

# Model Management in Xtend (first part)

Mathieu Acher

@acherm

Maître de Conférences

[mathieu.acher@irisa.fr](mailto:mathieu.acher@irisa.fr)

# Plan

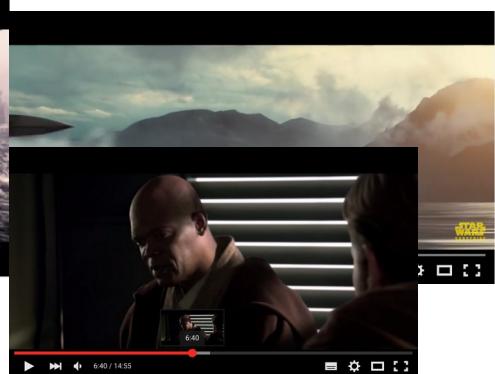
- Model Management in a nutshell
  - Loading, serializing, transforming models
- Xtend
  - Java 10, cheatsheet
  - Advanced features: extension methods, active annotations, template expressions
  - Xtend: behind the magic (Xtext+MDE)
- Model Management + Xtend
  - Model transformations
  - @Aspect annotation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



**Generator**  
~ composition of  
video sequences

**video  
variants**





## Generator ~ composition of video sequences



video  
variants





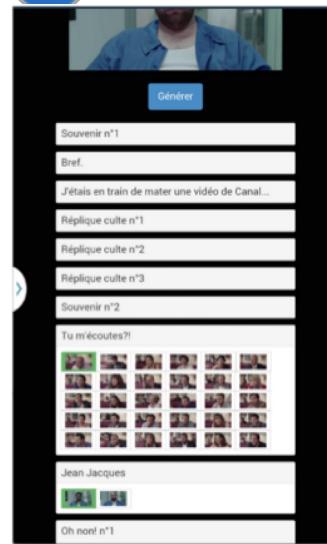
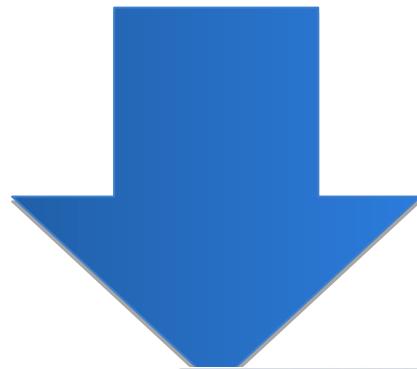
```

foo1.videogen ✘

mandatory videoseq v1 "https://www.youtube.com/watch?v=PJNi1uYhV5w"
optional videoseq v2 "v2Folder/v2.mp4"
alternatives v3 {
    videoseq v31 "v3/seq1.mp4"
    videoseq v32 "v3/seq1.mp4"
    videoseq v33 "v3/seq1.mp4"
}

alternatives v4 {
    videoseq v41 "v4/seq1.mp4"
    videoseq v42 "v4/seq1.mp4"
}
mandatory videoseq v5 "https://www.youtube.com/watch?v=ezKx-S0LiNQ"

```



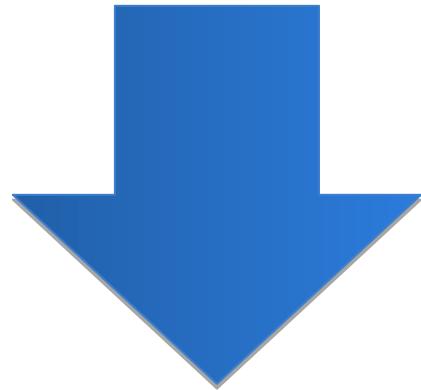
- ## Website/online
- Random generation
  - Configurator
  - Game
  - ...



```
foo1.videogen ✘

mandatory videoseq v1 "https://www.youtube.com/watch?v=PjNi1uYhV5w"
optional videoseq v2 "v2Folder/v2.mp4"
alternatives v3 {
    videoseq v31 "v3/seq1.mp4"
    videoseq v32 "v3/seq1.mp4"
    videoseq v33 "v3/seq1.mp4"
}

alternatives v4 {
    videoseq v41 "v4/seq1.mp4"
    videoseq v42 "v4/seq1.mp4"
}
mandatory videoseq v5 "https://www.youtube.com/watch?v=ezKx-S0LiNQ"
```



 FFmpeg

foo1.videogen

```
mandatory videoseq v1 "https://www.youtube.com/watch?v=PJNi1uYhV5w"
optional videoseq v2 "v2Folder/v2.mp4"
alternatives v3 {
    videoseq v31 "v3/seq1.mp4"
    videoseq v32 "v3/seq1.mp4"
    videoseq v33 "v3/seq1.mp4"
}

alternatives v4 {
    videoseq v41 "v4/seq1.mp4"
    videoseq v42 "v4/seq1.mp4"
}
mandatory videoseq v5 "https://www.youtube.com/watch?v=ezKx-S0LiNQ"
```

#1 How to design,  
create, and support  
dedicated languages  
(DSLs)?

#2 How to  
transform  
models/progra  
ms?



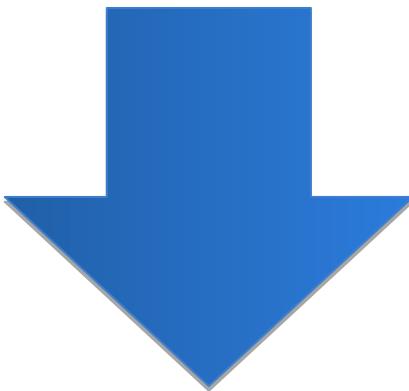
#3 How to manage  
variability/variants?

#4 How do  
frameworks  
internally work?

foo1.videogen

```
mandatory videoseq v1 "https://www.youtube.com/watch?v=PJNi1uYhV5w"
optional videoseq v2 "v2Folder/v2.mp4"
alternatives v3 {
    videoseq v31 "v3/seq1.mp4"
    videoseq v32 "v3/seq1.mp4"
    videoseq v33 "v3/seq1.mp4"
}

alternatives v4 {
    videoseq v41 "v4/seq1.mp4"
    videoseq v42 "v4/seq1.mp4"
}
mandatory videoseq v5 "https://www.youtube.com/watch?v=ezKx-S0LiNQ"
```



model-to-text

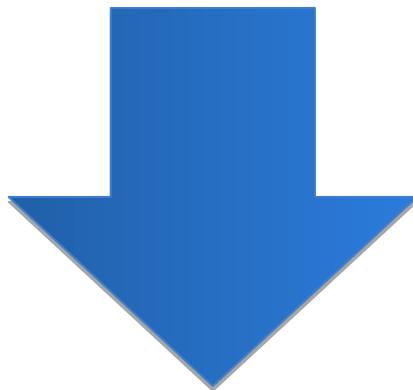
```
# this is a comment
file 'v3/seq1.avi'
file '/path/to/video2.avi'
file '/path/to/video3.avi'
```

 FFmpeg

### foo1.videogen

```
mandatory videoseq v1 "https://www.youtube.com/watch?v=PJNi1uYhV5w"
optional videoseq v2 "v2Folder/v2.mp4"
alternatives v3 {
    videoseq v31 "v3/seq1.mp4"
    videoseq v32 "v3/seq1.mp4"
    videoseq v33 "v3/seq1.mp4"
}

alternatives v4 {
    videoseq v41 "v4/seq1.mp4"
    videoseq v42 "v4/seq1.mp4"
}
mandatory videoseq v5 "https://www.youtube.com/watch?v=ezKx-S0LiNQ"
```



model-to-text

.m3u

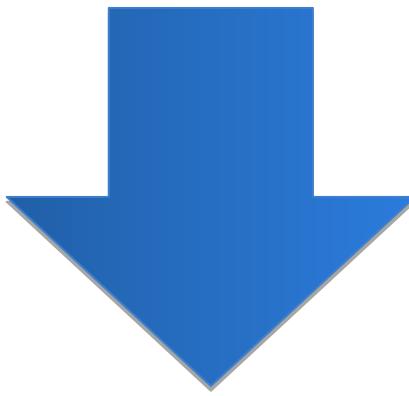
```
v3/seq1.avi
/path/to/video2.avi
/path/to/video3.avi
```



## foo1.videogen

```
mandatory videoseq v1 "https://www.youtube.com/watch?v=PJNi1uYhV5w"
optional videoseq v2 "v2Folder/v2.mp4"
alternatives v3 {
    videoseq v31 "v3/seq1.mp4"
    videoseq v32 "v3/seq1.mp4"
    videoseq v33 "v3/seq1.mp4"
}

alternatives v4 {
    videoseq v41 "v4/seq1.mp4"
    videoseq v42 "v4/seq1.mp4"
}
mandatory videoseq v5 "https://www.youtube.com/watch?v=ezKx-S0LiNQ"
```



model-to-text

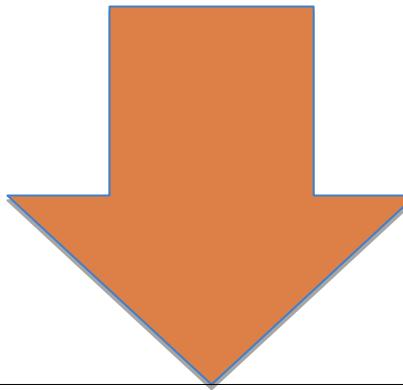
.m3u  
(extended)

```
#EXTM3U
#EXT-X-DISCONTINUITY
#EXTINF:3
resources/videos/vp0-logo/logo_start.ts
#EXT-X-DISCONTINUITY
#EXTINF:12
resources/videos/vp1-QR/QR05_1.ts
#EXT-X-DISCONTINUITY
#EXTINF:2
resources/videos/vp2-intro-fluide-glacial/
EtPendantCeTempsLaEn1975_processed.ts
```

flowplayer flash

```
foo1.videogen &gt;
mandatory videoseq v1 "https://www.youtube.com/watch?v=PjNi1uYhV5w"
optional videoseq v2 "v2folder/v2.mp4"
alternatives v3 {
    videoseq v31 "v3/seq1.mp4"
    videoseq v32 "v3/seq1.mp4"
    videoseq v33 "v3/seq1.mp4"
}

alternatives v4 {
    videoseq v41 "v4/seq1.mp4"
    videoseq v42 "v4/seq1.mp4"
}
mandatory videoseq v5 "https://www.youtube.com/watch?v=ezKx-S0LiNQ"
```

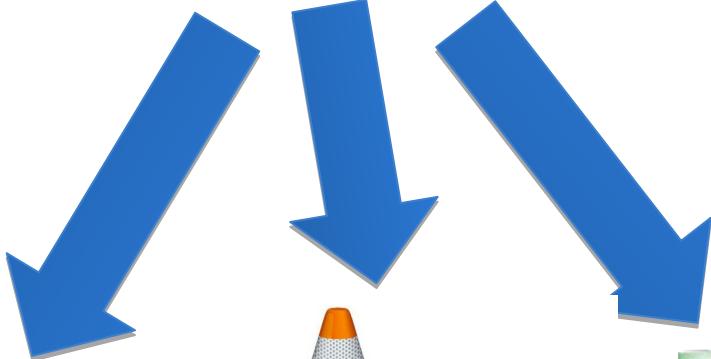


**model-to-model**

**playlist  
metamodel**

**playlist model**

**model-to-text**



flowplayer **flash**

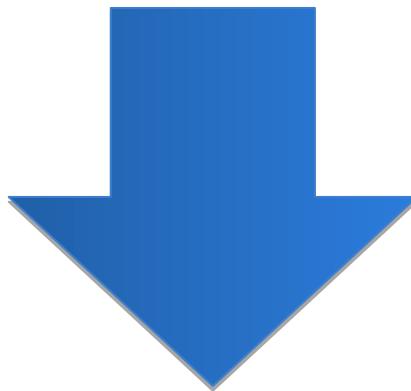


**FFmpeg**

## foo1.videogen

```
mandatory videoseq v1 "https://www.youtube.com/watch?v=PJNi1uYhV5w"
optional videoseq v2 "v2Folder/v2.mp4"
alternatives v3 {
    videoseq v31 "v3/seq1.mp4"
    videoseq v32 "v3/seq1.mp4"
    videoseq v33 "v3/seq1.mp4"
}

alternatives v4 {
    videoseq v41 "v4/seq1.mp4"
    videoseq v42 "v4/seq1.mp4"
}
mandatory videoseq v5 "https://www.youtube.com/watch?v=ezKx-S0LiNQ"
```



model-to-\*



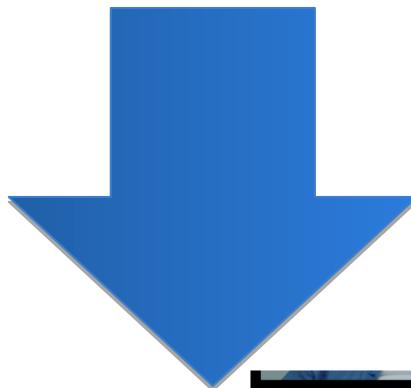
Thumbnails  
(vignettes) of  
each video  
sequence  
(e.g., PGN  
format)



## foo1.videogen

```
mandatory videoseq v1 "https://www.youtube.com/watch?v=PJNi1uYhV5w"
optional videoseq v2 "v2Folder/v2.mp4"
alternatives v3 {
    videoseq v31 "v3/seq1.mp4"
    videoseq v32 "v3/seq1.mp4"
    videoseq v33 "v3/seq1.mp4"
}

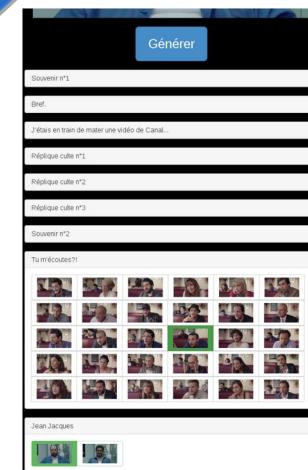
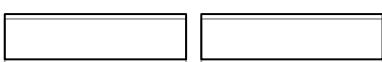
alternatives v4 {
    videoseq v41 "v4/seq1.mp4"
    videoseq v42 "v4/seq1.mp4"
}
mandatory videoseq v5 "https://www.youtube.com/watch?v=ezKx-S0LiNQ"
```



model-to-\*



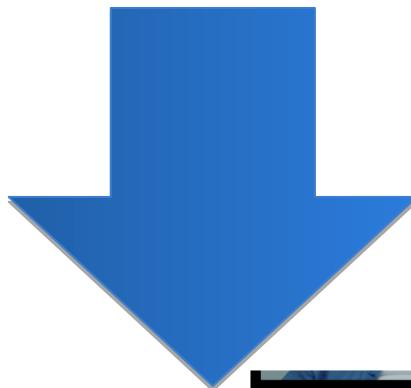
Thumbnails (vignettes) of each video sequence (e.g., PGN format)



## foo1.videogen

```
mandatory videoseq v1 "https://www.youtube.com/watch?v=PJNi1uYhV5w"
optional videoseq v2 "v2Folder/v2.mp4"
alternatives v3 {
    videoseq v31 "v3/seq1.mp4"
    videoseq v32 "v3/seq1.mp4"
    videoseq v33 "v3/seq1.mp4"
}

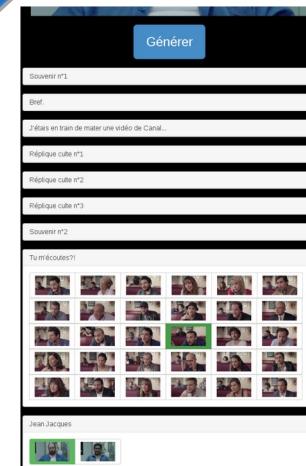
alternatives v4 {
    videoseq v41 "v4/seq1.mp4"
    videoseq v42 "v4/seq1.mp4"
}
mandatory videoseq v5 "https://www.youtube.com/watch?v=ezKx-S0LiNQ"
```



model-to-\*

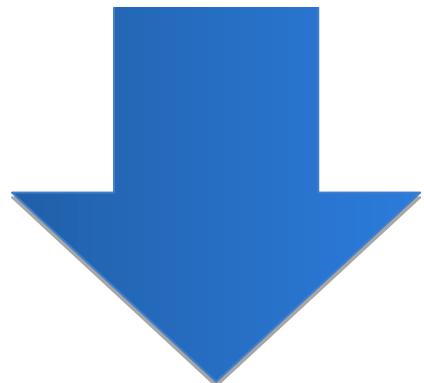


Thumbnails (vignettes) of each video sequence (e.g., PGN format)



foo1.videogen

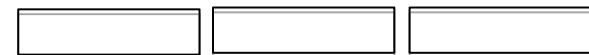
```
VideoGen {  
    mandatory videoseq v1 "V1/v1.mp4"  
    optional videoseq v2 "v2folder/v2.mp4" {  
        probability 25  
    }  
    alternatives v3 {  
        videoseq v31 "v3/seq1.mp4" {  
            duration 12  
            probability 25  
            description "a"  
        }  
        videoseq v31 "v3/seq2.mp4"  
        videoseq v32 "v3/seq3.mp4"  
    }  
    alternatives v4 {  
        videoseq v41 "v4/seq1.mp4"  
        videoseq v42 "v4/seq2.mp4"  
    }  
    mandatory videoseq v5 "v5.mp4"  
  
    optional videoseq v8 "v8.avi"  
    alternatives v9 {  
        videoseq v81 "V81.avi"  
    }  
}
```



## foo1.videogen

```
mandatory videoseq v1 "https://www.youtube.com/watch?v=PJNi1uYhV5w"
optional videoseq v2 "v2Folder/v2.mp4"
alternatives v3 {
    videoseq v31 "v3/seq1.mp4"
    videoseq v32 "v3/seq1.mp4"
    videoseq v33 "v3/seq1.mp4"
}

alternatives v4 {
    videoseq v41 "v4/seq1.mp4"
    videoseq v42 "v4/seq1.mp4"
}
mandatory videoseq v5 "https://www.youtube.com/watch?v=ezKx-S0LiNQ"
```



18, 167, 899



## foo1.videogen

```
mandatory videoseq v1 "https://www.youtube.com/watch?v=PJNi1uYhV5w"
optional videoseq v2 "v2folder/v2.mp4"
alternatives v3 {
    videoseq v31 "v3/seq1.mp4"
    videoseq v32 "v3/seq1.mp4"
    videoseq v33 "v3/seq1.mp4"
}

alternatives v4 {
    videoseq v41 "v4/seq1.mp4"
    videoseq v42 "v4/seq1.mp4"
}
mandatory videoseq v5 "https://www.youtube.com/watch?v=ezKx-S0LiNQ"
```



### Website/online

- Random generation
- Configurator
- Game
- ...

## foo1.videogen

```
mandatory videoseq v1 "https://www.youtube.com/watch?v=PJNi1uYhV5w"
optional videoseq v2 "v2folder/v2.mp4"
@alternatives v3 {
    videoseq v31 "v3/seq1.mp4"
    videoseq v32 "v3/seq1.mp4"
    videoseq v33 "v3/seq1.mp4"
}

@alternatives v4 {
    videoseq v41 "v4/seq1.mp4"
    videoseq v42 "v4/seq1.mp4"
}
mandatory videoseq v5 "https://www.youtube.com/watch?v=ezKx-S0LiNQ"
```

**Feature model:** another model for modeling “features” of your Web site (eg ability to save the video; mode=generation with frequencies)

### Website/online

- Random generation
- Configurator
- Game
- ...

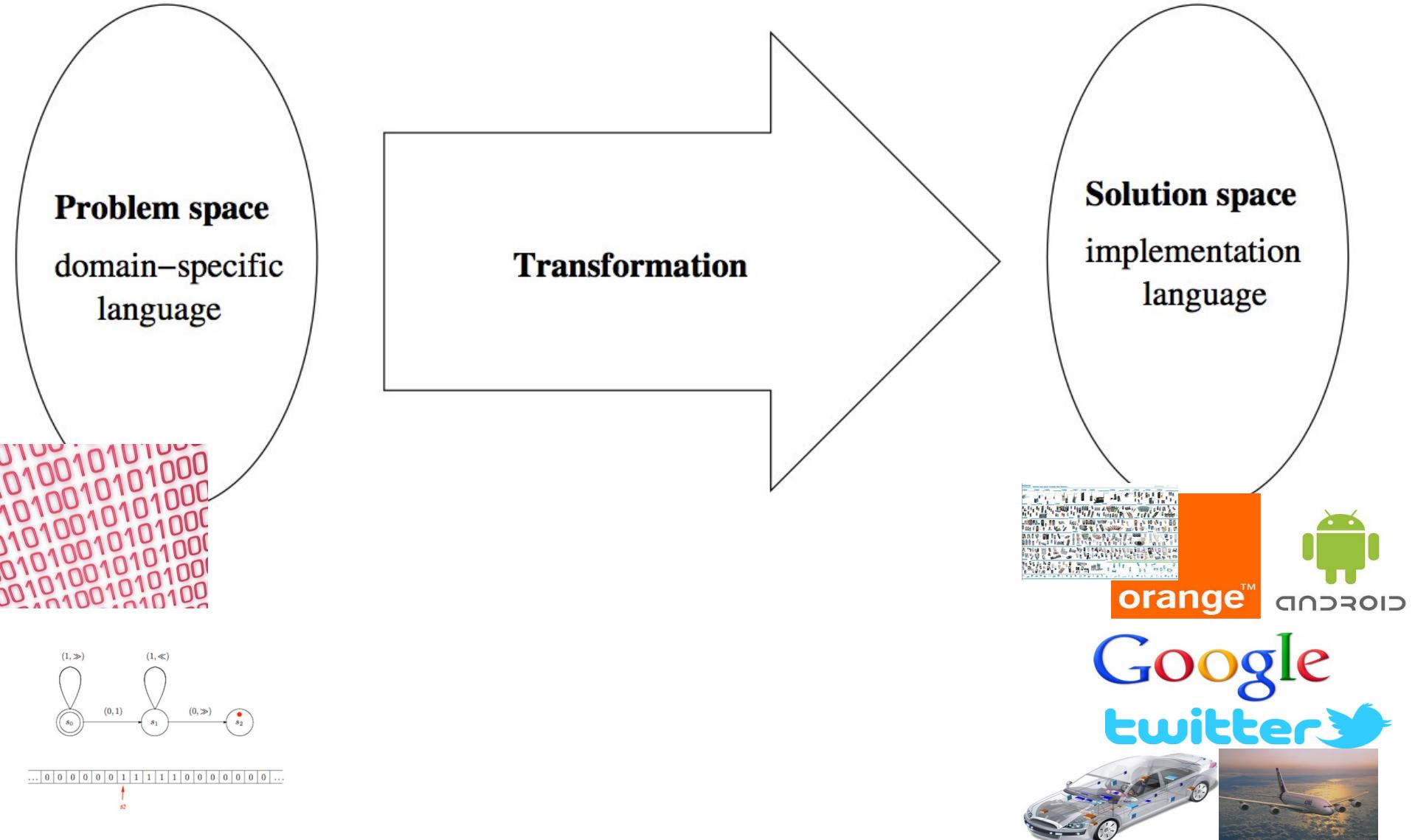


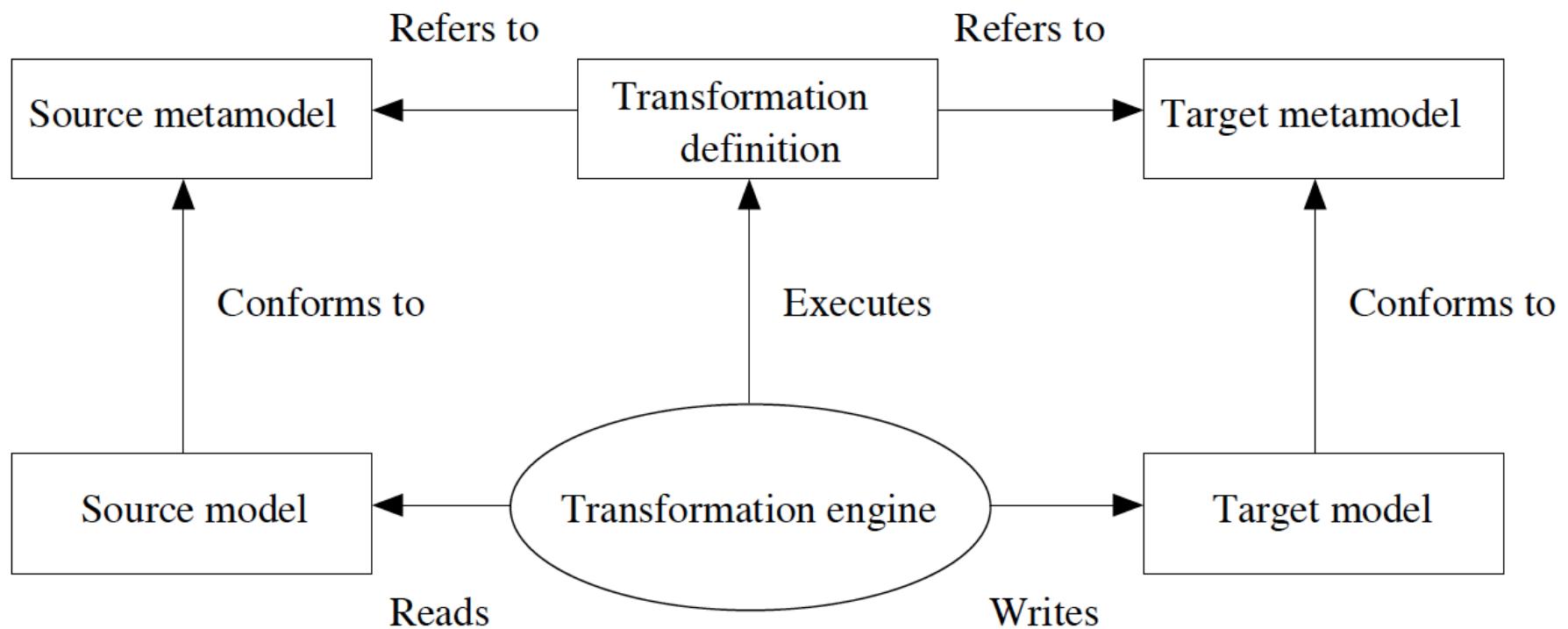
# *Model Transformations*

*Taxonomy + Examples*

# Abstraction Gap

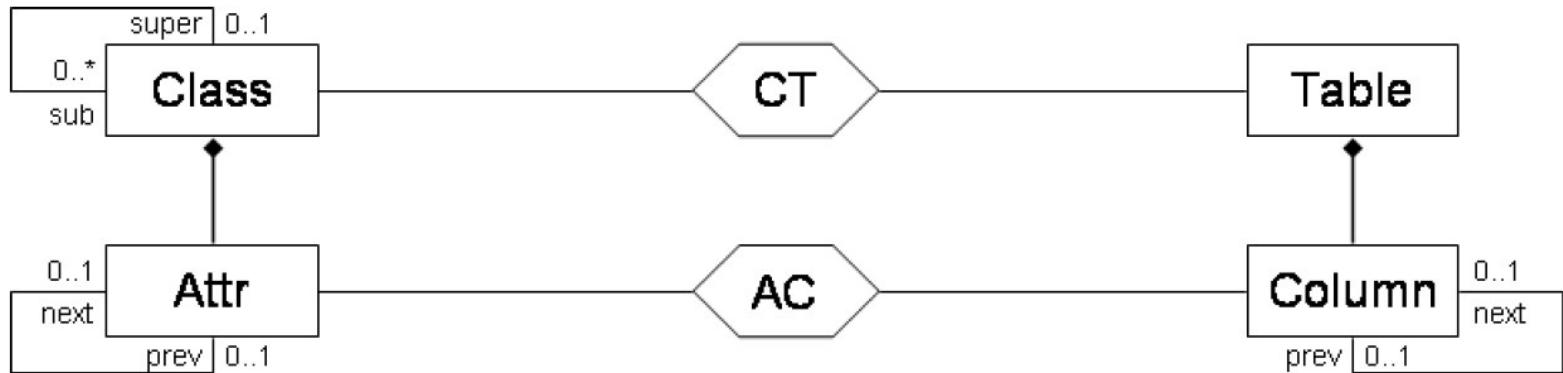
## Transformation is the key



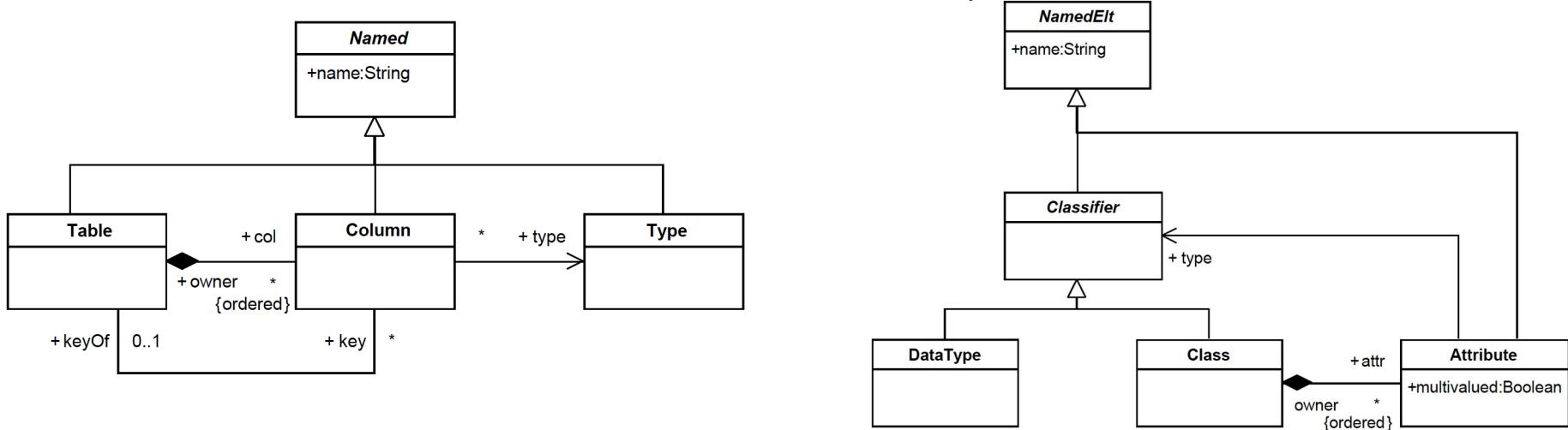


# Andy Schürr, Felix Klar “15 Years of Triple Graph Grammars.” ICGT 2008

(declarative; bi-directionnal; model-to-model)



# ATL (<http://www.eclipse.org/atl/atlTransformations/>)



```
rule Class2Table {
    from           -- source pattern
        c : Class!Class
    to             -- target pattern
        t : Relational!Table
    }
```

# Feature-Based Survey of Model Transformation Approaches

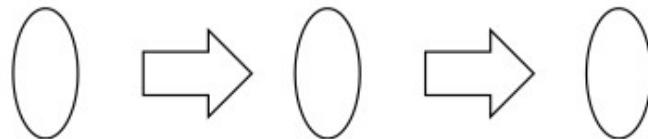
Krzysztof Czarnecki  
University of Waterloo  
Waterloo, Canada

Simon Helsen  
SAP AG  
Walldorf, Germany

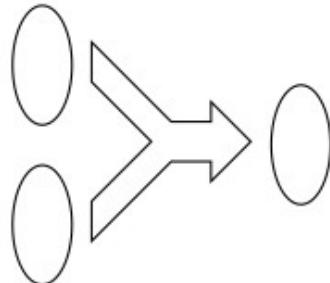
March 15, 2006

# Overview of Generative Software Development

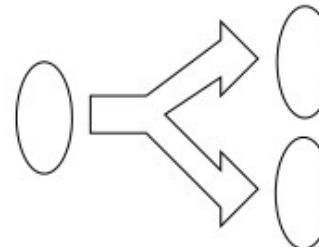
Krzysztof Czarnecki



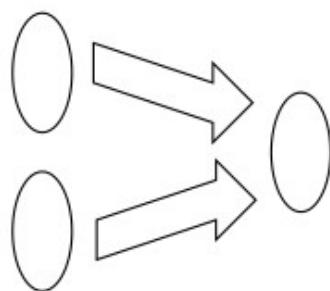
*a. Chaining of mappings*



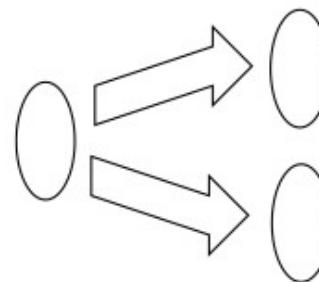
*b. Multiple problem spaces*



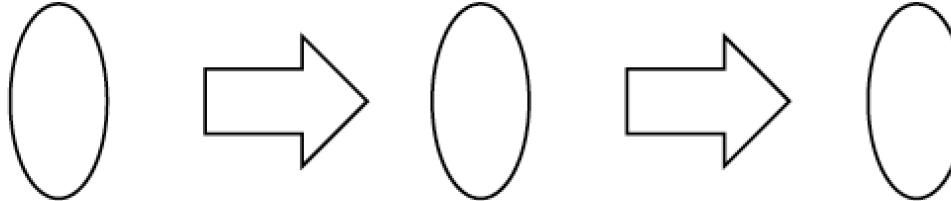
*c. Multiple solution spaces*



*d. Alternative problem spaces e. Alternative solution spaces*



# One step/stage transformation hardly the case



a. Chaining of mappings



```
foo1.videogen <input>
mandatory videoseq v1 "https://www.youtube.com/watch?v=PjNi1uYhV5w"
optional videoseq v2 "v2Folder/v2.mp4"
alternatives v3 {
    videoseq v31 "v3/seq1.mp4"
    videoseq v32 "v3/seq1.mp4"
    videoseq v33 "v3/seq1.mp4"
}

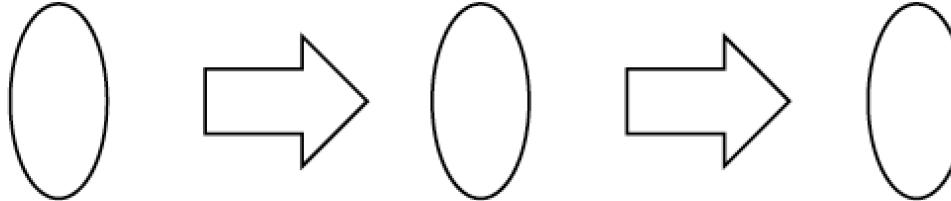
alternatives v4 {
    videoseq v41 "v4/seq1.mp4"
    videoseq v42 "v4/seq1.mp4"
}
mandatory videoseq v5 "https://www.youtube.com/watch?v=ezKx-S0LiNQ"
```



```
#EXTM3U
#EXT-X-DISCONTINUITY
#EXTINF:3
resources/videos/vp0-logo/logo_start.ts
#EXT-X-DISCONTINUITY
#EXTINF:12
resources/videos/vp1-QR/QR05_1.ts
#EXT-X-DISCONTINUITY
#EXTINF:2
resources/videos/vp2-intro-fluide-glacial/
EtPendantCeTempsLaEn1975_processed.ts
```

.m3u (extended)

# One step/stage transformation hardly the case



a. Chaining of mappings



```
foo1.videogen <>
mandatory videoseq v1 "https://www.youtube.com/watch?v=PjNi1uYhV5w"
optional videoseq v2 "v2Folder/v2.mp4"
@alternatives v3 {
    videoseq v31 "v3/seq1.mp4"
    videoseq v32 "v3/seq1.mp4"
    videoseq v33 "v3/seq1.mp4"
}

@alternatives v4 {
    videoseq v41 "v4/seq1.mp4"
    videoseq v42 "v4/seq1.mp4"
}
mandatory videoseq v5 "https://www.youtube.com/watch?v=ezKx-S0LiNQ"
```



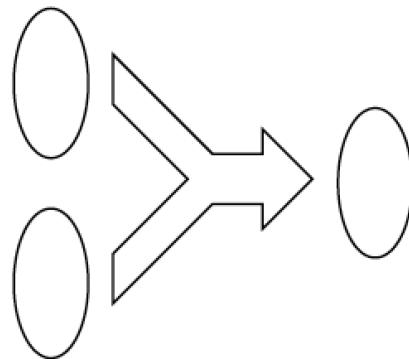
Playlist  
model

```
#EXTM3U
#EXT-X-DISCONTINUITY
#EXTINF:3
resources/videos/vp0-logo/logo_start.ts
#EXT-X-DISCONTINUITY
#EXTINF:12
resources/videos/vp1-QR/QR05_1.ts
#EXT-X-DISCONTINUITY
#EXTINF:2
resources/videos/vp2-intro-fluide-glacial/
EtPendantCeTempsLaEn1975_processed.ts
```

.m3u (extended)

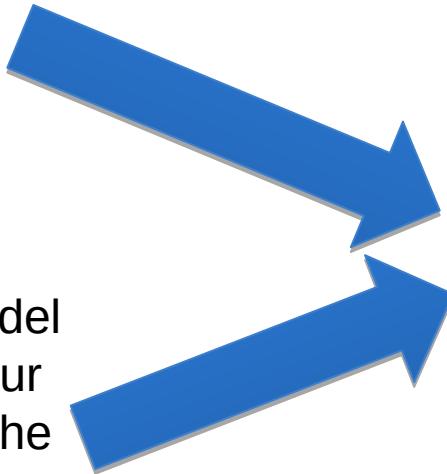
# Problem space

## Combination of expertises/aspects/DSLs



b. *Multiple problem spaces*

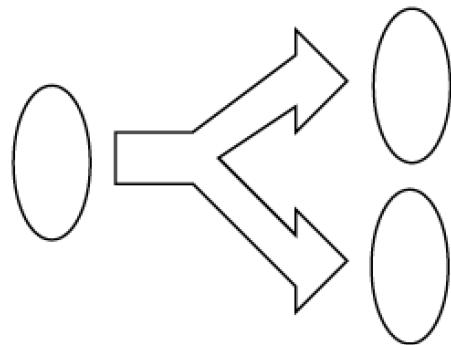
```
foo1.videogen ✘
mandatory videoseq v1 "https://www.youtube.com/watch?v=PJNi1uYhV5w"
optional videoseq v2 "v2folder/v2.mp4"
alternatives v3 {
    videoseq v31 "v3/seq1.mp4"
    videoseq v32 "v3/seq1.mp4"
    videoseq v33 "v3/seq1.mp4"
}
alternatives v4 {
    videoseq v41 "v4/seq1.mp4"
    videoseq v42 "v4/seq1.mp4"
}
mandatory videoseq v5 "https://www.youtube.com/watch?v=ezKx-S0LiNQ"
```



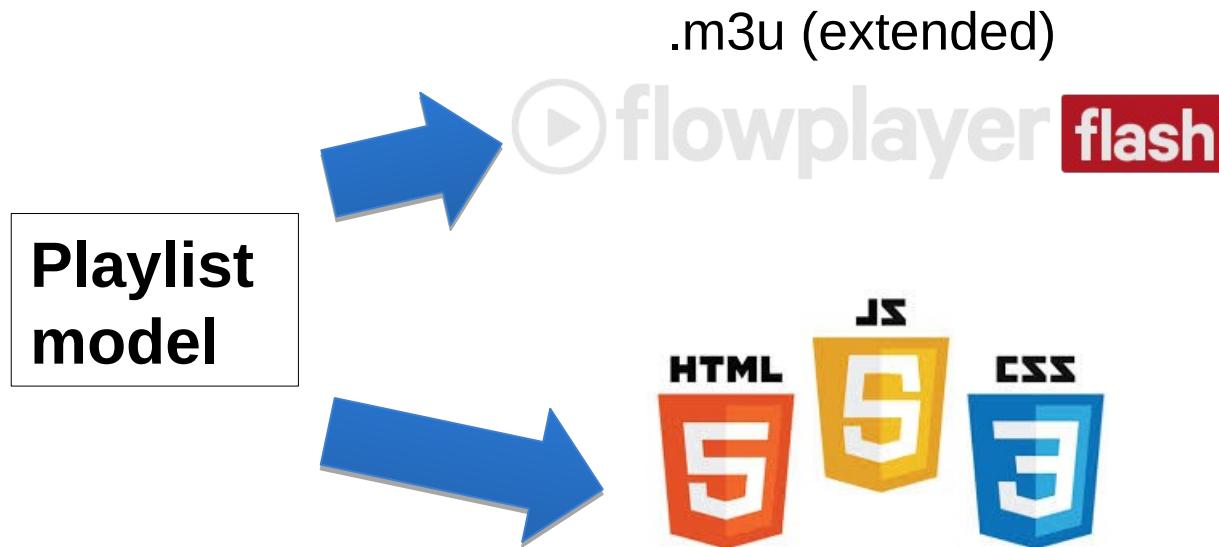
**Feature model:** another model for modeling “features” of your Web site (eg ability to save the video; mode=generation with frequencies)

# Solution space

## Different targets (e.g., technological platforms)

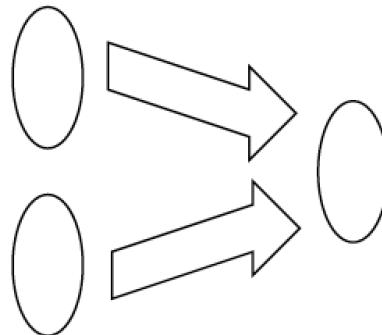


c. *Multiple solution spaces*



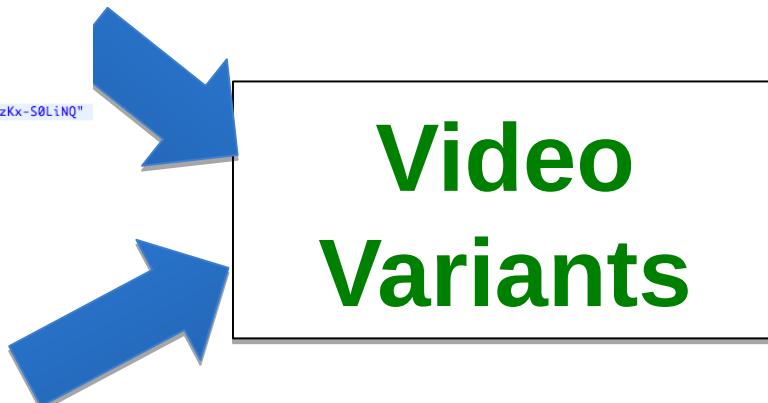
# Problem space

e.g., different concrete syntaxes



```
foo1.videogen ✘
mandatory videoseq v1 "https://www.youtube.com/watch?v=PJNi1uYhV5w"
optional videoseq v2 "v2folder/v2.mp4"
alternatives v3 {
    videoseq v31 "v3/seq1.mp4"
    videoseq v32 "v3/seq1.mp4"
    videoseq v33 "v3/seq1.mp4"
}
alternatives v4 {
    videoseq v41 "v4/seq1.mp4"
    videoseq v42 "v4/seq1.mp4"
}
mandatory videoseq v5 "https://www.youtube.com/watch?v=ezKx-S0LiNQ"
```

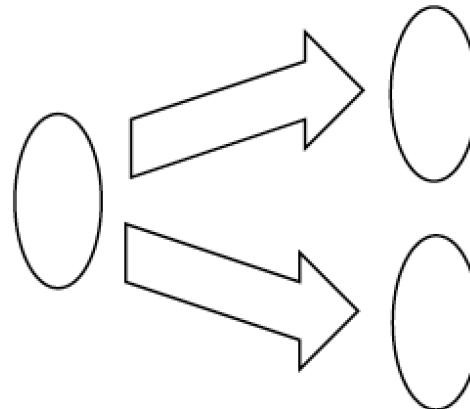
d. Alternative problem spaces



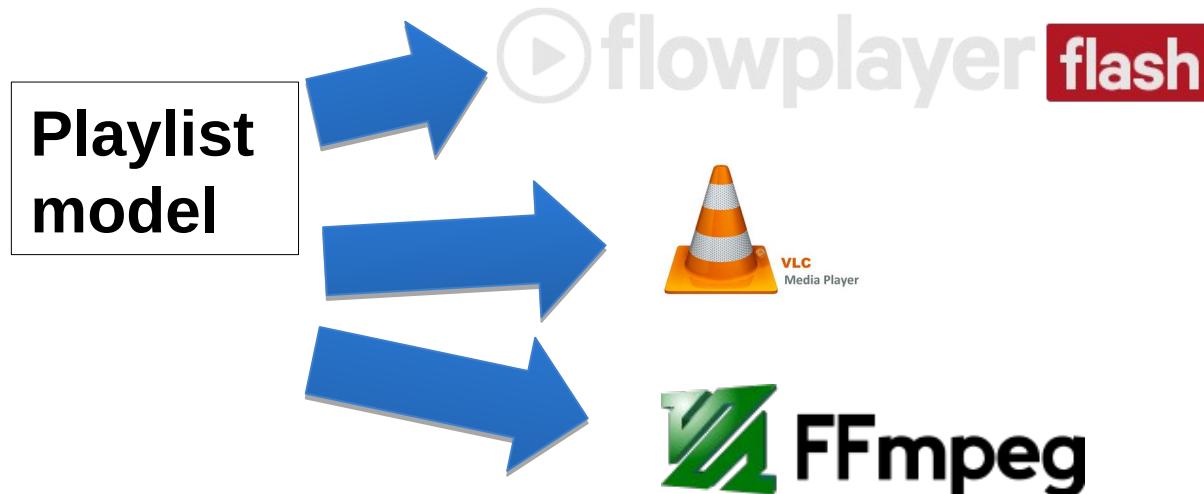
Feature Model  
(see next courses)

# Solution space

## Different targets (e.g., technological platforms)



e. Alternative solution spaces



# Model Transformation: Taxonomy

	<p>PIM → PIM PSM → PSM</p> <p><i>Horizontale</i> <i>Verticale</i></p>	
<p>Transformation endogène</p>	<p>Restructuration Normalisation intégration de patrons</p>	<p>Raffinement</p>
<p>Transformation exogène</p>	<p>Migration de logiciel Fusion de modèles</p>	<p>PIM vers PSM Rétro-conception</p>

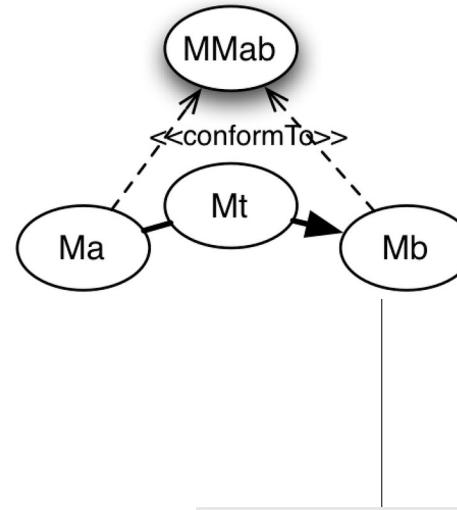
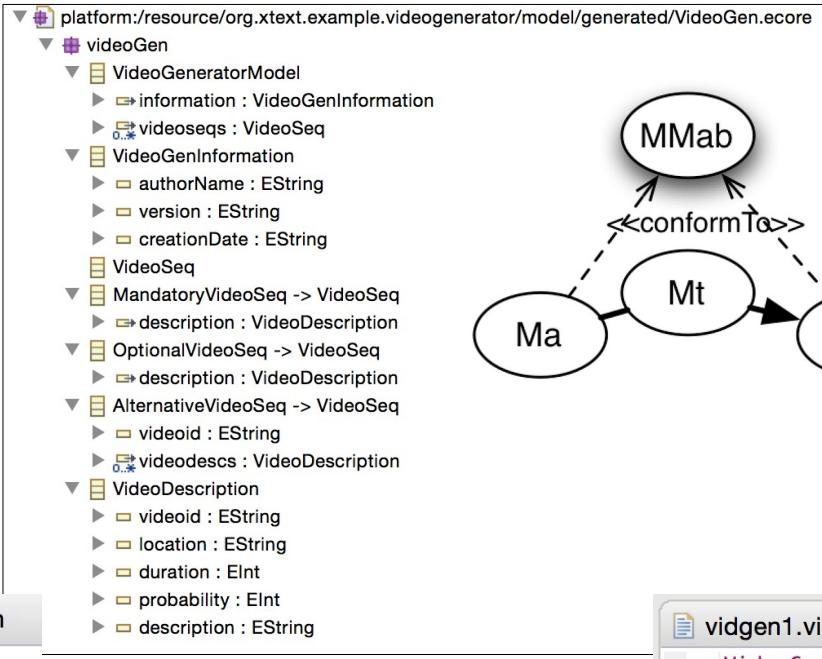
# Endogenous Transformation

```
vidgen1.videogen  vidgen1-bis.videogen
VideoGen {

    mandatory videoseq "V1/v1.mp4"
    optional videoseq "v2folder/v2.mp4" {
        probability 25
    }
    alternatives vid3 {
        videoseq "v3/seq1.mp4"
        videoseq vid31 "v3/seq2.mp4"
        videoseq vid32 "v3/seq3.mp4"
    }

    alternatives vid4 {
        videoseq vid41 "v4/seq1.mp4"
        videoseq vid42 "v4/seq2.mp4"
    }
    mandatory videoseq vid5 "v5.mp4"

    optional videoseq vid8 "v8.avi"
    alternatives vid9 {
        videoseq vid81 "V81.avi"
    }
}
```



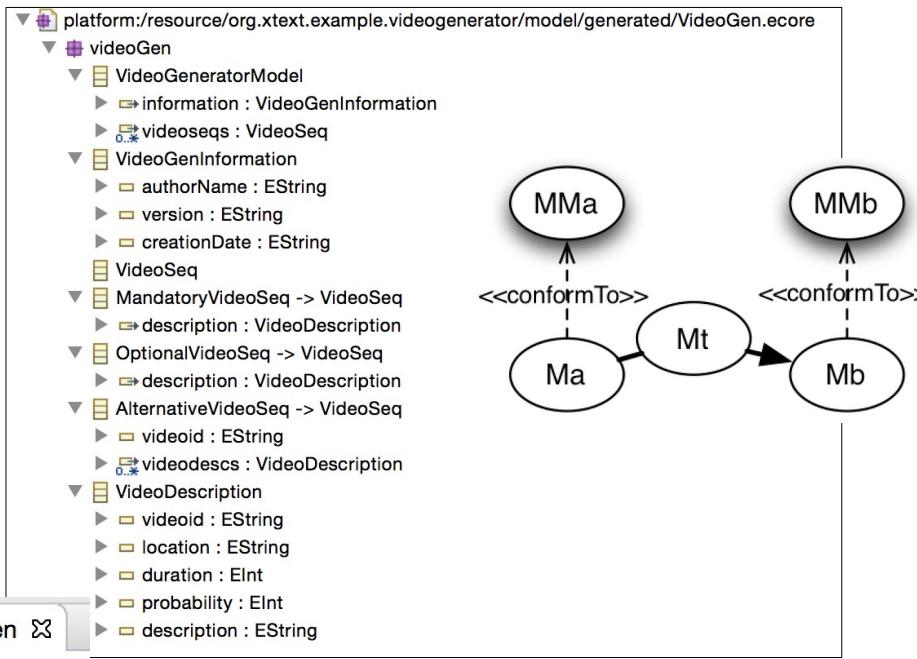
```
vidgen1.videogen  vidgen1-bis.videogen
VideoGen {
    VideoGen1/vidgen1-bis.videogen
    mandatory videoseq v0 "v1/v1.mp4"
    optional videoseq v1 "v2folder/v2.mp4" {
        probability 25
    }
    alternatives vid3 {
        videoseq v2 "v3/seq1.mp4"
        videoseq vid31 "v3/seq2.mp4"
        videoseq vid32 "v3/seq3.mp4"
    }

    alternatives vid4 {
        videoseq vid41 "v4/seq1.mp4"
        videoseq vid42 "v4/seq2.mp4"
    }
    mandatory videoseq vid5 "v5.mp4"

    optional videoseq vid8 "v8.avi"
    alternatives vid9 {
        videoseq vid81 "V81.avi"
    }
}
```

# Exogeneous Transformation

(metamodel)



vidgen1.videoegen vidgen1-bis.videoegen

```
VideoGen {
    VideoGen1/vidgen1-bis.videoegen
        mandatory videoseq v0 "v1/v1.mp4"
        optional videoseq v1 "v2folder/v2.mp4" {
            probability 25
        }
        alternatives vid3 {
            videoseq v2 "v3/seq1.mp4"
            videoseq vid31 "v3/seq2.mp4"
            videoseq vid32 "v3/seq3.mp4"
        }

        alternatives vid4 {
            videoseq vid41 "v4/seq1.mp4"
            videoseq vid42 "v4/seq2.mp4"
        }
        mandatory videoseq vid5 "v5.mp4"

        optional videoseq vid8 "v8.avi"
        alternatives vid9 {
            videoseq vid81 "V81.avi"
        }
}
```

```
<ul>
<li>v0</li>
<li>v1</li>
<li>vid3</li>
<ul>
<li>v2</li>
<li>vid31</li>
<li>vid32</li>
</ul>
<li>vid4</li>
<ul>
<li>vid41</li>
<li>vid42</li>
</ul>
<li>vid5</li>
<li>vid8</li>
<li>vid9</li>
<ul>
<li>vid81</li>
</ul>
</ul>
```

# Vertical Transformation

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

The screenshot shows the Xtext IDE interface with two code editors and a navigation view.

**Navigation View:**

- platform:/resource/org.xtext.example.videogenerator/model/generated/VideoGen.ecore
- videoGen
- VideoGeneratorModel
  - information : VideoGenInformation
  - videoseqs : VideoSeq
- VideoGenInformation
  - authorName : EString
  - version : EString
  - creationDate : EString
- VideoSeq
- MandatoryVideoSeq -> VideoSeq
  - description : VideoDescription
- OptionalVideoSeq -> VideoSeq
  - description : VideoDescription
- AlternativeVideoSeq -> VideoSeq
  - videoid : EString
  - videodescs : VideoDescription
- VideoDescription
  - videoid : EString
  - location : EString
  - duration : EInt
  - probability : EInt
  - description : EString

**Code Editors:**

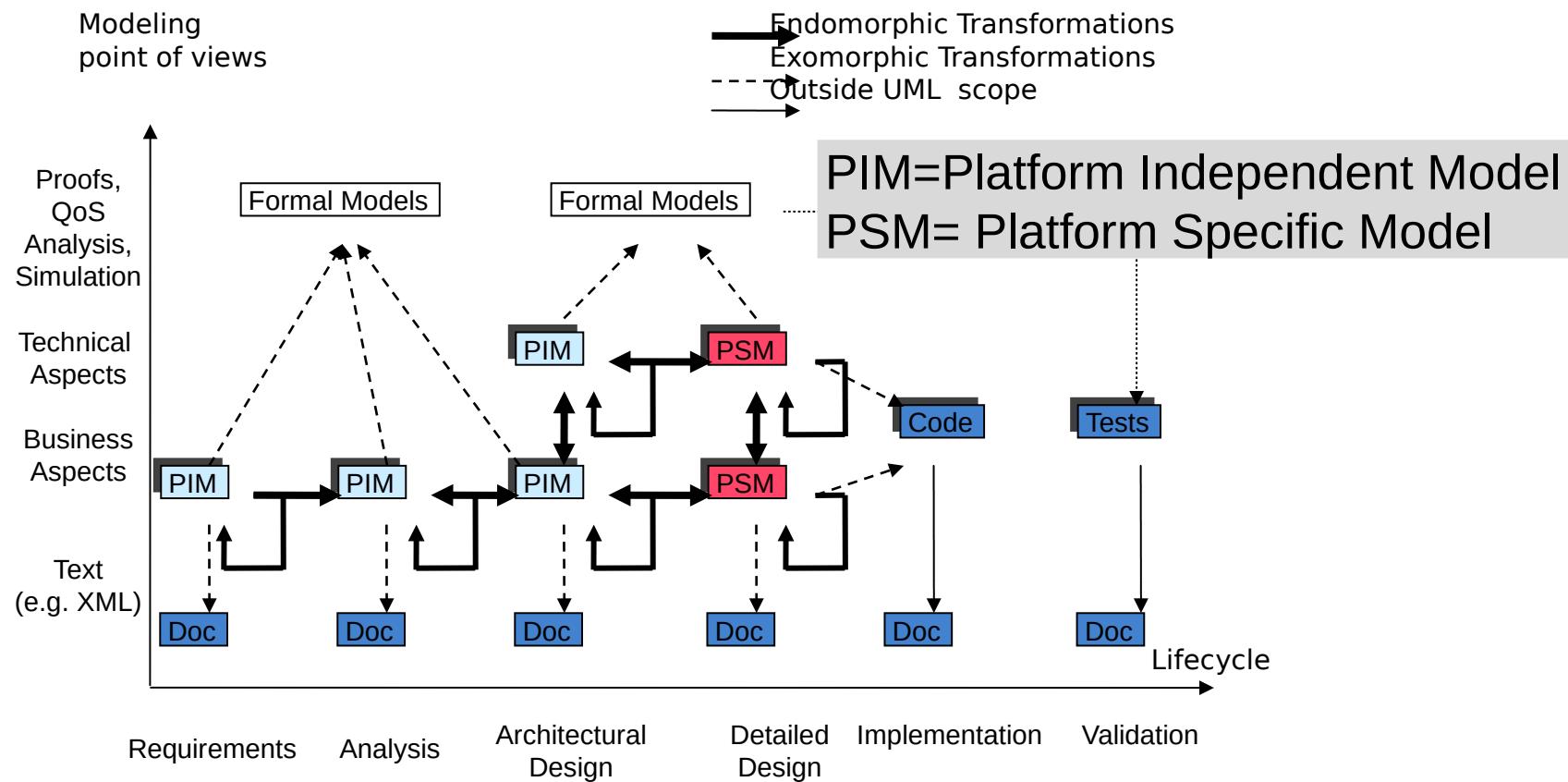
- vidgen1.videogen**:

```
VideoGen {  
    mandatory videoseq "V1/v1.mp4"  
    optional videoseq "v2folder/v2.mp4" {  
        probability 25  
    }  
    alternatives vid3 {  
        videoseq "v3/seq1.mp4"  
        videoseq vid31 "v3/seq2.mp4"  
        videoseq vid32 "v3/seq3.mp4"  
    }  
    alternatives vid4 {  
        videoseq vid41 "v4/seq1.mp4"  
        videoseq vid42 "v4/seq2.mp4"  
    }  
    mandatory videoseq vid5 "v5.mp4"  
  
    optional videoseq vid8 "v8.avi"  
    alternatives vid9 {  
        videoseq vid81 "V81.avi"  
    }  
}
```
- vidgen1-bis.videogen**:

```
VideoGen [  
VideoGen1/vidgen1-bis.videogen  
    mandatory videoseq v0 "v1/v1.mp4"  
    optional videoseq v1 "v2folder/v2.mp4" {  
        probability 25  
    }  
    alternatives vid3 {  
        videoseq v2 "v3/seq1.mp4"  
        videoseq vid31 "v3/seq2.mp4"  
        videoseq vid32 "v3/seq3.mp4"  
    }  
    alternatives vid4 {  
        videoseq vid41 "v4/seq1.mp4"  
        videoseq vid42 "v4/seq2.mp4"  
    }  
    mandatory videoseq vid5 "v5.mp4"  
  
    optional videoseq vid8 "v8.avi"  
    alternatives vid9 {  
        videoseq vid81 "V81.avi"  
    }  
}
```

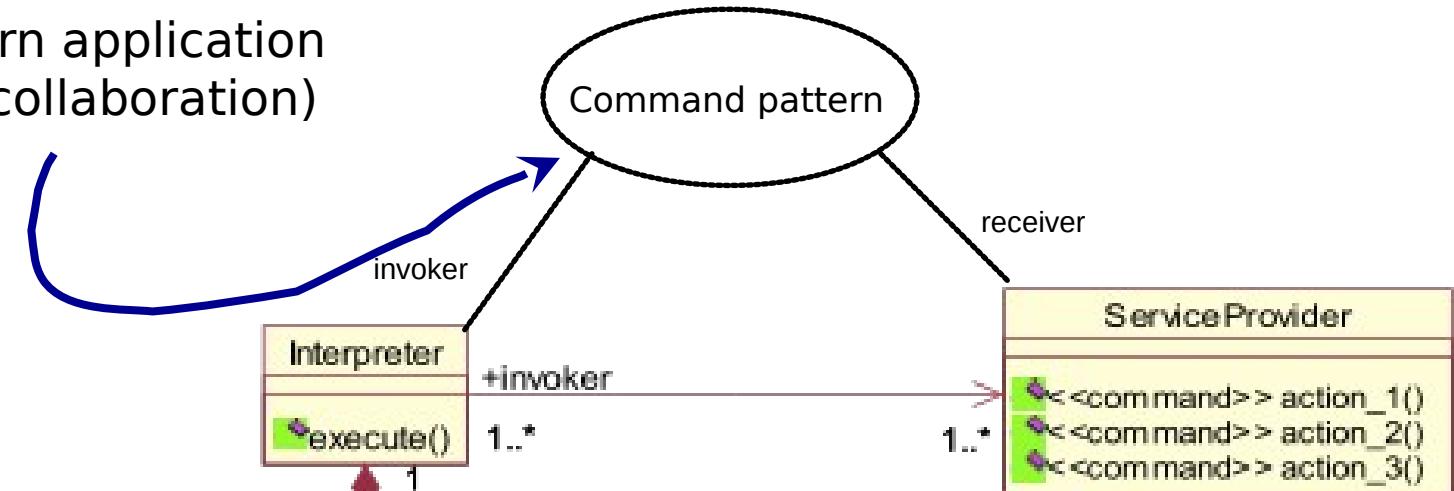
# Chaining of Transformations

Back to the first courses



# Embedding implicit semantics into a model

Design pattern application  
(parametric collaboration)

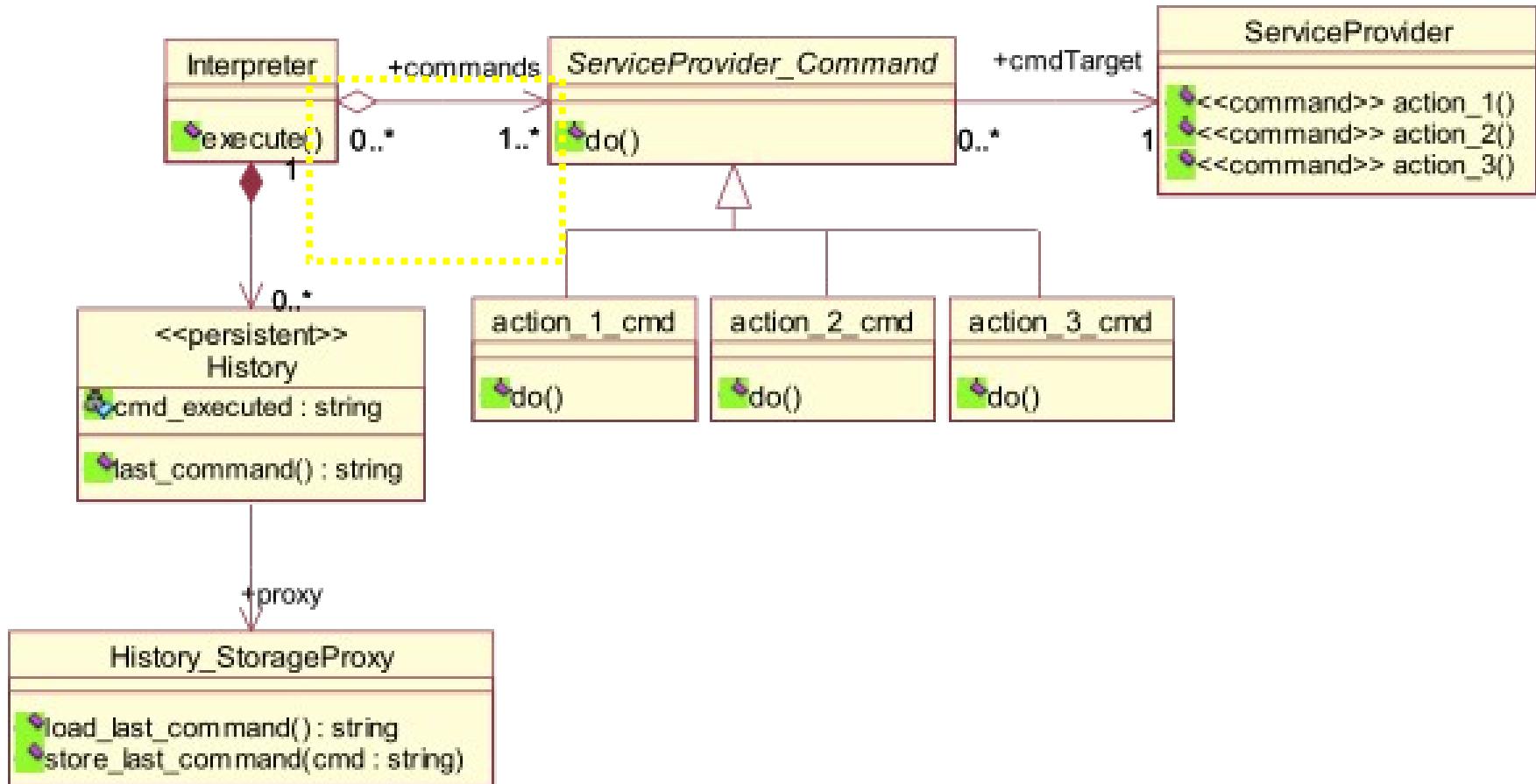


Element stereotype

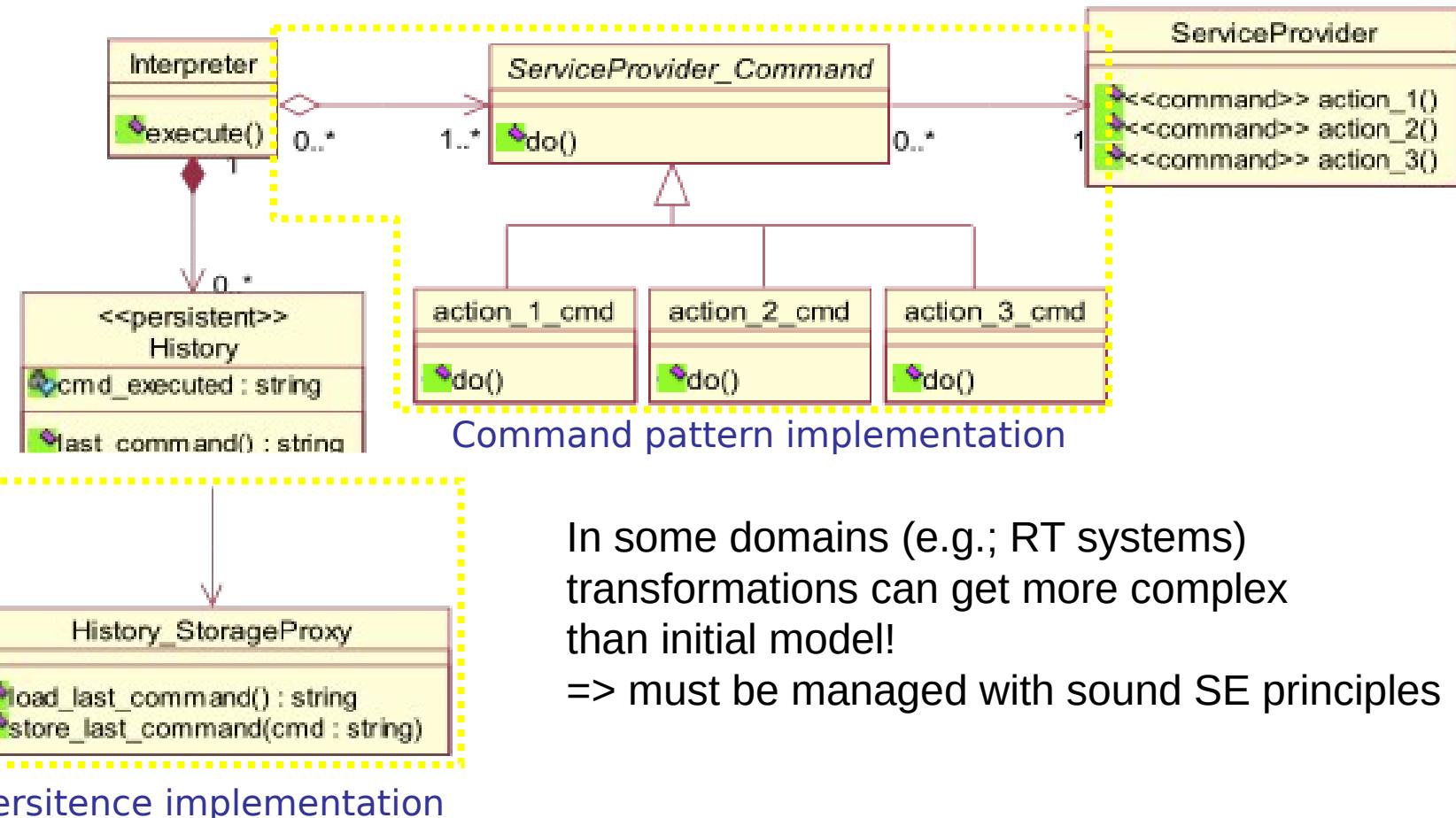


...and also  
Tagged values  
& Contracts

# ...and the result we want...



# How To: Automatic Model Transformations



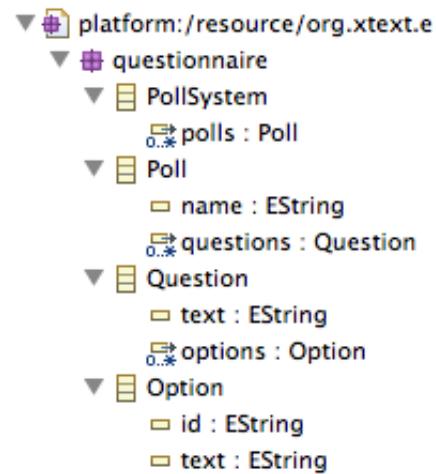
# Quizz Time

Characterize the following model transformations

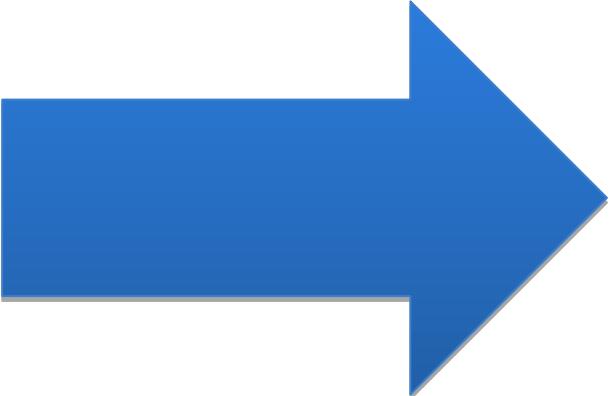
Endogeneous? Exogeneous?

Vertical? Horizontal?

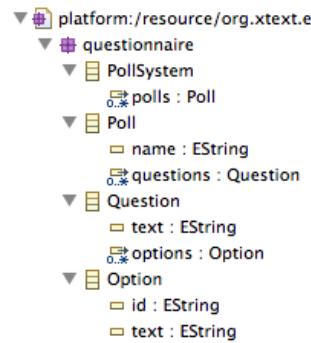
Model-to-text? Model-to-Model?



```
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"
        }
    }
}
```



```
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"
        }
    }
}
```



# HTML



foo.q

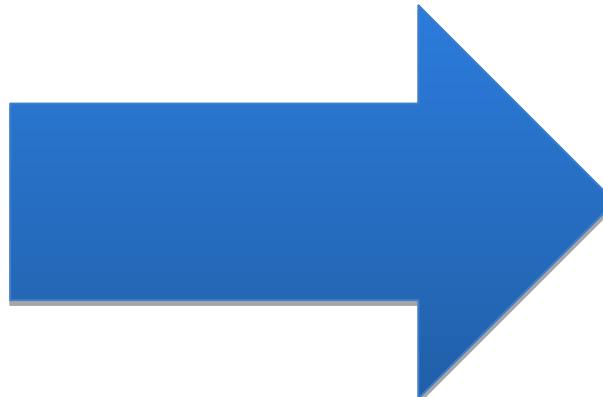
```

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

# *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 loadVideoGenerator(URI uri) {
    new VideoGenStandaloneSetupGenerated().createInjectorAndDoEMFRegistration()
    var res = new ResourceSetImpl().getResource(uri, true);
    res.contents.get(0) as VideoGeneratorModel
}

```

```

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

```

```

@Test
def test1() {
    // loading
    var videoGen = loadVideoGenerator(URI.createURI("foo2.videogen"))
    assertNotNull(videoGen)
    assertEquals(3, videoGen.videoseqs.size)
    // MODEL MANAGEMENT (ANALYSIS, TRANSFORMATION)
    videoGen.videoseqs.forEach[videoseq | 
        if (videoseq instanceof MandatoryVideoSeq) {
            val desc = (videoseq as MandatoryVideoSeq).description
            if(desc.videoid.isNullOrEmpty) desc.videoid = genID()
        }
        else if (videoseq instanceof OptionalVideoSeq) {
            val desc = (videoseq as OptionalVideoSeq).description
            if(desc.videoid.isNullOrEmpty) desc.videoid = genID()
        }
        else {
            val altvid = (videoseq as AlternativeVideoSeq)
            if(altvid.videoid.isNullOrEmpty) altvid.videoid = genID()
            for (vdesc : altvid.videodescs) {
                if(vdesc.videoid.isNullOrEmpty) vdesc.videoid = genID()
            }
        }
    ]
    // serializing
    saveVideoGenerator(URI.createURI("foo2bis.xmi"), videoGen)
    saveVideoGenerator(URI.createURI("foo2bis.videogen"), videoGen)
}

```

```

foo2.videogen
VideoGen {
    mandatory videoseq v1 "V1/v1.mp4"
    optional videoseq v2 "v2folder/v2.mp4"
    alternatives v3 {
        videoseq v31 "v31.mp4"
        videoseq v32 "v32.mp4"
    }
}

```

```

platform:/resource/org.xtext.example.videogenerator/model
videoGen
    VideoGeneratorModel
        information : VideoGenInformation
        videoseqs : VideoSeq
    VideoGenInformation
        authorName : EString
        version : EString
        creationDate : EString
        VideoSeq
    MandatoryVideoSeq -> VideoSeq
        description : VideoDescription
    OptionalVideoSeq -> VideoSeq
        description : VideoDescription
    AlternativeVideoSeq -> VideoSeq
        videoid : EString
        videodescs : VideoDescription
    VideoDescription
        videoid : EString
        location : EString
        duration : EInt
        probability : EInt
        description : EString

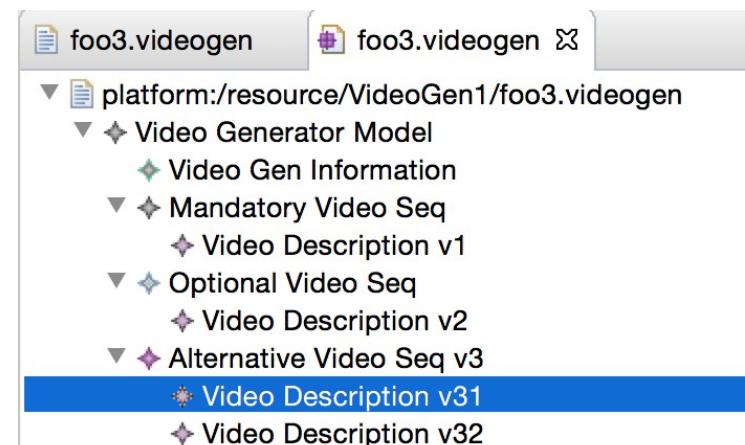
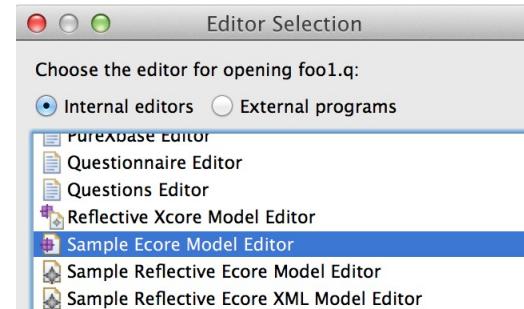
```

# Loading Models (1)



foo3.videogen

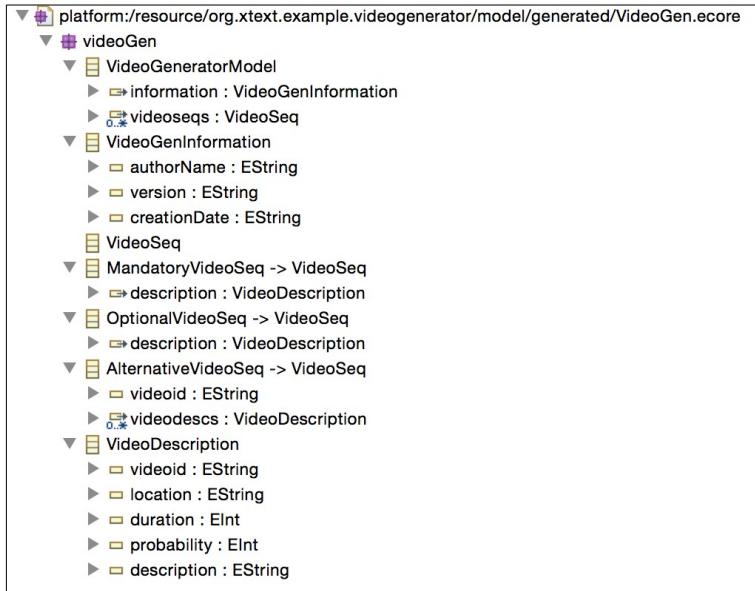
```
VideoGen {  
    mandatory videoseq v1 "V1/v1.mp4"  
    optional videoseq v2 "v2folder/v2.mp4"  
    alternatives v3 {  
        videoseq v31 "v31.mp4"  
        videoseq v32 "v32.mp4"  
    }  
}
```



Properties

Property	Value
Description	
Duration	0
Location	v31.mp4
Probability	0
Videoid	v31

# Loading Models (2)



## Persistence of Models in XMI (XML Metadata Interchange)

The screenshot shows the Xtext IDE interface with two tabs:

- foo3.videogen**: Shows the Xtext grammar definition:VideoGen {  
 mandatory videoseq v1 "V1/v1.mp4"  
 optional videoseq v2 "v2folder/v2.mp4"  
 alternatives v3 {  
 videoseq v31 "v31.mp4"  
 videoseq v32 "v32.mp4"  
 }  
}
- foo3.videogen**: Shows the generated XMI representation of the model. The tree view shows the structure of the Video Generator Model, including Video Gen Information, Mandatory Video Seq, Optional Video Seq, and Alternative Video Seq. The properties view shows the values for Description, Duration, Location, Probability, and Videoid.

```
<?xml version="1.0" encoding="ASCII"?>  
<videoGen:VideoGeneratorModel xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI">  
  <information/>  
  <videoseqs xsi:type="videoGen:MandatoryVideoSeq">  
    <description videoid="v1" location="V1/v1.mp4"/>  
  </videoseqs>  
  <videoseqs xsi:type="videoGen:OptionalVideoSeq">  
    <description videoid="v2" location="v2folder/v2.mp4"/>  
  </videoseqs>  
  <videoseqs xsi:type="videoGen:AlternativeVideoSeq" videoid="v3">  
    <videodescs videoid="v31" location="v31.mp4"/>  
    <videodescs videoid="v32" location="v32.mp4"/>  
  </videoseqs>  
</videoGen:VideoGeneratorModel>
```

```

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());
}

```

```

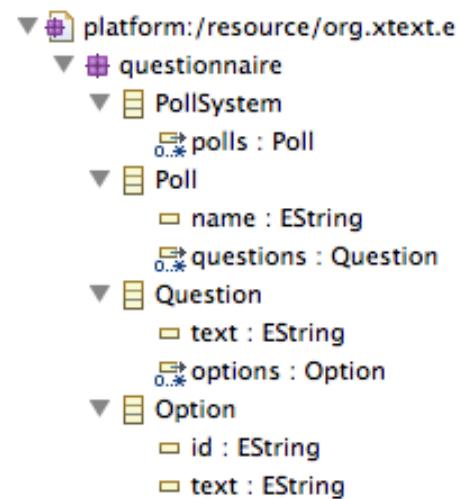
@Test
def test1() {

    // 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)

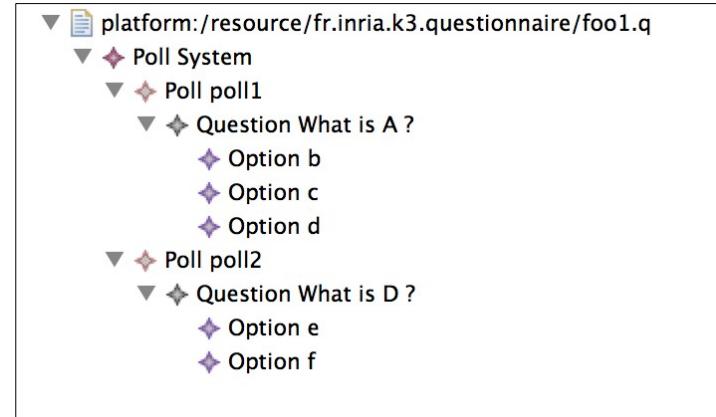
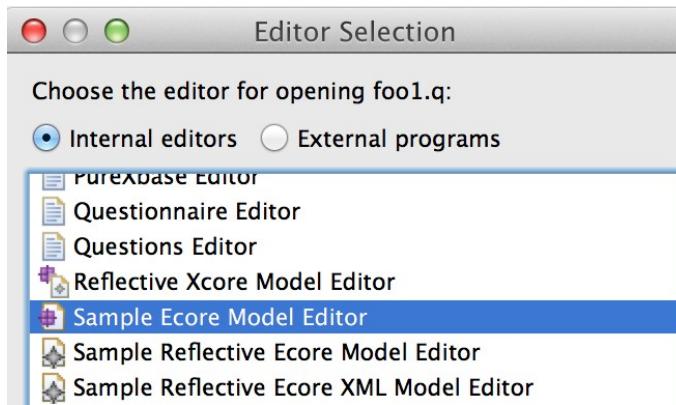
The screenshot illustrates the process of loading a UML-like questionnaire model into a tool.

**Top Right:** A tree view of the model structure under `platform:/resource/org.xtext.e`:

- `questionnaire`
  - `PollSystem`
    - `polls : Poll`
  - `Poll`
    - `name : EString`
    - `questions : Question`
  - `Question`
    - `text : EString`
    - `options : Option`
  - `Option`
    - `id : EString`
    - `text : EString`

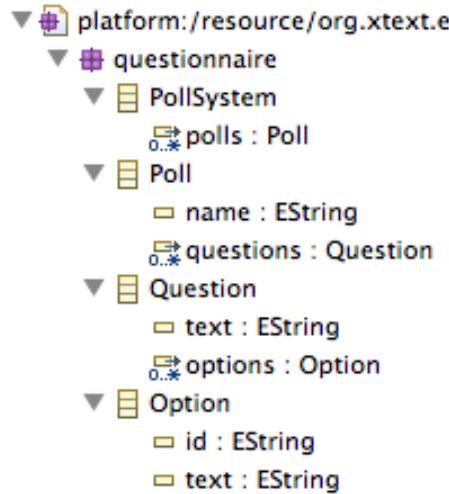
```
foo.q
-----
PollSystem {
    Poll poll1 {
        Question A {
            "What is A ?"
            options
                b : "B"
                c : "C"
                d : "D"
        }
    }

    Poll poll2 {
        Question D {
            "What is D ?"
            options
                e : "E"
                f : "F"
        }
    }
}
```

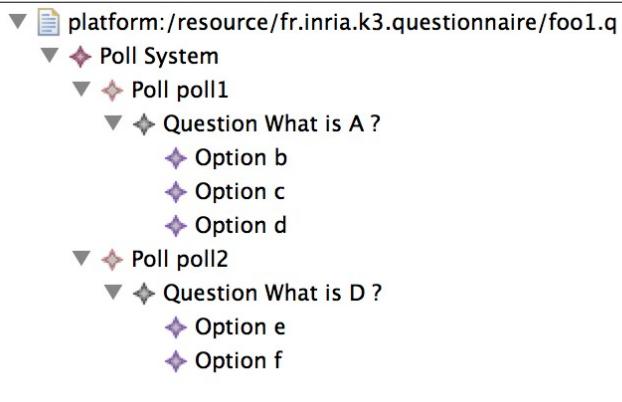


# Loading Models (2)

```
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"
        }
    }
}
```

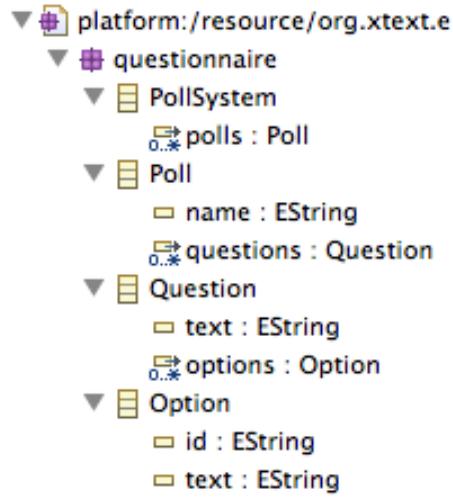


XMI



```
<?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)

```
foo1.q ☒
foolSystem {
    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"
        }
    }
}
```

```
<?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

platform:/resource/org.xtext.e

questionnaire

PollSystem

o.\* polls : Poll

Poll

name : EString

o.\* questions : Question

Question

text : EString

o.\* options : Option

Option

id : EString

text : EString

org.xtext.example.mydsl.questionnaire

Option.java

Poll.java

PollSystem.java

Question.java

QuestionnaireFactory.java

QuestionnairePackage.java

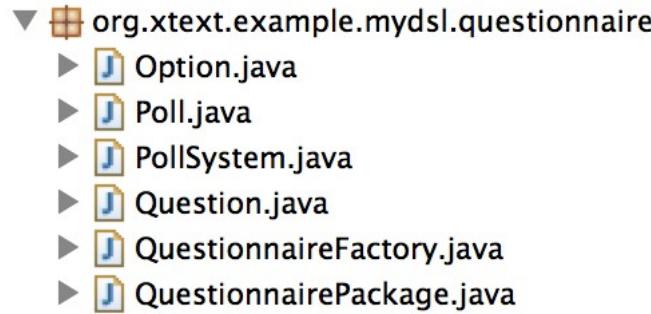
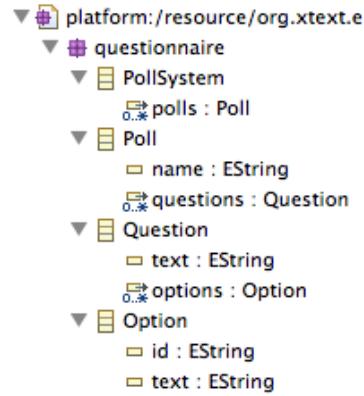
```
public interface Poll extends EObject
{
    /**
     * Returns the value of the '<em><b>Name</b></em>' attribute.
     * <!-- begin-user-doc -->
     * <p>
     * If the meaning of the '<em>Name</em>' attribute isn't clear,
     * there really should be more of a description here...
     * </p>
     * <!-- 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

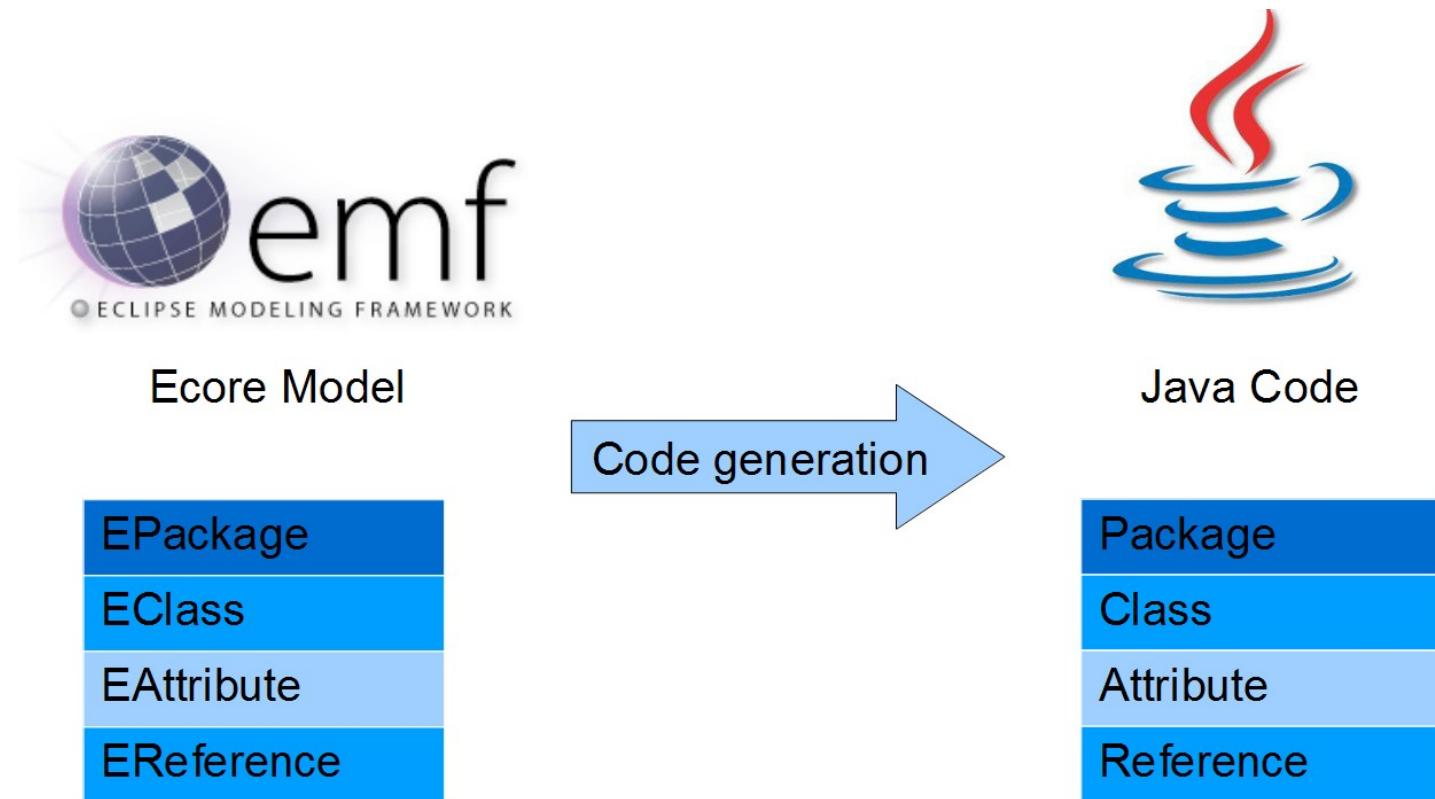


```
public interface Poll extends EObject {
    /**
     * Returns the value of the '<em><em><b>name</b></em></em>' attribute.
     * <br/><!-- begin-user-doc -->
     * <!-- The following method is provided for generated classes. An equivalent method may be present in the generated class<br/><!-- end-user-doc -->
     * <br/>Return the value of the '<em><em><b>name</b></em></em>' attribute.
     * <br/>#setname(String)
     * <!-- org.xtext.example.mydsl.questionnaire.QuestionnairePackage#getPoll_Name() -->
     * <!-- #model
     * <!-- #generated
     * String getName();
     */
    /**
     * Sets the value of the '<em><em><b>name</b></em></em>' attribute.
     * <br/><!-- begin-user-doc -->
     * <!-- <!-- end-user-doc -->
     * <br/>Return the new value of the '<em><em><b>name</b></em></em>' attribute.
     * <br/>#setname(String)
     * <!-- org.xtext.example.mydsl.questionnaire.QuestionnairePackage#getPoll_Name() -->
     * <!-- #model
     * <!-- #generated
     * void setName(String value);
     */
    /**
     * Returns the value of the '<em><em><b>questions</b></em></em>' containment reference list.
     * <br/>The list contents are of type '#link org.xtext.example.mydsl.questionnaire.Question'.
     * <br/><!-- begin-user-doc -->
     * <!-- <!-- end-user-doc -->
     * <br/>Return the value of the '<em><em><b>questions</b></em></em>' containment reference list isn't clear,
     * <br/>there should be more of a description here...
     * <br/><!-- begin-user-doc -->
     * <!-- <!-- end-user-doc -->
     * <br/>Return the value of the '<em><em><b>questions</b></em></em>' containment reference list.
     * <br/>#size org.xtext.example.mydsl.questionnaire.QuestionnairePackage#getPoll_Questions()
     * <!-- <!-- contains="true" -->
     * <!-- #generated
     * EList<Question> getQuestions();
     */
    // ...
}
```

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



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



```
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"
        }

    }

}


```

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

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

@Test
def testIO {
    // loading
    var polls = loadPollSystem(URI.createURI("fool.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)
}
```

```
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"
        }

    }

}


```

## Questionnaire MM (ecore)

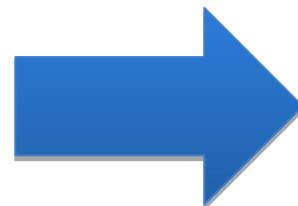
## Questionnaire MM (ecore)

## 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
}
```

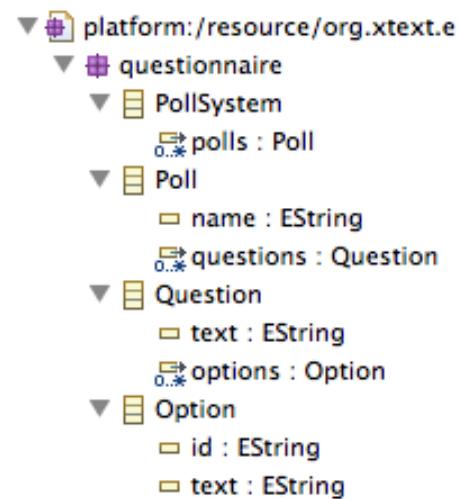
```
def savePollSystem(URI uri, PollSystem pollS) {
    var Resource rs = new ResourceSetImpl().createResource(uri);
    rs.getContents.add(pollS);
    rs.save(new HashMap());
}
```

```
@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"))

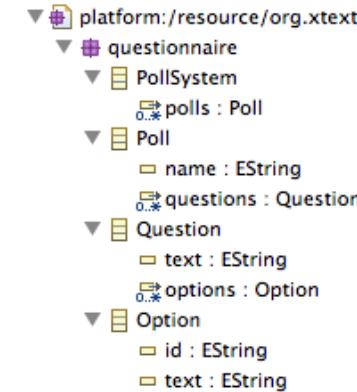
    // MODEL MANAGEMENT (ANALYSIS, TRANSFORMATION)
    var html = toPolls(pollS.polls)
    assertNotNull(html)

    // 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 ?**

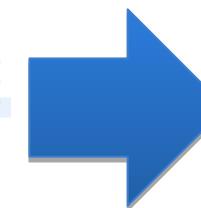
- B
- C
- D

foo1.q

```

class 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"
            }
        }
    }
}

```



**poll2**

**What is D ?**

- E
- 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)

# What are the problems with this Xtext grammar?

```
grammar org.xtext.example.mydsl.VideoGen with org.eclipse.xtext.common.Terminals

generate videoGen "http://www.xtext.org/example/mydsl/VideoGen"

VideoGeneratorModel:
    'VideoGen' LEFT_BRACKET
    videoseqs+=VideoSeq+
    RIGHT_BRACKET
    ;

VideoSeq: MandatoryVideoSeq | OptionalVideoSeq | AlternativeVideoSeq ;

MandatoryVideoSeq : 'mandatory' VideoDescription;
OptionalVideoSeq : 'optional' VideoDescription;
AlternativeVideoSeq : 'alternatives' (videoid=ID)? LEFT_BRACKET
    videodescs+=VideoDescription+ RIGHT_BRACKET
    ;

VideoDescription : 'videoseq' videoid=ID STRING
    ;

terminal LEFT_BRACKET: '{' ;
terminal RIGHT_BRACKET: '}' ;
```

```

grammar org.xtext.example.mydsl.VideoGen with org.eclipse.xtext.common.Terminals

generate videoGen "http://www.xtext.org/example/mydsl/VideoGen"

VideoGeneratorModel:
    'VideoGen' LEFT_BRACKET
    videoseqs+=VideoSeq+
    RIGHT_BRACKET
;

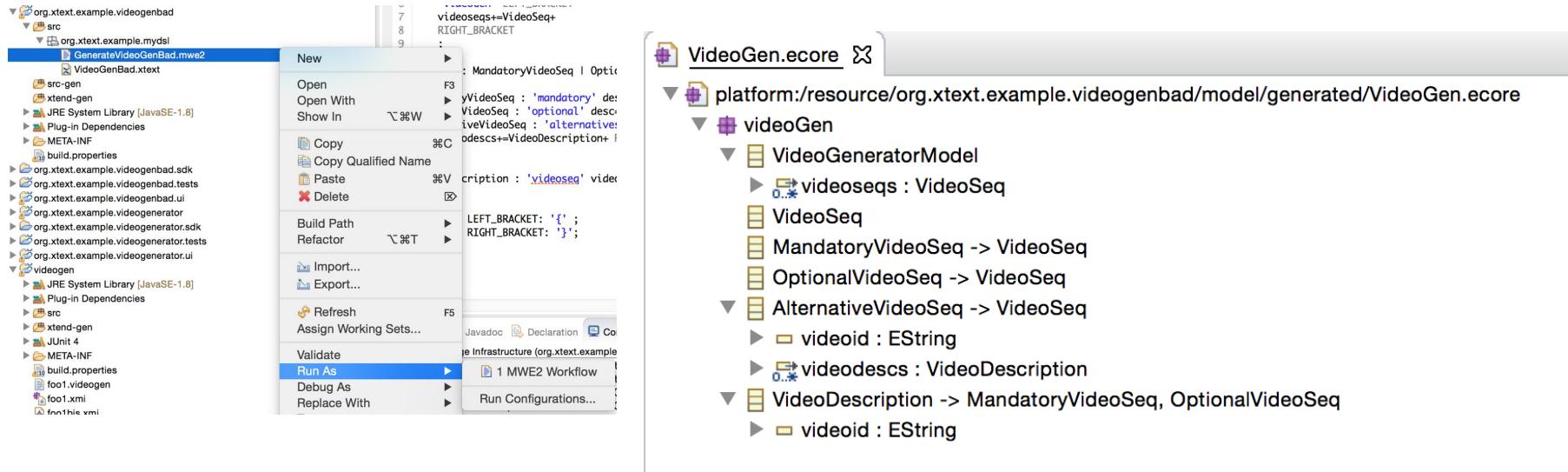
VideoSeq: MandatoryVideoSeq | OptionalVideoSeq | AlternativeVideoSeq ;

MandatoryVideoSeq : 'mandatory' VideoDescription;
OptionalVideoSeq : 'optional' VideoDescription;
AlternativeVideoSeq : 'alternatives' (videoid=ID)? LEFT_BRACKET
    videodescs+=VideoDescription+ RIGHT_BRACKET
;

VideoDescription : 'videoseq' videoid=ID STRING
;

terminal LEFT_BRACKET: '{';
terminal RIGHT_BRACKET: '}';

```



```

grammar org.xtext.example.mydsl.VideoGen with org.eclipse.xtext.common.Terminals

generate videoGen "http://www.xtext.org/example/mydsl/VideoGen"

VideoGeneratorModel:
    'VideoGen' LEFT_BRACKET
    videoseqs+=VideoSeq+
    RIGHT_BRACKET
;

VideoSeq: MandatoryVideoSeq | OptionalVideoSeq | AlternativeVideoSeq ;

MandatoryVideoSeq : 'mandatory' VideoDescription;
OptionalVideoSeq : 'optional' VideoDescription;
AlternativeVideoSeq : 'alternatives' (videoid=ID)? LEFT_BRACKET
    videodescs+=VideoDescription+
    RIGHT_BRACKET
;

VideoDescription : 'videoseq' videoid=ID STRING
;

terminal LEFT_BRACKET: '{';
terminal RIGHT_BRACKET: '}';

```



**foo1.videogenbad**

```

VideoGen {
    mandatory videoseq v1 "v1.avi"
    optional videoseq v2 "v2.mp4"
}

```



**foo1.videogenbad**

```

platform:/resource/FooVideoBad/foo1.videogenbad
    ▾ Video Generator Model
        * Video Description v1
        ♦ Video Description v2

```

**Properties**

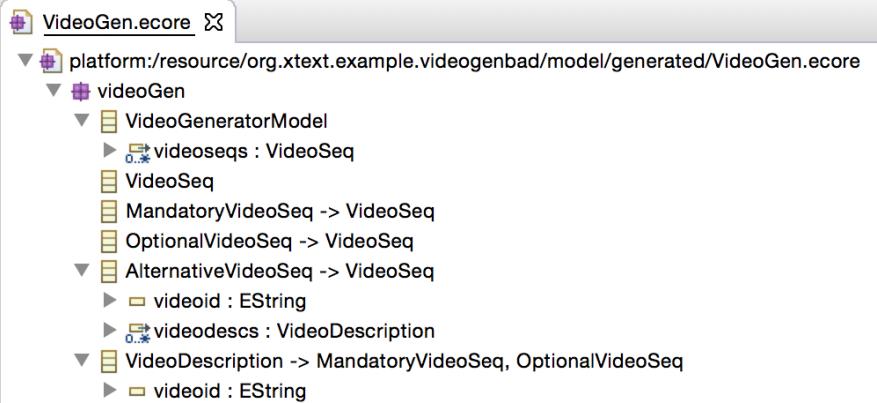
Property	Value
Videoid	v1

foo1.videoogenbad

```

VideoGen {
    mandatory videoseq v1 "v1.avi"
    optional videoseq v2 "v2.mp4"
}

```



```

def loadVideoGenerator(URI uri) {
    new VideoGenStandaloneSetupGenerated().createInjectorAndDoEMFRegistration()
    var res = new ResourceSetImpl().getResource(uri, true);
    res.contents.get(0) as VideoGeneratorModel
}

@Test
def test1() {

    // loading
    var videoGen = loadVideoGenerator(URI.createURI("foo1.videoogenbad"))
    assertNotNull(videoGen)

    // MODEL MANAGEMENT (ANALYSIS, TRANSFORMATION)
    assertEquals(2, videoGen.videoseqs.size)

    // MODEL MANAGEMENT (ANALYSIS, TRANSFORMATION)
    videoGen.videoseqs.forEach[videoseq] {
        if (videoseq instanceof MandatoryVideoSeq) {
            val desc = (videoseq as MandatoryVideoSeq).description
            // ...
        }
    }
}

```

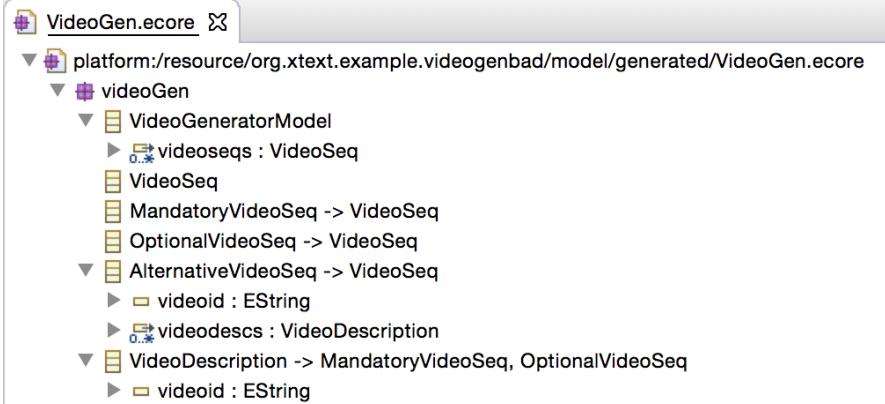
The method description is undefined....

foo1.videoogenbad

```

VideoGen {
    mandatory videoseq v1 "v1.avi"
    optional videoseq v2 "v2.mp4"
}

```



```

def loadVideoGenerator(URI uri) {
    new VideoGenStandaloneSetupGenerated().createInjectorAndDoEMFRegistration()
    var res = new ResourceSetImpl().getResource(uri, true);
    res.contents.get(0) as VideoGeneratorModel
}

@Test
def test1() {

    // loading
    var videoGen = loadVideoGenerator(URI.createURI("foo1.videoogenbad"))
    assertNotNull(videoGen)

    // MODEL MANAGEMENT (ANALYSIS, TRANSFORMATION)
    assertEquals(2, videoGen.videoseqs.size)

    // MODEL MANAGEMENT (ANALYSIS, TRANSFORMATION)
    videoGen.videoseqs.forEach[videoseq] {
        if (videoseq instanceof MandatoryVideoSeq) {
            val desc = (videoseq as MandatoryVideoSeq).description
            // ...
        }
    }
}

```

The method description is undefined....

VideoGen.ecore

platform:/resource/org.xtext.example.videogenbad/model/generated/VideoGen.ecore

videoGen

- VideoGeneratorModel
  - videoseqs : VideoSeq
  - VideoSeq
  - MandatoryVideoSeq -> VideoSeq
  - OptionalVideoSeq -> VideoSeq
- AlternativeVideoSeq -> VideoSeq
  - videoid : EString
  - videodescs : VideoDescription
- VideoDescription -> MandatoryVideoSeq, OptionalVideoSeq
  - videoid : EString

```
1+ /**
2 package org.xtext.example.mydsl.videoGen;
3
4
5
6 /**
7 * <!-- begin-user-doc -->
8 * A representation of the model object 'Mandatory Video Seq'.
9 * <!-- end-user-doc -->
10 *
11 *
12 * @see org.xtext.example.mydsl.videoGen.VideoGenPackage#getMandatoryVideoSeq()
13 * @model
14 * @generated
15 */
16 public interface MandatoryVideoSeq| extends VideoSeq
17 {
18 } // MandatoryVideoSeq
19
```

```
grammar org.xtext.example.mydsl.VideoGen with org.eclipse.xtext.common.Terminals
```

```
generate videoGen "http://www.xtext.org/example/mydsl/VideoGen"
```

```
VideoGeneratorModel:
```

```
    'VideoGen' LEFT_BRACKET  
    videoseqs+=VideoSeq+  
    RIGHT_BRACKET  
    ;
```

```
VideoSeq: MandatoryVideoSeq | OptionalVideoSeq | AlternativeVideoSeq ;
```

```
MandatoryVideoSeq : 'mandatory' VideoDescription;
```

```
OptionalVideoSeq : 'optional' VideoDescription;
```

```
AlternativeVideoSeq : 'alternatives' (videoid=ID)? LEFT_BRACKET  
    videodescs+=VideoDescription+ RIGHT_BRACKET
```

```
;
```

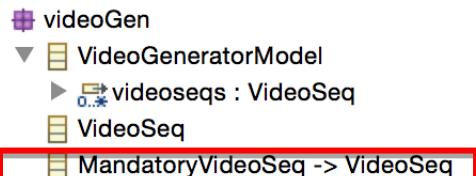
```
VideoDescription : 'videoseq' videoid=ID STRING  
    ;
```

```
terminal LEFT_BRACKET: '{' ;
```

```
terminal RIGHT_BRACKET: '}' ;
```

deoGen.ecore

| platform:/resource/org.xtext.example.videogenbad/model/generated/VideoGen.ecore



```
1  /**  
2  package org.xtext.example.mydsl.videoGen;  
3  
4  
5  
6  /**  
7  * <!-- begin-user-doc -->  
8  * A representation of the model object 'Mandatory Video Seq'.  
9  * <!-- end-user-doc -->  
10 *  
11 *  
12 * @see org.xtext.example.mydsl.videoGen.VideoGenPackage#getMandatoryVideoSeq()  
13 * @model  
14 * @generated  
15 */  
16 public interface MandatoryVideoSeq extends VideoSeq  
17 {  
18 } // MandatoryVideoSeq  
19
```

# Fixing the grammar

```
grammar org.xtext.example.mydsl.VideoGen with org.eclipse.xtext.common.Terminals

generate videoGen "http://www.xtext.org/example/mydsl/VideoGen"

VideoGeneratorModel:
    'VideoGen' LEFT_BRACKET
    videoseqs+=VideoSeq+
    RIGHT_BRACKET
;

VideoSeq: MandatoryVideoSeq | OptionalVideoSeq | AlternativeVideoSeq ;

MandatoryVideoSeq : 'mandatory' description=VideoDescription;
OptionalVideoSeq : 'optional' VideoDescription;
AlternativeVideoSeq : 'alternatives' (videoid=ID)? LEFT_BRACKET
    videodescs+=VideoDescription+ RIGHT_BRACKET
;

VideoDescription : 'videoseq' videoid=ID STRING
;

terminal LEFT_BRACKET: '{' ;
terminal RIGHT_BRACKET: '}' ;
```

▼ platform:/resource/org.xtext.example.videogenbad/model/generated/VideoGen.ecore

▼ videoGen

    ▼ VideoGeneratorModel

        ► 0..\* videoseqs : VideoSeq

    ▼ VideoSeq

        ▼ MandatoryVideoSeq -> VideoSeq

        ► description : VideoDescription

        ▼ OptionalVideoSeq -> VideoSeq

        ▼ AlternativeVideoSeq -> VideoSeq

            ► videoid : EString

            ► 0..\* videodescs : VideoDescription

        ▼ VideoDescription -> OptionalVideoSeq

            ► videoid : EString

```
1 /**
2 */
3 package org.xtext.example.mydsl.videoGen;
4
5 /**
6 * <!-- begin-user-doc -->
7 * A representation of the model object 'Mandatory Video Seq'.
8 * <!-- end-user-doc -->
9 *
10 * <p>
11 * The following features are supported:
12 * </p>
13 * <ul>
14 * <li>{@link org.xtext.example.mydsl.videoGen.MandatoryVideoSeq#getDescription Description}</li>
15 * </ul>
16 *
17 * @see org.xtext.example.mydsl.videoGen.VideoGenPackage#getMandatoryVideoSeq()
18 * @model
19 * @generated
20 */
21
22 public interface MandatoryVideoSeq extends VideoSeq
23 {
24 /**
25 * Returns the value of the 'Description' containment reference.
26 * <!-- begin-user-doc -->
27 * <p>
28 * If the meaning of the 'Description' containment reference isn't clear,
29 * there really should be more of a description here...
30 * </p>
31 * <!-- end-user-doc -->
32 * @return the value of the 'Description' containment reference.
33 * @see #setDescription(VideoDescription)
34 * @see org.xtext.example.mydsl.videoGen.VideoGenPackage#getMandatoryVideoSeq_Description()
35 * @model containment="true"
36 * @generated
37 */
38 VideoDescription getDescription();
39
40 /**
41 * Sets the value of the '{@link org.xtext.example.mydsl.videoGen.MandatoryVideoSeq#getDescription Description}' containment reference.
42 * <!-- begin-user-doc -->
43 * <!-- end-user-doc -->
44 * @param newValue the new value of the 'Description' containment reference.
45 * @see #getDescription()
46 * @generated
47 */
48 void setDescription(VideoDescription value);
49 } // MandatoryVideoSeq
```

// loading  
var videoGen = loadVideoGenerator(URI.createURI("foo1.videogenbad"))  
assertNotNull(videoGen)

// MODEL MANAGEMENT (ANALYSIS, TRANSFORMATION)  
assertEquals(2, videoGen.videoseqs.size)

// MODEL MANAGEMENT (ANALYSIS, TRANSFORMATION)  
videoGen.videoseqs.forEach[videoseq |  
    if (videoseq instanceof MandatoryVideoSeq) {  
        val desc = (videoseq as MandatoryVideoSeq).description  
        // ...  
    }  
]

# Grammar and Metamodel

- Model transformations are defined on top of metamodel constructs
- **Co-design** of grammar and metamodel
  - Grammar defines the syntax
  - Metamodel defines the structure
  - Xtext facilitates the metamodel design with
    - Default rules for inferring the metamodel from the grammar
    - Facilities to parameterize the inference
- Some transformations may be difficult to express. In this case two possible attitudes:
  - Revise the Xtext grammar and the underlying metamodel
  - Design another metamodel (from scratch) and write a model-to-model transformation

# Plan

- Model Management in a nutshell
  - Loading, serializing, transforming models: scenarios
  - Taxonomy
  - Model transformation in Xtend
- Xtend: a case study for GPL/DSL, MDE, and model transformation
  - Advanced features: extension methods, active annotations, template expressions
  - Xtend: behing the magic (Xtext+MDE)
  - Xtend + Xtext (breathing life into DSLs)
  - @Aspect annotation

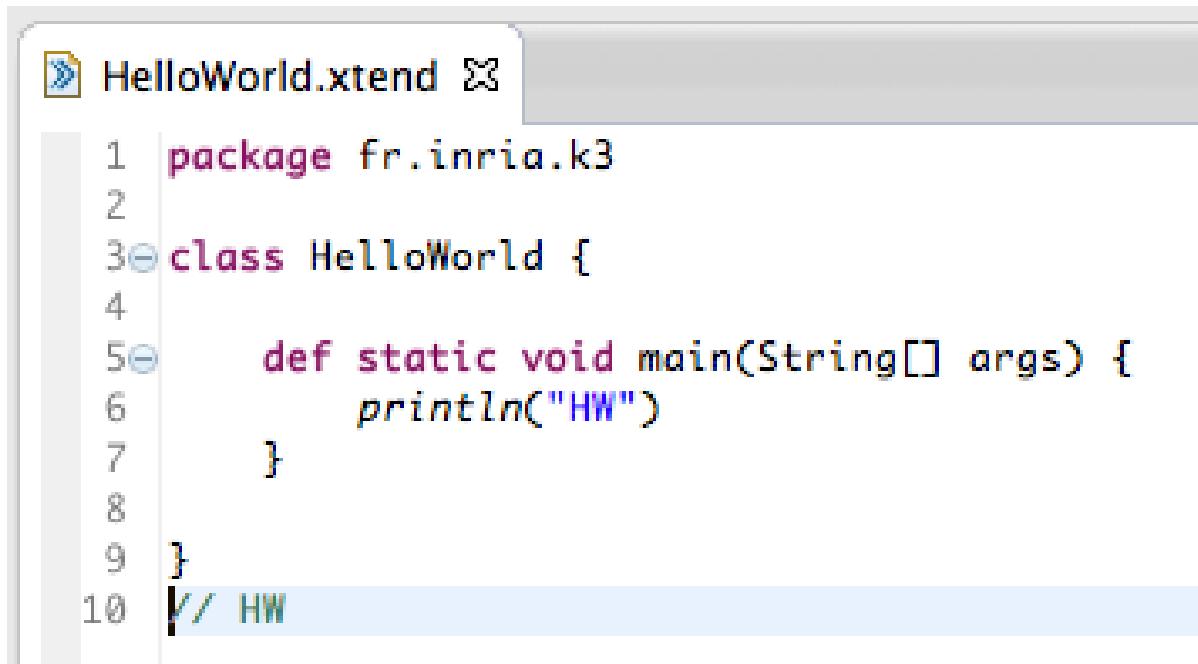
# Contract

- Practical foundations of model management
- Model transformations
  - Model-to-Text
  - Model-to-Model
  - Metaprogramming
- DSLs and model management: all together (Xtext + Xtend)

# *Xtend*

*The Basics  
(~ Java cheatsheet)*

# Hello World



The image shows a screenshot of a code editor window titled "HelloWorld.xtend". The code editor displays the following Java code:

```
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

A screenshot of a code editor showing the `HelloWorld.xtend` file. The code is:1 package fr.inria.k3  
2  
3 class HelloWorld {  
4  
5 def static void main(String[] args) {  
6 println("HW")  
7 }  
8  
9 }  
10 // HWThe tab bar at the top shows `HelloWorld.xtend` is the active tab.

Semi-colon ';' is optional

A screenshot of a code editor showing the `HelloWorld.xtend` and `HelloWorld.java` files. The tabs at the top show `HelloWorld.xtend` is active. The code is:1 package fr.inria.k3.^def  
2  
3 class PackageExploder {  
4  
5 }The `HelloWorld.java` tab is also visible.

'^' for avoiding keyword conflicts

# Methods

```
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|
```

A code editor window showing a Scala file. It contains a class `FooMethod` with four methods: `foo1`, `foo2`, `foo3` (private), and `foo4` (public). Below it is a class `FooMethodUses` with a method `fooUse` that creates instances of `FooMethod` and calls `foo1` and `foo3`. A tooltip appears over the call to `foo3`, listing three options: `foo1 : String - FooMethod.foo10`, `foo2 : int - FooMethod.foo20`, and `foo4 : int - FooMethod.foo40`. Line numbers 1 through 35 are visible on the left.

By default:  
visibility  
conditions set  
to public

```
>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
```

A code editor window showing a Java file named `HelloWorld.xtend`. It contains a single class `HelloWorld` with a static main method that prints "HW". Line numbers 1 through 10 are visible on the left.

# Methods

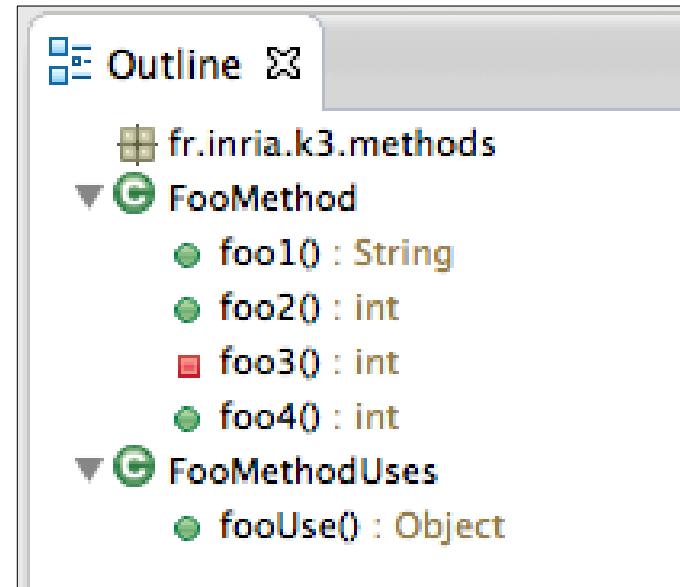
```
1 package fr.inria.k3.methods
2
3 class FooMethod {
4
5     def foo1() {
6         "A"
7     }
8
9     def foo2() {
10        6 + 3
11    }
12
13     def private foo3() {
14        6 + 3
15    }
16
17     def public foo4() {
18        foo3() * 8
19    }
20 }
21
22 class FooMethodUses {
23
24    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

new FooMethod().f|

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

```
31
32
33     }
34
35 }
```

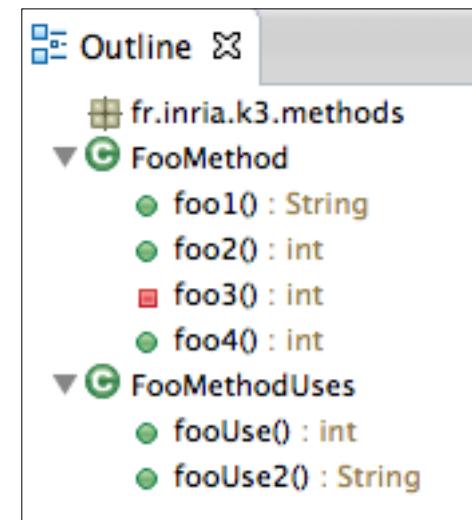
## Type inference (return type)



# Method Calling

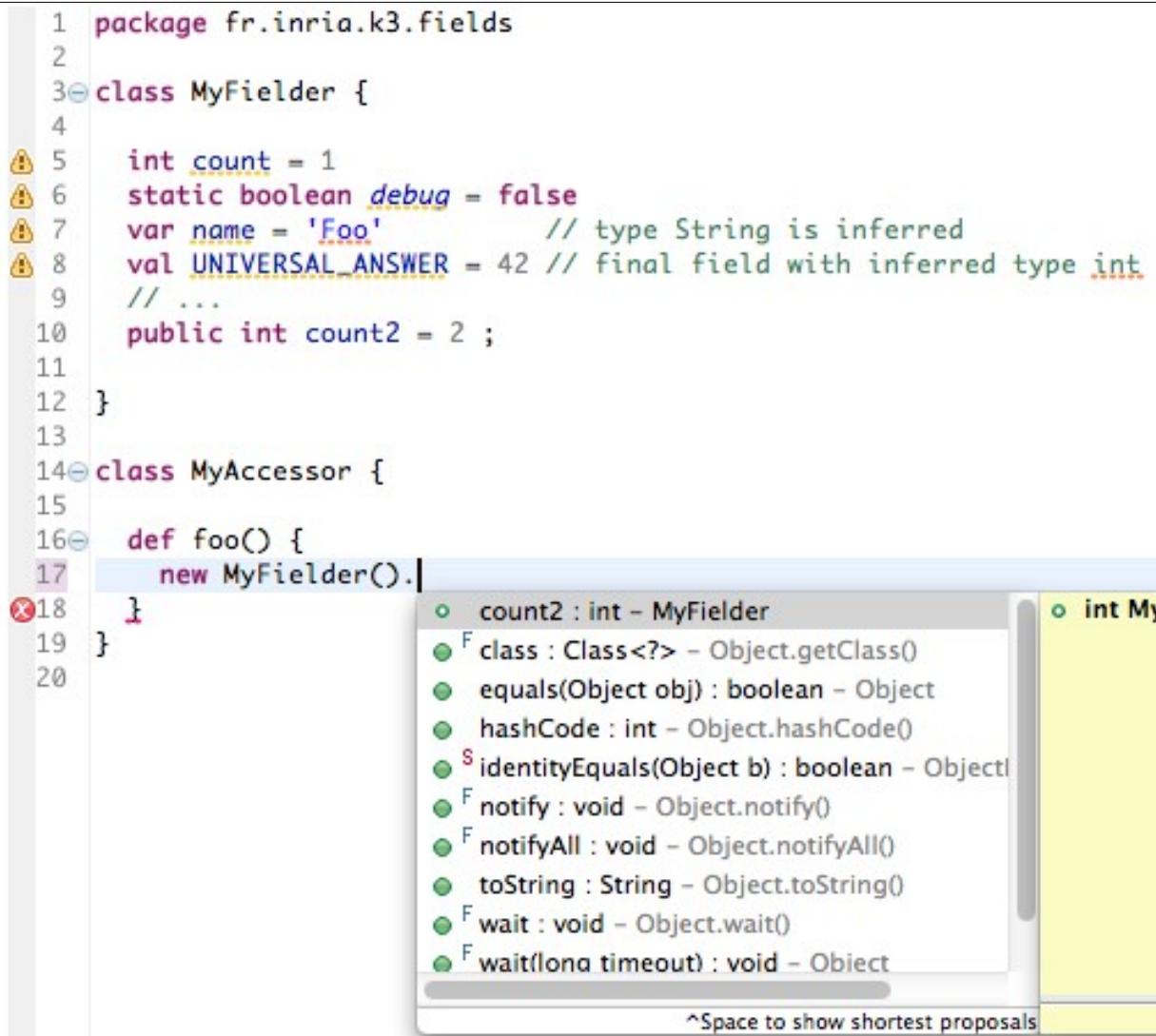
You can omit  
parentheses

```
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

```
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
```



The screenshot shows a code editor with Java-like syntax highlighting. A completion dropdown is open over the line `new MyFielder().|`. The dropdown lists several methods:

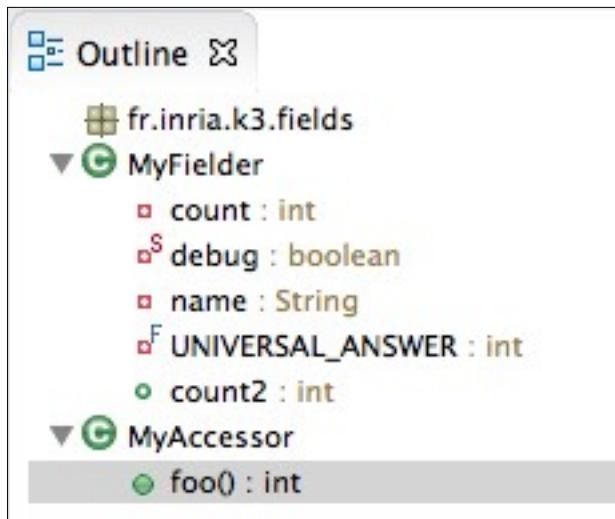
- count2 : int - MyFielder
- class : Class<?> - Object.getClass()
- equals(Object obj) : boolean - Object
- hashCode : int - Object.hashCode()
- identityEquals(Object b) : boolean - Object
- notify : void - Object.notify()
- notifyAll : void - Object.notifyAll()
- toString : String - Object.toString()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

At the bottom of the dropdown, there is a note: "Space to show shortest proposals".

By default:  
visibility  
conditions set  
to private

# 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]
```

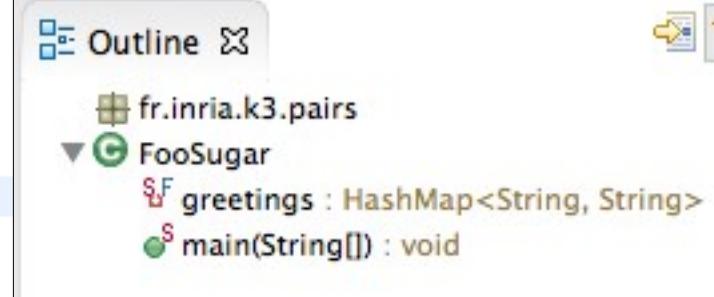
# Pairs

```
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  
▼ G 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 }
```



The screenshot shows the IntelliJ IDEA interface with the 'Console' tool window open. The console output shows the results of the program execution:

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

# Immutable data structure

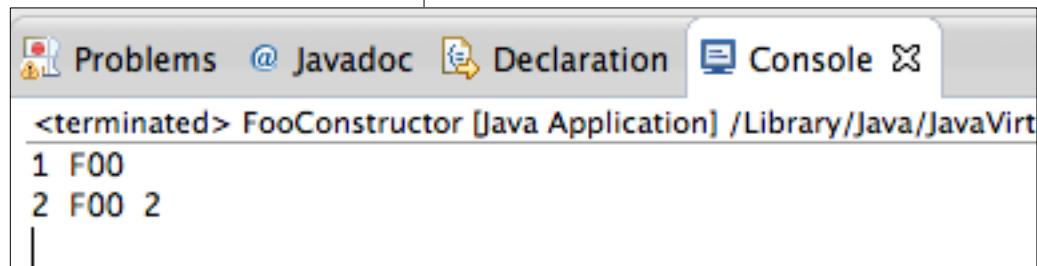
```
1 package fr.inria.k3.stat
2
3 import java.util.Collections
4
5 class FooStati {
6
7
8     static var colors = #[46, 76, 89, 53] // newArrayList(46, 76, 89, 53)
9
10
11 def static void main(String... args) {
12     println("B " + colors)
13     Collections::sort(colors)
14     println("A " + colors)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

```
<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
5     var String l
6
7     new() {
8         this("FOO")
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("FOO 2"))
23     }
24 }
```



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")
```

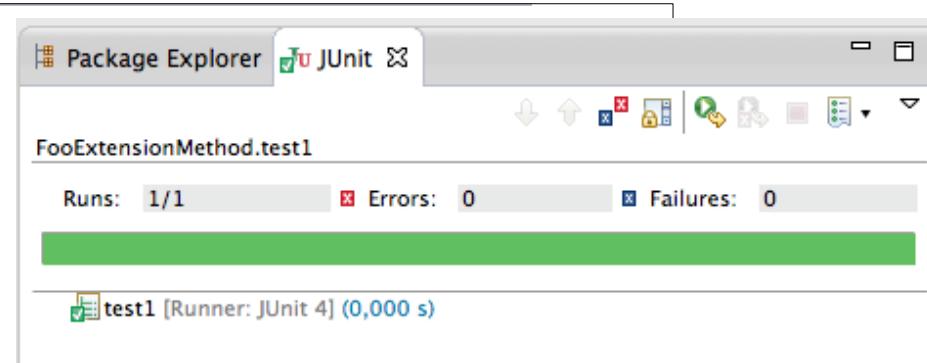
or as an extension method of String:

```
"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 <code>null</code>.  
 * @param transformation  
 *        the transformation. May not be <code>null</code>.  
 * @return a list that effectively contains the results of the transformation. Never <code>null</code>.  
 */  
@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 <code>null</code>.
 * @return the {@link String} {@code s} with an upper case first character or <code>null</code> if the input
 *      {@link String} {@code s} was <code>null</code>.
 */
@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.text = "Something happened!"
3. ])
```

You can even  
ommit e

```
1. textField.addActionListener([
2.     textField.text = "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 referring to the first parameter
    var (String)=>int l3 = [length] // we can even omit 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
9     def someHTML(String content) '''<html><body>«content»</body></html>'''
10
11
12@    @Test
13    def test1() {
14        assertEquals("<html><body>HW</body></html>", someHTML('HW').toString)
15    }
16
17 }
```

```

@Test
def test2() {

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

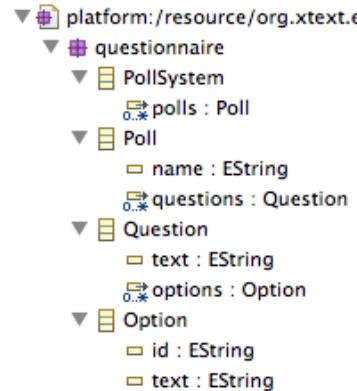
    // MODEL MANAGEMENT (ANALYSIS, TRANSFORMATION)
    var html = toPolls(polls.polls)
    assertNotNull(html)

    // 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>
...

```

# Templates (2)



**poll1**

**What is A ?**

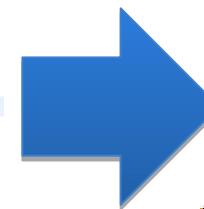
- B
- C
- D

foo1.q

```

class 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"
        }
    }
}

```



**poll2**

**What is D ?**

- E
- F

# Templates (3)

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

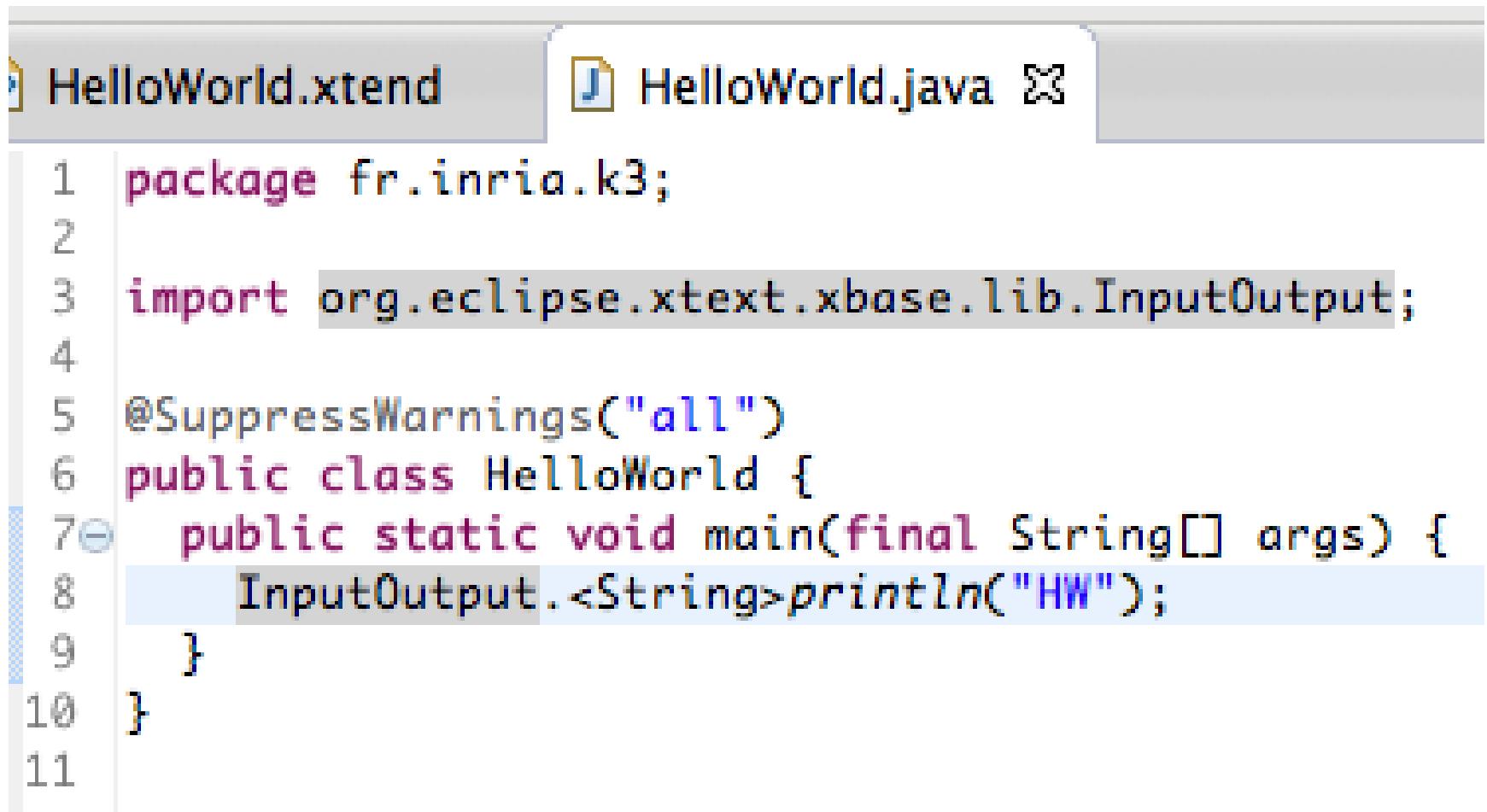
```
<h1>Exemple de page JSP</h1>
<%-- Impression de variables --%>
<p>Au moment de l'exécution de ce script, nous sommes le <%= date %>. </p>
<p>Cette page a été affichée <%= nombreVisites %> fois !</p>
</body>
</html>
```

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



- Xtend: seamless integration into a general purpose language

# Xtend to Java



The screenshot shows an IDE interface with two tabs. The active tab is "HelloWorld.java", which contains the following Java code:

```
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 }
```

The "HelloWorld.xtend" tab is visible in the background, indicating that the Xtend file is also present in the project.

# 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}
```



```
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 <code>System.out.println()</code>
     * @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: <code>println(1 + println(2)) + 3</code>
     *
     * @param object
     *          the to-be-printed object
     * @return the printed object.
     */
    public static <T> T println(T object) {
        System.out.println(object);
        return object;
    }
}
```

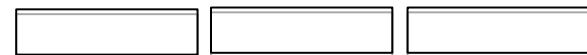
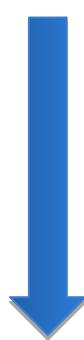
# *Xtend/Xtext*

*Back to our scenarios*

## foo1.videogen

```
mandatory videoseq v1 "https://www.youtube.com/watch?v=PJNi1uYhV5w"
optional videoseq v2 "v2Folder/v2.mp4"
alternatives v3 {
    videoseq v31 "v3/seq1.mp4"
    videoseq v32 "v3/seq1.mp4"
    videoseq v33 "v3/seq1.mp4"
}

alternatives v4 {
    videoseq v41 "v4/seq1.mp4"
    videoseq v42 "v4/seq1.mp4"
}
mandatory videoseq v5 "https://www.youtube.com/watch?v=ezKx-S0LiNQ"
```



18, 167, 899



# Refactoring

```

def loadVideoGenerator(URI uri) {
    new VideoGenStandaloneSetupGenerated().createInjectorAndDoEMFRegistration()
    var res = new ResourceSetImpl().getResource(uri, true);
    res.contents.get(0) as VideoGeneratorModel
}

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

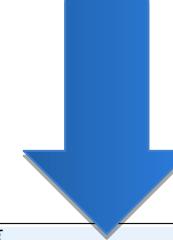
@Test
def test1() {
    // loading
    var videoGen = loadVideoGenerator(URI.createURI("foo2.videogen"))
    assertNotNull(videoGen)
    assertEquals(3, videoGen.videoseqs.size)
    // MODEL MANAGEMENT (ANALYSIS, TRANSFORMATION)
    videoGen.videoseqs.forEach[videoseq] {
        if (videoseq instanceof MandatoryVideoSeq) {
            val desc = (videoseq as MandatoryVideoSeq).description
            if(desc.videoid.isNullOrEmpty) desc.videoid = genID()
        }
        else if (videoseq instanceof OptionalVideoSeq) {
            val desc = (videoseq as OptionalVideoSeq).description
            if(desc.videoid.isNullOrEmpty) desc.videoid = genID()
        }
        else {
            val altvid = (videoseq as AlternativeVideoSeq)
            if(altvid.videoid.isNullOrEmpty) altvid.videoid = genID()
            for (vdesc : altvid.videodescs) {
                if(vdesc.videoid.isNullOrEmpty) vdesc.videoid = genID()
            }
        }
    }
    // serializing
    saveVideoGenerator(URI.createURI("foo2bis.xmi"), videoGen)
    saveVideoGenerator(URI.createURI("foo2bis.videogen"), videoGen)
}

```

```

VideoGen {
    mandatory videoseq "V1/v1.mp4"
    optional videoseq "v2folder/v2.mp4" {
        probability 25
    }
    alternatives vid3 {
        videoseq "v3/seq1.mp4"
        videoseq vid31 "v3/seq2.mp4"
        videoseq vid32 "v3/seq3.mp4"
    }
    alternatives vid4 {
        videoseq vid41 "v4/seq1.mp4"
        videoseq vid42 "v4/seq2.mp4"
    }
    mandatory videoseq vid5 "v5.mp4"
    optional videoseq vid8 "v8.avi"
    alternatives vid9 {
        videoseq vid81 "V81.avi"
    }
}

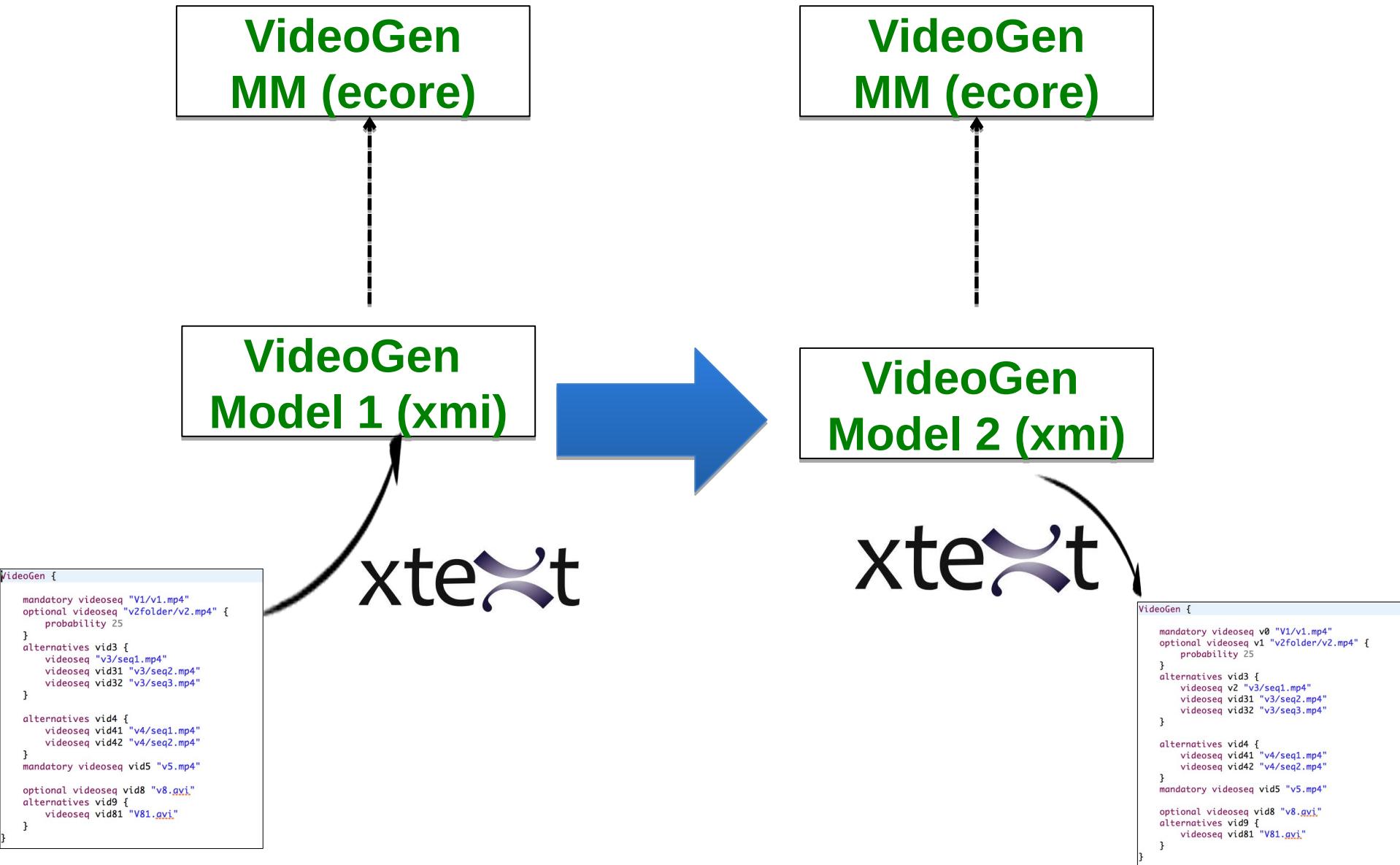
```



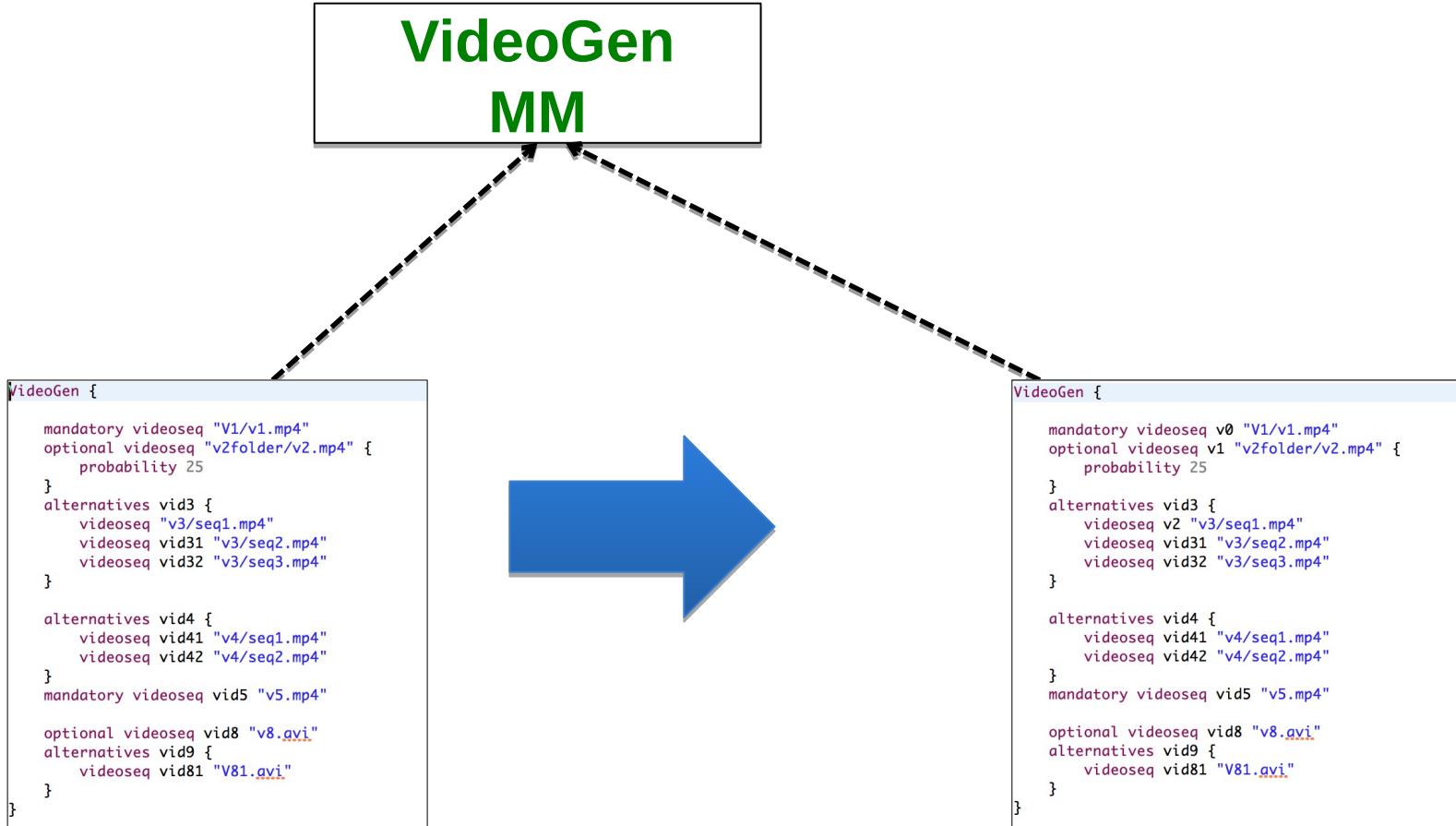
```

VideoGen {
    mandatory videoseq v0 "V1/v1.mp4"
    optional videoseq v1 "v2folder/v2.mp4" {
        probability 25
    }
    alternatives vid3 {
        videoseq v2 "v3/seq1.mp4"
        videoseq vid31 "v3/seq2.mp4"
        videoseq vid32 "v3/seq3.mp4"
    }
    alternatives vid4 {
        videoseq vid41 "v4/seq1.mp4"
        videoseq vid42 "v4/seq2.mp4"
    }
    mandatory videoseq vid5 "v5.mp4"
    optional videoseq vid8 "v8.avi"
    alternatives vid9 {
        videoseq vid81 "V81.avi"
    }
}

```



# Endogeneous Transformation



```
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
```

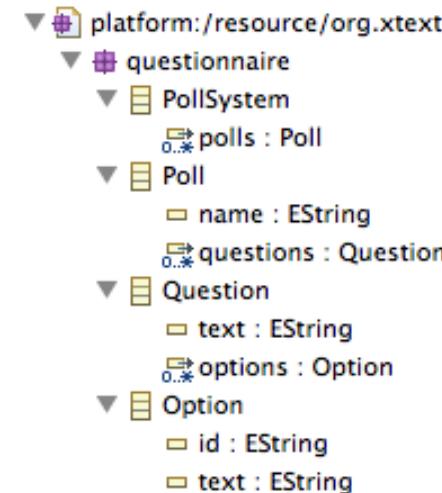
```
def test1() {
```

```
// 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)
```

```
}
```



# 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
9     def someHTML(String content) '''<html><body>«content»</body></html>'''
10
11
12@    @Test
13    def test1() {
14        assertEquals("<html><body>HW</body></html>", someHTML('HW').toString)
15    }
16
17 }
```

```

@Test
def test2() {

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

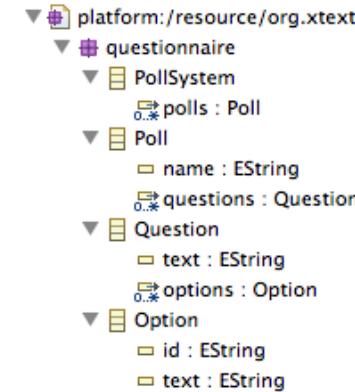
    // MODEL MANAGEMENT (ANALYSIS, TRANSFORMATION)
    var html = toPolls(pollS.polls)
    assertNotNull(html)

    // 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 ?**

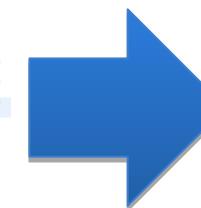
- B
- C
- D

foo1.q

```

class 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"
            }
        }
    }
}

```



**poll2**

**What is D ?**

- E
- F

# Facilities to create objects in a programmatic way



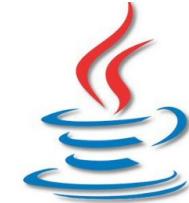
```
platform:/resource/org.xtext.e
  questionnaire
    PollSystem
      polls : Poll
    Poll
      name : EString
      questions : Question
    Question
      text : EString
      options : Option
    Option
      id : EString
      text : EString
```



Ecore Model

EPackage
EClass
EAttribute
EReference

Code generation



Java Code

Package
Class
Attribute
Reference

```
@Test
def test2() {

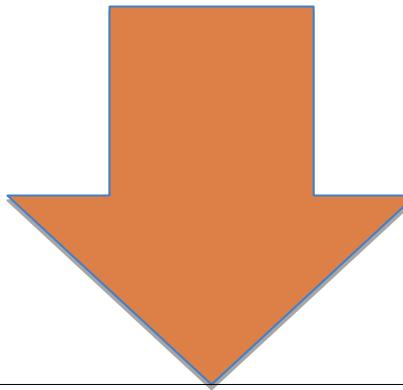
    var pollSystem = QuestionnaireFactory.eINSTANCE.createPollSystem ;
    var p1 = QuestionnaireFactory.eINSTANCE.createPoll() ;
    p1.setName("p1");
    pollSystem.polls.add(p1)
    //
```

# *Lab Sessions*

*Overview*

```
foo1.videogen &gt;
mandatory videoseq v1 "https://www.youtube.com/watch?v=PjNi1uYhV5w"
optional videoseq v2 "v2folder/v2.mp4"
alternatives v3 {
    videoseq v31 "v3/seq1.mp4"
    videoseq v32 "v3/seq1.mp4"
    videoseq v33 "v3/seq1.mp4"
}

alternatives v4 {
    videoseq v41 "v4/seq1.mp4"
    videoseq v42 "v4/seq1.mp4"
}
mandatory videoseq v5 "https://www.youtube.com/watch?v=ezKx-S0LiNQ"
```

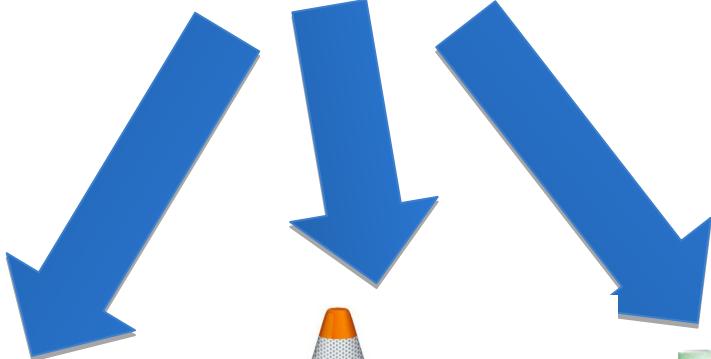


**model-to-model**

**playlist  
metamodel**

**playlist model**

**model-to-text**



flowplayer **flash**

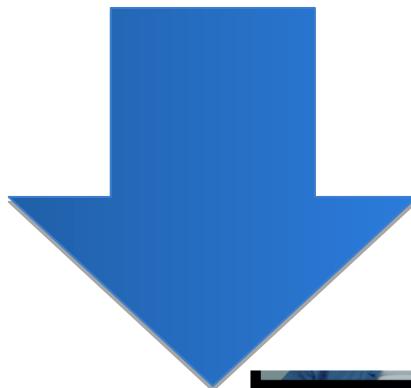


**FFmpeg**

## foo1.videogen

```
mandatory videoseq v1 "https://www.youtube.com/watch?v=PJNi1uYhV5w"
optional videoseq v2 "v2Folder/v2.mp4"
alternatives v3 {
    videoseq v31 "v3/seq1.mp4"
    videoseq v32 "v3/seq1.mp4"
    videoseq v33 "v3/seq1.mp4"
}

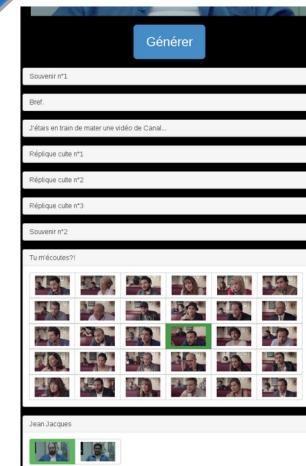
alternatives v4 {
    videoseq v41 "v4/seq1.mp4"
    videoseq v42 "v4/seq1.mp4"
}
mandatory videoseq v5 "https://www.youtube.com/watch?v=ezKx-S0LiNQ"
```



model-to-\*



Thumbnails (vignettes) of each video sequence (e.g., PGN format)



## foo1.videogen

```
mandatory videoseq v1 "https://www.youtube.com/watch?v=PJNi1uYhV5w"
optional videoseq v2 "v2folder/v2.mp4"
alternatives v3 {
    videoseq v31 "v3/seq1.mp4"
    videoseq v32 "v3/seq1.mp4"
    videoseq v33 "v3/seq1.mp4"
}

alternatives v4 {
    videoseq v41 "v4/seq1.mp4"
    videoseq v42 "v4/seq1.mp4"
}
mandatory videoseq v5 "https://www.youtube.com/watch?v=ezKx-S0LiNQ"
```



### Website/online

- Random generation
- Configurator
- Game
- ...



# *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/>



*Xtext.Xtext?*