

# Model Management

# 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)

# Model Management

## Scenarios: illustrative examples

# Bref

bref30ans.canalplus.fr/#c

## **ETAPE 2 : CHOISIS 3 BONS SOUVENIRS**



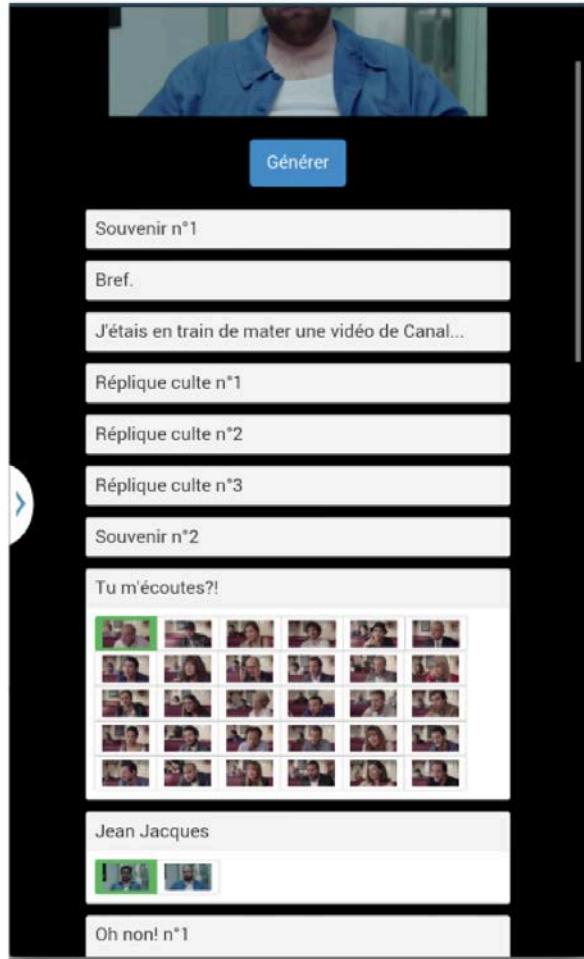
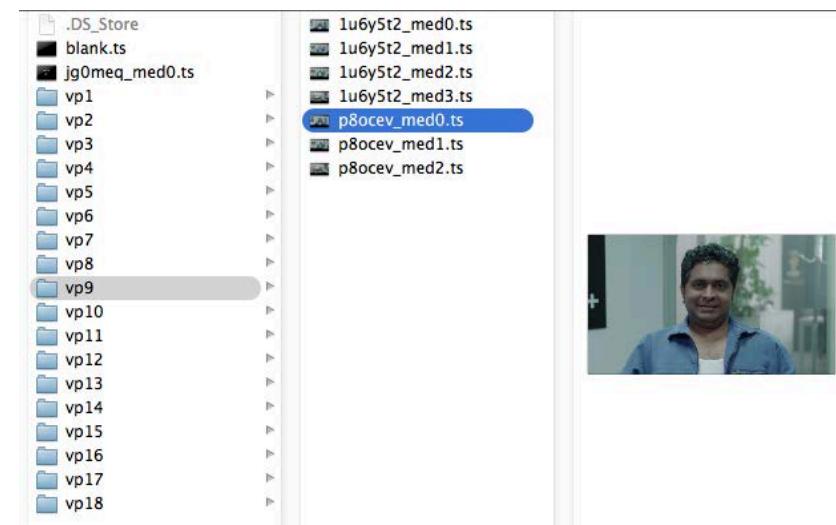


Figure 2: Re-engineering of the novel configurator (excerpt): users can now select a specific video for the 18 variation points identified during the reverse engineering of the original generator

```
"sq": ["dwlcjv", "1y60t3z", "1lyfhk", "wqzv0y",
"1xxivi2", "1oxnvtu", "lolbe9", "wvo06o",
"1u6y5t2", "1eqb8bw", "1j9aij7", "nr7jom",
"1jmvl1y", "1qgn9dh", "1bv7rka", "19ykyyw",
"5znrg7", "116hv1k"]
```



```
#EXTINF:06.40,
http://bref30cdn.wildmoka.com/vidv2/116hv1k_med0.ts
#EXTINF:03.96,
http://bref30cdn.wildmoka.com/vidv2/116hv1k_med1.ts
#EXTINF:04.52,
http://bref30cdn.wildmoka.com/vidv2/116hv1k_med2.ts
#EXTINF:03.08,
http://bref30cdn.wildmoka.com/vidv2/116hv1k_med3.ts
```





**Generator**  
~ composition of  
video sequences

**video  
variants**





**Generator**  
~ composition of  
video sequences

**video  
variants**





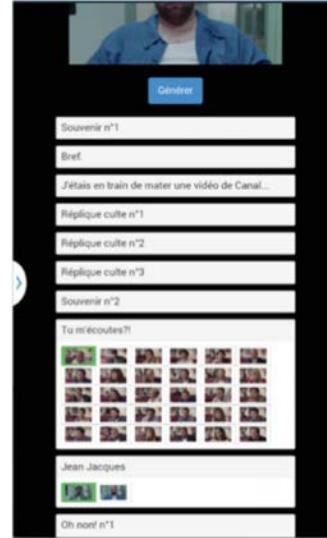
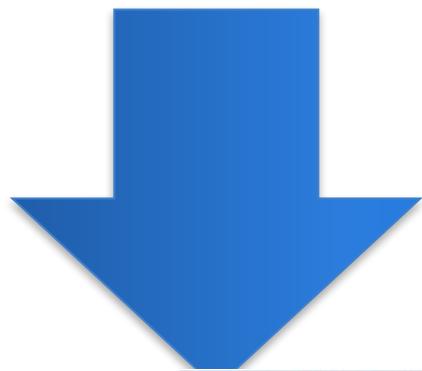
```

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"

```



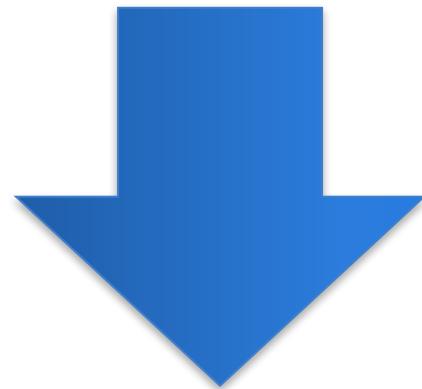
- ## Website/online
- Random generation
  - Configurator
  - Game
  - ...



```
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"
```



# Quizz Time

Write a Xtext grammar so that the specification below is conformant

```
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"
```

# Quizz Time (2)

Write a Xtext grammar so that the specification below is conformant; **what is the metamodel generated by Xtext?**

```
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"
```

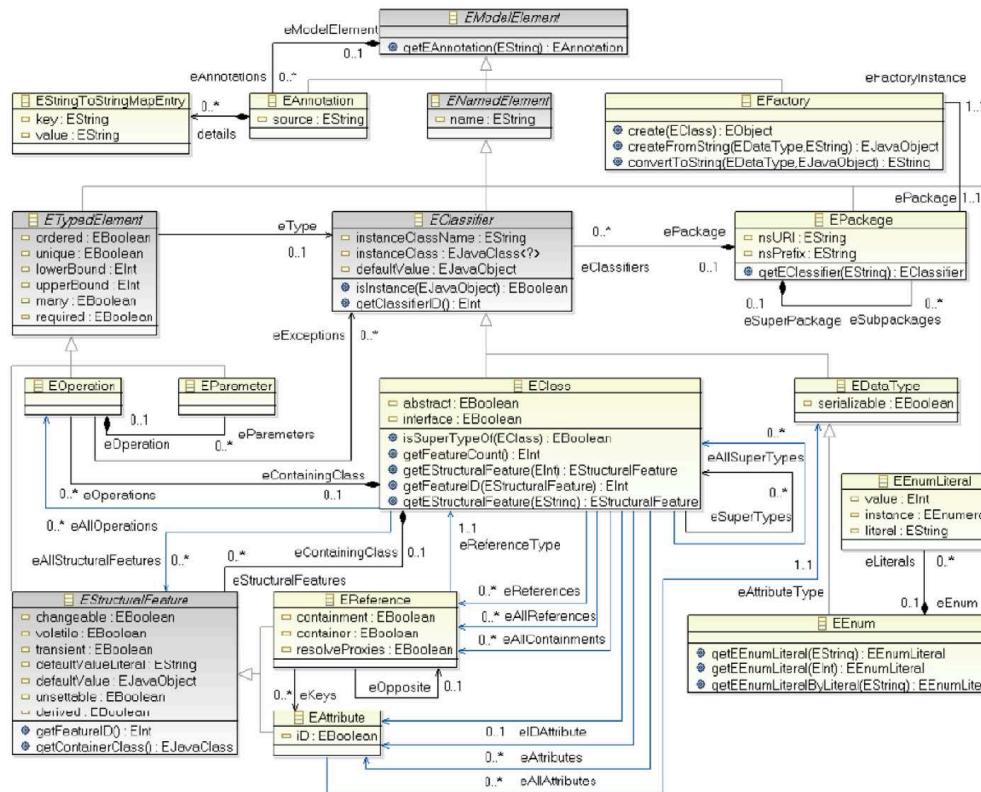
# Quizz Time (3)

Write a Xtext grammar so that the specification below is conformant; what is the metamodel MM generated by Xtext?; **use the notation of object diagram for encoding the specification below as a model conformant to MM**

```
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"
```

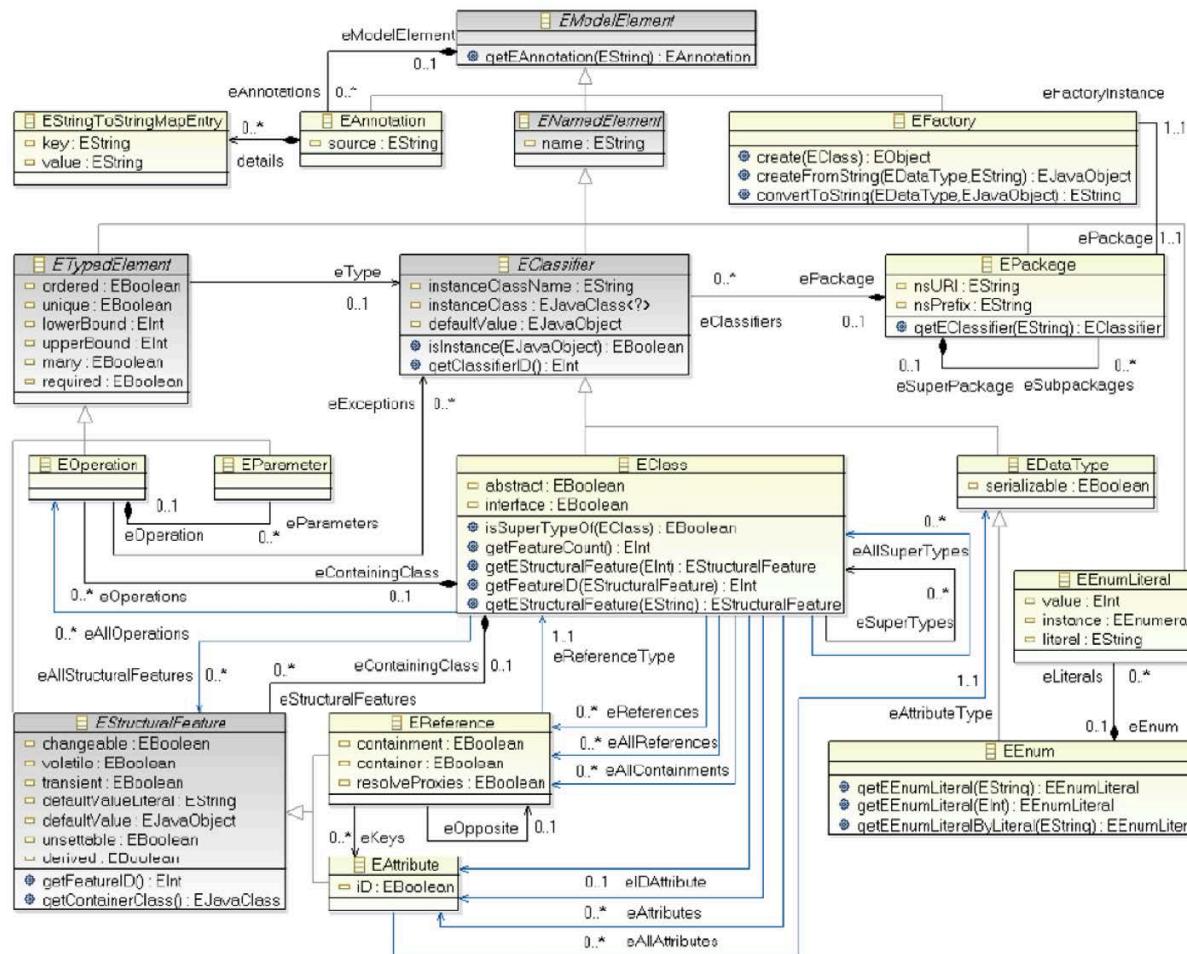
# Quizz Time (4)

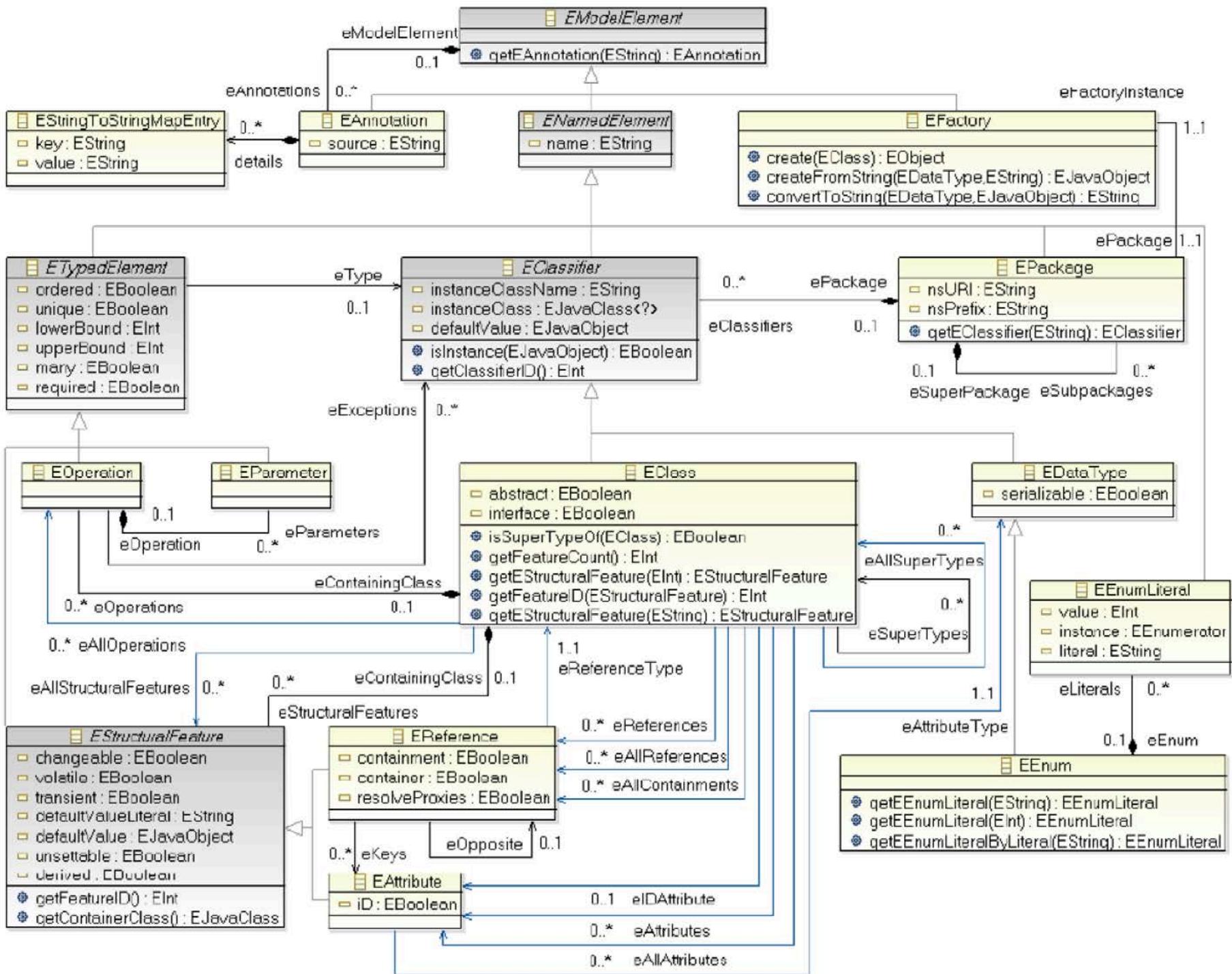
Write a Xtext grammar so that the specification below is conformant; what is the metamodel MM generated by Xtext?; **use the notation of object diagram for encoding the metamodel MM as a model conformant to Ecore**



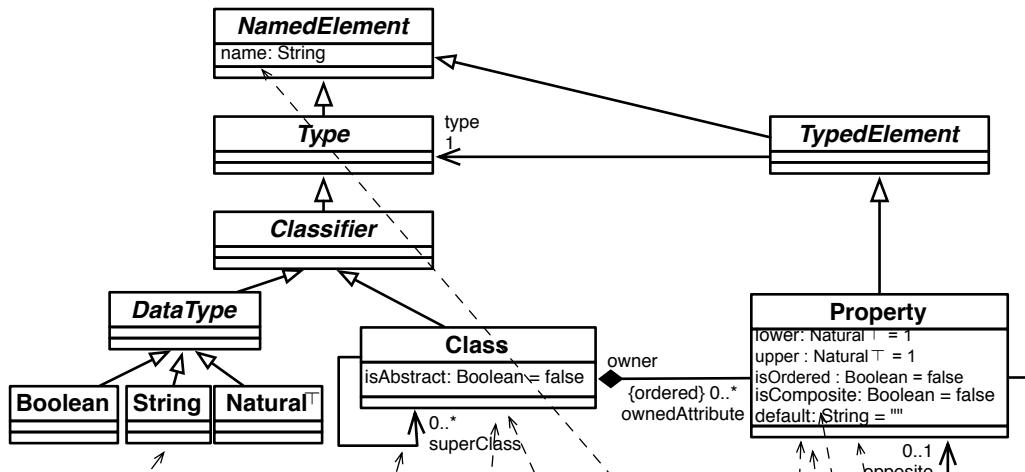
# Quizz Time (5)

# Use the notation of object diagram for encoding the Ecore metamodel as a model conformant to Ecore

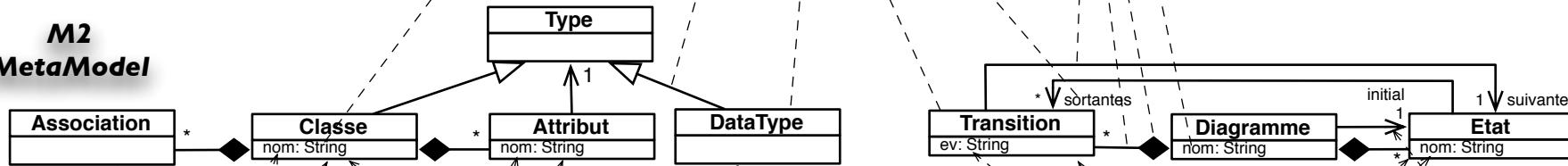




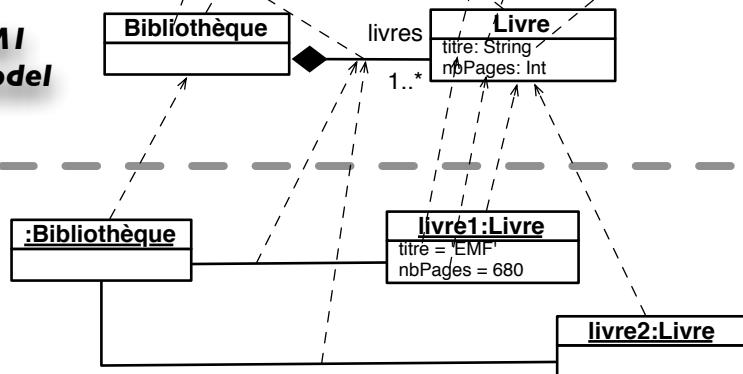
## M3 *MetaMetaModel*



M2  
MetaModel



MI  
Model



```
mandatory videooseq v1 "https://www.youtube.com/watch?v=PjNi1uYhV5w"  
optional videooseq v2 "v2Folder/v2.mp4"  
alternatives v3 {  
    videooseq v31 "v3/seq1.mp4"  
    videooseq v32 "v3/seq1.mp4"  
    videooseq v33 "v3/seq1.mp4"  
}  
  
alternatives v4 {  
    videooseq v41 "v4/seq1.mp4"  
    videooseq v42 "v4/seq1.mp4"  
}  
mandatory videooseq v5 "https://www.youtube.com/watch?v=ezKx-S0LiNQ"
```

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

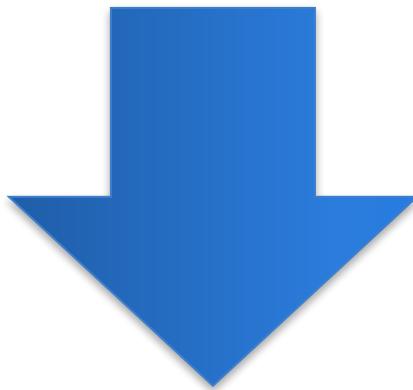
#2 How to  
transform  
models/  
programs?



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'
```

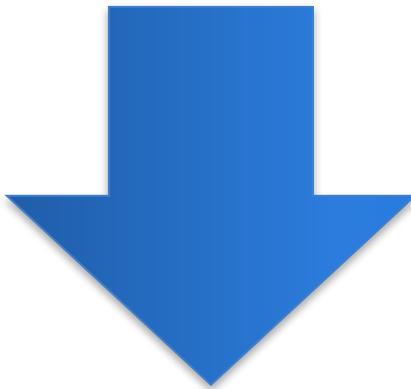
 FFmpeg

The FFmpeg logo, featuring a stylized green 'F' and 'M' followed by the word 'FFmpeg' in a bold black font.

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

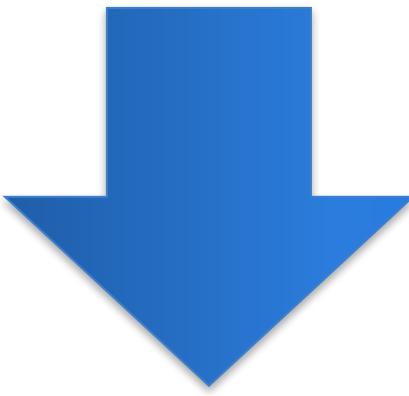
```
v3/seq1.avi
/path/to/video2.avi
/path/to/video3.avi
```



## 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

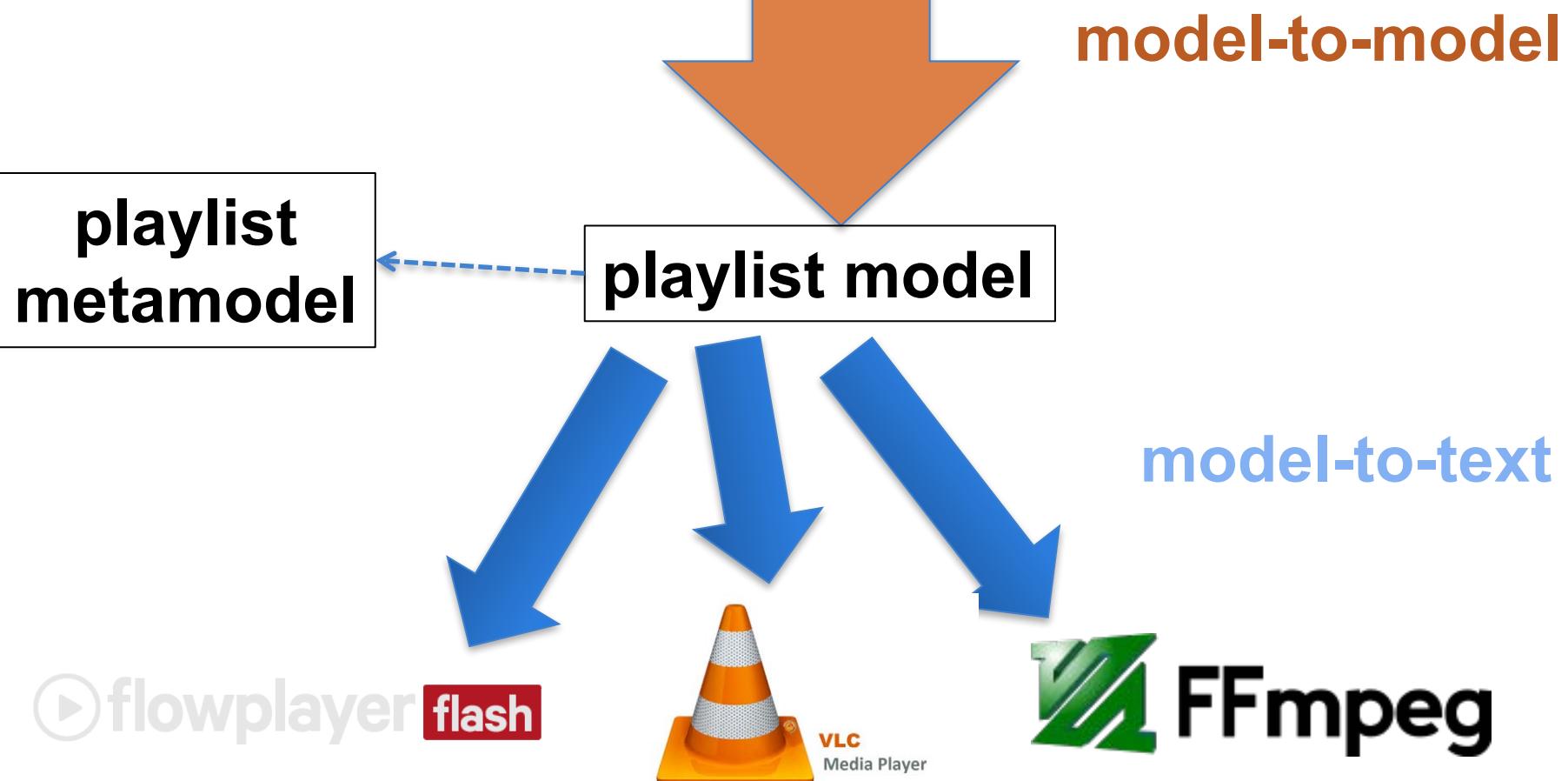
.m3u  
(extended)

```
#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

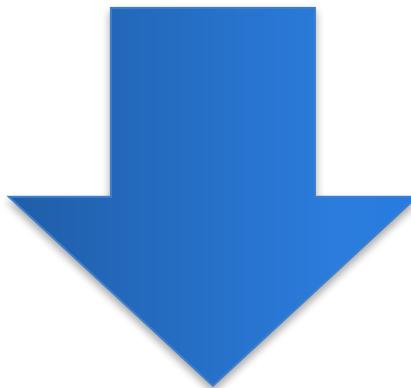
```
foo1.videogen &gt;
mandatory videoseq v1 "https://www.youtube.com/watch?v=PjNi1uYhV5w"
optional videoseq v2 "v2folder/v2.mp4"
alternatives v3 {
    videoseq v31 "v3/seq1.mp4"
    videoseq v32 "v3/seq1.mp4"
    videoseq v33 "v3/seq1.mp4"
}

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



## 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-\*



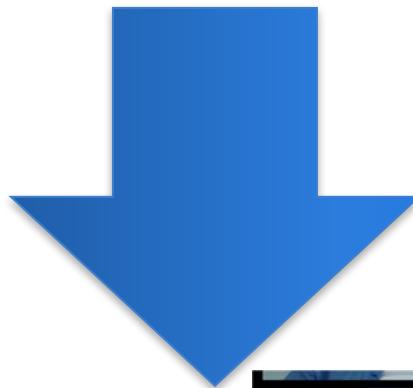
Thumbnails  
(vignettes) of  
each video  
sequence  
(e.g., PGN  
format)



## 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"
}

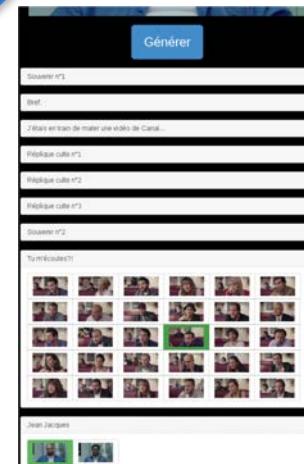
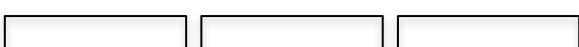
alternatives v4 {
    videoseq v41 "v4/seq1.mp4"
    videoseq v42 "v4/seq1.mp4"
}
mandatory videoseq v5 "https://www.youtube.com/watch?v=ezKx-S0LiNQ"
```



model-to-\*

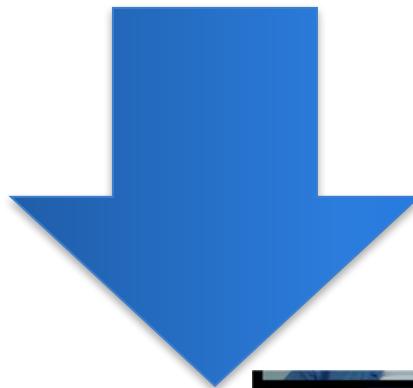


Thumbnails (vignettes) of each video sequence (e.g., PGN format)



## foo1.videogen

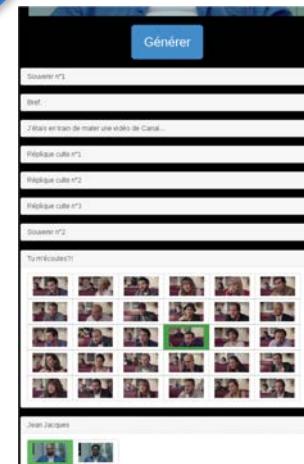
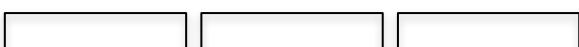
```
mandatory videoseq v1 "https://www.youtube.com/watch?v=PJNi1uYhV5w"  
optional videoseq v2 "v2Folder/v2.mp4"  
alternatives v3 {  
    videoseq v31 "v3/seq1.mp4"  
    videoseq v32 "v3/seq1.mp4"  
    videoseq v33 "v3/seq1.mp4"  
}  
  
alternatives v4 {  
    videoseq v41 "v4/seq1.mp4"  
    videoseq 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)

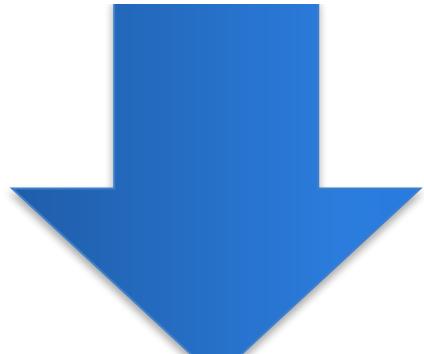


JS



foo1.videogen

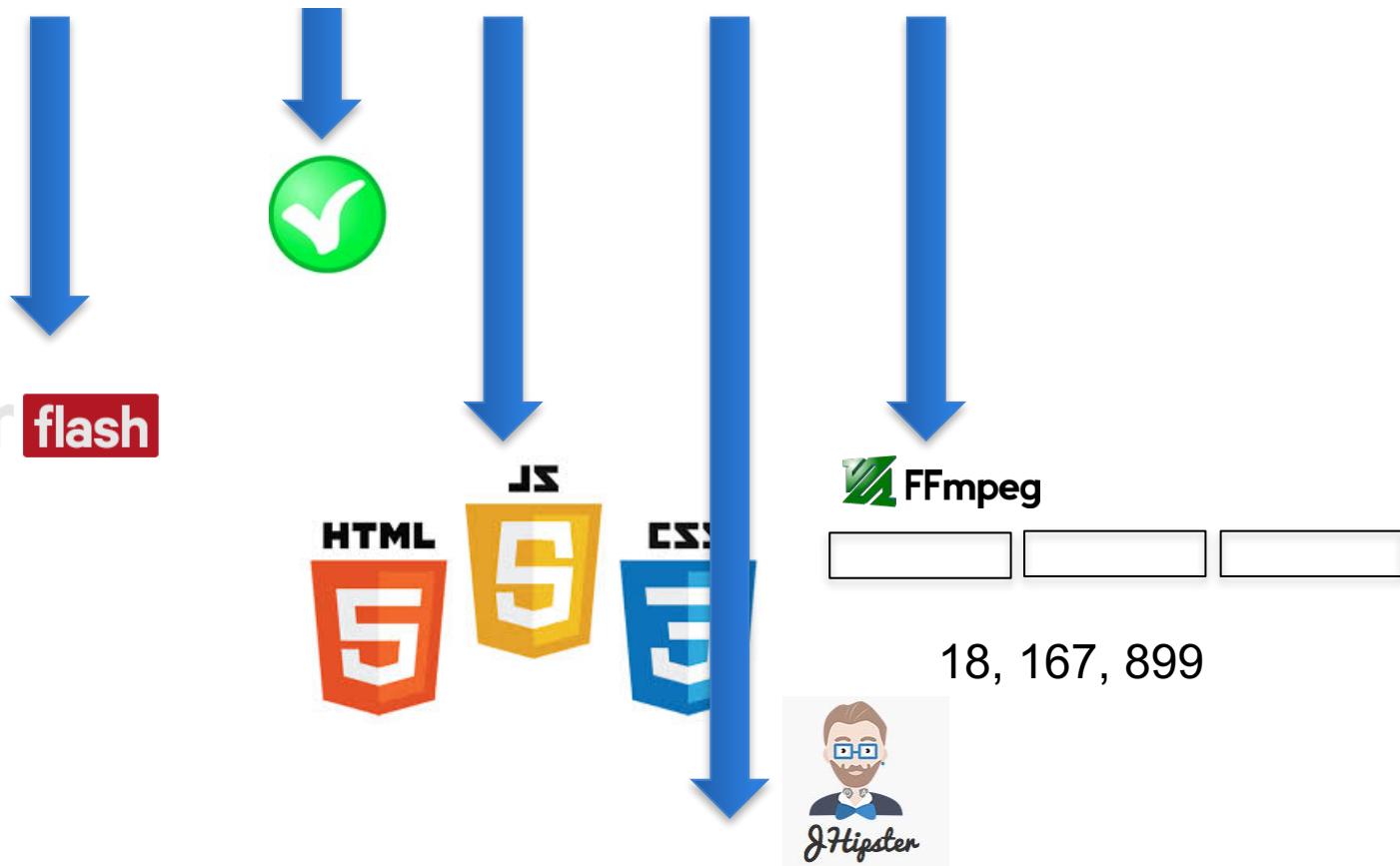
```
VideoGen {  
    mandatory videoseq v1 "V1/v1.mp4"  
    optional videoseq v2 "v2folder/v2.mp4" {  
        probability 25  
    }  
    alternatives v3 {  
        videoseq v31 "v3/seq1.mp4" {  
            duration 12  
            probability 25  
            description "a"  
        }  
        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.gvi"  
    alternatives v9 {  
        videoseq v81 "V81.gvi"  
    }  
}
```



## foo1.videoogen

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

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



## 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
- ...

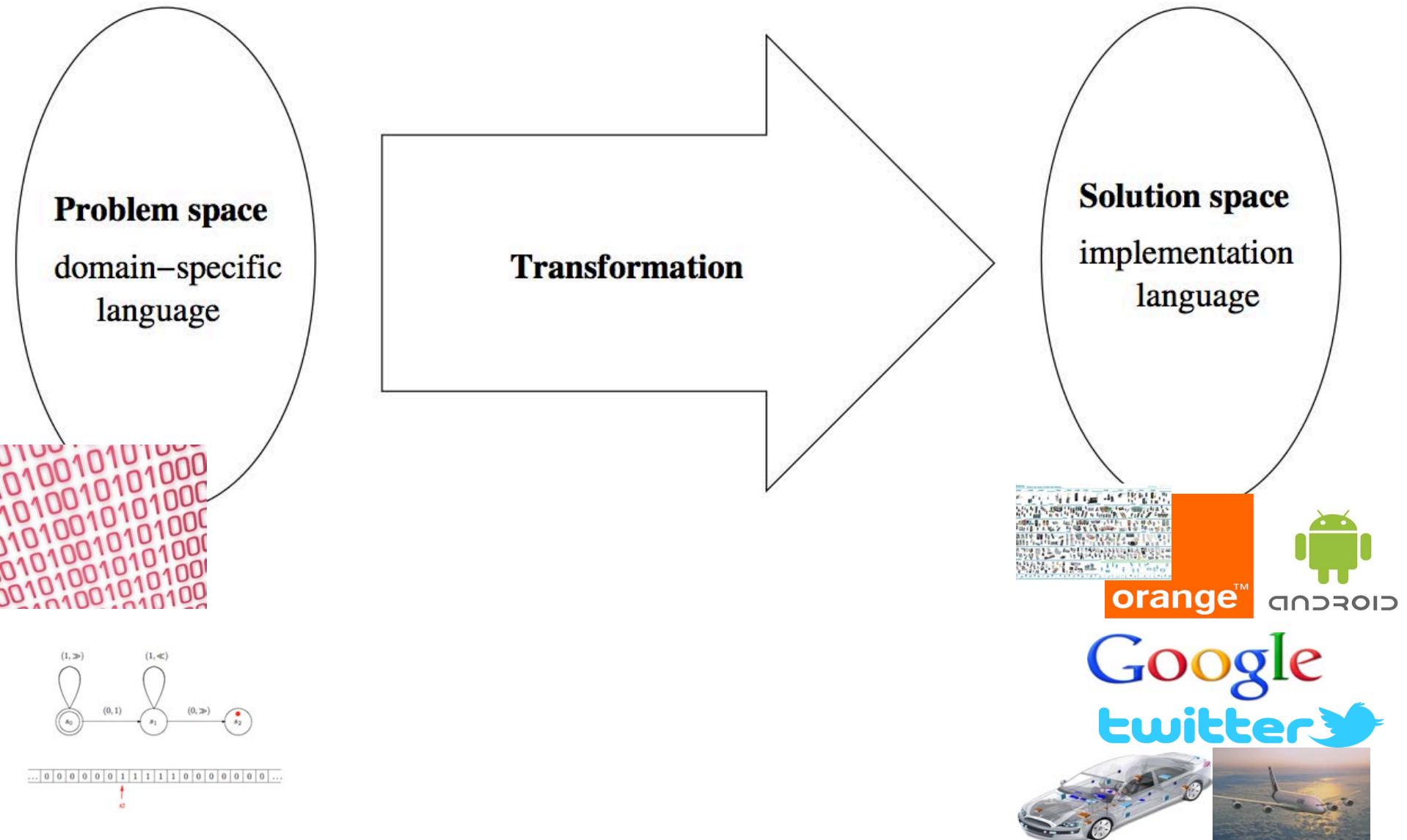


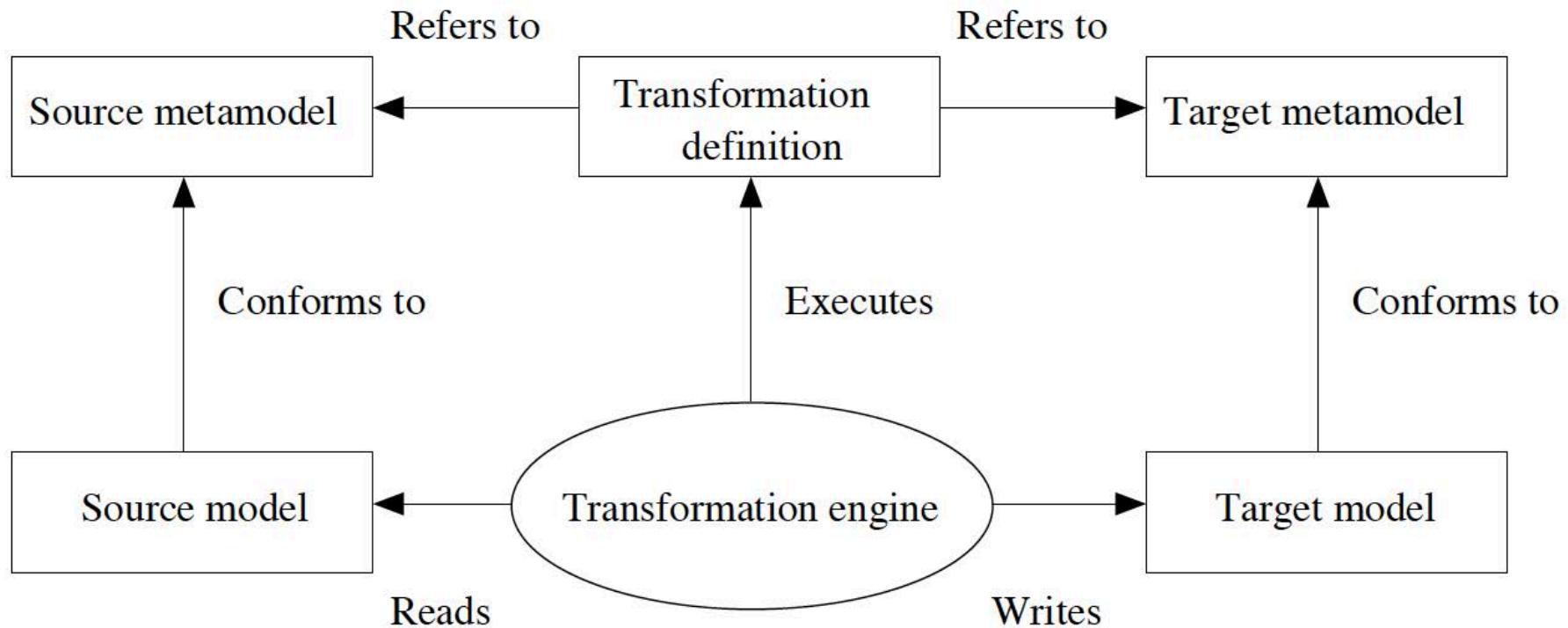
# Model Transformations

Taxonomy + Examples

# Abstraction Gap

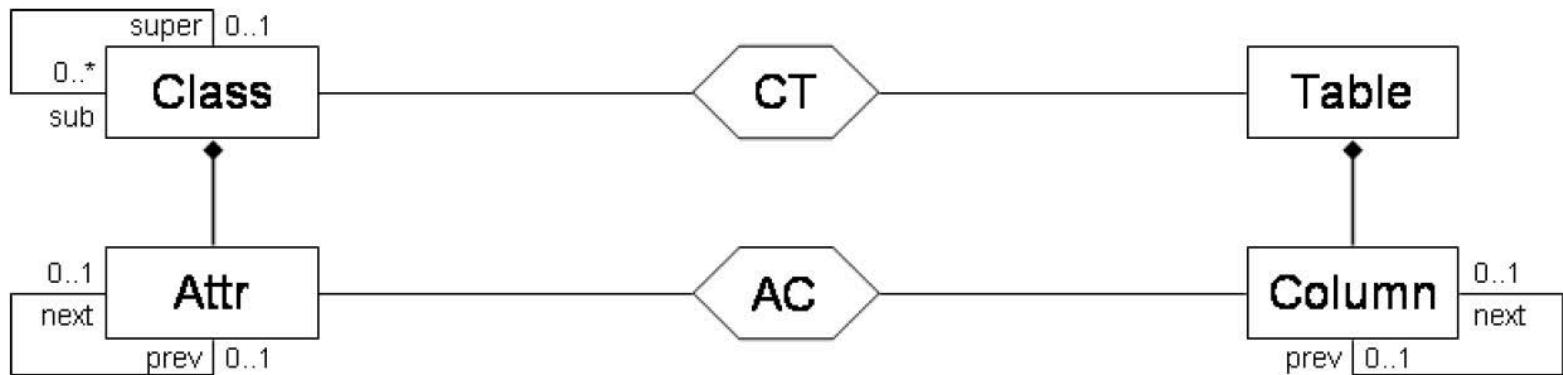
## Transformation is the key



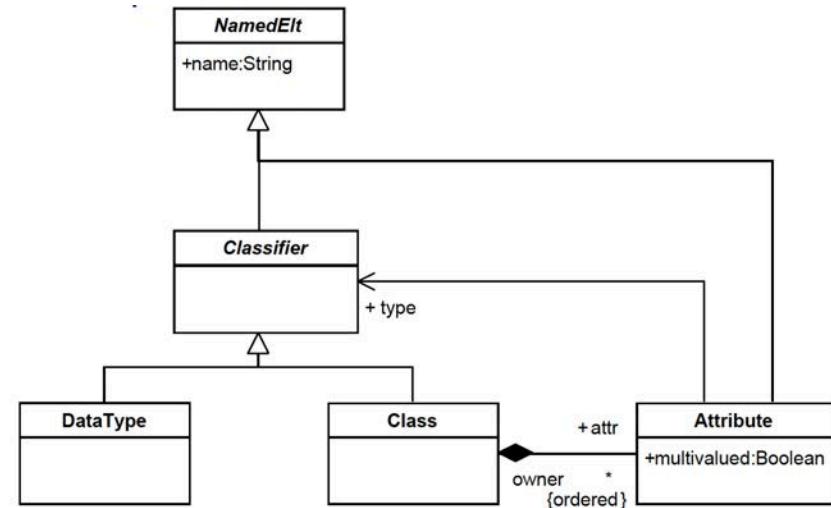
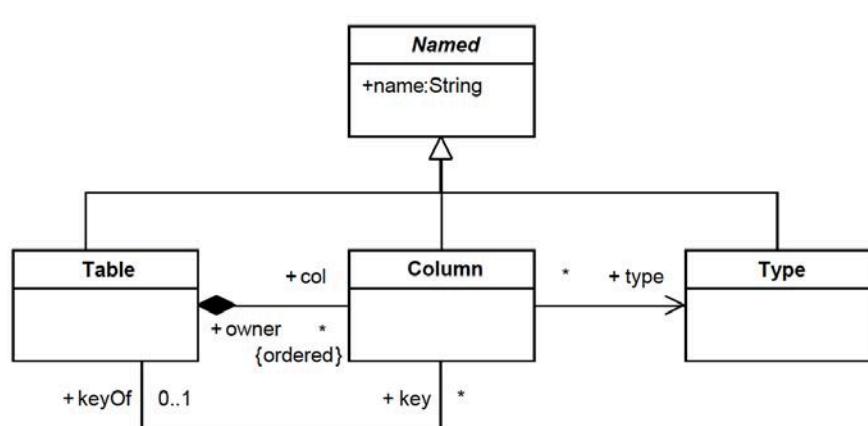


# Andy Schürr, Felix Klar “15 Years of Triple Graph Grammars.” ICGT 2008

(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

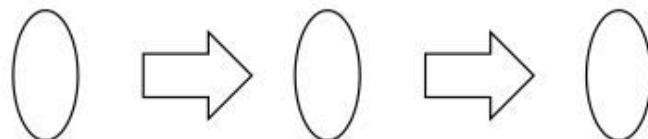
Krzysztof Czarnecki  
University of Waterloo  
Waterloo, Canada

Simon Helsen  
SAP AG  
Walldorf, Germany

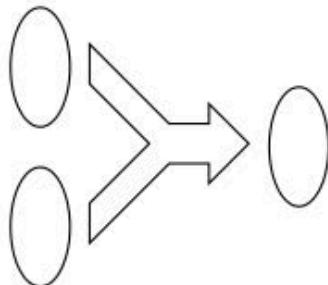
March 15, 2006

# Overview of Generative Software Development

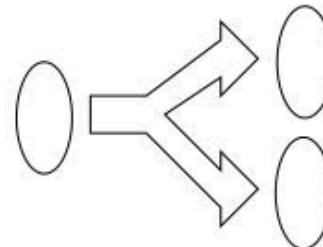
Krzysztof Czarnecki



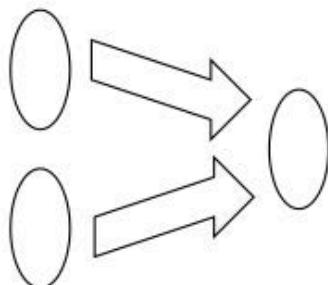
*a. Chaining of mappings*



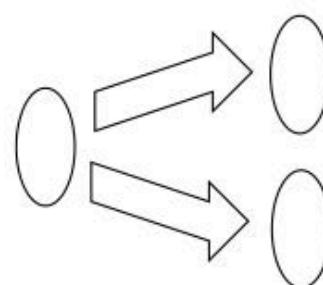
*b. Multiple problem spaces*



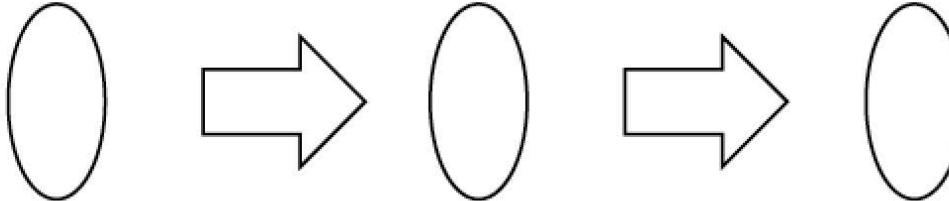
*c. Multiple solution spaces*



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



# One step/stage transformation hardly the case



a. *Chaining of mappings*

flowplayer flash

```
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"
}

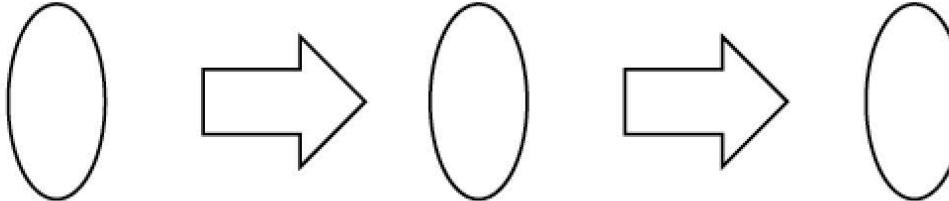
@alternatives v4 {
    videoseq v41 "v4/seq1.mp4"
    videoseq v42 "v4/seq1.mp4"
}
mandatory videoseq v5 "https://www.youtube.com/watch?v=ezKx-S0LiNQ"
```



```
#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_proce
ssed.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"
```



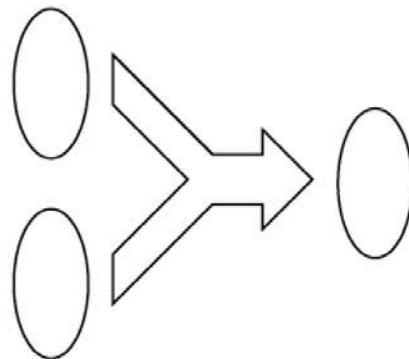
Playlist  
model

```
#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_proce
ssed.ts
```

.m3u (extended)

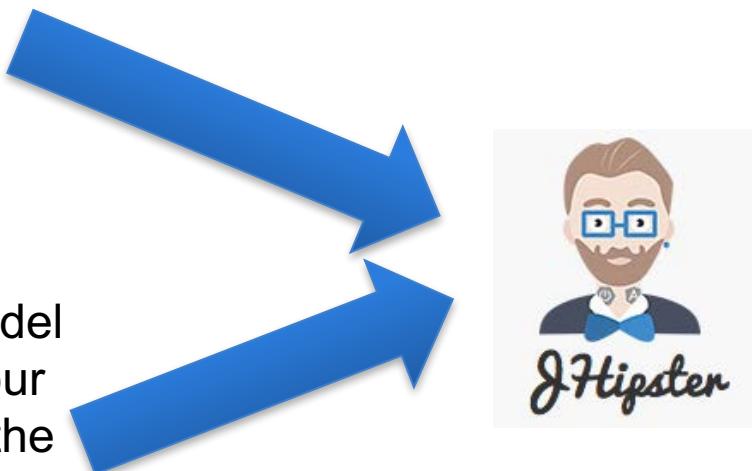
# Problem space

## Combination of expertises/aspects/DSLs



b. *Multiple problem spaces*

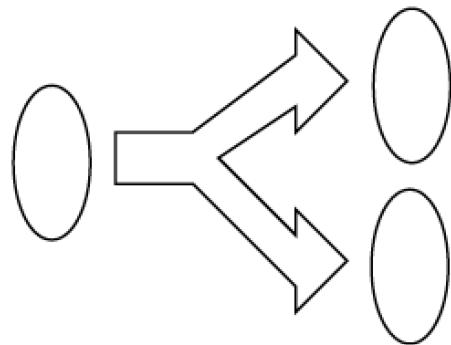
```
foo1.videogen ✘
mandatory videoseq v1 "https://www.youtube.com/watch?v=PJNi1uYhV5w"
optional videoseq v2 "v2folder/v2.mp4"
alternatives v3 {
    videoseq v31 "v3/seq1.mp4"
    videoseq v32 "v3/seq1.mp4"
    videoseq v33 "v3/seq1.mp4"
}
alternatives v4 {
    videoseq v41 "v4/seq1.mp4"
    videoseq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



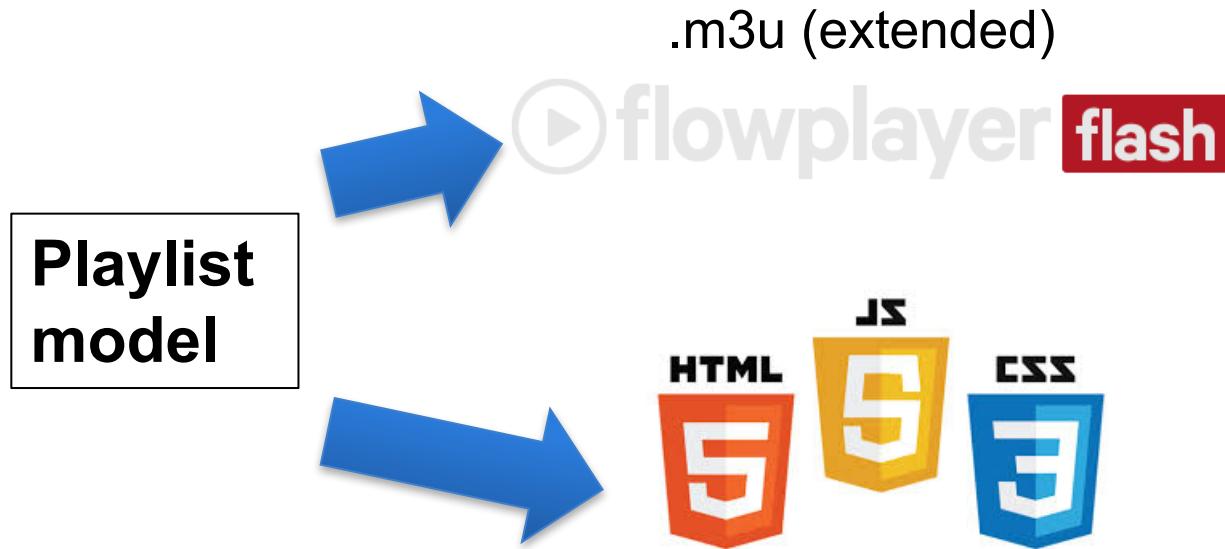
**Feature model:** another model for modeling “features” of your Web site (eg ability to save the video; mode=generation with frequencies)

# Solution space

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

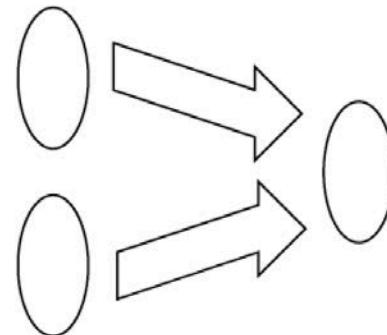


c. *Multiple solution spaces*



# Problem space

## e.g., different concrete syntaxes



d. Alternative problem spaces

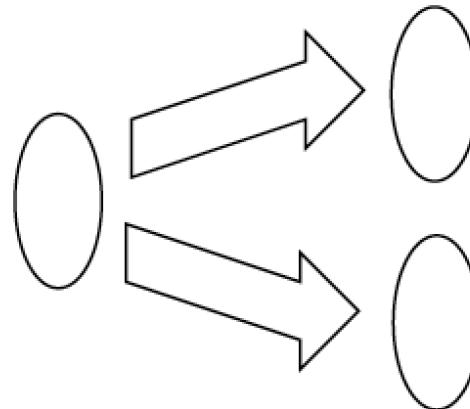
```
foo1.videogen ✘
mandatory videoseq v1 "https://www.youtube.com/watch?v=PJNi1uYhV5w"
optional videoseq v2 "v2folder/v2.mp4"
alternatives v3 {
    videoseq v31 "v3/seq1.mp4"
    videoseq v32 "v3/seq1.mp4"
    videoseq v33 "v3/seq1.mp4"
}
alternatives v4 {
    videoseq v41 "v4/seq1.mp4"
    videoseq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



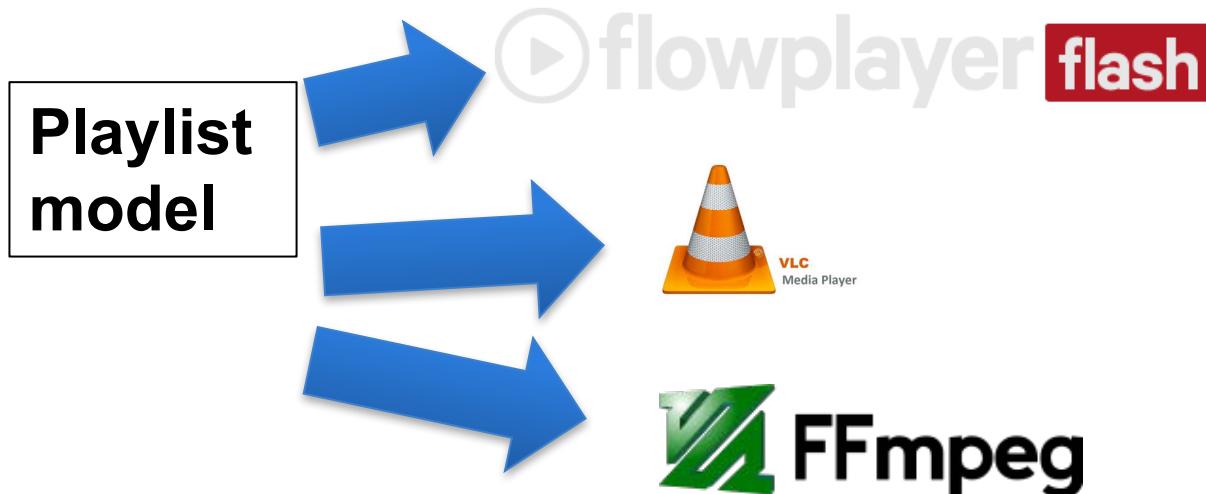
Feature Model

# Solution space

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



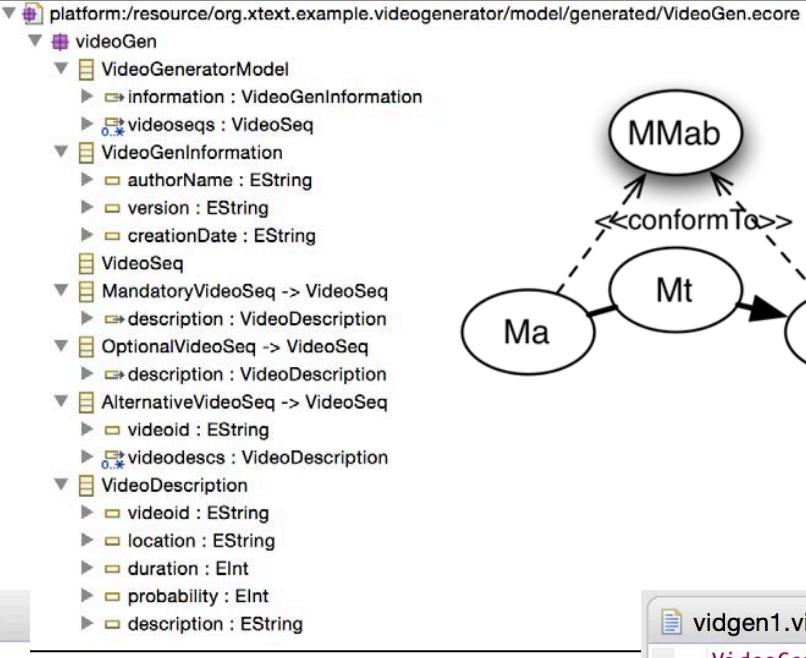
e. Alternative solution spaces



# 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>

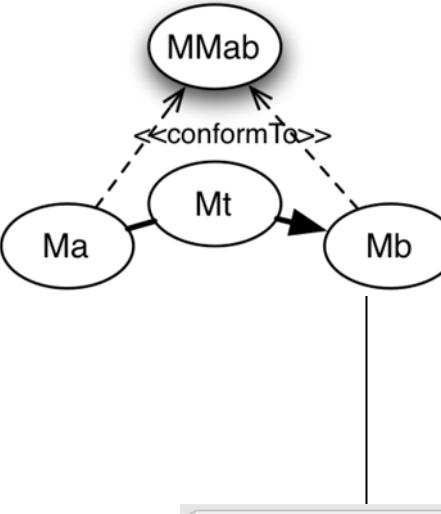
# Endogeneous Transformation



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

vidgen1.videogen

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



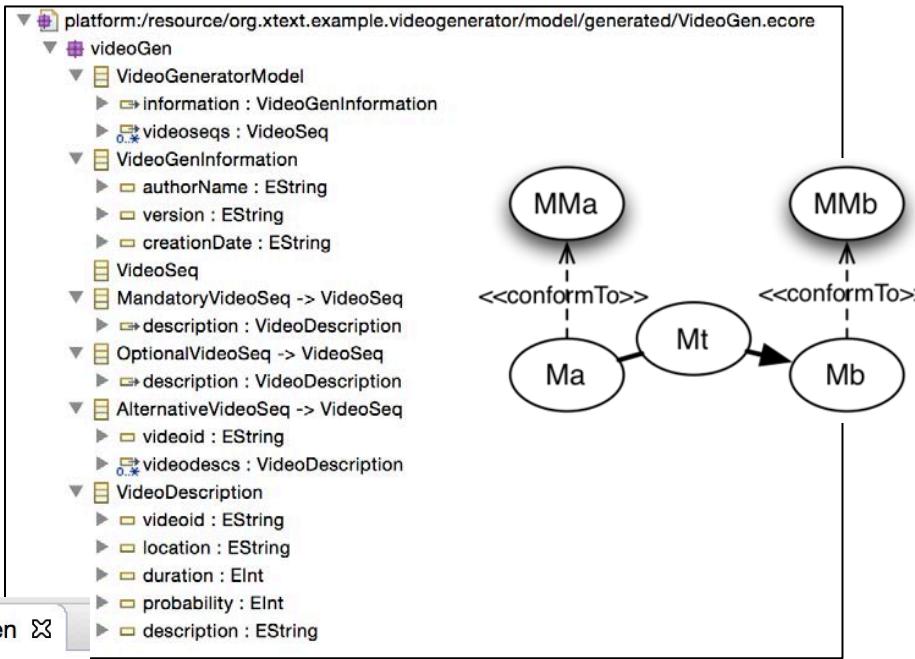
```
classDiagram  
  class MMab  
  class Ma  
  class Mt  
  class Mb  
  Ma --> Mt : <<conformTo>>  
  Mt --> Mb  
  Mb --> MMab
```

vidgen1.videogen

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

# Exogeneous Transformation

(metamodel)



```
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"
        }
]
```

```
<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>
```

# Horizontal Transformation

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

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

vidgen1.videogen

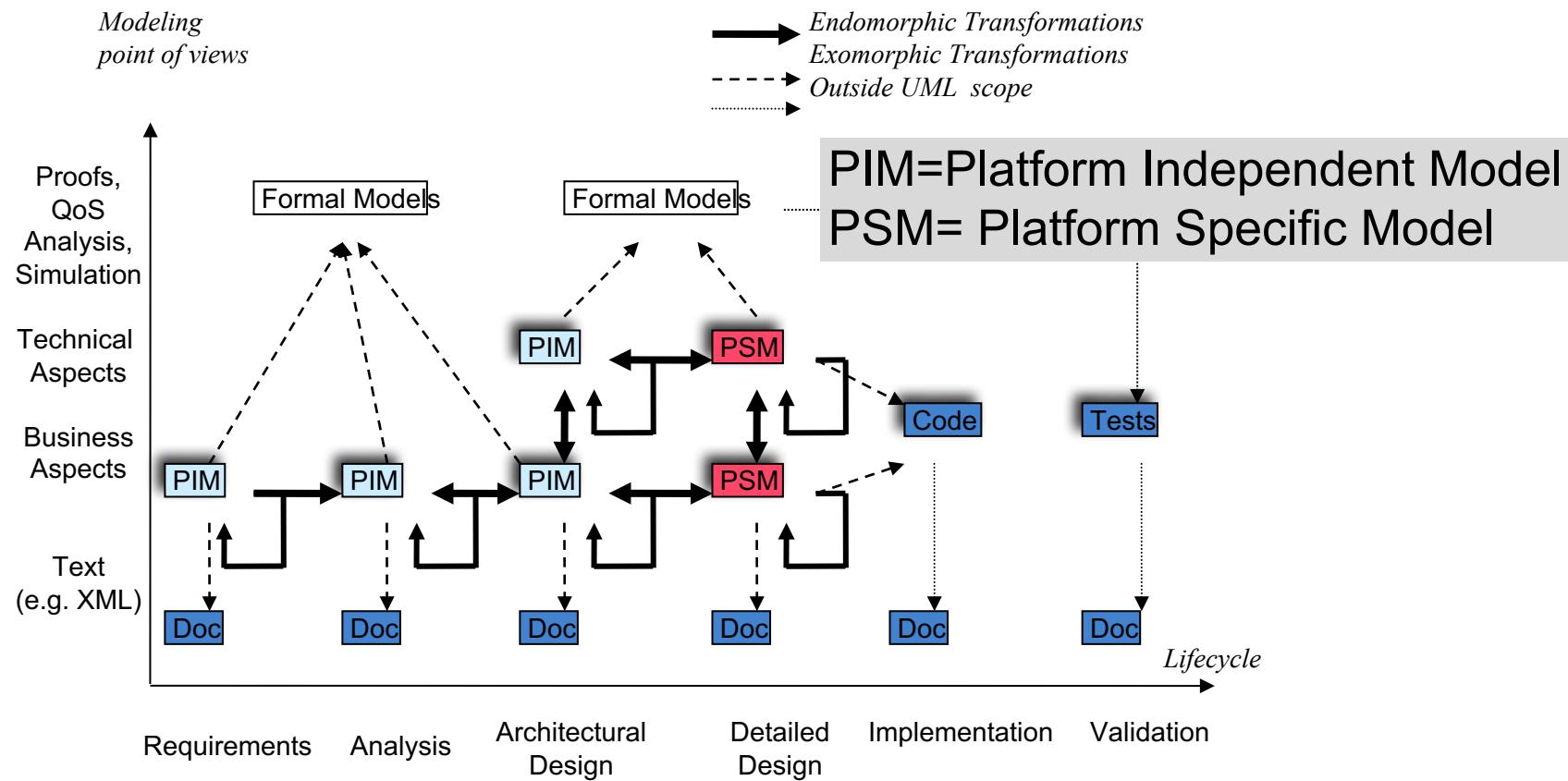
```
VideoGen {  
    mandatory videoseq "V1/v1.mp4"  
    optional videoseq "v2folder/v2.mp4" {  
        probability 25  
    }  
    alternatives vid3 {  
        videoseq "v3/seq1.mp4"  
        videoseq vid31 "v3/seq2.mp4"  
        videoseq vid32 "v3/seq3.mp4"  
    }  
    alternatives vid4 {  
        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

```
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"  
}
```

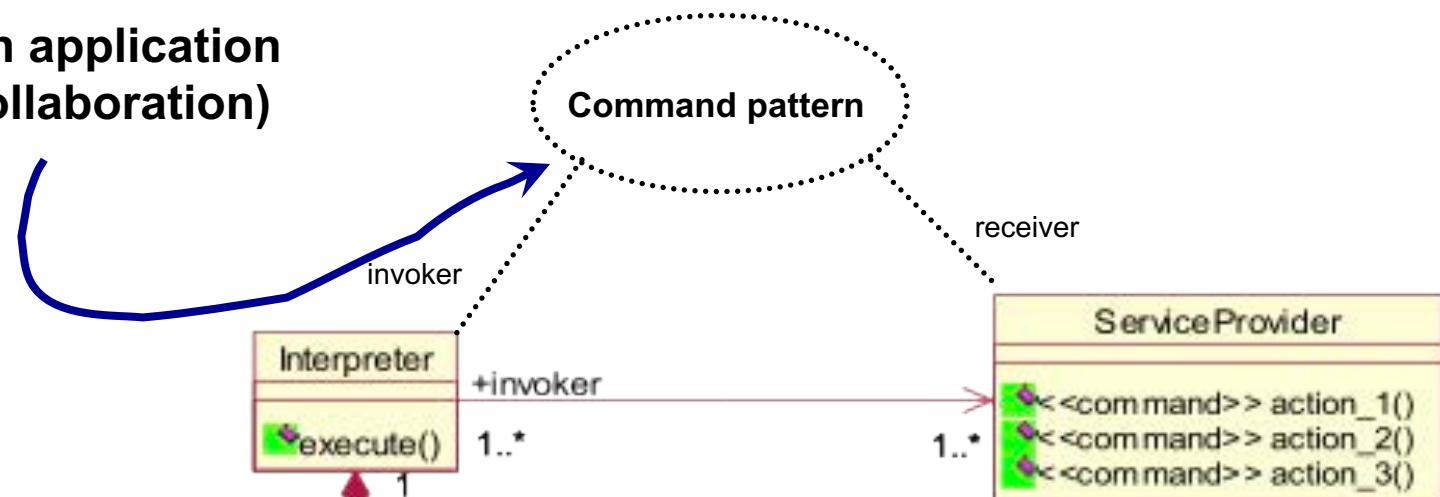
# Chaining of (Vertical) Transformations

## Back to the first courses

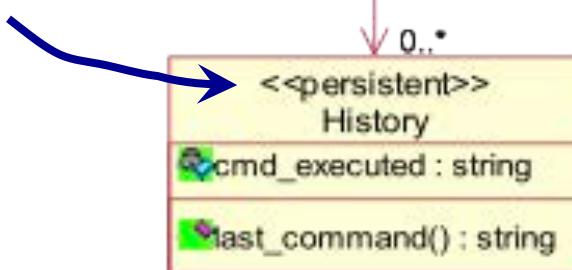


# Embedding implicit semantics into a model

Design pattern application  
(parametric collaboration)

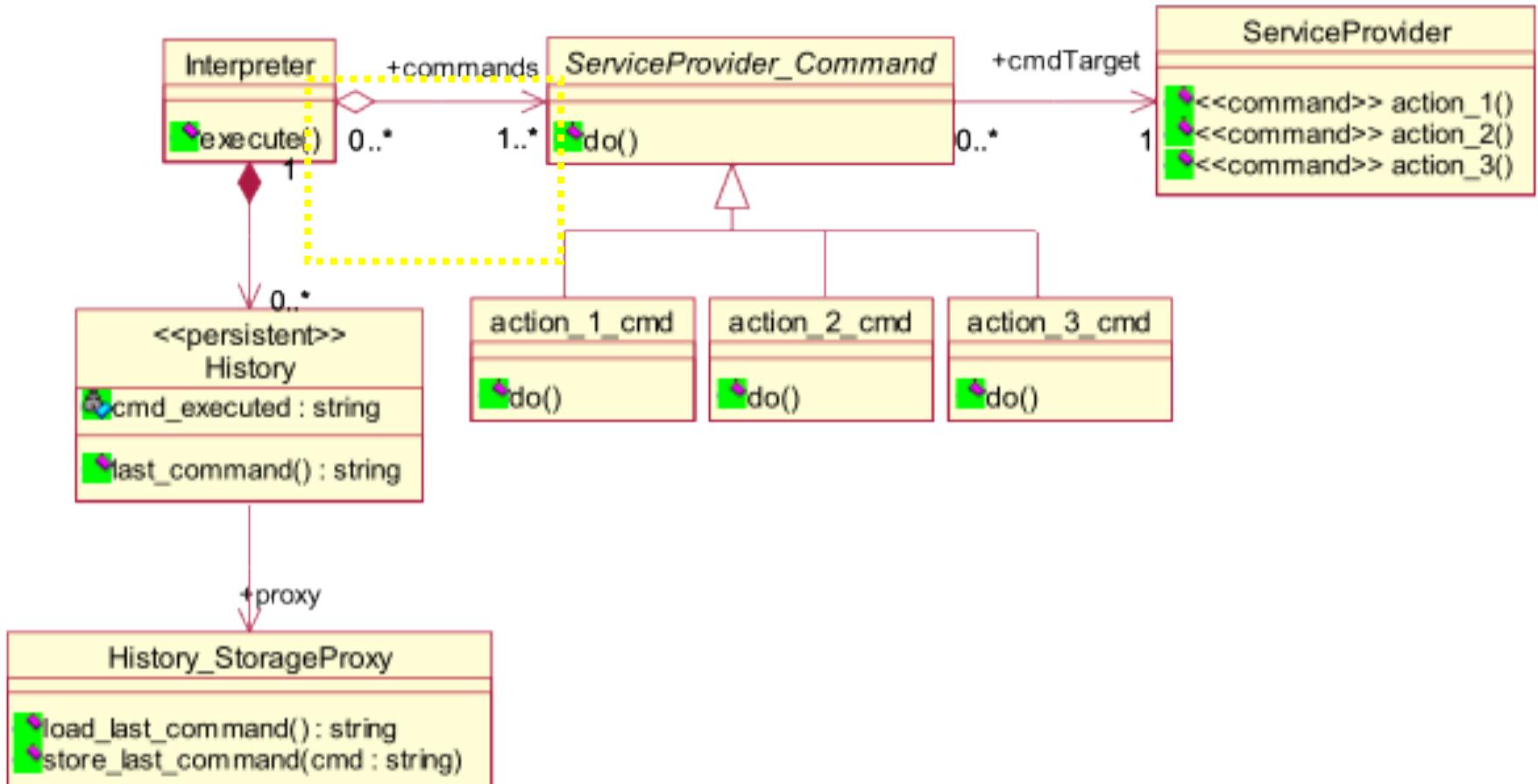


Element stereotype

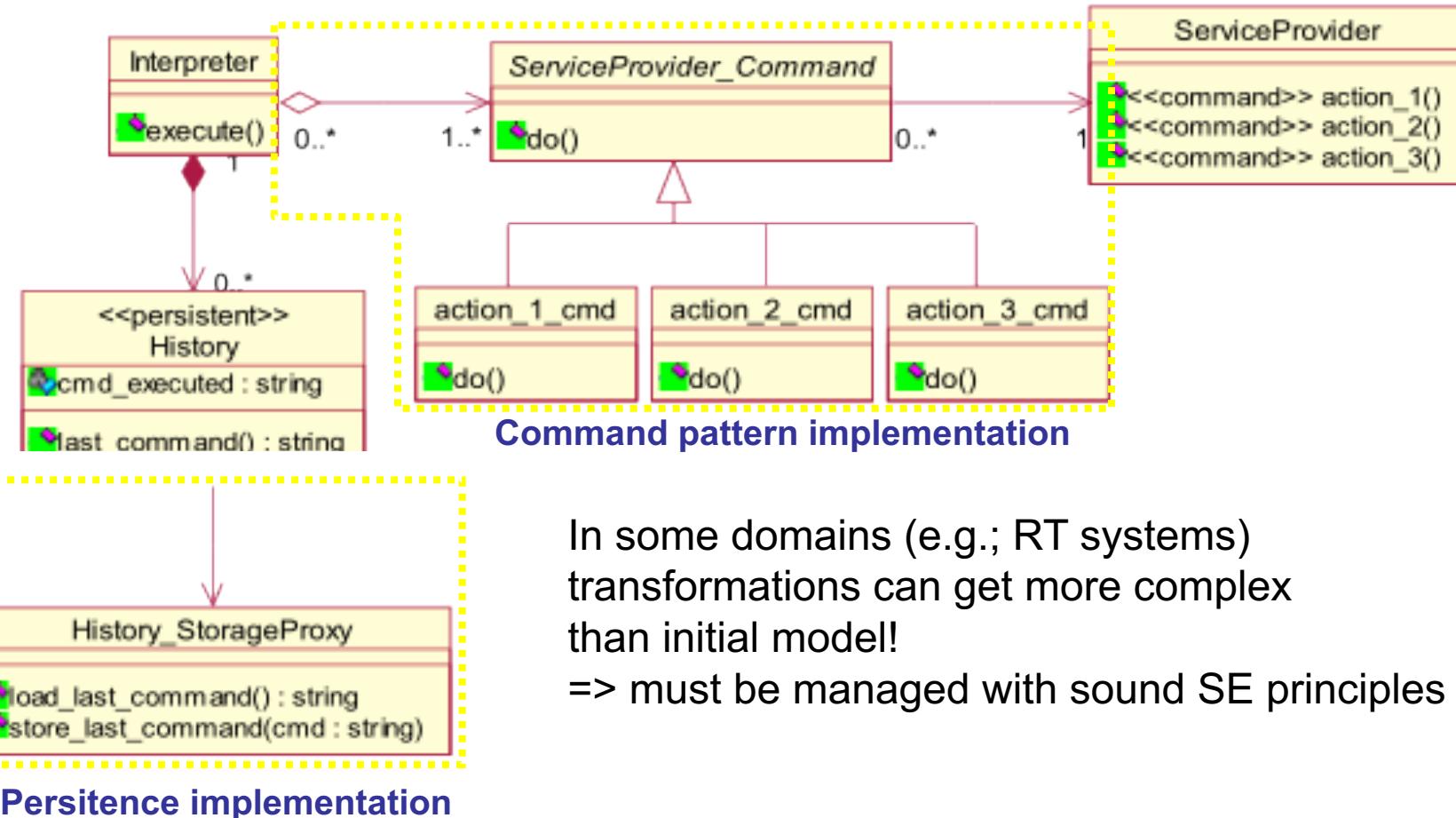


...and also  
Tagged values  
& Contracts

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



# How To: Automatic Model Transformations



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

Persitence implementation

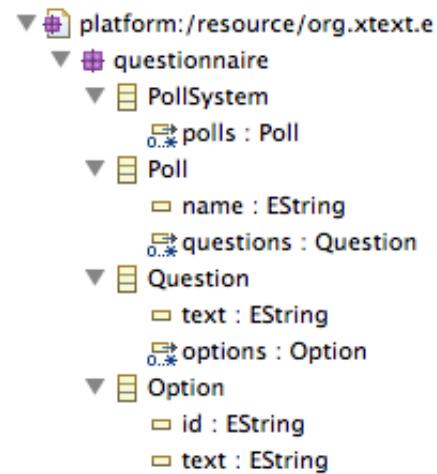
# Quizz Time

Characterize the following model transformations

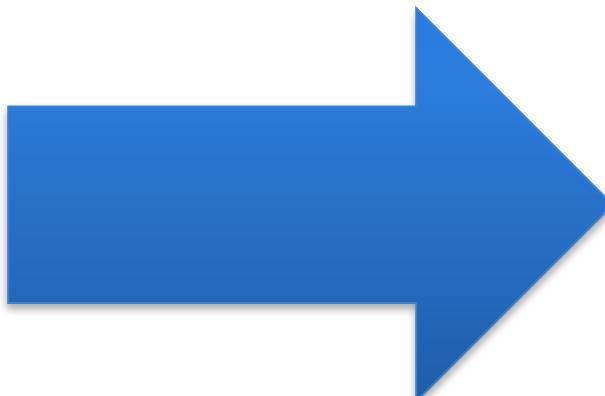
Endogeneous? Exogeneous?

Vertical? Horizontal?

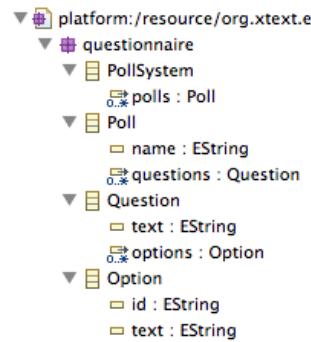
Model-to-text? Model-to-Model?



```
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

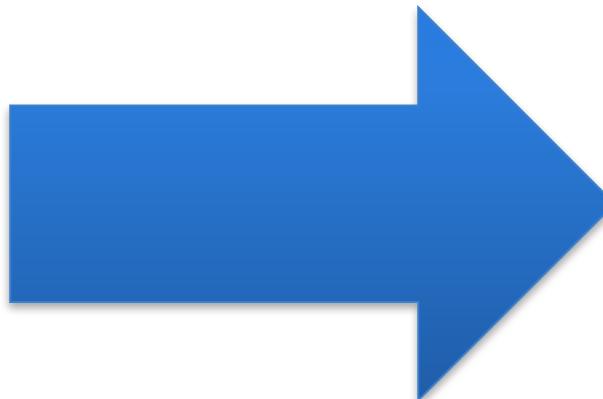


foo.q

```

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"
        }
    }
}
  
```



## poll1

**What is A ?**

- B
- C
- D

## poll2

**What is D ?**

- E
- F

# Quizz Time

Think about the different artefacts generated by Xtext.

**What are the different transformations implemented in Xtext? Characterize them.**

# Model Transformation in Java/Xtend

# 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"
    }
}

```

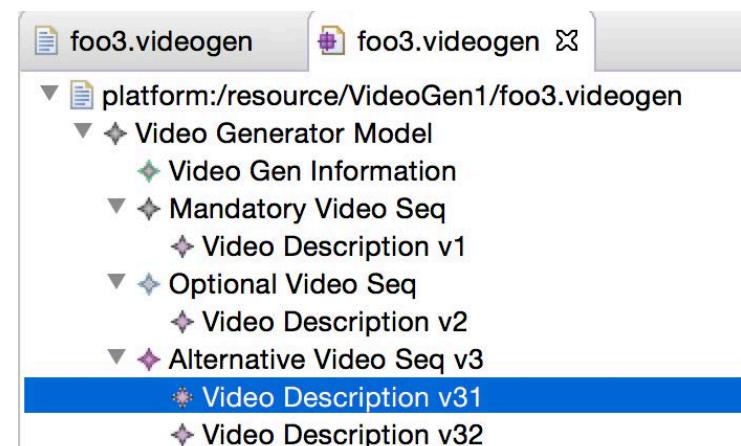
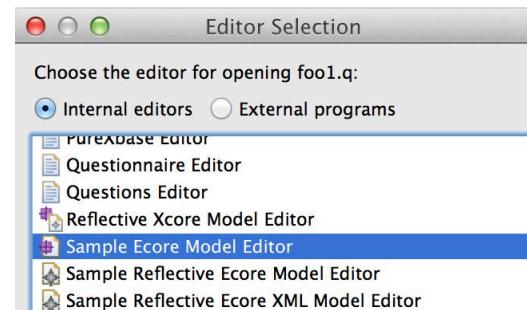
platform:/resource/org.xtext.example.videogenerator/model

```

videoGen
└─ VideoGeneratorModel
    └─ VideoGenInformation
        └─ VideoSeq
            └─ MandatoryVideoSeq -> VideoSeq
                └─ description : VideoDescription
            └─ OptionalVideoSeq -> VideoSeq
                └─ description : VideoDescription
            └─ AlternativeVideoSeq -> VideoSeq
                └─ videoid : EString
                └─ videodescs : VideoDescription
        └─ VideoDescription
            └─ videoid : EString
            └─ location : EString
            └─ duration : Elnt
            └─ probability : Elnt
            └─ description : EString

```

# Loading Models (1)

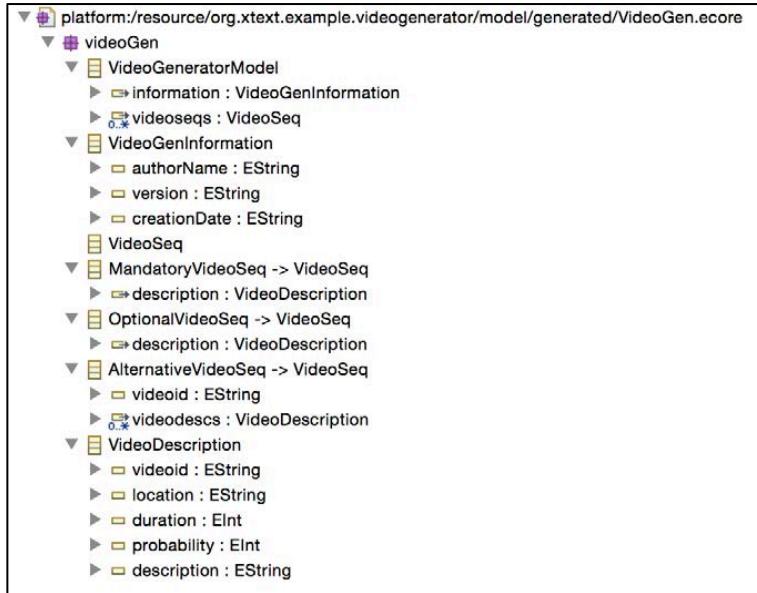


Properties	
Property	Value
Description	
Duration	0
Location	v31.mp4
Probability	0
Videoid	v31

```
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"  
    }  
}
```

# Loading Models (2)



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

The screenshot shows a modeling environment. On the left, there is an Xtext editor with the following code:

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

On the right, there is a UML diagram for a `Video Generator Model` with nodes for `Video Gen Information`, `Mandatory Video Seq`, `Optional Video Seq`, and `Alternative Video Seq`. The `Mandatory Video Seq` node is expanded, showing `Video Description v1`. The `Optional Video Seq` node is expanded, showing `Video Description v2`. The `Alternative Video Seq` node is expanded, showing `Video Description v31` and `Video Description v32`. Below the diagram is a `Properties` table:

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

```
<?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
}
```

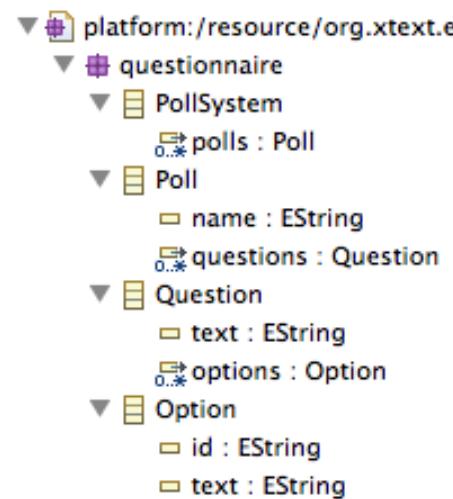
```
def savePollSystem(URI uri, PollSystem pollS) {
    var Resource rs = new ResourceSetImpl().createResource(uri);
    rs.getContents.add(pollS);
    rs.save(new HashMap());
}
```

```
@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)
}
```



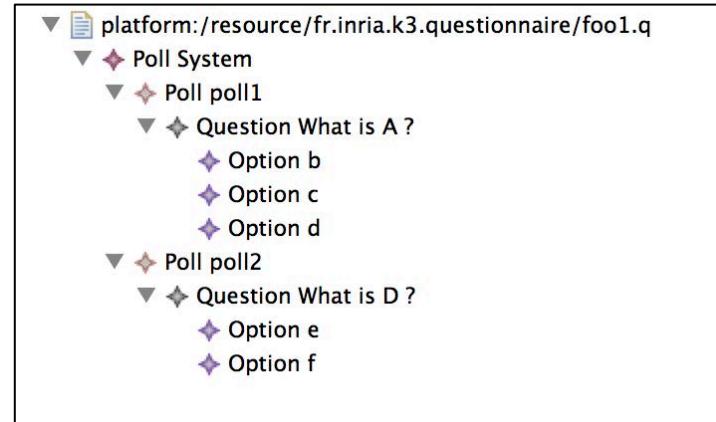
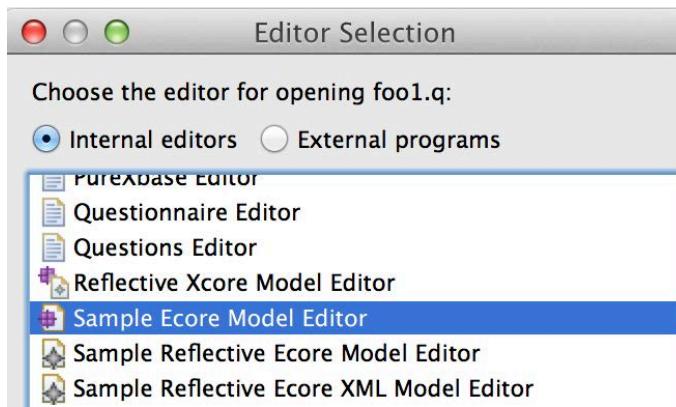
# Loading Models (1)

The screenshot illustrates the process of loading a PollSystem model named `foo1.q`.

**Model Structure:**

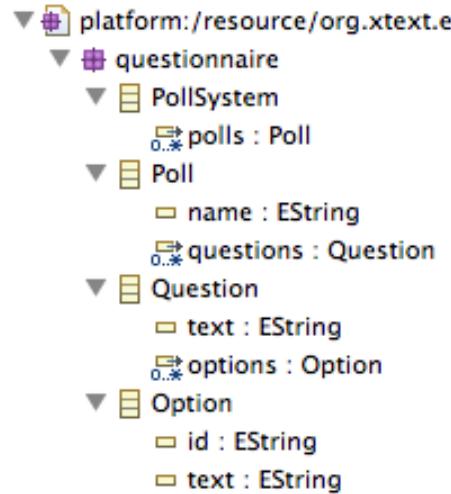
- `platform:/resource/org.xtext.e`
  - `questionnaire`
    - `PollSystem`
      - `polls : Poll`
    - `Poll`
      - `name : EString`
      - `questions : Question`
    - `Question`
      - `text : EString`
      - `options : Option`
    - `Option`
      - `id : EString`
      - `text : EString`

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

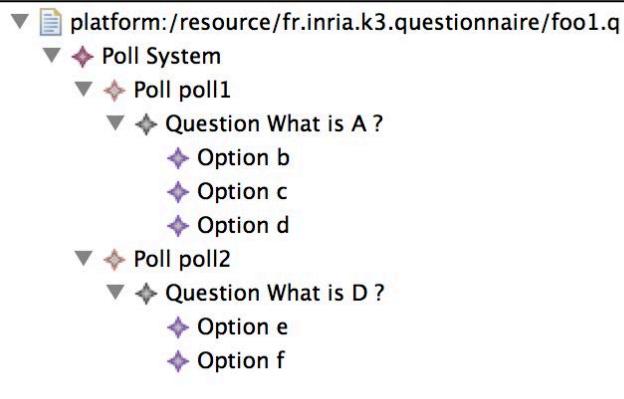


# Loading Models (2)

```
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"
        }
    }
}
```

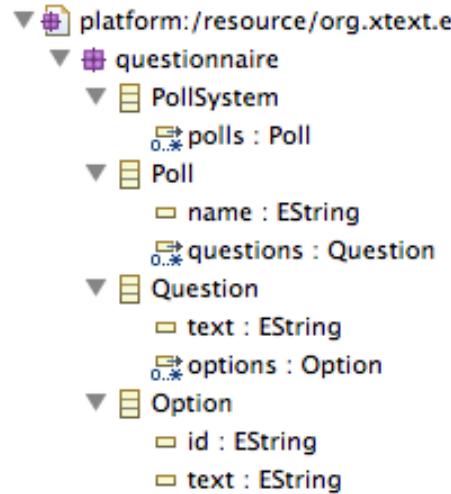


XMI



```
<?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/mydata">
<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)



```
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"
        }
    }
}
```

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

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

# Meta(models) and Java

The screenshot shows a code editor with two panes. The left pane displays the Xtext metamodel structure under the path `platform:/resource/org.xtext.e`. The right pane shows the corresponding generated Java code.

**Metamodel Structure (Left Pane):**

- questionnaire**
  - PollSystem**
    - polls** : Poll (0..\*)
  - Poll**
    - name** : EString
    - questions** : Question (0..\*)
  - Question**
    - text** : EString
    - options** : Option (0..\*)
  - Option**
    - id** : EString
    - text** : EString

**Generated Java Code (Right Pane):**

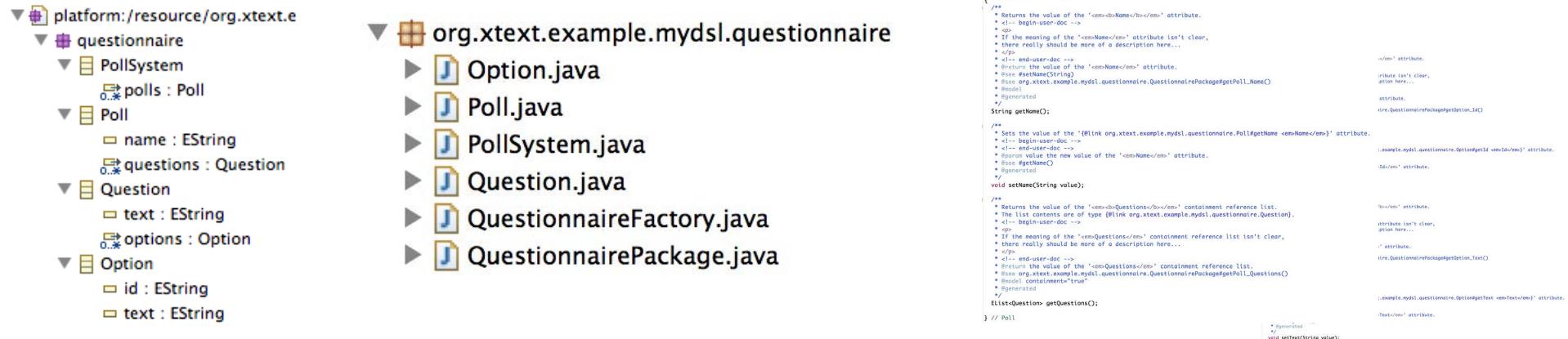
```
public interface Poll extends EObject
{
    /**
     * Returns the value of the '<em><b>Name</b></em>' attribute.
     * <!-- begin-user-doc -->
     * <p>
     * If the meaning of the '<em>Name</em>' attribute isn't clear,
     * there really should be more of a description here...
     * </p>
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

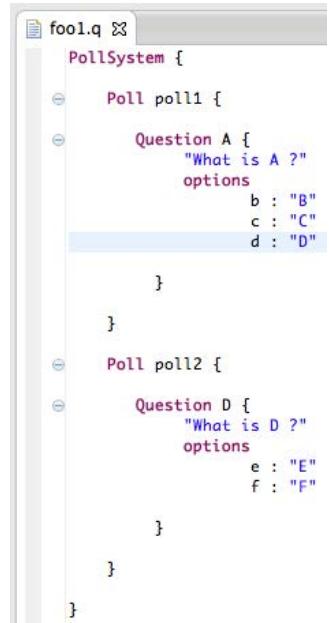
# Meta(models) and Java



The screenshot shows the generated Java code for the Poll class. The code is annotated with comments explaining the generated code based on the Xtext metamodel. The annotations include descriptions for the getName() and getQuestions() methods, as well as the setNome() method.

```
public interface Poll extends EObject {
    /**
     * Returns the value of the '<em><bn>Name</bn></em>' attribute.
     * <!-- begin-user-doc -->
     * <!-- The meaning of the '<em><bn>Name</bn></em>' attribute isn't clear,
     * there really should be more of a description here...
     * <!-- end-user-doc -->
     * @return the value of the '<em><bn>Name</bn></em>' attribute.
     */
    String getName();
    /**
     * Sets the value of the '<em><bn>Name</bn></em>' attribute.
     * <!-- begin-user-doc -->
     * <!-- end-user-doc -->
     * @param value the new value of the '<em><bn>Name</bn></em>' attribute.
     */
    void setName(String value);
    /**
     * Returns the value of the '<em><bn>Questions</bn></em>' containment reference list.
     * The list contents are of type {@link org.xtext.example.mydsl.questionnaire.Question}.
     * <!-- begin-user-doc -->
     * <!-- The meaning of the '<em><bn>Questions</bn></em>' containment reference list isn't clear,
     * there really should be more of a description here...
     * <!-- end-user-doc -->
     * @return the value of the '<em><bn>Questions</bn></em>' containment reference list.
     */
    EList<Question> getQuestions();
}
// Poll
```

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



```
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"
        }
    }
}
```



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



ECLIPSE MODELING FRAMEWORK

Ecore Model

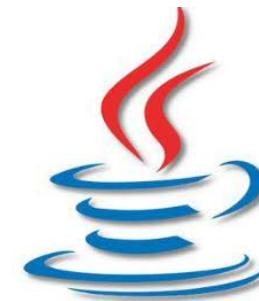
EPackage

EClass

EAttribute

EReference

Code generation



Java Code

Package

Class

Attribute

Reference

```
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"
        }

    }

}

}
```

```
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"
        }

    }

}

}


```

```
def loadPollSystem(URL uri) {
    new QuestionnaireStandaloneSetupGenerated().createInjectorAndDoEMFRegistration()
    var res = new ResourceSetImpl().getResources(uri, true);
    res.getContents().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 + ".poll1" }
    // serializing
    savePollSystem(URI.createURI("foo2.q"), polls)
}
```

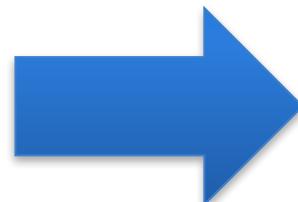
## Questionnaire MM (ecore)

## 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
}
```

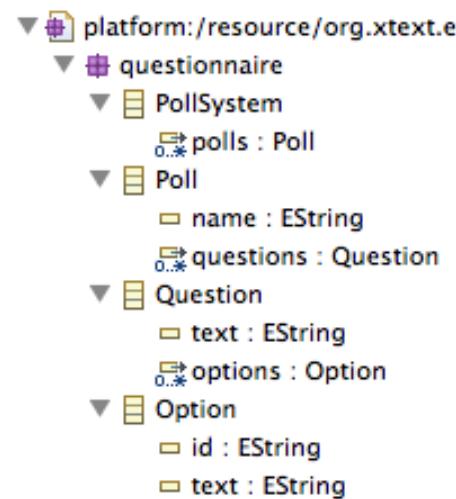
```
def savePollSystem(URI uri, PollSystem pollS) {
    var Resource rs = new ResourceSetImpl().createResource(uri);
    rs.getContents.add(pollS);
    rs.save(new HashMap());
}
```

```
@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"))

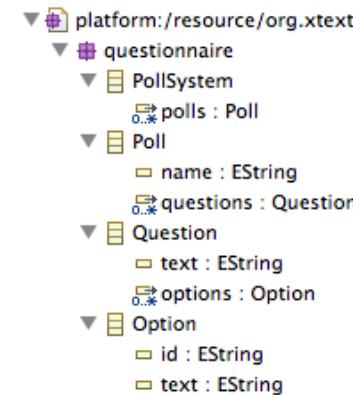
    // MODEL MANAGEMENT (ANALYSIS, TRANSFORMATION)
    var html = toPolls(pollS.polls)
    assertNotNull(html)

    // 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 ?**

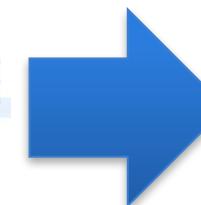
- B
- C
- D

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

# Quizz: 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
    ;

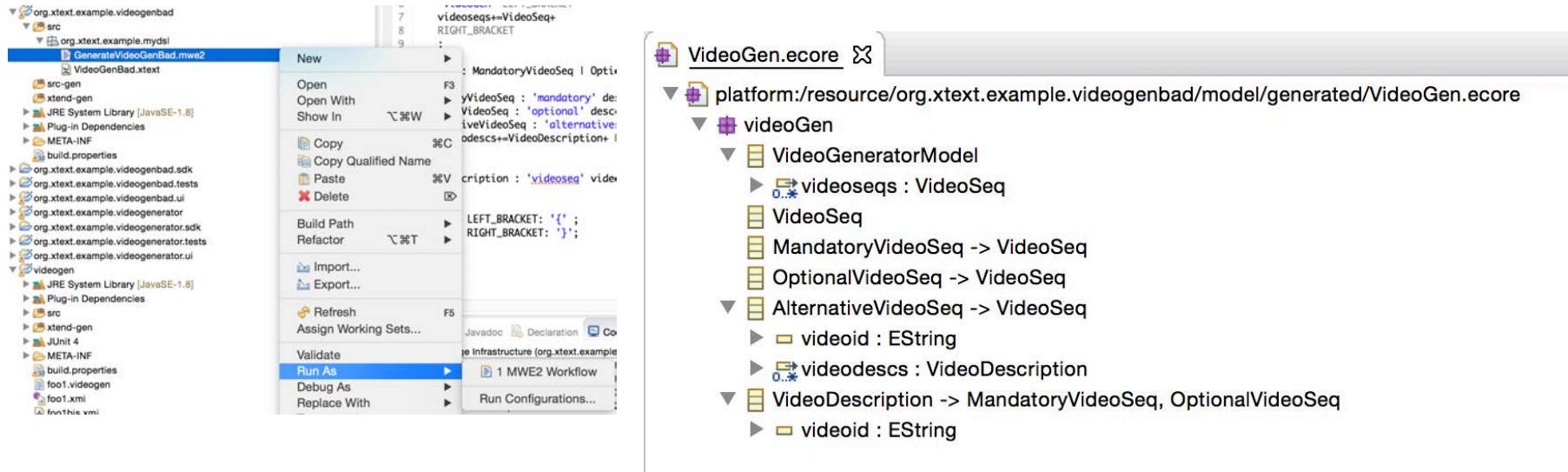
VideoSeq: MandatoryVideoSeq | OptionalVideoSeq | AlternativeVideoSeq ;

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
;

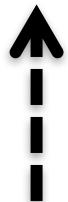
VideoSeq: MandatoryVideoSeq | OptionalVideoSeq | AlternativeVideoSeq ;

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: '}';

```

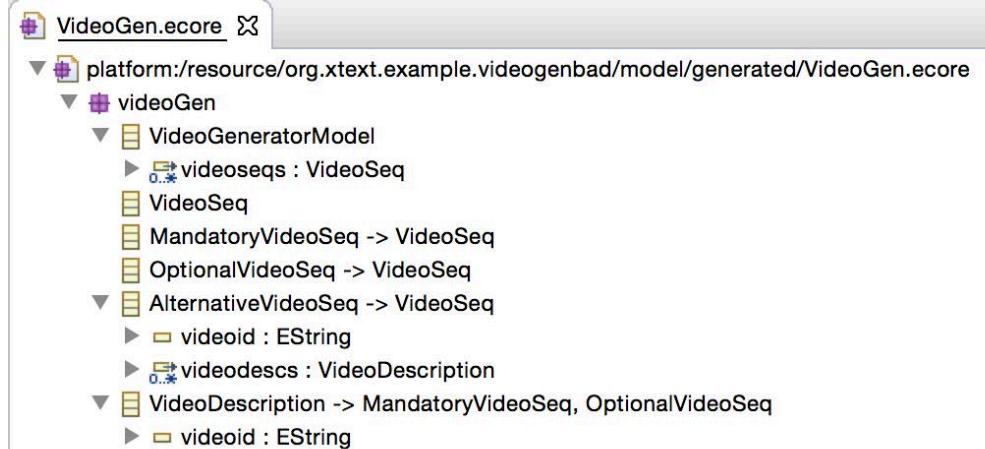


**foo1.videogenbad**

```

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

```



**foo1.videogenbad**

```

Video Generator Model
    * Video Description v1
    Video Description v2

```

**Properties**

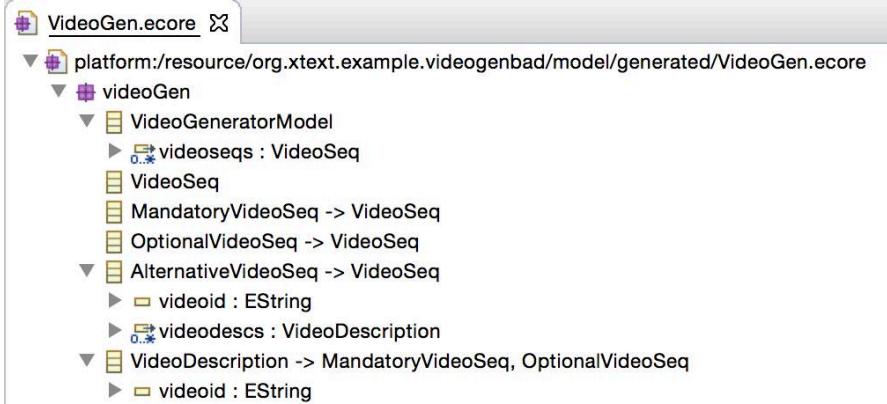
Property	Value
Videoid	v1

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
            // ...
        }
    }
}

```

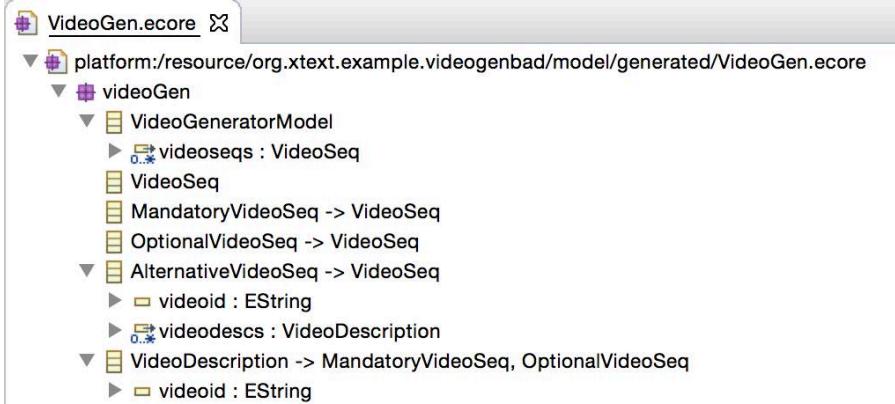
The method `description` is undefined....

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
```

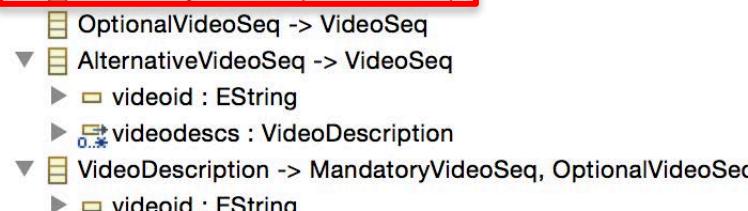
```
;
```

```
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

        ► videoseqs : VideoSeq  
            0..\*

    ▼ VideoSeq

        ▼ MandatoryVideoSeq -> VideoSeq

            ► description : VideoDescription

            ▼ OptionalVideoSeq -> VideoSeq

        ▼ AlternativeVideoSeq -> VideoSeq

            ► videoid : EString

            ► videodescs : VideoDescription  
                0..\*

        ▼ 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 <b>Description</b>}</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 <b>Description</b>}' containment reference.
42 * <!-- begin-user-doc -->
43 * <!-- end-user-doc -->
44 * @param value the new value of the 'Description' containment reference.
45 * @see #getDescription()
46 * @generated
47 */
48 void setDescription(VideoDescription value);
49
50 } // 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: behind the magic (Xtext+MDE)
  - Xtend + Xtext (breathing life into DSLs)
  - @Aspect annotation

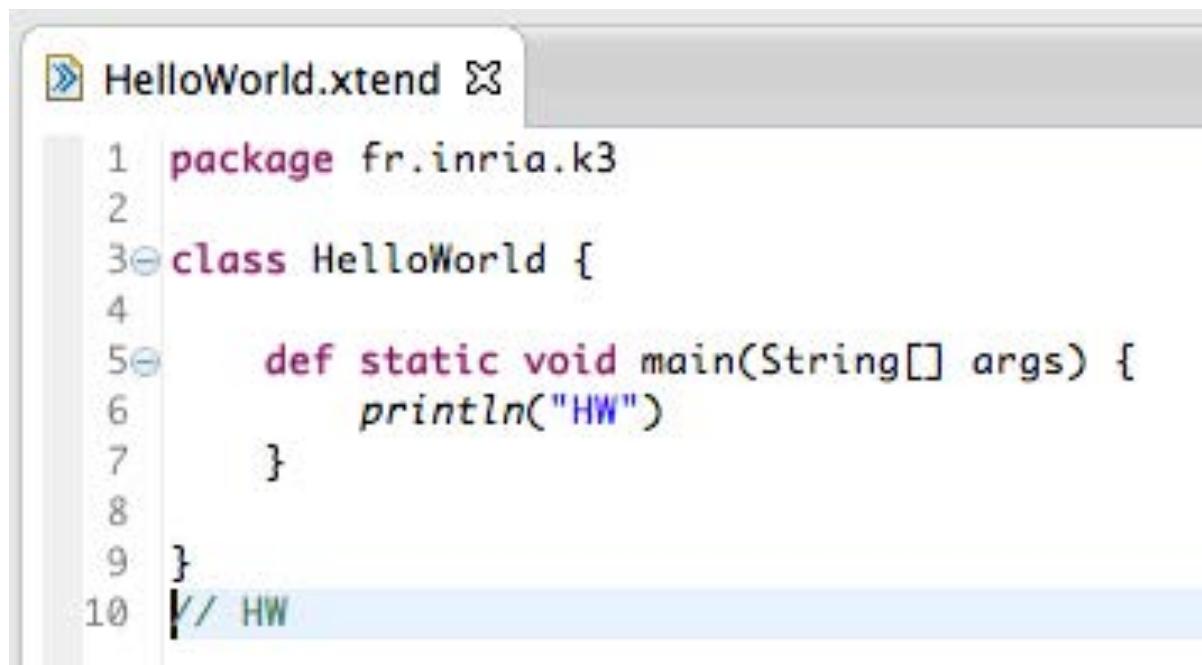
# Contract

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

# Xtend

A pragmatic general-purpose language  
implemented using MDE  
techniques

# Hello World



The screenshot shows a code editor window with a tab labeled "HelloWorld.xtend". The content of the file is as follows:

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

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

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

# Package Declaration

A screenshot of a code editor showing the file `HelloWorld.xtend`. 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 above shows `HelloWorld.xtend` is the active tab.

Semi-colon ';' is optional

A screenshot of a code editor showing three tabs: `HelloWorld.xtend`, `HelloWorld.java`, and `PackageExploder.xtend`. The `HelloWorld.xtend` tab is active. The code in `HelloWorld.xtend` is:1 package fr.inria.k3.^def  
2  
3 class PackageExploder {  
4  
5 }The other tabs show empty files.

'^' 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|
```

new FooMethod().f|

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

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

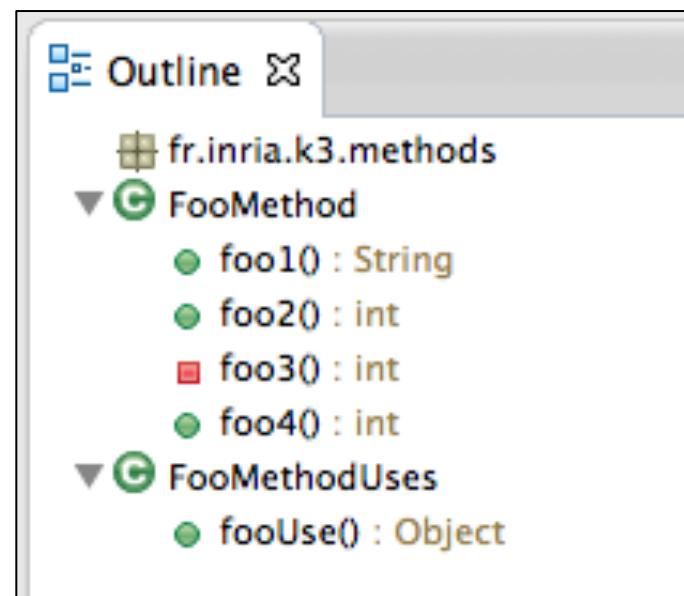
**By default:  
visibility  
conditions set  
to public**

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

# Methods

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

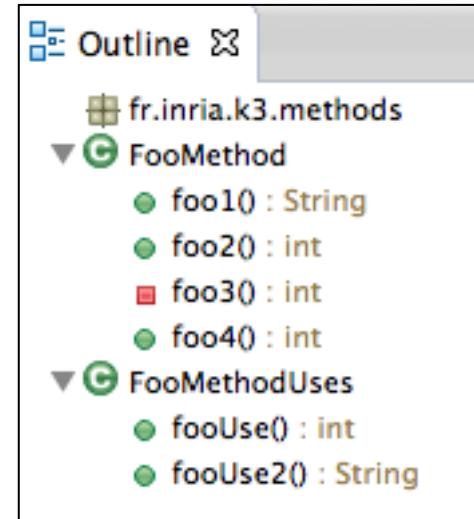
## Type inference (return type)



# Method Calling

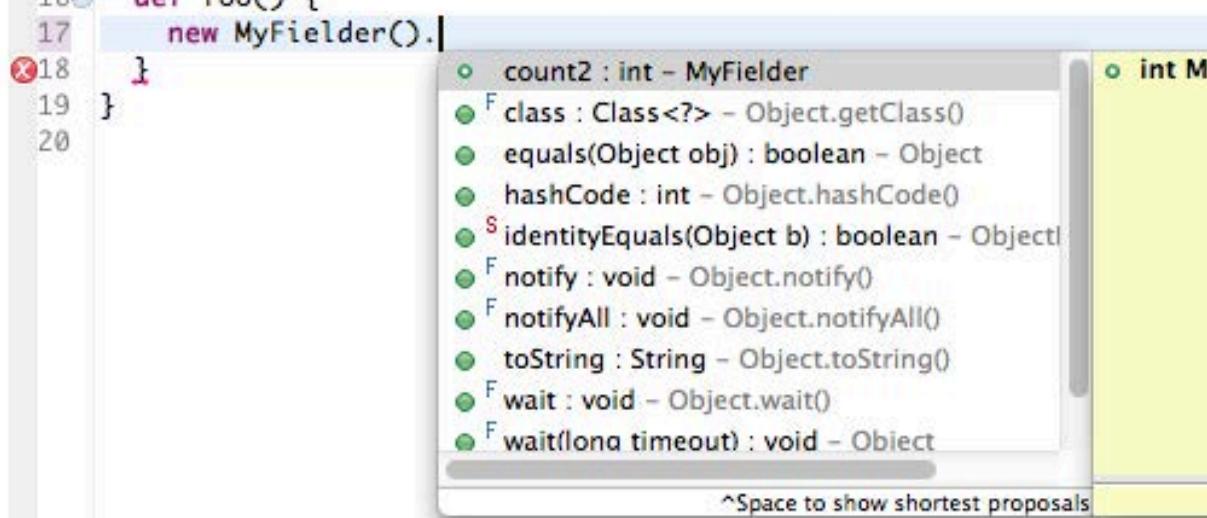
You can omit  
parentheses

```
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 Java code editor with the following code:

```
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 }
```

A code completion dropdown is open at the end of the line `new MyFielder().|`. The dropdown lists the following methods:

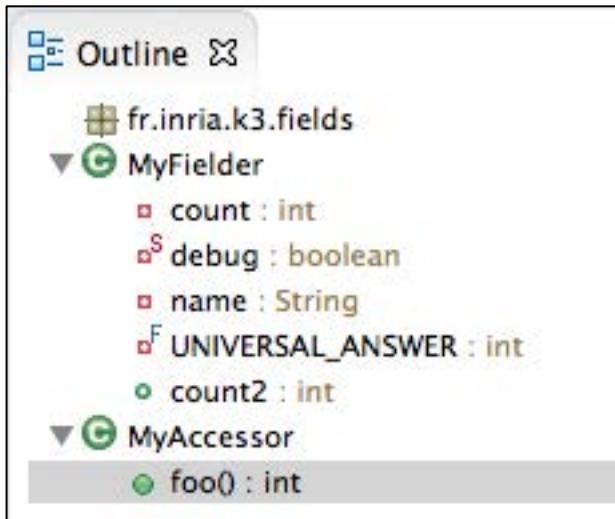
- count2 : int - MyFielder
- getClass : Class<?> - Object.getClass()
- equals(Object obj) : boolean - Object
- hashCode : int - Object.hashCode()
- identityEquals(Object b) : boolean - Object
- notify : void - Object.notify()
- notifyAll : void - Object.notifyAll()
- toString : String - Object.toString()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

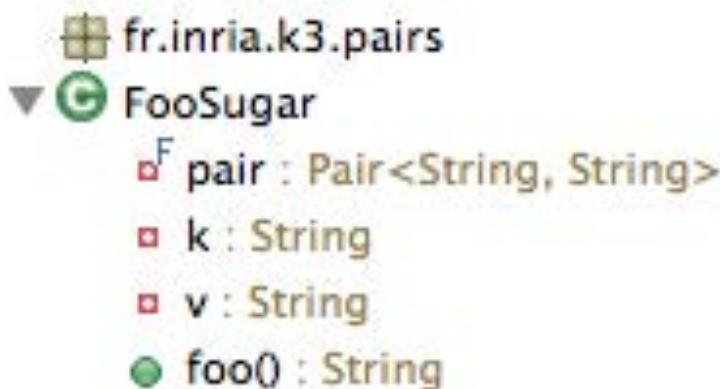
# Static Methods (::)

```
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]
```

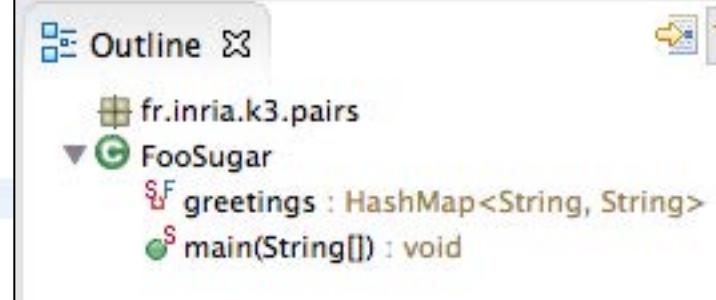
# Pairs

```
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 }
```



# 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 Eclipse IDE's Console view. The output window displays the results of the `main` method 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")
```

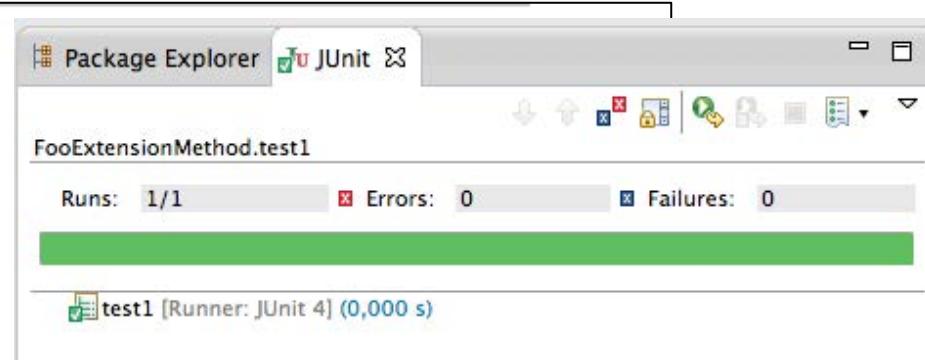
or as an extension method of String:

```
"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"))

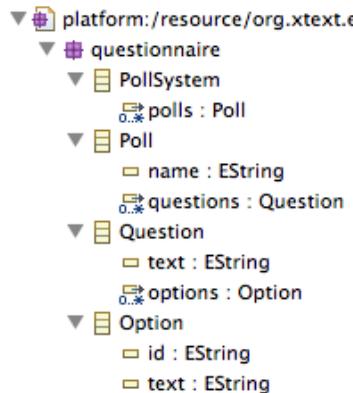
    // MODEL MANAGEMENT (ANALYSIS, TRANSFORMATION)
    var html = toPolls(pollS.polls)
    assertNotNull(html)

    // 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>
}

```

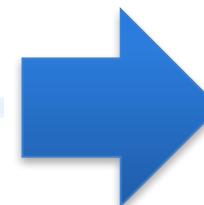
# Templates (2)



**poll1**

**What is A ?**

- B
- C
- D



**poll2**

**What is D ?**

- E
- F

```

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

```

# 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



```
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 }
```

# Xtend to Java (2)

more after

```
>HelloWorld.xtend  J 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;
    }
}
```

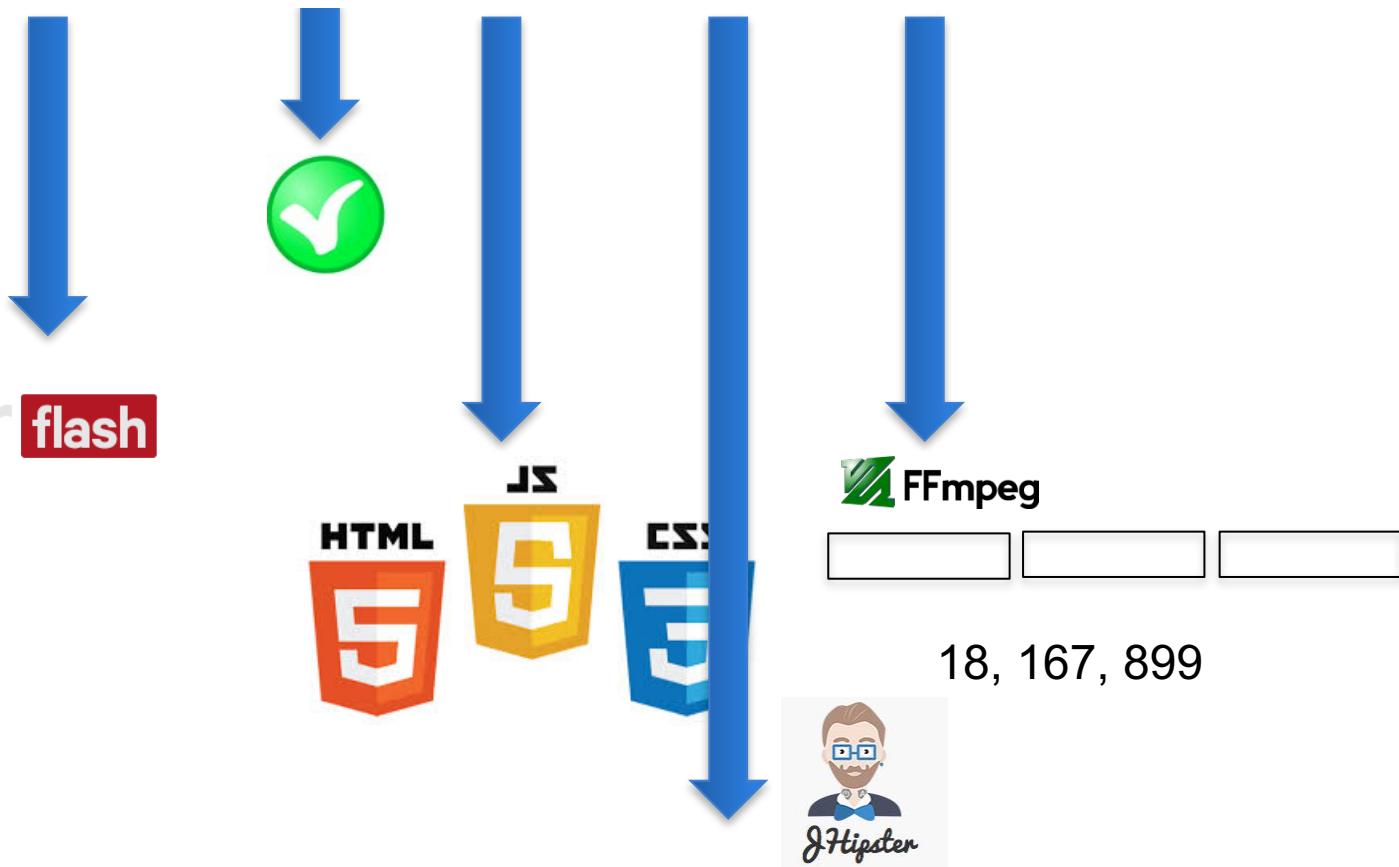
# Xtend/Xtext

Back to our scenarios

## foo1.videoogen

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

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



# 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());
}

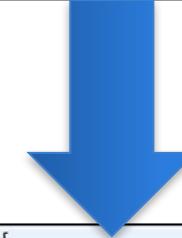
@Test
def test1() {
    // loading
    var videoGen = loadVideoGenerator(URI.createURI("foo2.videogen"))
    assertNotNull(videoGen)
    assertEquals(3, videoGen.videoseqs.size)
    // MODEL MANAGEMENT (ANALYSIS, TRANSFORMATION)
    videoGen.videoseqs.forEach[videoseq | 
        if (videoseq instanceof 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"
    }
}

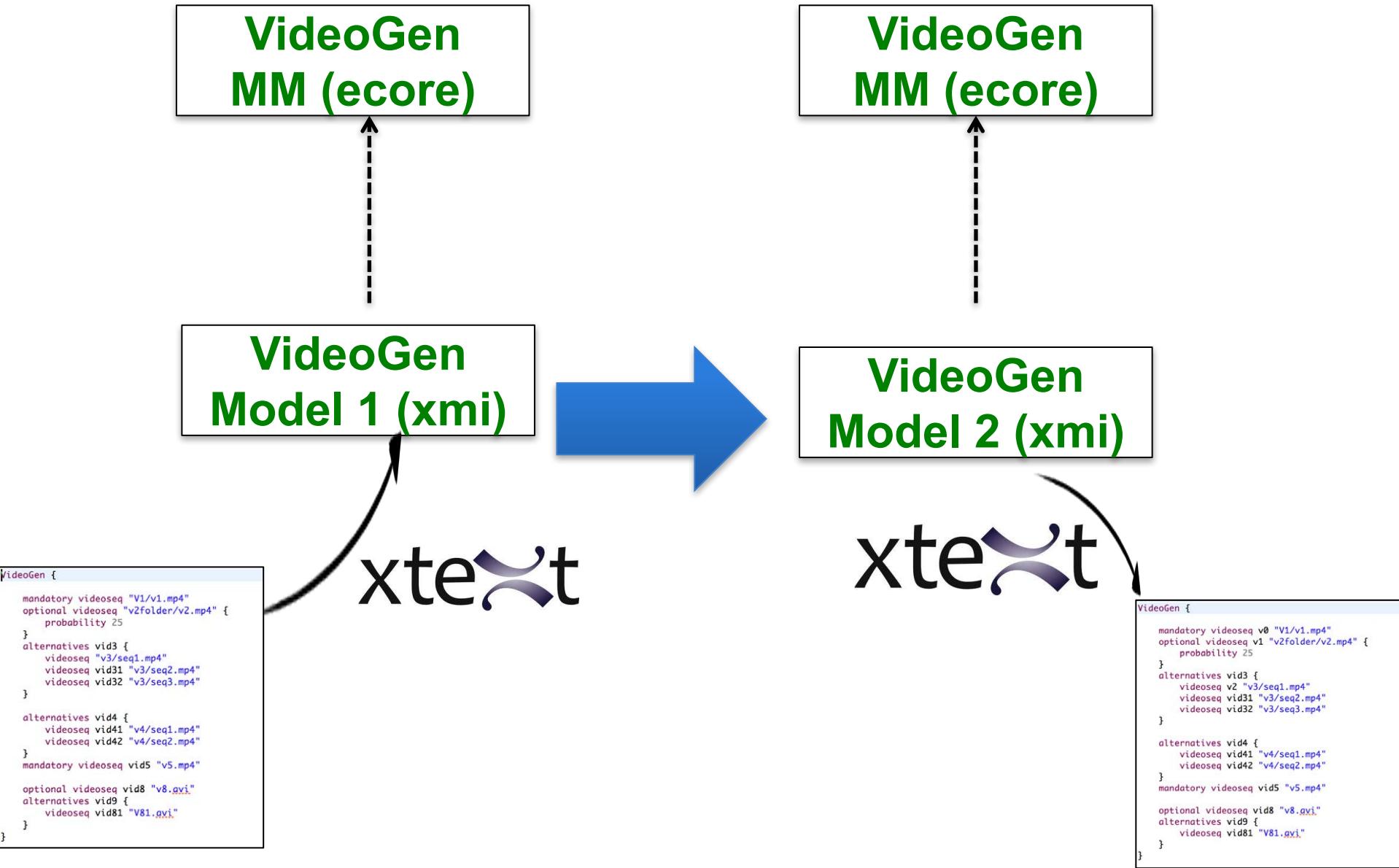
```



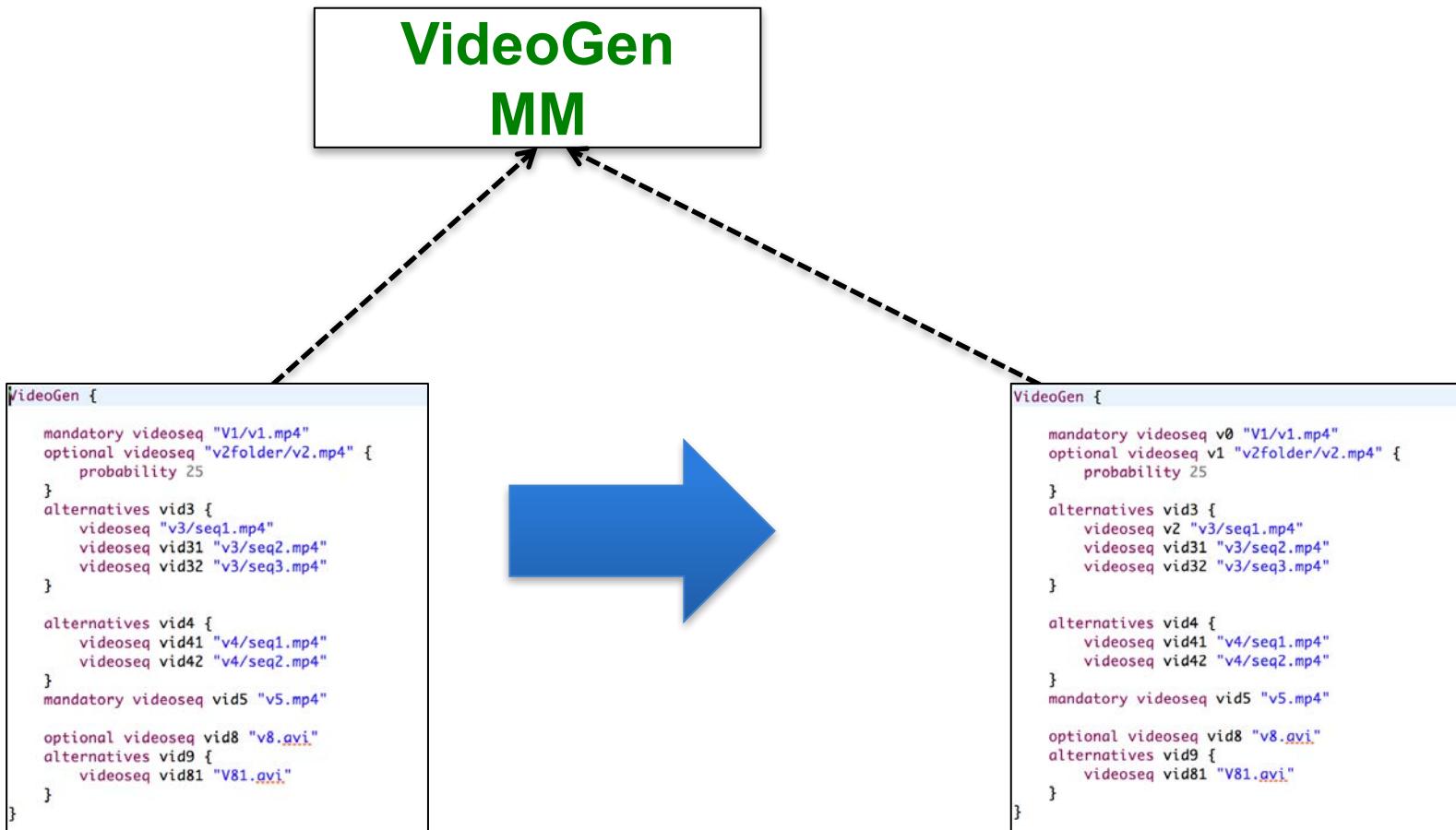
```

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"
    }
}

```

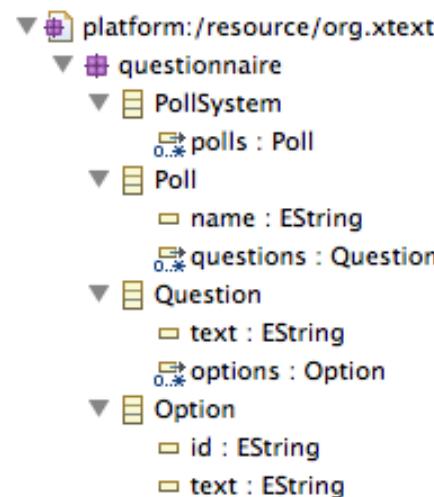


# Endogeneous Transformation





```
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"))

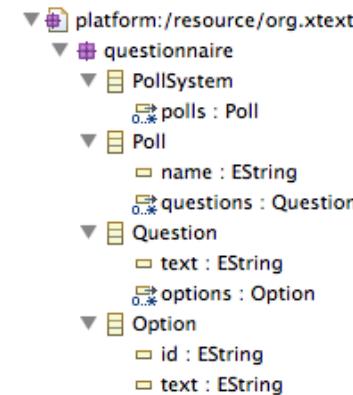
    // MODEL MANAGEMENT (ANALYSIS, TRANSFORMATION)
    var html = toPolls(pollS.polls)
    assertNotNull(html)

    // 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 ?**

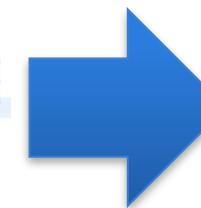
- B
- C
- D

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

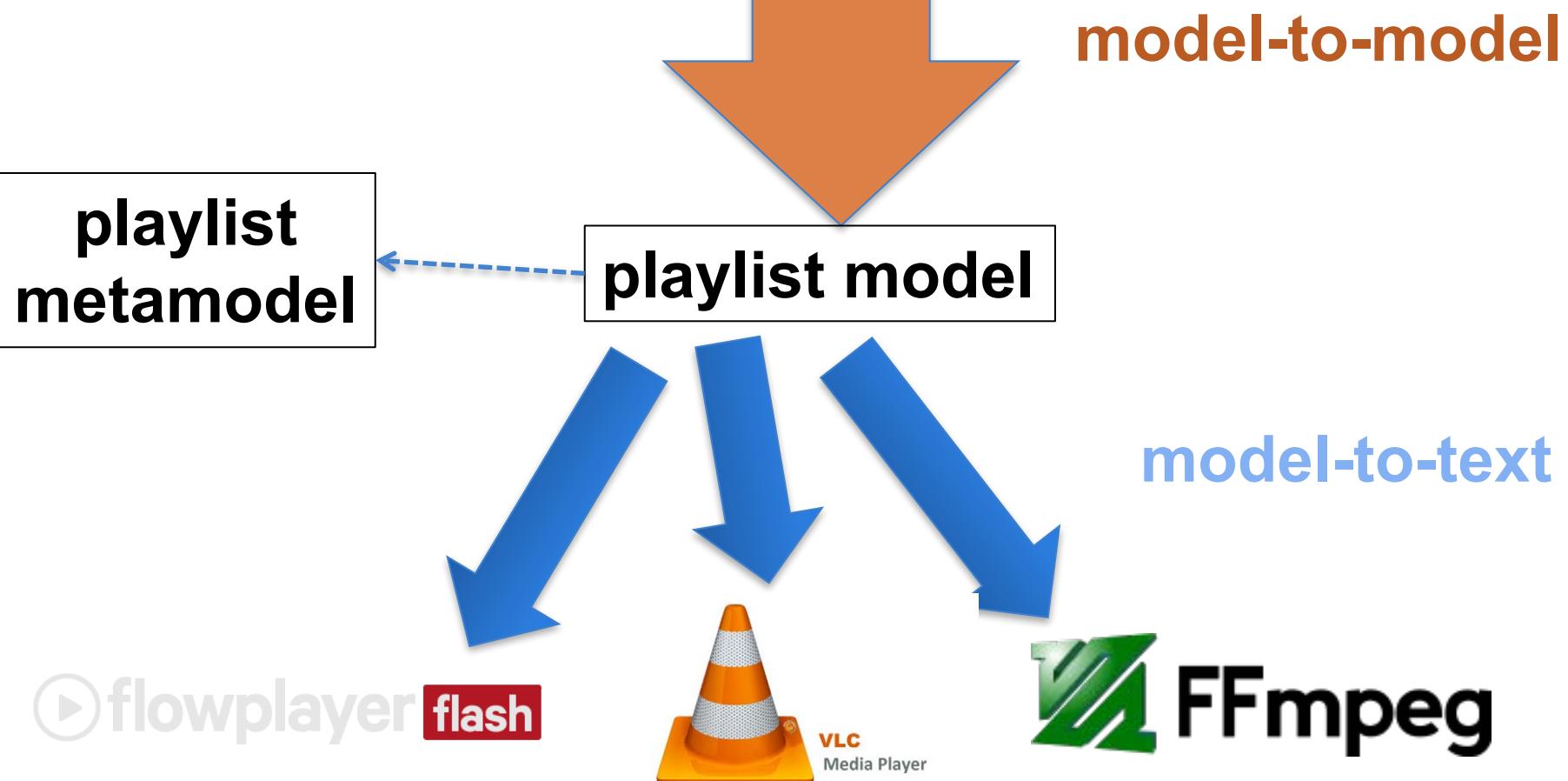
Package
Class
Attribute
Reference

```
@Test
def test2() {

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

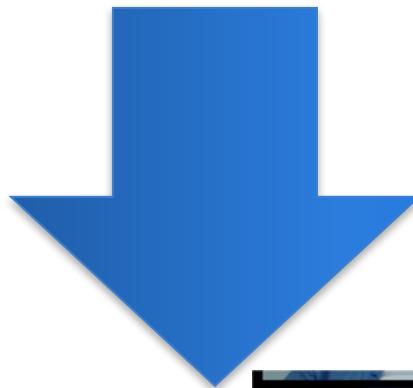
```
foo1.videogen &gt;
mandatory videoseq v1 "https://www.youtube.com/watch?v=PjNi1uYhV5w"
optional videoseq v2 "v2folder/v2.mp4"
alternatives v3 {
    videoseq v31 "v3/seq1.mp4"
    videoseq v32 "v3/seq1.mp4"
    videoseq v33 "v3/seq1.mp4"
}

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



## foo1.videogen

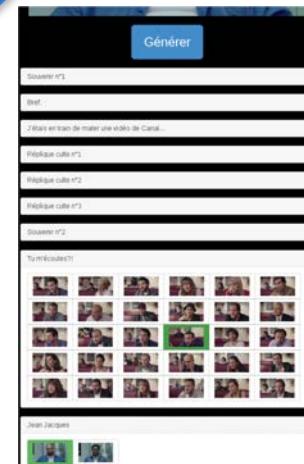
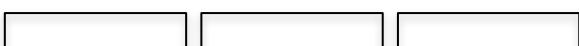
```
mandatory videoseq v1 "https://www.youtube.com/watch?v=PJNi1uYhV5w"  
optional videoseq v2 "v2Folder/v2.mp4"  
alternatives v3 {  
    videoseq v31 "v3/seq1.mp4"  
    videoseq v32 "v3/seq1.mp4"  
    videoseq v33 "v3/seq1.mp4"  
}  
  
alternatives v4 {  
    videoseq v41 "v4/seq1.mp4"  
    videoseq 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)



JS



## 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

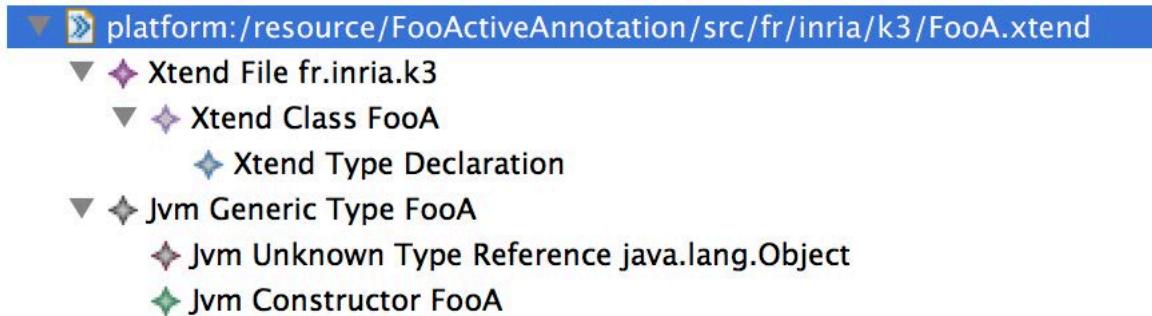
- Random generation
- Configurator
- Game
- ...

Xtend is  
implemented using  
MDE principles

```
package fr.inria.k3
```

```
class FooA {  
    //|  
}
```

Model



<https://github.com/eclipse/xtext/blob/master/plugins/org.eclipse.xtext/src/org/eclipse/xtext/core/Xtend.xtext>

```
grammar org.eclipse.xtext.core.Xtend with org.eclipse.xtext.xbase.annotations.XbaseWithAnnotations

import "http://www.eclipse.org/xtend"
import "http://www.eclipse.org/xtext/xbase/Xbase" as xbase
import "http://www.eclipse.org/xtext/xbase/Xtype" as xtype
import "http://www.eclipse.org/Xtext/Xbase/XAnnotations" as annotations
import "http://www.eclipse.org/xtext/common/JavaVMTypes" as types

File returns XtendFile :
    ('package' package=QualifiedName ';'?)?
        importSection=XImportSection?
        (xtendTypes+=Type)*
;

Type returns XtendTypeDeclaration :
    {XtendTypeDeclaration} annotations+=XAnnotation*
    (
        {XtendClass.annotationInfo = current}
        modifiers+=CommonModifier*
        'class' name=ValidID ('<' typeParameters+=JvmTypeParameter (',' typeParameters+=JvmTypeParameter)* '>')?
        ('extends' extends=JvmParameterizedTypeReference)?
        ('implements' implements+=JvmParameterizedTypeReference (',' implements+=JvmParameterizedTypeReference)*)? '{'
            (members+=Member)*
        '}'
    |
        {XtendInterface.annotationInfo = current}
        modifiers+=CommonModifier*
        'interface' name=ValidID ('<' typeParameters+=JvmTypeParameter (',' typeParameters+=JvmTypeParameter)* '>')?
        ('extends' extends+=JvmParameterizedTypeReference (',' extends+=JvmParameterizedTypeReference)*)? '{'
            (members+=Member)*
        '}'
    |
        {XtendEnum.annotationInfo = current}
        modifiers+=CommonModifier*
        'enum' name=ValidID '{'
            (members+=XtendEnumLiteral (',' members+=XtendEnumLiteral)*)? ';'?
        '}'
    |
        {XtendAnnotationType.annotationInfo = current}
        modifiers+=CommonModifier*
        'annotation' name=ValidID '{'
            (members+=AnnotationField)*
        '}'
    )
;
```



```
public class XtendCompiler extends XbaseCompiler {  
  
    @Override  
    public void acceptForLoop(JvmFormalParameter parameter, @Nullable XExpression expression) {  
        currentAppendable = null;  
        super.acceptForLoop(parameter, expression);  
        if (expression == null)  
            throw new IllegalArgumentException("expression may not be null");  
        RichStringForLoop forLoop = (RichStringForLoop) expression.eContainer();  
        forStack.add(forLoop);  
        appendable.newLine();  
        pushAppendable(forLoop);  
        appendable.append("{").increaseIndentation();  
  
        ITreeAppendable debugAppendable = appendable.trace(forLoop, true);  
        internalToJavaStatement(expression, debugAppendable, true);  
        String variableName = null;  
        if (forLoop.getBefore() != null || forLoop.getSeparator() != null || forLoop.getAfter() != null) {  
            variableName = debugAppendable.declareSyntheticVariable(forLoop, "_hasElements");  
            debugAppendable.newLine();  
            debugAppendable.append("boolean ");  
            debugAppendable.append(variableName);  
            debugAppendable.append(" = false;");  
        }  
        debugAppendable.newLine();  
        debugAppendable.append("for(final ");  
        JvmTypeReference paramType = getTypeProvider().getTypeForIdentifiable(parameter);  
        serialise(paramType, parameter, debugAppendable);  
        debugAppendable.append(" ");  
        String loopParam = debugAppendable.declareVariable(parameter, parameter.getName());  
        debugAppendable.append(loopParam);  
        debugAppendable.append(" : ");  
        internalToJavaExpression(expression, debugAppendable);  
        debugAppendable.append(") {").increaseIndentation();  
    }  
}
```

Model Transformation

# Xtend to Java

```
HelloWorld.xtend    J HelloWorld.java ✎
```

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

```
public final class GUIWindow {
    private GUIWindow() {
        // singleton
    }

    private int x;

    private int y;

    private final static GUIWindow INSTANCE = new GUIWindow();

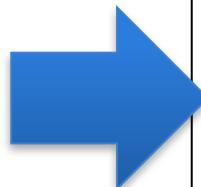
    public static GUIWindow getINSTANCE() {
        return INSTANCE;
    }
}
```

```
package fr.inria.k3

@Singleton
class GUIWindow {

    int x;
    int y;

}
```



# Xtend

Advanced features (active  
annotation)

Model transformation in depth

# Active Annotations

(a practical way to transform your  
data, programs, models)

# Do You know Java Annotations ?



@Override

@SuppressWarnings



Guice (pronounced 'juice') is a lightweight dependency injection framework for Java 5 and above, brought to you by Google.

# JUnit

```
package com.vogella.junit.first;

import org.junit.runner.RunWith;
import org.junit.runners.Suite;
import org.junit.runners.Suite.SuiteClasses;

@RunWith(Suite.class)
@SuiteClasses({ MyClassTest.class, MySecondClassTest.class })
public class AllTests {

}
```

```
public class MyClassTest {

    @BeforeClass
    public static void testSetup() {
    }

    @AfterClass
    public static void testCleanup() {
        // Teardown for data used by the unit tests
    }

    @Test(expected = IllegalArgumentException.class)
    public void testExceptionIsThrown() {
        MyClass tester = new MyClass();
        tester.multiply(1000, 5);
    }

    @Test
    public void testMultiply() {
        MyClass tester = new MyClass();
        assertEquals("10 x 5 must be 50", 50, tester.multiply(10, 5));
    }
}
```

# Annotations (JUnit 4)

@Test public void method()	The <b>@Test</b> annotation identifies a method as a test method.
@Test(expected = Exception.class)	Fails, if the method does not throw the named exception.
@Test(timeout=100)	Fails, if the method takes longer than 100 milliseconds.
@Before public void method()	This method is executed before each test. It is used to can prepare the test environment (e.g. read input data, initialize the class).
@After public void method()	This method is executed after each test. It is used to cleanup the test environment (e.g. delete temporary data, restore defaults). It can also save memory by cleaning up expensive memory structures.
@BeforeClass public static void method()	This method is executed once, before the start of all tests. It is used to perform time intensive activities, for example to connect to a database. Methods annotated with this annotation need to be defined as <b>static</b> to work with JUnit.
@AfterClass public static void method()	This method is executed once, after all tests have been finished. It is used to perform clean-up activities, for example to disconnect from a database. Methods annotated with this annotation need to be defined as <b>static</b> to work with JUnit.

[http://www.vogella.com/articles/JUnit/article.html#usingjunit\\_annotations](http://www.vogella.com/articles/JUnit/article.html#usingjunit_annotations)

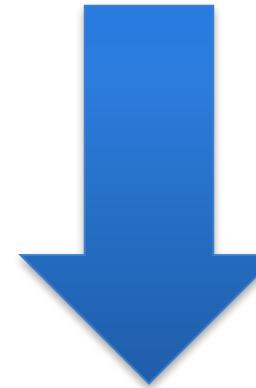
```
@XmlRootElement  
public class Customer {  
  
    String name;  
    int age;  
    int id;  
  
    public String getName() {  
        return name;  
    }  
  
    @XmlElement  
    public void setName(String name) {  
        this.name = name;  
    }  
  
    public int getAge() {  
        return age;  
    }  
  
    @XmlElement  
    public void setAge(int age) {  
        this.age = age;  
    }  
  
    public int getId() {  
        return id;  
    }  
  
    @XmlAttribute  
    public void setId(int id) {  
        this.id = id;  
    }  
}
```



**JAXB**

# Java Annotations

```
Customer customer = new Customer();  
customer.setId(100);  
customer.setName("mkyong");  
customer.setAge(29);
```



```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>  
<customer id="100">  
    <age>29</age>  
    <name>mkyong</name>  
</customer>
```



# HIBERNATE

## 2.2.1. Marking a POJO as persistent entity

Every persistent POJO class is an entity and is declared using the `@Entity` annotation (at the class level):

```
@Entity
public class Flight implements Serializable {
    Long id;

    @Id
    public Long getId() { return id; }

    public void setId(Long id) { this.id = id; }
}
```

`@Entity` declares the class as an entity (i.e. a persistent POJO class), `@Id` declares the identifier property of this entity. The other mapping declarations are implicit. The class `Flight` is mapped to the `Flight` table, using the column `id` as its primary key column.

```
@Entity
class MedicalHistory implements Serializable {
    @Id @OneToOne
    @JoinColumn(name = "person_id")
    Person patient;
}

@Entity
public class Person implements Serializable {
    @Id @GeneratedValue Integer id;
}
```

# Javadoc (old fashion, not real annotations)

```
/**  
 * Returns an Image object that can then be painted on the screen.  
 * The url argument must specify an absolute {@link URL}. The name  
 * argument is a specifier that is relative to the url argument.  
 * <p>  
 * This method always returns immediately, whether or not the  
 * image exists. When this applet attempts to draw the image on  
 * the screen, the data will be loaded. The graphics primitives  
 * that draw the image will incrementally paint on the screen.  
 *  
 * @param url an absolute URL giving the base location of the image  
 * @param name the location of the image, relative to the url argument  
 * @return the image at the specified URL  
 * @see Image  
 */  
public Image getImage(URL url, String name) {  
    try {  
        return getImage(new URL(url, name));  
    } catch (MalformedURLException e) {  
        return null;  
    }  
}
```

## Disclaimer

- @AhaMoment
- @BossMadeMeDoIt
- @HandsOff
- @IAmAwesome
- @LegacySucks

## Enforceable

- @CantTouchThis
- @ImaLetYouFinishBut

## Literary Verse (new subcategory)

- @Burma Shave
- @Clerihew
- @DoubleDactyl
- @Haiku (moved to this subcategory)
- @Limerick
- @Sonnet

## Remarks

- @Fail
- @OhNoYouDidnt
- @RTFM
- @Win



The Google Annotations Gallery is an exciting new Java open source library that provides a rich set of annotations for developers to express themselves.

Do you find the standard Java annotations dry and lackluster? Have you ever resorted to leaving messages to fellow developers with the `@Deprecated` annotation? Wouldn't you rather leave a `@LOL` or `@Facepalm` instead?

Not only can you leave expressive remarks in your code, you can use these annotations to draw attention to your poetic endeavors. How many times have you written a palindromic or synecdochal line of code and wished you could annotate it for future readers to admire? Look no further than `@Palindrome` and `@Synecdoche`.

But wait, there's more. The Google Annotations Gallery comes complete with dynamic bytecode instrumentation. By using the `gag-agent.jar` Java agent, you can have your annotations behavior-enforced at runtime. For example, if you want to ensure that a method parameter is non-zero, try `@ThisHadBetterNotBe(Property.ZERO)`. Want to completely inhibit a method's implementation? Try `@Noop`.

# **Annotations for...**

- Documentation
  - Javadoc like
- Information to the Compiler
  - Suppress warnings, error detections
- Generation
  - Code (Java, SQL, etc.)
  - Configuration files (e.g., XML-like)
- Runtime processing

⇒ **Transformation of programs, datas, models**

⇒ **You can define your own**

# Annotations: How does it work?

The screenshot shows a browser window displaying the Javadoc for the `Test` interface. The URL is `junit.sourceforge.net/javadoc/org/junit/Test.html`. The page title is "Annotation Type Test". The navigation bar includes links for Overview, Package, Class, Tree, Deprecated, Index, and Help. Below the navigation bar are links for PREV CLASS and NEXT CLASS, and a summary section with REQUIRED and OPTIONAL links.

org.junit

## Annotation Type Test

```
@Retention(value=RUNTIME)
@Target(value=METHOD)
public @interface Test
```

The `Test` annotation tells JUnit that the `public void` method to which it is applied has no side effects. If no exceptions are thrown, the test is assumed to have succeeded.

A simple test looks like this:

```
public class Example {
    @Test
    public void method() {
        org.junit.Assert.assertTrue( new ArrayList().isEmpty());
    }
}
```

The `Test` annotation supports two optional parameters. The first, `expected`,

```
@Test(expected=IndexOutOfBoundsException.class) public
    new ArrayList<Object>().get(1);
}
```

The second optional parameter, `timeout`, causes a test to fail if it takes longer than the specified time.

```
@Test(timeout=100) public void infinity() {
    while(true);
}
```

# Annotations: How does it work?

GitHub, Inc. [US] <https://github.com/junit-team/junit/blob/master/src/main/java/org/junit/Test.java>

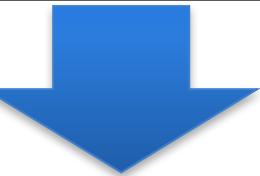
```
60  @Retention(RetentionPolicy.RUNTIME)
61  @Target({ElementType.METHOD})
62  public @interface Test {
63
64      /**
65      * Default empty exception
66      */
67      static class None extends Throwable {
68          private static final long serialVersionUID = 1L;
69
70          private None() {
71              }
72      }
73
74      /**
75      * Optionally specify <code>expected</code>, a Throwable, to cause a
76      * and only if an exception of the specified class is thrown by the i
77      */
78      Class<? extends Throwable> expected() default None.class;
79
80      /**
81      * Optionally specify <code>timeout</code> in milliseconds to cause i
82      * takes longer than that number of milliseconds.
83      * <p>
84      * <b>THREAD SAFETY WARNING:</b> Test methods with a timeout paramet
85      * thread which runs the fixture's @Before and @After methods. This i
86      * code that is not thread safe when compared to the same test method
87      * <b>Consider using the {@link org.junit.rules.Timeout} rule instead
88      * same thread as the fixture's @Before and @After methods.
89      * </p>
90      */
91      long timeout() default 0L;
92  }
```

## Java Build Path

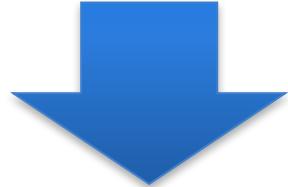
Source | Projects | Libraries | Order and Export

JARs and class folders on the build path:

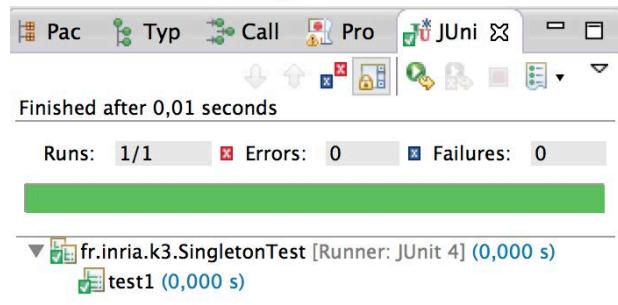
- ▶ JRE System Library [JavaSE-1.6]
- ▶ JUnit 4



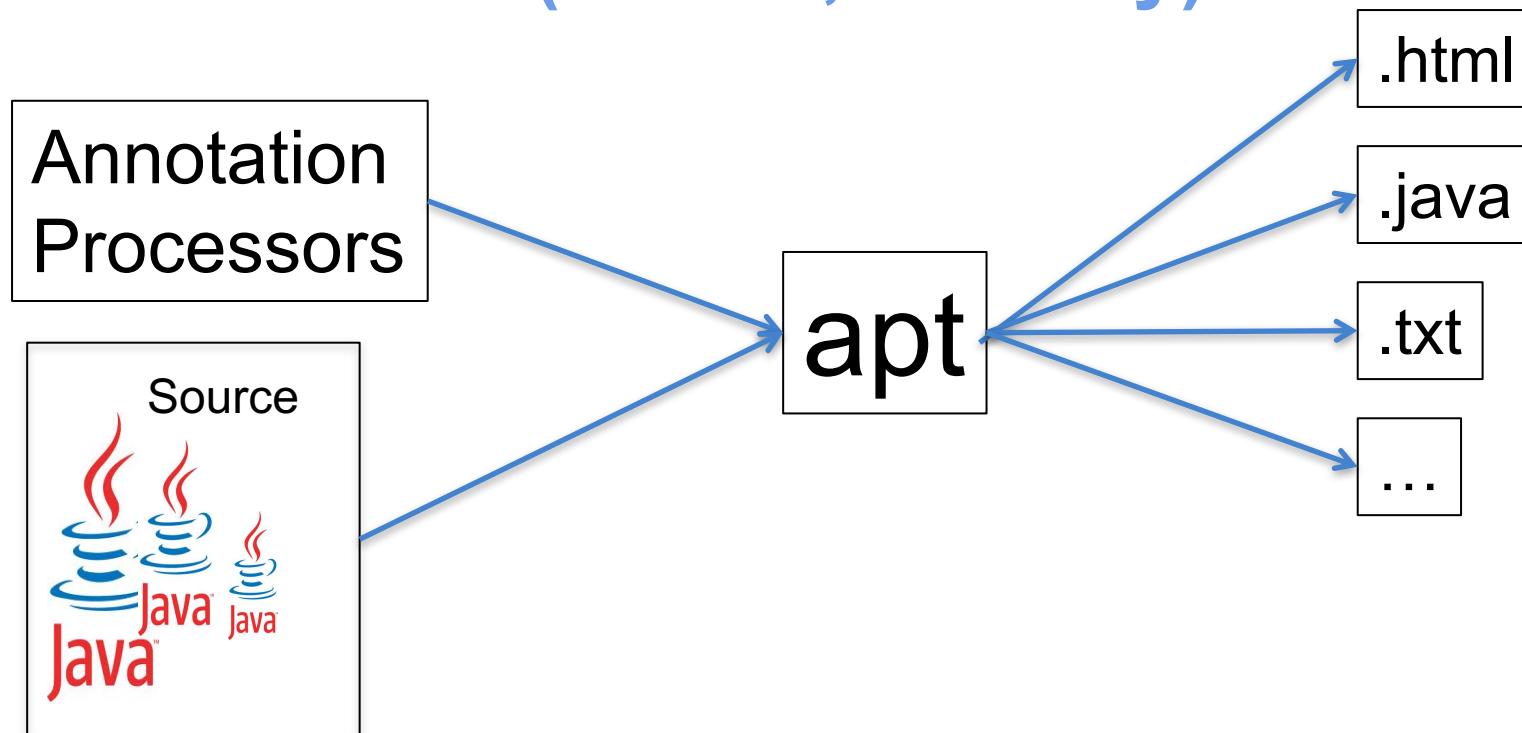
```
package com.vogella.junit.first;  
  
import org.junit.runner.RunWith;  
import org.junit.runners.Suite;  
import org.junit.runners.SuiteClasses;  
  
@RunWith(Suite.class)  
@SuiteClasses({ MyClassTest.class, MySecondClassTest.class })  
public class AllTests {  
}
```



Transformation of Java code



# Annotations and Transformations (Java 5, old way)



← → C docs.oracle.com/javase/1.5.0/docs/guide/apt/GettingStarted.html



Getting Started with  
the Annotation Processing Tool (apt)

What is apt?

The command-line utility `apt`, annotation processing tool, finds and executes *annotation processors* based on the annotations present in the set of specified source files being examined. The annotation

# Annotations and Transformations (Java 5, old way)

## Annotation Processors

```
/*
 * This class is used to run an annotation processor that lists class
 * names. The functionality of the processor is analogous to the
 * ListClass doclet in the Doclet Overview.
 */
public class ListClassApf implements AnnotationProcessorFactory {
    // Process any set of annotations
    private static final Collection<String> supportedAnnotations
        = unmodifiableCollection(Arrays.asList("*"));

    // No supported options
    private static final Collection<String> supportedOptions = emptySet();

    public Collection<String> supportedAnnotationTypes() {
        return supportedAnnotations;
    }

    public Collection<String> supportedOptions() {
        return supportedOptions;
    }

    public AnnotationProcessor getProcessorFor(
        Set<AnnotationTypeDeclaration> atds,
        AnnotationProcessorEnvironment env) {
        return new ListClassAp(env);
    }

    private static class ListClassAp implements AnnotationProcessor {
        private final AnnotationProcessorEnvironment env;
        ListClassAp(AnnotationProcessorEnvironment env) {
            this.env = env;
        }

        public void process() {
            for (TypeDeclaration typeDecl : env.getSpecifiedTypeDeclarations())
                typeDecl.accept(getDeclarationScanner(new ListClassVisitor(),
                    NO_OP));
        }

        private static class ListClassVisitor extends SimpleDeclarationVisitor {
            public void visitClassDeclaration(ClassDeclaration d) {
                System.out.println(d.getQualifiedName());
            }
        }
    }
}
```

The logo for the apt tool, consisting of the lowercase letters "apt" in a bold, sans-serif font, enclosed within a thin black rectangular border.

### The `apt` Command Line

In addition to its own options, the `apt` tool accepts all of the command-line options accepted by `javac`.

The `apt` specific options are:

- s *dir*  
Specify the directory root under which processor-generated source files will be placed.
- nocompile  
Do not compile source files to class files.
- print  
Print out textual representation of specified types; perform no annotation processing.
- A[key[=val]]  
Options to pass to annotation processors -- these are not interpreted by `apt` directly.
- factorypath *path*  
Specify where to find annotation processor factories; if this option is used, the classpath is ignored.
- factory *classname*  
Name of `AnnotationProcessorFactory` to use; bypasses default discovery procedure.

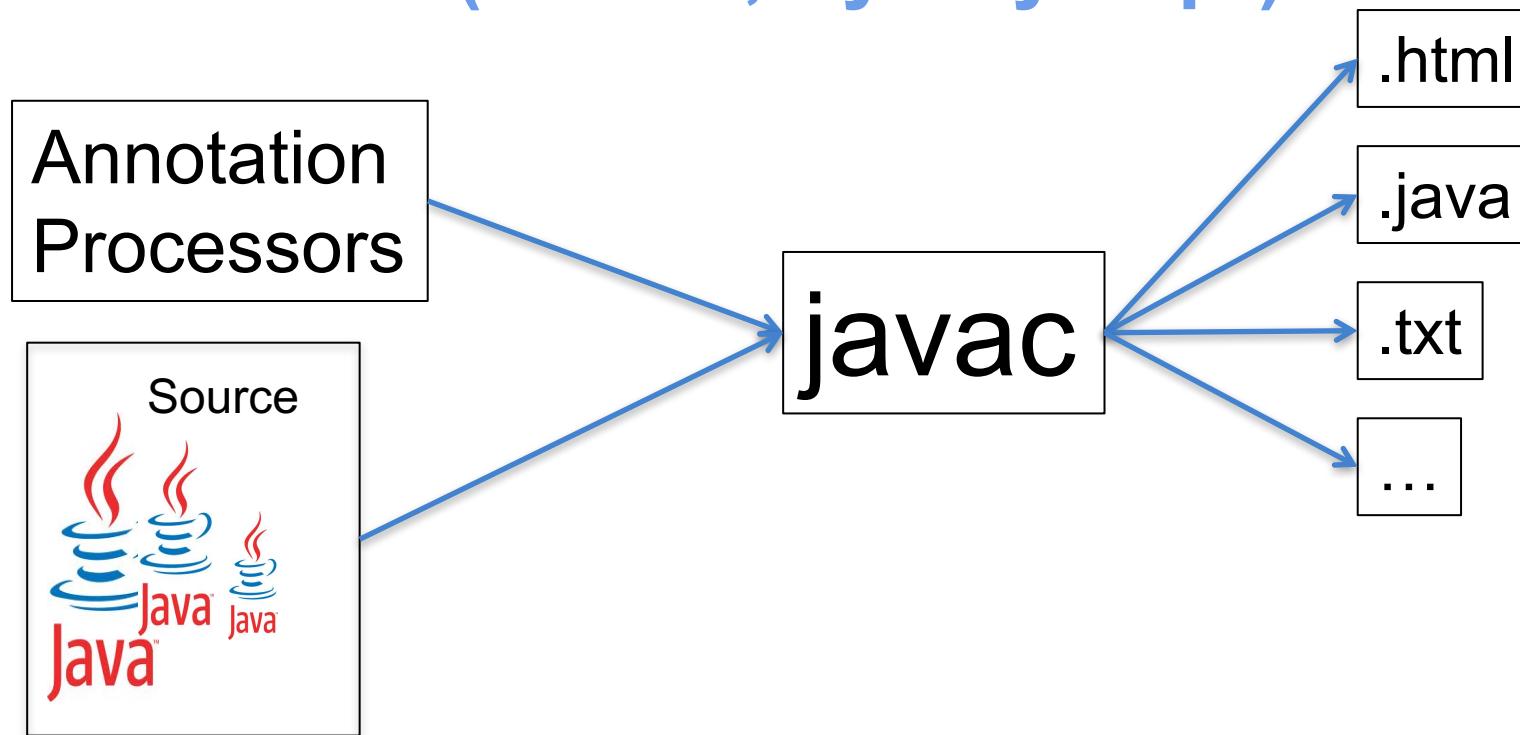
How `apt` shares some of `javac`'s options:

- d *dir*  
Specify where to place processor and javac generated class files.
- cp *path* or -classpath *path*  
Specify where to find user class files and annotation processor factories. If -factorypath is specified, this option is ignored.

There are a few `apt` hidden options that may be useful for debugging:

- XListAnnotationTypes  
List found annotation types.
- XListDeclarations  
List specified and included declarations.
- XPrintAptRounds  
Print information about initial and recursive `apt` rounds.
- XPrintFactoryInfo  
Print information about which annotations a factory is asked to process.

# Annotations and Transformations (Java 6, bye bye apt)



**Integrated into the Java compiler (javac)**  
**New API: Pluggable Annotation Processing**

# Annotations and Transformations (Java 6, bye bye apt)

## Annotation

Pr

```
import java.util.*;
import javax.annotation.processing.*;
import javax.lang.model.*;
import javax.lang.model.element.*;
```

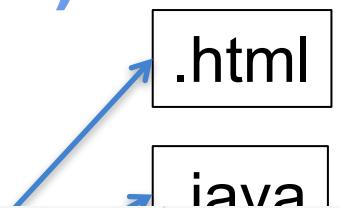


```
@SupportedAnnotationTypes(value= {"*"})
@SupportedSourceVersion(SourceVersion.RELEASE_6)

public class TestAnnotationProcessor extends AbstractProcessor {

    @Override
    public boolean process(
        Set<?> extends TypeElement> annotations, RoundEnvironment roundEnv){

        for (TypeElement element : annotations){
            System.out.println(element.getQualifiedName());
        }
        return true;
    }
}
```



javac –processor ...

# Alternative: Java Reflection

```
import java.lang.annotation.Documented;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;

@Documented
@Retention(RetentionPolicy.RUNTIME)
public @interface Todo {

    public enum Importance {
        MINEURE, IMPORTANT, MAJEUR, CRITIQUE
    }

    Importance importance() default Importance.MINEURE;

    String[] description();

    String assigneA();

    String dateAssignment();
}
```

```
@Todo(importance = Importance.CRITIQUE,
       description = "Corriger le bug dans le calcul",
       assigneA = "JMD",
       dateAssignment = "11-11-2007")
public class TestInstrospectionAnnotation {

    public static void main(
        String[] args) {
        Todo todo = null;

        // traitement annotation sur la classe
        Class classe = TestInstrospectionAnnotation.class;
        todo = (Todo) classe.getAnnotation(Todo.class);
        if (todo != null) {
            System.out.println("classe " + classe.getName());
            System.out.println(" [" + todo.importance() + "]" + ("+" + todo.assigneA()
                + " le " + todo.dateAssignment() + ")");
            for(String desc : todo.description()) {
                System.out.println("      _ " + desc);
            }
        }

        // traitement annotation sur les méthodes de la classe
        for(Method m : TestInstrospectionAnnotation.class.getMethods()) {
            todo = (Todo) m.getAnnotation(Todo.class);
            if (todo != null) {
                System.out.println("methode " + m.getName());
                System.out.println(" [" + todo.importance() + "]" + ("+" + todo.assigneA()
                    + " le " + todo.dateAssignment() + ")");
                for(String desc : todo.description()) {
                    System.out.println("      _ " + desc);
                }
            }
        }
    }

    @Todo(importance = Importance.MAJEUR,
          description = "Implementer la methode",
          assigneA = "JMD",
          dateAssignment = "11-11-2007")
    public void methode1() {
    }

    @Todo(importance = Importance.MINEURE,
          description = {"Completer la methode", "Ameliorer les logs"},
          assigneA = "JMD",
          dateAssignment = "12-11-2007")
    public void methode2() {
    }
}
```

<http://www.jmdoudoux.fr/java/dej/chap-annotations.htm#annotations-7>

# You can define your own annotations

- Specification
  - At the Class, Field, Method level
  - Annotations can be combined
  - Annotations can have parameters
- Transformation (compilation)
  - Introspection
  - Compiler (javac/apt) and definition of « processors »
- Widely used
  - Generation, verification, etc.

# Back to Xtend

- Active Annotations
  - Facilities to specify Annotations and their treatment (API)
  - Seamless integration in the IDE
    - On-the-fly compilation to Java allows proper type checking and auto-completion

# Example

```
package fr.inria.k3  
@Singleton  
class GUIWindow {  
    int x ;  
    int y ;  
}
```

# Example

```
package fr.inria.k3
```

```
@Singleton
```

```
class GUIWindow {
```

```
    int x ;
```

```
    int y ;
```

```
public final class GUIWindow {
    private GUIWindow() {
        // singleton
    }

    private int x;

    private int y;

    private final static GUIWindow INSTANCE = new GUIWindow();

    public static GUIWindow getINSTANCE() {
        return INSTANCE;
    }
}
```

```
package fr.inria.k3
```

```
@Singleton
```

```
class GUIWindow {
```

```
    int x;
```

```
    int y;
```

```
}
```

```
public final class GUIWindow {
    private GUIWindow() {
        // singleton
    }

    private int x;
    private int y;

    private final static GUIWindow INSTANCE = new GUIWindow();
    public static GUIWindow getINSTANCE() {
        return INSTANCE;
    }
}
```

```
class SingletonProcessor extends AbstractClassProcessor {

    override doTransform(MutableClassDeclaration annotatedClass, extension TransformationContext context) {

        annotatedClass.final = true

        if (annotatedClass.declaredConstructors.size > 1)
            annotatedClass.addError("More than one constructor is defined")

        val constructor = annotatedClass.declaredConstructors.head
        if (constructor.parameters.size > 0)
            constructor.addError("Constructor has arguments")

        if (constructor.body == null) {

            // no constructor defined in the annotated class
            constructor.visibility = Visibility::PRIVATE
            constructor.body = ["'// singleton'"]
        } else {
            if (constructor.visibility != Visibility::PRIVATE)
                constructor.addError("Constructor is not private")
        }

        annotatedClass.addField('INSTANCE') [
            visibility = Visibility::PRIVATE
            static = true
            final = true
            type = annotatedClass.newTypeReference
            initializer = [
                "'new «annotatedClass.simpleName»()'"
            ]
        ]

        annotatedClass.addMethod('getINSTANCE') [
            visibility = Visibility::PUBLIC
            static = true
            returnType = annotatedClass.newTypeReference
            body = [
                "'return INSTANCE;'"
            ]
        ]
    }
}
```

# Example (2)

```
package fr.inria.k3

@Extract
class ExtractA {  
}
```

```
package fr.inria.k3;

import fr.inria.k3.Extract;..  
  
@Extract
@SuppressWarnings("all")
public class ExtractA implements ExtractAInterface {  
}
```

```
package fr.inria.k3
```

```
@Extract  
class ExtractA {  
}
```



```
package fr.inria.k3;
```

```
import fr.inria.k3.Extract;
```

```
@Extract  
@SuppressWarnings("all")  
public class ExtractA implements ExtractAInterface {  
}
```

```
/**  
 * Extracts an interface for all locally declared public methods.  
 */  
@Target(ElementType.TYPE)  
@Active(ExtractProcessor)  
annotation Extract {}  
  
class ExtractProcessor extends AbstractClassProcessor {  
  
    override doRegisterGlobals(ClassDeclaration annotatedClass, RegisterGlobalsContext context) {  
        context.registerInterface(annotatedClass.interfaceName)  
    }  
  
    def getInterfaceName(ClassDeclaration annotatedClass) {  
        annotatedClass.qualifiedName+"Interface"  
    }  
  
    override doTransform(MutableClassDeclaration annotatedClass, extension TransformationContext context) {  
        val interfaceType = findInterface(annotatedClass.interfaceName)  
  
        // add the interface to the list of implemented interfaces  
        annotatedClass.implementedInterfaces = annotatedClass.implementedInterfaces + #[interfaceType.newTypeReference]  
  
        // add the public methods to the interface  
        for (method : annotatedClass.declaredMethods) {  
            if (method.visibility == Visibility.PUBLIC) {  
                interfaceType.addMethod(method.simpleName) [  
                    docComment = method.docComment  
                    returnType = method.returnType  
                    for (p : method.parameters) {  
                        addParameter(p.simpleName, p.type)  
                    }  
                    exceptions = method.exceptions  
                ]  
            }  
        }  
    }  
}
```

# Predefined Annotations

```
@Singleton  
class SingletonA {  
  
    @Property  
    int a = 13 ;  
  
    @Property  
    int b ;  
  
    @Property  
    String c ;  
  
}
```

```
@Singleton  
@SuppressWarnings("all")  
public final class SingletonA {  
    private SingletonA() {  
        // singleton  
    }  
  
    private int _a = 13;  
  
    public int getA() {  
        return this._a;  
    }  
  
    public void setA(final int a) {  
        this._a = a;  
    }  
  
    private int _b;  
  
    public int getB() {  
        return this._b;  
    }  
  
    public void setB(final int b) {  
        this._b = b;  
    }  
  
    private String _c;  
  
    public String getC() {  
        return this._c;  
    }  
  
    public void setC(final String c) {  
        this._c = c;  
    }  
  
    private final static SingletonA INSTANCE = new SingletonA();  
  
    public static SingletonA getINSTANCE() {  
        return INSTANCE;  
    }  
}
```

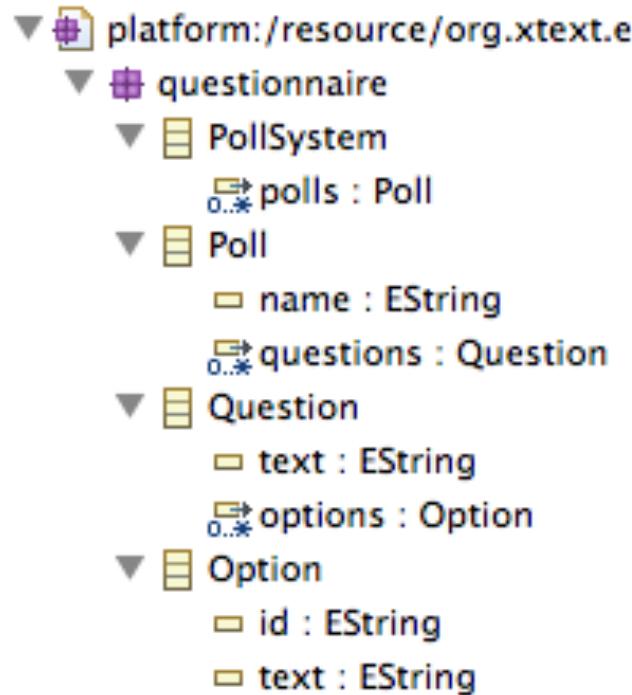
# Visitors, EMF, and Xtend

(key to M2M or M2T:  
iterate  
over the model)

```

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"
            }
        }
    }
}

```



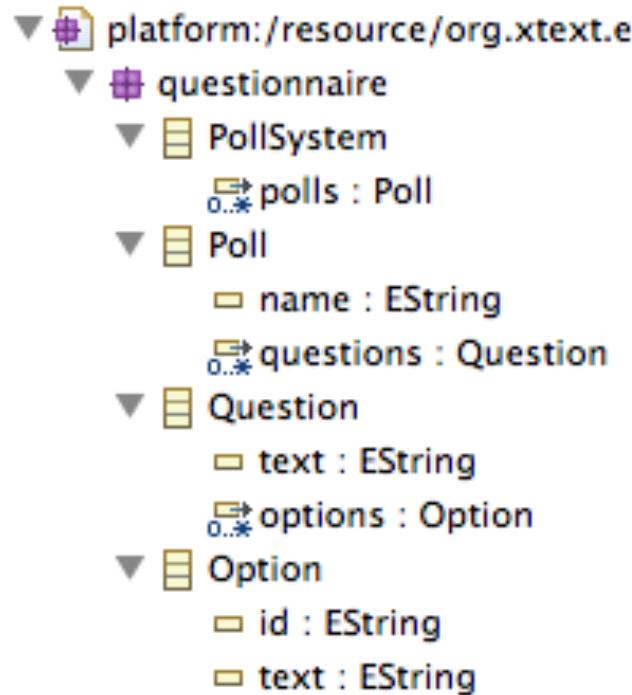
We already give examples of transformation, defined over the metamodel...

## Common point: the need to visit the model (graph)

```

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"
            }
        }
    }
}

```



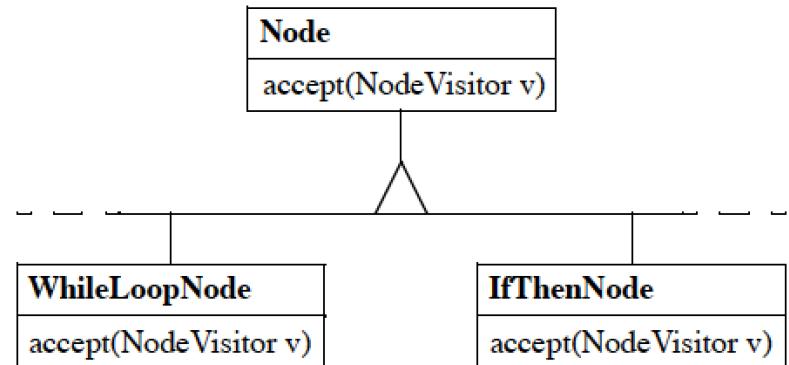
## Visit the model (graph)

Possible solution: a series of casts (lots of if-statements and traversal loops)

# Visitor Pattern

separating an algorithm from an object structure on which it operates

```
public class WhileLoopNode extends Node {  
    protected Node condition, body;  
    /* ... */  
    public void accept(NodeVisitor v) {  
        v.visitWhileLoop(this);  
    }  
}  
  
public class IfThenNode extends Node {  
    protected Node condition, thenBranch;  
    /* ... */  
    public void accept(NodeVisitor v) {  
        v.visitIfThen(this);  
    }  
}
```

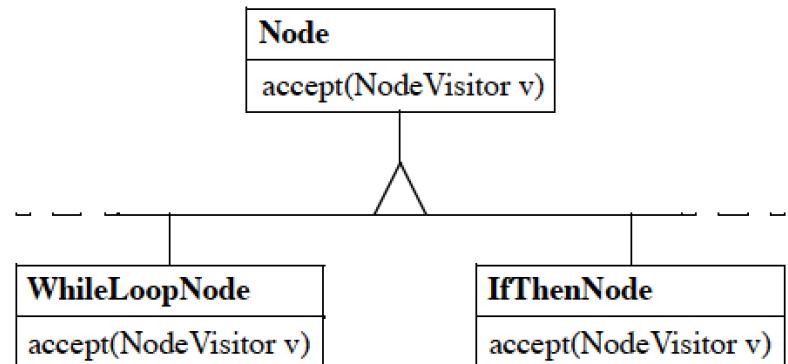


```
public abstract class NodeVisitor {  
    /* ... */  
    public abstract void visitWhileLoop(WhileLoopNode n);  
    public abstract void visitIfThen(IfThenNode n);  
}  
  
public class TypeCheckingVisitor extends NodeVisitor {  
    /* ... */  
    public void visitWhileLoop(WhileLoopNode n) { n.getCondition().accept(this); /* ... */ }  
    public void visitIfThen(IfThenNode n) { /* ... */ }  
}
```

new operations can be added modularly, without needing to edit any of the Node subclasses: the programmer simply defines a new NodeVisitor subclass containing methods for visiting each class in the Node hierarchy.

# Visitor Pattern (problems)

```
public class WhileLoopNode extends Node {  
    protected Node condition, body;  
    /* ... */  
    public void accept(NodeVisitor v) {  
        v.visitWhileLoop(this);  
    }  
}  
  
public class IfThenNode extends Node {  
    protected Node condition, thenBranch;  
    /* ... */  
    public void accept(NodeVisitor v) {  
        v.visitIfThen(this);  
    }  
}
```



```
public abstract class NodeVisitor {  
    /* ... */  
    public abstract void visitWhileLoop(WhileLoopNode n);  
    public abstract void visitIfThen(IfThenNode n);  
}  
  
public class TypeCheckingVisitor extends NodeVisitor {  
    /* ... */  
    public void visitWhileLoop(WhileLoopNode n) { n.getCondition().accept(this); /* ... */ }  
    public void visitIfThen(IfThenNode n) { /* ... */ }  
}
```

#1 stylized double-dispatching code is tedious to write and prone to error.

# Visitor Pattern (problems)

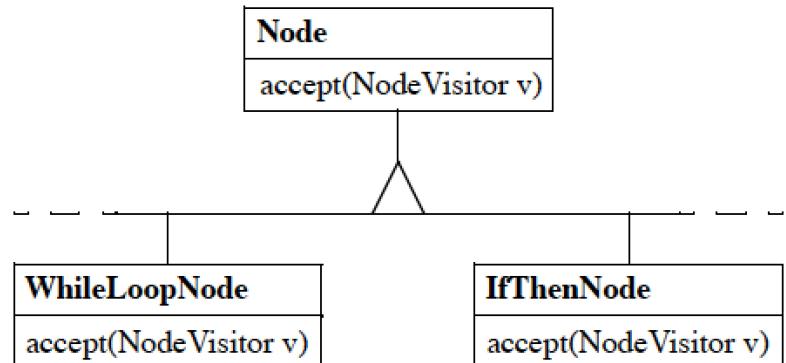
```
public class WhileLoopNode extends Node {  
    protected Node condition, body;  
    /* ... */  
    public void accept(NodeVisitor v) {  
        v.visitWhileLoop(this);  
    }  
}
```

```
public class IfThenNode extends Node {  
    protected Node condition, thenBranch;  
    /* ... */  
    public void accept(NodeVisitor v) {  
        v.visitIfThen(this);  
    }  
}
```

---

```
public abstract class NodeVisitor {  
    /* ... */  
    public abstract void visitWhileLoop(WhileLoopNode n);  
    public abstract void visitIfThen(IfThenNode n);  
}
```

```
public class TypeCheckingVisitor extends NodeVisitor {  
    /* ... */  
    public void visitWhileLoop(WhileLoopNode n) { n.getCondition().accept(this); /* ... */ }  
    public void visitIfThen(IfThenNode n) { /* ... */ }  
}
```



#2 the need for the Visitor pattern must be anticipated ahead of time, when the Node class is first implemented

# Visitor Pattern (problems)

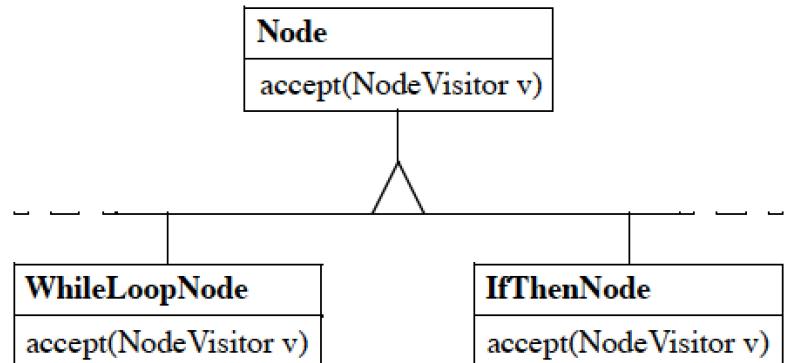
```
public class WhileLoopNode extends Node {  
    protected Node condition, body;  
    /* ... */  
    public void accept(NodeVisitor v) {  
        v.visitWhileLoop(this);  
    }  
}
```

```
public class IfThenNode extends Node {  
    protected Node condition, thenBranch;  
    /* ... */  
    public void accept(NodeVisitor v) {  
        v.visitIfThen(this);  
    }  
}
```

---

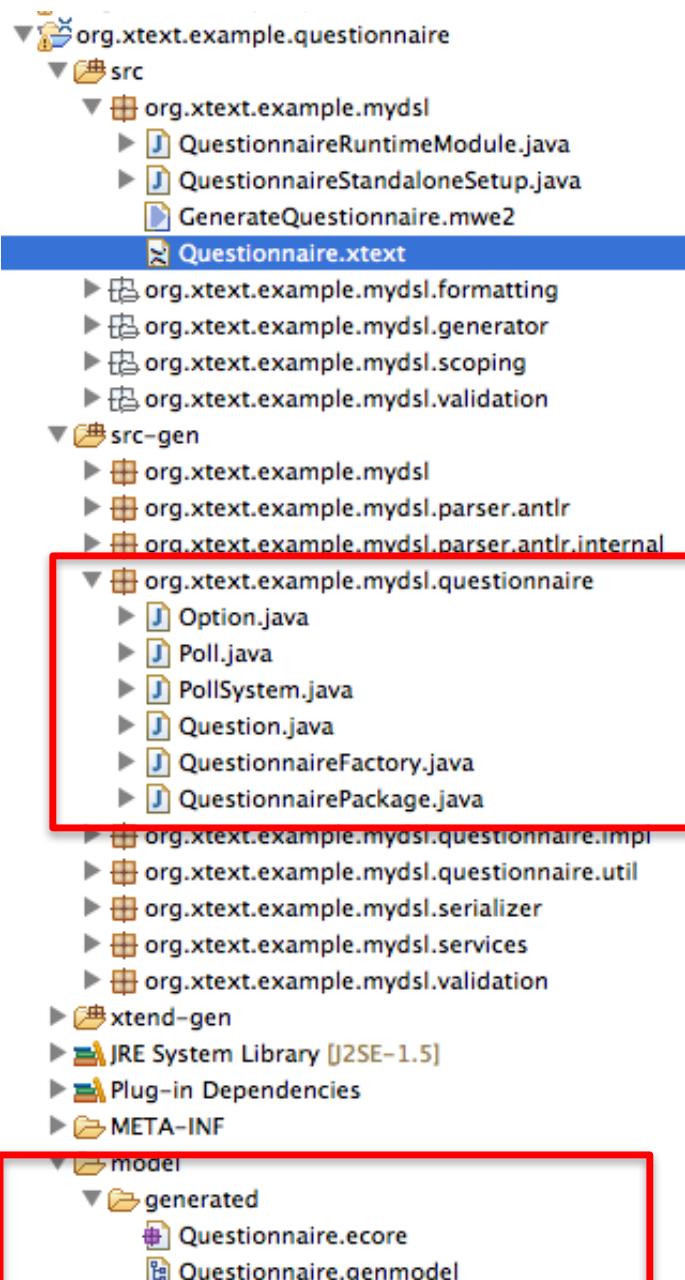
```
public abstract class NodeVisitor {  
    /* ... */  
    public abstract void visitWhileLoop(WhileLoopNode n);  
    public abstract void visitIfThen(IfThenNode n);  
}
```

```
public class TypeCheckingVisitor extends NodeVisitor {  
    /* ... */  
    public void visitWhileLoop(WhileLoopNode n) { n.getCondition().accept(this); /* ... */ }  
    public void visitIfThen(IfThenNode n) { /* ... */ }  
}
```



#3 class hierarchy evolution (e.g., new **Node** subclass) forces us to rewrite **NodeVisitor**

# Visitor Pattern (impact of the problem)



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

generate questionnaire "http://www.xtext.org/example/mydsl/Questionnaire"

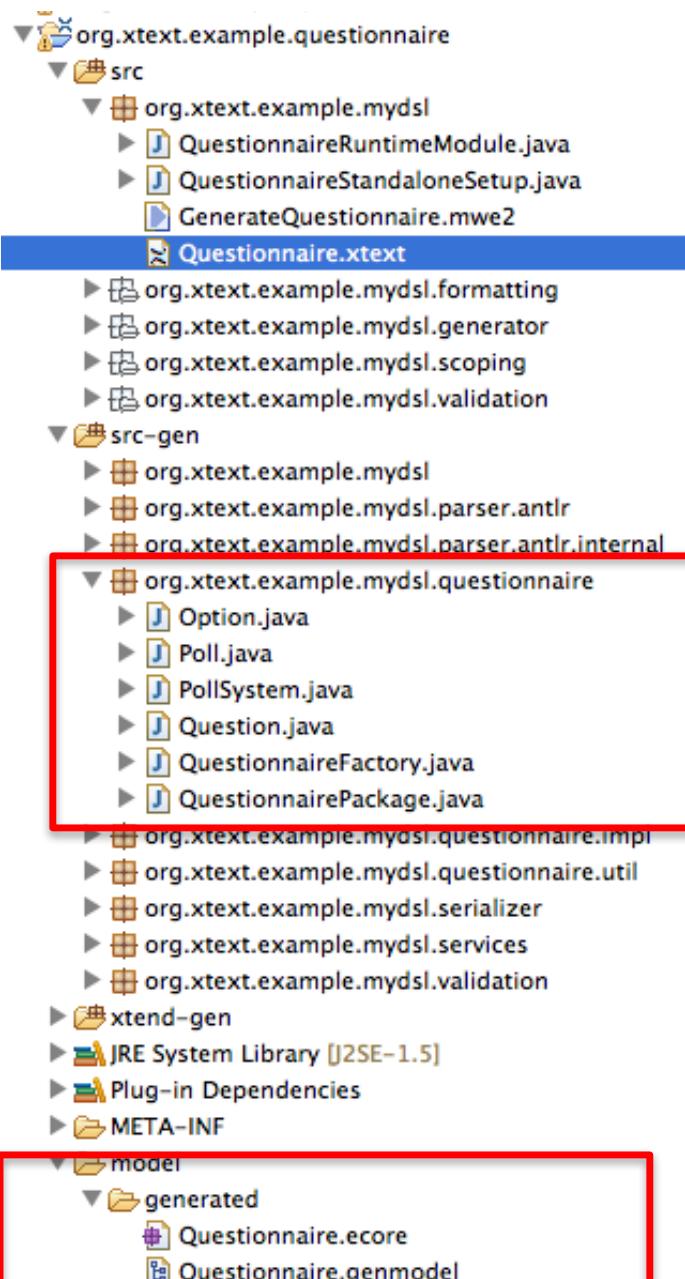
@PollSystem:
    'PollSystem' '{' polls+=Poll+ '}';

@Poll:
    'Poll' name=ID '{' questions+=Question+ '}';

Question : 'Question' ID? '{' text=STRING 'options'=options+=Option+ '}';

Option : id=ID ':' text=STRING ;
```

# Visitor Pattern (impact of the problem)



```
public interface Question extends EObject
{
    /**
     * Returns the value of the '{@link org.xtext.example.mydsl.questionnaire.Question#getText Question.getText}' attribute.
     * <!-- begin-user-doc -->
     * <p>
     * If the meaning of the '<er>' token really should be more ...
     * </p>
     * <!-- end-user-doc -->
     * @return the value of the ...
     * @see #setText(String)
     * @see org.xtext.example.mydsl.questionnaire.QuestionnairePackage#getQuestion_Text()
     * @model
     * @generated
     */
    String getText();

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

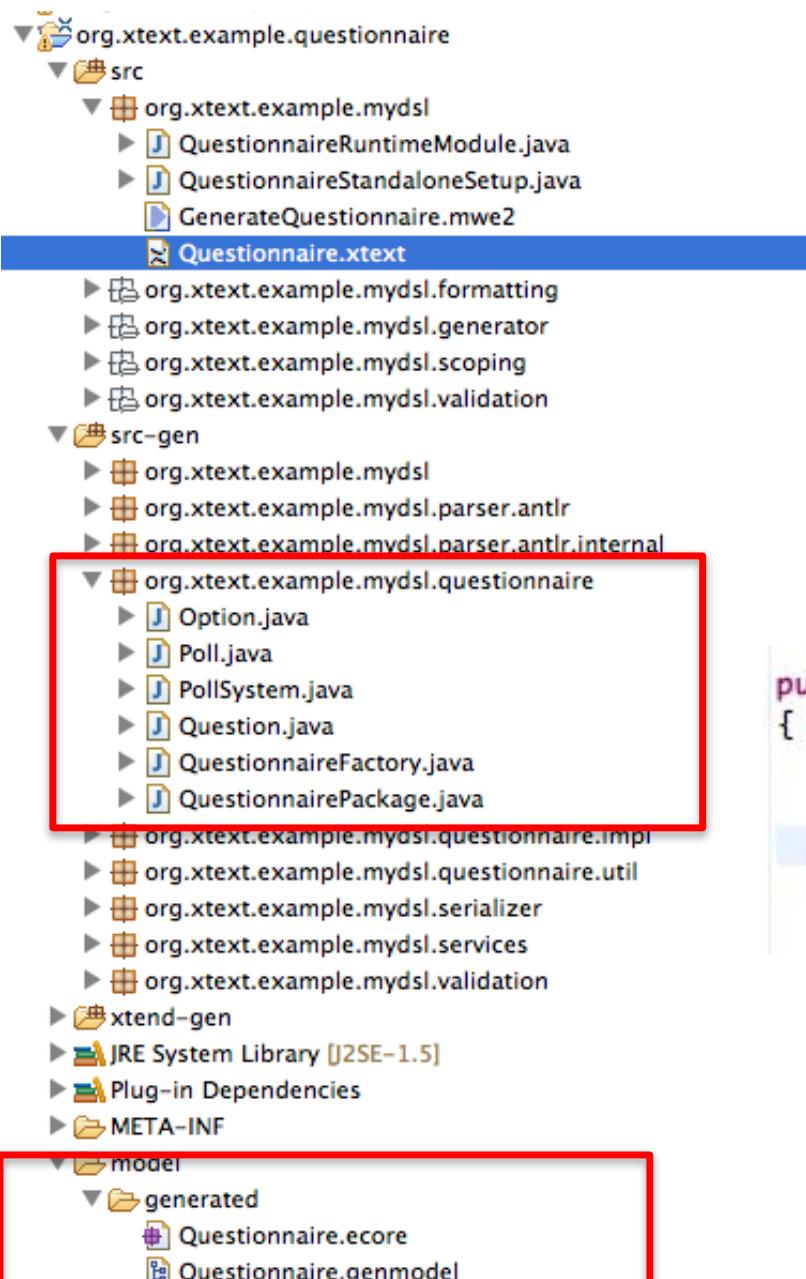
## Outline

```
org.xtext.example.mydsl.questionnaire
└ Question
    • getText() : String
    • setText(String) : void
    • getOptions() : EList<Option>
```

# No accept method

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

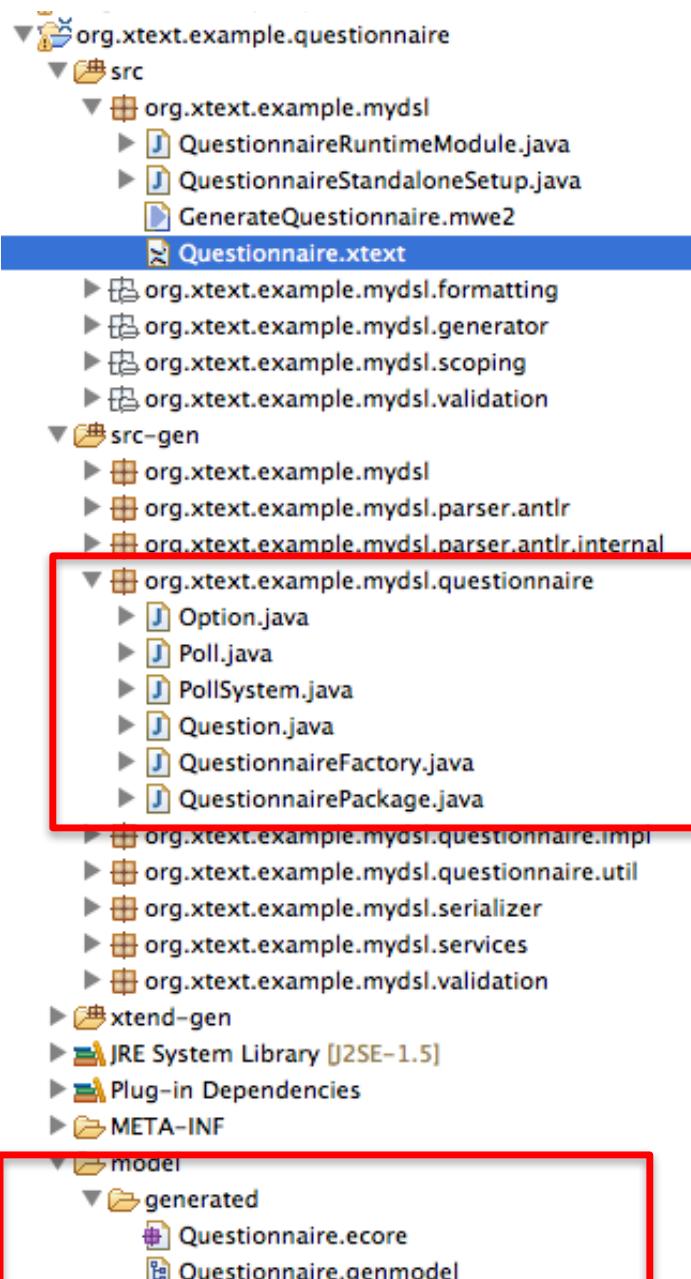
# Visitor Pattern (impact of the problem)



## Handcrafted code?

```
public interface Question extends EObject
{
    public void accept(QuestionnaireVisitor vis);
}
```

# Visitor Pattern (impact of the problem)



⇒ Manual  
⇒ Some classes are not concerned by the visit...

```
public interface Question extends EObject
{
    public void accept(QuestionnaireVisitor vis);
}
```

⇒ If Xtext Grammar changes,  
you can restart again

# Visitor Pattern (requirements)

#1 stylized double-dispatching code is tedious to write and prone to error.

## Automation

#2 the need for the Visitor pattern must be anticipated ahead of time, when the Node class is first implemented

**No accept method**

**Violation of open/close principle: no way**

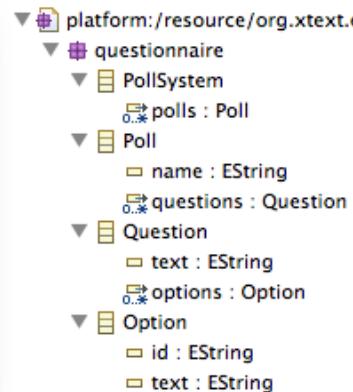
#3 class hierarchy evolution (e.g., new Node subclass) forces us to (completely) rewrite NodeVisitor

## Automation

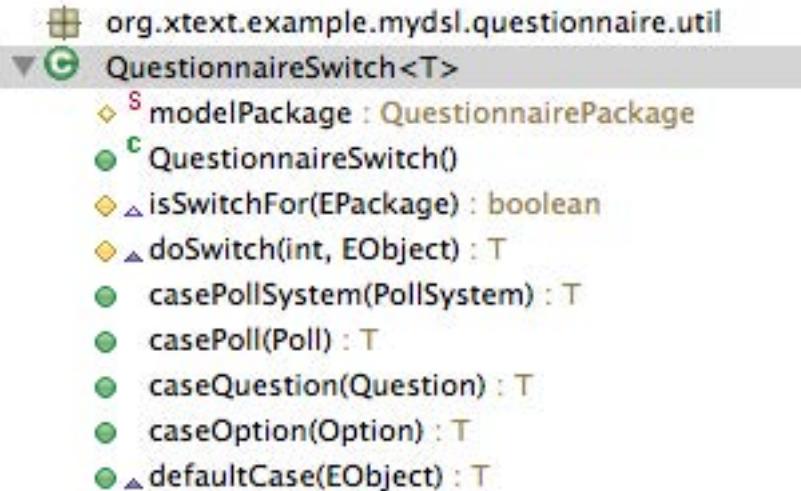
```

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"
            }
        }
    }
}

```



# Possible solution (1): « \*Switch » generated by... EMF



```

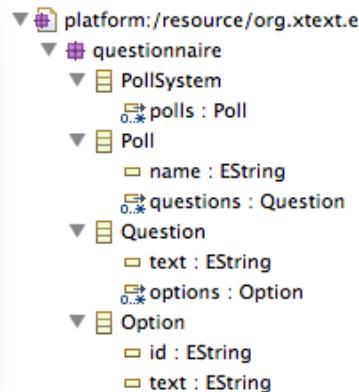
/**
 * The switch that delegates to the <code>createXXX</code> methods.
 * <!-- begin-user-doc -->
 * <!-- end-user-doc -->
 * @generated
 */
protected QuestionnaireSwitch<Adapter> modelSwitch =
    new QuestionnaireSwitch<Adapter>()
{
    @Override
    public Adapter casePollSystem(PollSystem object)
    {
        return createPollSystemAdapter();
    }
    @Override
    public Adapter casePoll(Poll object)
    {
        return createPollAdapter();
    }
    @Override
    public Adapter caseQuestion(Question object)
    {
        return createQuestionAdapter();
    }
    @Override
    public Adapter caseOption(Option object)
    {
        return createOptionAdapter();
    }
    @Override
    public Adapter defaultCase(EObject object)
    {
        return createEObjectAdapter();
    }
};

```

```

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 foo(PollSystem sys, Context c) {
    // treatment
}

```

`pollSystem.foo (new Context)`

## Possible solution (2): Extension Methods of Xtend

Context (classical with the Visitor)

Can be seen as a way  
to avoid a (very) long list of  
parameters and record  
the « state » of the visit

# @Aspect

(Active Annotations  
for implementing Visitors)

```
class A {  
    def boolean testReplacement() {  
        return false  
    }  
}
```

## Weaving methods

AspectA can handle a context in a proper way

```
@Aspect(className=typeof(A))  
abstract class AspectA {  
  
    def String foo() {  
        return "A"  
    }  
  
    abstract def String foofoo()  
}
```

```
@Test  
def void testA() {  
    val l = new A  
    l.foofoo  
}
```

```

override def doTransform(List<? extends MutableClassDeclaration> classes, extension TransformationContext context) {

    //Method name_parameterLengths,
    val Map<MutableClassDeclaration, List<MutableClassDeclaration>> superclass = new HashMap<MutableClassDeclaration, List<MutableClassDeclaration>>()
    val Map<MutableMethodDeclaration, Set<MutableMethodDeclaration>> dispatchmethod = new HashMap<MutableMethodDeclaration, Set<MutableMethodDeclaration>>()
    init_superclass(classes, context, superclass)
    init_dispatchmethod(superclass, dispatchmethod, context)

    for (clazz : classes) {

        //var List<String> inheritList1 = new ArrayList<String>() //sortByClassInheritance(clazz)

        var List<MutableClassDeclaration> listRes = sortByClassInheritance(clazz, classes,context)
        val List<String> inheritList = new ArrayList<String>()
        listRes.forEach[c| inheritList.add(c.simpleName)]
        listResMap.put(clazz,listRes)
        //sortByClassInheritance(clazz, inheritList1,context)

        /*val StringBuffer Log = new StringBuffer
        log.append("before ")
        inheritlist.forEach[ s | log.append(" " + s)]
        log.append("\n after ")
        inheritList1.forEach[ s | log.append(" " + s)]
        */
        //clazzaddError(log .toString)

        var classNam = clazz.annotations.findFirst[getValue('className') != null].getValue('className')
        //addError(clazz, classNam.class.toString)

        //var simpleNameF = classNam.eClass.ALLStructuralFeatures.findFirst[name == "simpleName"]
        //val className = classNam.eGet(simpleNameF) as String
        val className = classNam.class.getMethod("getSimpleName").invoke(classNam) as String
        //var identF = classNam.eClass.getAllStructuralFeatures().findFirst[name == "identifier"]
        //val identifier = classNam.eGet(identF) as String
        val identifier = classNam.class.getMethod("getIdentifier").invoke(classNam) as String
        val Map<MutableMethodDeclaration, String> bodies = new HashMap<MutableMethodDeclaration, String>()

        //clazzaddError(className)
        //MOVE non static fields
        fields_processing(context, clazz, className, identifier, bodies)

        //Transform method to static
        methods_processing(clazz, context, identifier, bodies, dispatchmethod, inheritList, className)

        aspectContextMaker(context, clazz, className, identifier)
    }
}

```

<https://github.com/diverse-project/k3/blob/master/k3-al/fr.inria.diverse.k3.al.annotationprocessor/src/main/java/fr/inria/diverse/k3/allannotationprocessor/Aspect.xtend>

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

[http://jnario.org/org/jnario/jnario/documentation/20Fac  
tsAboutXtendSpec.html](http://jnario.org/org/jnario/jnario/documentation/20Fac<br/>tsAboutXtendSpec.html)

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

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



# #1 Model Transformations

(importance, taxonomy, and  
some techniques -- templates,  
visitors, annotation processors)

# #2 Xtend

(A general purpose language  
with advanced features; an  
illustration on how to transform  
models in practice; Xtend  
written in Xtext, using MDE  
principles)

# #3 All together

Grammar, Metamodel,  
models/specifications, DSL/GPL,  
model transformations, meta-  
programming

## For breathing life into models!

# References

- Krzysztof Czarnecki, Simon Helsen: Feature-based survey of model transformation approaches. IBM Systems Journal 45(3): 621-646 (2006)
- Krzysztof Czarnecki and Lurich Eisenecker “Generative Programming: Methods, Tools, and Applications”
- Pierre-Alain Muller, Franck Fleurey and Jean-Marc Jézéquel “Weaving Executability into Object-Oriented Meta-Languages” MODELS’05
- Pierre-Alain Muller , Frédéric Fondement, Franck Fleurey, Michel Hassenforder, Rémi Schnakenburger, Sébastien Gérard, Jean-Marc Jézéquel “Model-driven analysis and synthesis of textual concrete syntax” SoSyM’08
- Sven Efftinge, Moritz Eysholdt, Jan Köhnlein, Sebastian Zarnekow, Robert von Massow, Wilhelm Hasselbring, and Michael Hanus. Xbase: Implementing domain-specific languages for java. GPCE ’12

# References

- M. Eysholdt and H. Behrens. “Xtext: Implement your language faster than the quick and dirty way.” In OOPSLA ’10
- Curtis Clifton, Gary T. Leavens, Craig Chambers, and Todd Millstein. “MultiJava: modular open classes and symmetric multiple dispatch for Java” OOPSLA’00
- Andy Schürr, Felix Klar “15 Years of Triple Graph Grammars.” ICGT 2008
- Mark Hills, Paul Klint, Tijs van der Storm, Jurgen J. Vinju “A Case of Visitor versus Interpreter Pattern.” TOOLS’11
- Thomas Degueule, Benoit Combemale, Arnaud Blouin, Olivier Barais and Jean-Marc Jézéquel “Melange: A Meta-language for Modular and Reusable Development of DSLs “ SLE’15