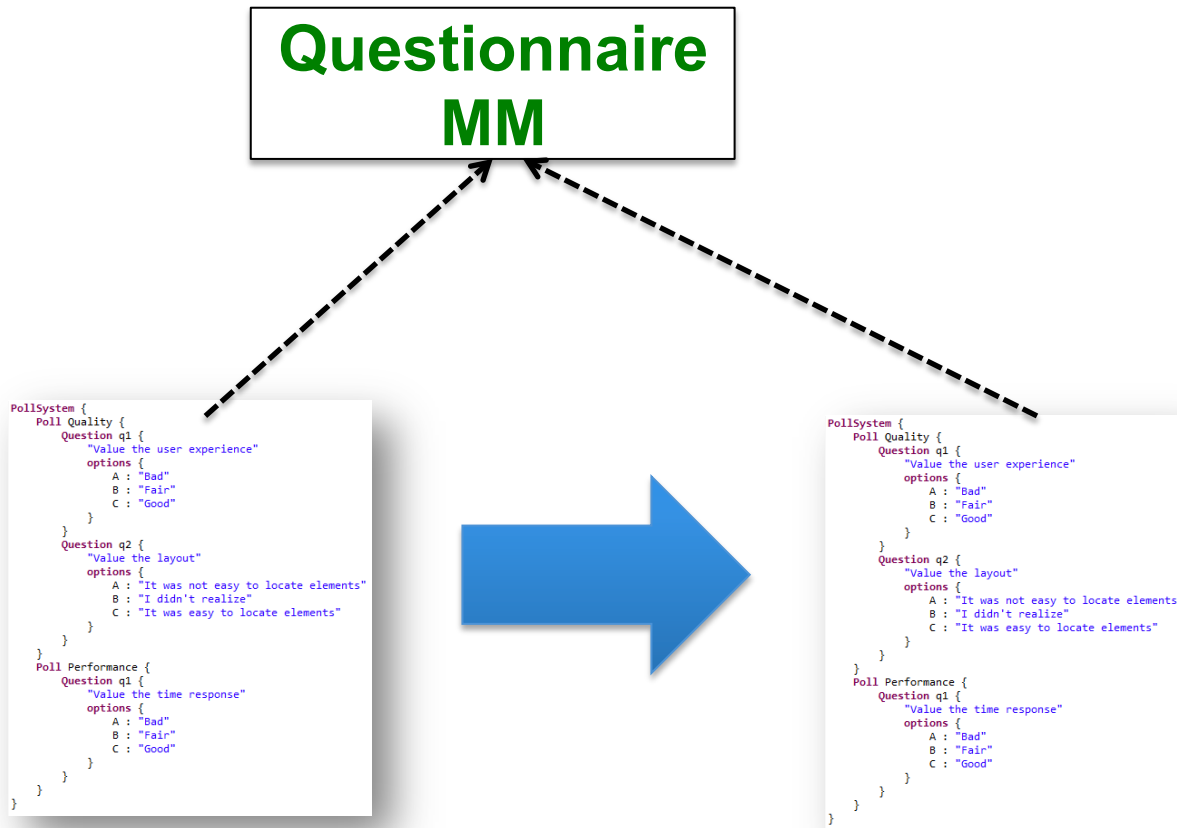
xtext

xtext

```
PollSystem {  
  Poll Quality {  
    Question q1 {  
      "Value the user experience"  
      options {  
        A : "Bad"  
        B : "Fair"  
        C : "Good"  
      }  
    }  
    Question q2 {  
      "Value the layout"  
      options {  
        A : "It was not easy to locate elements"  
        B : "I didn't realize"  
        C : "It was easy to locate elements"  
      }  
    }  
  }  
  Poll Performance {  
    Question q1 {  
      "Value the time response"  
      options {  
        A : "Bad"  
        B : "Fair"  
        C : "Good"  
      }  
    }  
  }  
}
```

```
PollSystem {  
  Poll Quality {  
    Question q1 {  
      "Value the user experience"  
      options {  
        A : "Bad"  
        B : "Fair"  
        C : "Good"  
      }  
    }  
    Question q2 {  
      "Value the layout"  
      options {  
        A : "It was not easy to locate elements"  
        B : "I didn't realize"  
        C : "It was easy to locate elements"  
      }  
    }  
  }  
  Poll Performance {  
    Question q1 {  
      "Value the time response"  
      options {  
        A : "Bad"  
        B : "Fair"  
        C : "Good"  
      }  
    }  
  }  
}
```


Endogeneous Transformation



xtext

```
def loadPollSystem(URI uri) {  
    new QuestionnaireStandaloneSetupGenerated().createInjectorAndDoEMFRegistration()  
    var res = new ResourceSetImpl().getResource(uri, true);  
    res.contents.get(0) as PollSystem  
}
```

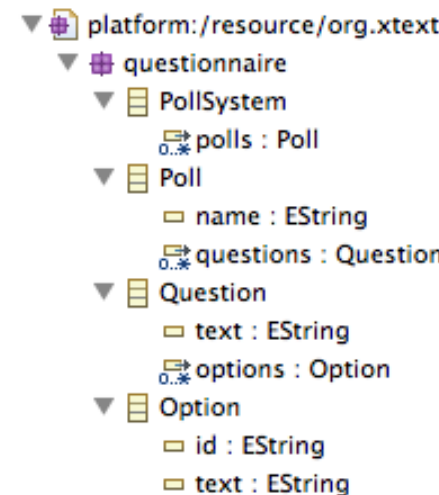
```
def savePollSystem(URI uri, PollSystem pollS) {  
    var Resource rs = new ResourceSetImpl().createResource(uri);  
    rs.getContents().add(pollS);  
    rs.save(new HashMap());  
}
```

@Test

```
def test1() {  
  
    // loading  
    var pollS = loadPollSystem(URI.createURI("foo1.q"))  
    assertNotNull(pollS)  
    assertEquals(2, pollS.polls.size)  
  
    // MODEL MANAGEMENT (ANALYSIS, TRANSFORMATION)  
    pollS.polls.forEach[p | p.name = p.name + "_poll"]  
  
    // serializing  
    savePollSystem(URI.createURI("foo2.q"), pollS)  
}
```

xtext

xtext



```

@Test
def test2() {

// loading
var pollS = loadPollSystem(URI.createURI("foo1.q"))

// MODEL MANAGEMENT (ANALYSIS, TRANSFORMATION)
var html = toPolls(pollS.polls)
assertNotNull(html)

// serializing (note: we could type check the HTML
// with Xtext by specifying the grammar for instance)
val fw = new FileWriter("foo1.html")
fw.write(html.toString)
fw.close

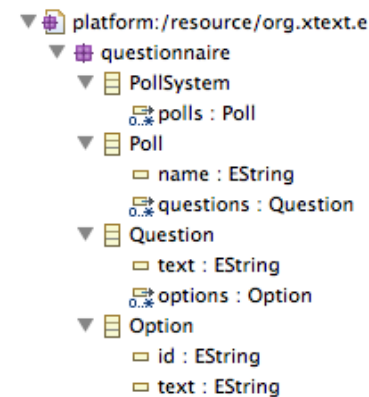
}

```

```

def toPolls(List<Poll> polls) '''
<html>
<body>
  «FOR p : polls»
    «IF p.name != null»
      <h1>«p.name»</h1>
    «ENDIF»
    «FOR q : p.questions»
      <p>
        <h2>«q.text»</h2>
        <ul>
          «FOR o : q.options»
            <li>«o.text»</li>
          «ENDFOR»
        </ul>
      </p>
    «ENDFOR»
  «ENDFOR»
</body>
</html>
'''

```



poll1

What is A ?

- B
- C
- D

poll2

What is D ?

- E
- F

```

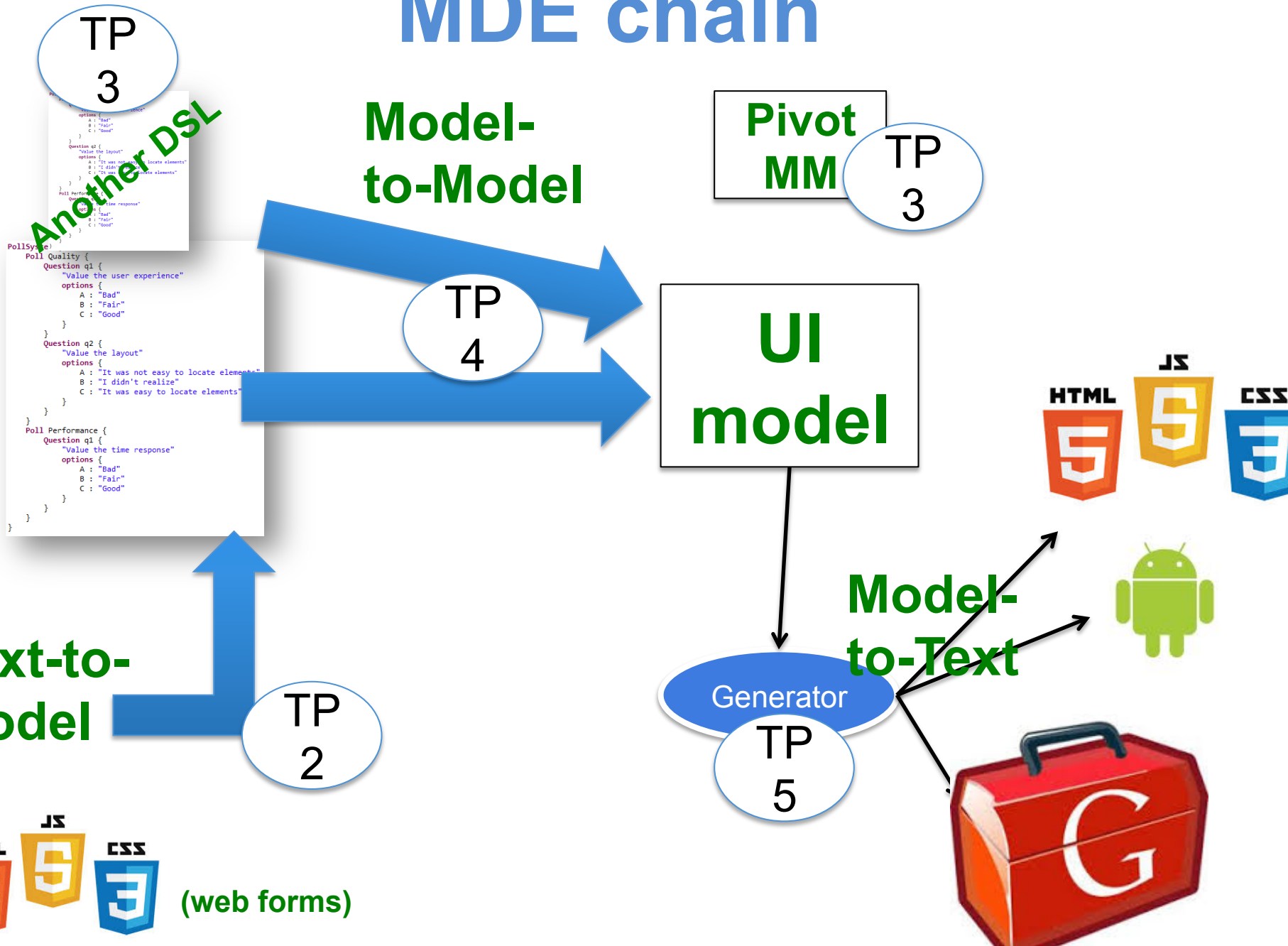
foo1.q
PollSystem {
  Poll poll1 {
    Question A {
      "What is A ?"
      options
        b : "B"
        c : "C"
        d : "D"
    }
  }
  Poll poll2 {
    Question D {
      "What is D ?"
      options
        e : "E"
        f : "F"
    }
  }
}

```

Lab Sessions

Overview

MDE chain



Xtend

Advanced features (active annotation)
Model transformation in depth
MDE enables Xtend

<http://www.eclipse.org/xtend/documentation.html>

<http://jnario.org/org/jnario/jnario/documentation/20FactsAboutXtendSpec.html>

<http://blog.efftinge.de/2012/12/java-8-vs-xtend.html>

<http://eclipsesource.com/blogs/tutorials/emf-tutorial/>

The logo for Xtend, featuring a stylized 'X' composed of two overlapping blue and purple chevron-like shapes, followed by the word 'tend' in a bold, black, sans-serif font.