

# Domain-Specific Languages and Xtext

Mathieu Acher

Maître de Conférences

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

# Material

[https://github.com/acherm/  
teaching-MDE1920-MIAGE](https://github.com/acherm/teaching-MDE1920-MIAGE)

# Plan

- Domain-Specific Languages (DSLs)
  - Languages and abstraction gap
  - Examples and rationale
  - DSLs vs General purpose languages, taxonomy
- **External DSLs**
  - Grammar and parsing
  - Xtext

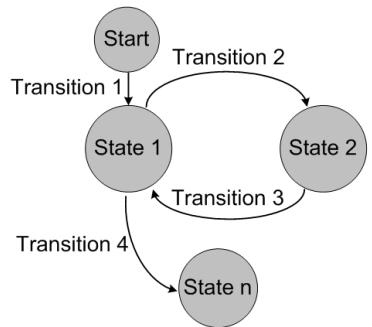
# Contract

- Better understanding/source of inspiration of software languages and DSLs
  - Revisit of history and existing languages
- Foundations and practice of Xtext
  - State-of-the-art language workbench (Most Innovative Eclipse Project in 2010, mature and used in a variety of industries)

# BIBTEX



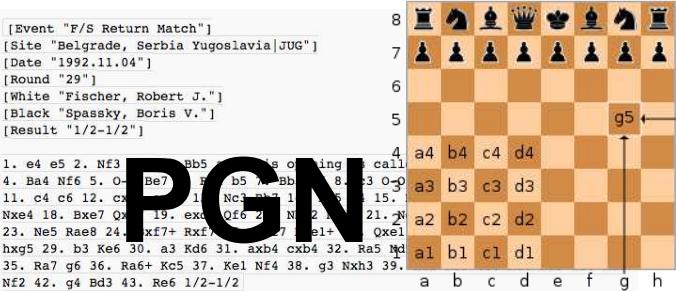
**Graphviz**



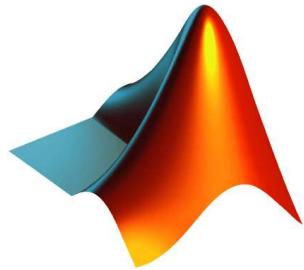
**Finite State  
Machine**



**Domain-Specific Languages (DSLs)**



**Make**



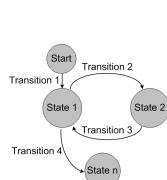
**Matlab**



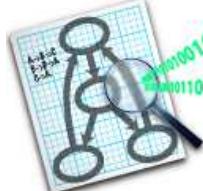
# DSL = Syntax + Services

## Specialized notation:

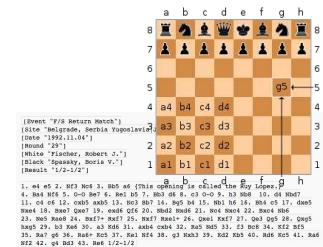
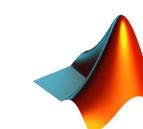
Textual or Graphical  
Specific Vocabulary  
Idiomatic constructs



BIBT<sub>E</sub>X



SQL



## Specialized tools/IDE:

Editor with auto-completion, syntax highlighting, etc.  
Compiler  
Interpreter  
Debugger  
Profiler  
Syntax/Type Checker  
...

# Language workbenches

- Tools for reducing the gap between the design and implementation of (external) domain-specific languages
- The Killer App for DSLs? <http://www.martinfowler.com/articles/languageWorkbench.html>

# Language Workbenches

Erdweg et al. SLE'13

		Ensō	Más	MetaEdit+	MPS	Onion	Rascal	Spoofax	SugarJ	Whole	Xtext
Notation	Textual	●	●		●	●	●	●	●	●	●
	Graphical	●	○	●			○			●	
	Tabular		●	●	●					●	
	Symbols			●	●					●	
Semantics	Model2Text		●	●	●	●	●	●	●	●	●
	Model2Model			●	●	●	●	●	●	●	●
	Concrete syntax			●	●	●	●	●	●		
	Interpretative	●		●	●		○	●		●	●
Validation	Structural	●	●	●	●	●	●	●	●	●	●
	Naming	○	●	●	●	●		●		●	○
	Types				●				●		●
	Programmatic	●			●	●	●	●	●		●
Testing	DSL testing				●		○	●		●	●
	DSL debugging	●		●	●		●			●	●
	DSL prog. debugging	●			●					●	●
Composability	Syntax/views	●		●	●	●	●	●	●	●	○
	Validation			●	●	●	●	●	●	●	●
	Semantics	●		●	●	●	●	●	●		●
	Editor services			●	●	●	●	●	●		●
Editing mode	Free-form	●		●		●	●	●	●		●
	Projectional		●		●	●				●	
Syntactic services	Highlighting	○	●	●	●	●	●	●	●	●	●
	Outline			●	●	●	●	●	●	●	●
	Folding	●	●	●	●	●	●	●	●	●	●
	Syntactic completion			●	●	●		●	●		●
	Diff	●		●	●	●	●	●	●		●
	Auto formatting	●	●	●	●	●	●	●		●	●
Semantic services	Reference resolution		●	●	●	●	●	●	●		●
	Semantic completion		●	●	●	●	●	●	●	●	●
	Refactoring	○	●	●	●		●	●		●	
	Error marking	●	●	●	●	●	●	●	●	●	●
	Quick fixes				●						●
	Origin tracking	●		●	●		●	●	●	●	●
	Live translation		●		●	●	○	●	●	●	●

Table 1: Language Workbench Features (● = full support, ○ = partial/limited support)

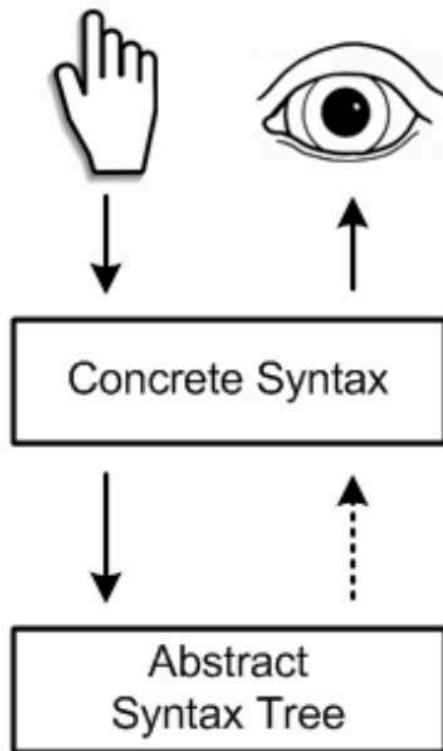
The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** Java – strategoxt-sugar-papers/test/BookHandler.sugj – Eclipse – /Users/seba/tmp/ecli...
- Toolbar:** Includes icons for file operations, search, and transform.
- Left Margin:** Shows error markers (yellow exclamation, red X) and a vertical scroll bar.
- Left View:** Displays two files: BookSchema.sugj and BookHandler.sugj. The BookHandler.sugj file contains Java code for a SAX ContentHandler named BookHandler. It includes imports for xml.Sugar, xml.Editor, and xml.schema.BookSchema. The class definition includes a method appendBook that creates an XML document fragment representing a book by Sidney W. Mintz, with editions from 1985 and 1986.
- Outline View:** Located on the right, it shows the XML schema structure:
  - BookHandler
    - appendBook
      - book
        - author
        - editions
          - edition
          - edit
  - Problems View:** Shows 1 error and 1 warning.
    - Errors (1 item):** expected element edition of namespace lib (at line 18)
    - Warnings (1 item):** skipping validation of quoted attribute value (at line 14)
  - Search View:** A search bar at the bottom of the outline view.
  - Bottom Bar:** Includes tabs for Writable and Smart Insert.

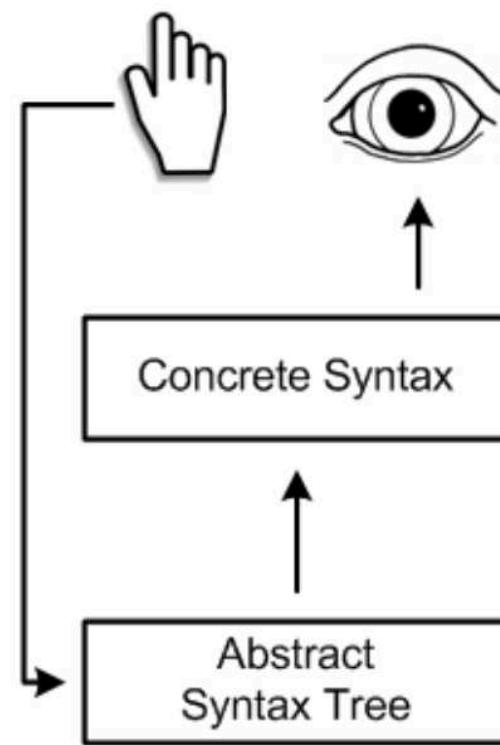
Sebastian Erdweg, Tillmann Rendel, Christian Kästner, and Klaus Ostermann. Sugarj: Library-based syntactic language extensibility. OOPSLA'11

# Projectional editing

**Parsing**



**Projection**



# Projectional editing

```
exported component Judge extends nothing {
    provides FlightJudger judger
    int16 points = 0;
    void judger_reset() <= op judger.reset {
        points = 0;
    } runnable judger_reset
    void judger_addTrackpoint(Trackpoint* tp) <= op judger.addTrackpoint {
        points += 0
            | tp->alt <= 2000 m | tp->alt >= 2000 m
            | tp->speed < 150 mps | 0 | 10
            | tp->speed >= 150 mps | 5 | 20
    } runnable judger_addTrackpoint
    int16 judger_getResult() <= op judger.getResult {
        return points;
    } runnable judger_getResult
} component Judge
```

# Projectional Editing

exported statemachine FlightAnalyzer initial = beforeFlight {		reset()
beforeFlight	next(Trackpoint* tp) [tp->alt == 0 m] -> airborne	
airborne	[tp->alt == 0 m && tp->speed == 0 mps] -> crashed [tp->alt == 0 m && tp->speed > 0 mps] -> landing [tp->speed > 200 mps && tp->alt == 0 m] -> airborne [tp->speed > 100 mps && tp->speed <= 200 mps && tp->alt == 0 m] -> airborne	[ ] -> beforeFlight
landing	[tp->speed == 0 mps] -> landed [tp->speed > 0 mps] -> landing	[ ] -> beforeFlight
landed		[ ] -> beforeFlight
crashed		
}		

```

SM.sdf3
System.Machine = [
  state machine [ID] [Extends]
  [{Element "\n"}*]
]

Extends.Extends =
[extends [ID]]

Extends.NoExtends = □

Element.State =
[state [ID]]

Element.Transition = [
  transition from [StateRef] to
  [Guard] [Actions]
]

names.nab
11 Machine(m, elems, extends) :
12   defines Machine m
13   scopes State, Variable
14
15 Extends(m) :
16   imports State, Variable from A
17
18 State(s) :
19   defines State s
20
21 StateRef(s) :
22   refers to State s
23
24 VarDef(x, c) :
25   defines Variable x of type t
26   where c has type t

types.ts
6 False() : BoolType()
7 True() : BoolType()
8
9
10 Var(x) : t
11 where definition of x : t
12
13 Or(e1, e2) + And(e1, e2) :
14   where e1 : BoolType()
15     else error "bool exp"
16     and e2 : BoolType()
17     else error "bool exp"
18
19 Eq(e1, e2) + Gt(e1, e2) :
20   where e1 : IntType()
21     else error "int exp"

generate.str
6 sm-to-java :
7   machine@Machine(m, exten
8   public class [m] [<ext
9     String current = [<
10    [vardefs]
11
12    String next(String e
13      [cond-stat*]
14      while(true) {
15        [uncond-stat*]
16      }
17    }
18  ]
19  ]
20
21 where ...

VendingMachine.aterm
1 Machine(
2   "VendingMachine"
3   , NoExtends()
4   , [VarDef("drinks", Int("10")))
5   , VarDef("sweets", Int("20")))
6   , State("Waiting"))

```

# The Spoofax Language Workbench

Spoofax is a platform for developing textual domain-specific languages with full-featured [Eclipse](#) editor plugins.

With the Spoofax language workbench, you can write the grammar of your language using the high-level SDF grammar formalism. Based on this grammar, basic editor services such as syntax highlighting and code folding are automatically provided. Using high-level descriptor languages, these services can be customized. More sophisticated services such as error marking and content completion can be specified using rewrite rules in the Stratego language.

## Meta Languages

Language definitions in Spoofax are constructed using the following meta-languages:

- The [SDF3](#) syntax definition formalism
- The [NaBL](#) name binding language
- The [TS](#) type specification language
- The [Stratego](#) transformation language

xtext

Xtext, a popular, easy-to-use model-based tool  
for developping DSLs

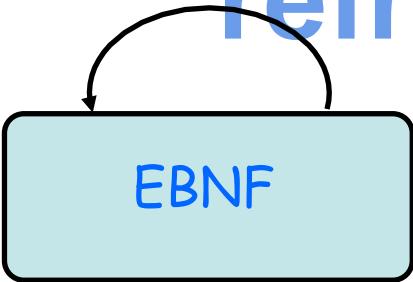
Your DSL in 5' (incl.  
editors and serializers)

Your DSL in 5'

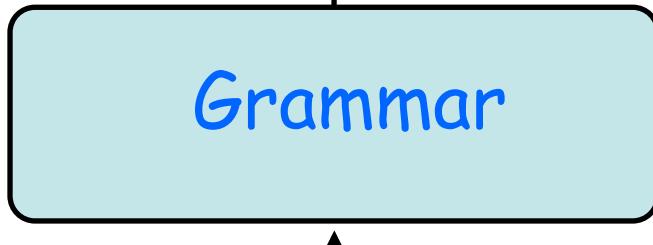
Short Demonstration

# Foundations (or some course refresh)

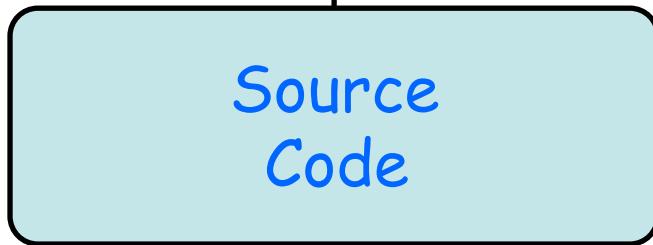
M<sup>3</sup>



M<sup>2</sup>



M<sup>1</sup>



Java Grammar

```
CHARLITERAL
: '\'' 
| '\"' EscapeSequence
| ~('\'' | '\"' | '\r' | '\n')
| '\\';

STRINGLITERAL
: '\"' 
| '\'' EscapeSequence
| ~('\"' | '\'' | '\r' | '\n')
| '\"'
| '\''* '\"';

fragment
EscapeSequence
: '\\\\' (
    'b'
| 't'
| 'n'
| 'r'
| '\"'
| '\'');
```

Java Program

```
/*
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World");
    }
}
```

```
classOrInterfaceDeclaration
: classDeclaration
| interfaceDeclaration
;

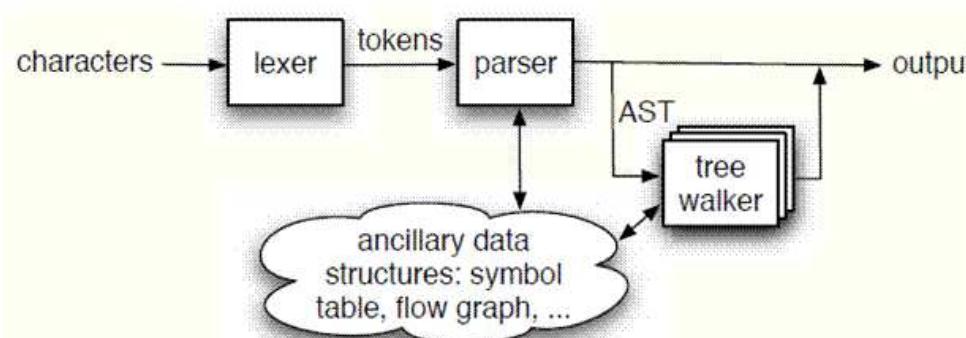
modifiers
: (
    annotation
| PUBLIC
| PROTECTED
| PRIVATE
| STATIC
| ABSTRACT
| FINAL
| NATIVE
| SYNCHRONIZED
| TRANSIENT
| VOLATILE
| STRICTFP
)*
;

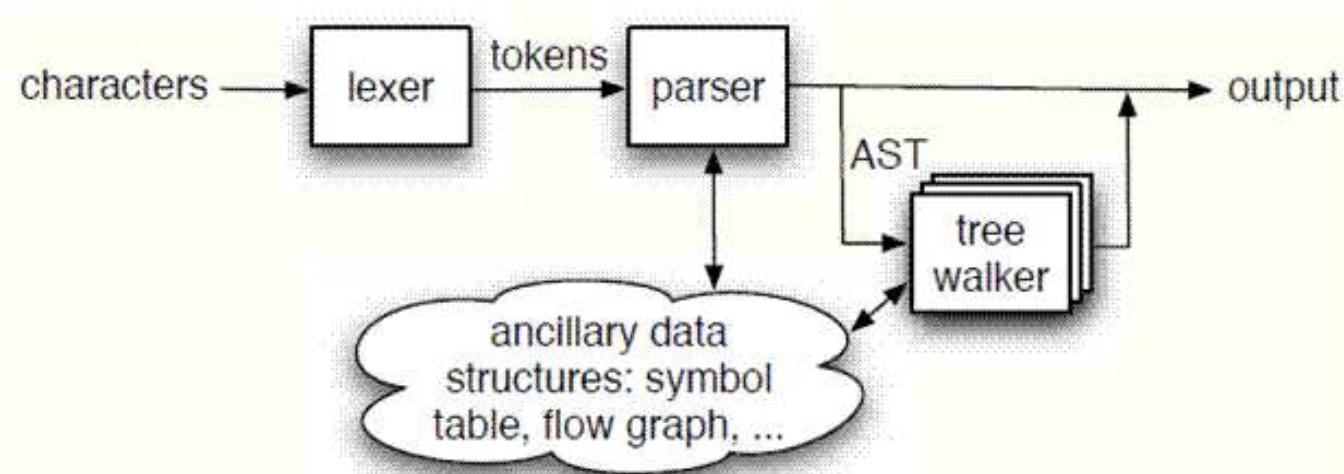
variableModifiers
: (
    FINAL
| annotation
)*
;

classDeclaration
: normalClassDeclaration
| enumDeclaration
```

# Compilation Process

- Source code
  - Concrete syntax used for specifying a program
  - Conformant to a grammar
- Lexical analysis
  - Converting a sequence of characters into a sequence of **tokens**
- Parsing (Syntactical analysis)
  - Abstract Syntax Tree (AST)





The Definitive  
ANTLR  
Reference

Building Domain-  
Specific Languages



Terence Parr

```

CHARLITERAL
:   '\'' 
|   ( EscapeSequence
|   ~('\''|'\\"'|'\r'|'\n')
)
'\''
;

STRINGLITERAL
:   """
|   ( EscapeSequence
|   ~('\''|'"'|'\r'|'\n')
)*
"""
;

fragment
EscapeSequence
:   '\\' (
    'b'
|   't'
|   'n'
|   'f'
|   'r'
|   '\"'
)
;
```

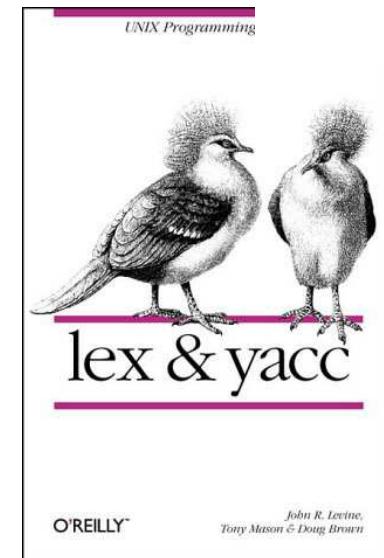
```

classOrInterfaceDeclaration
:   classDeclaration
|   interfaceDeclaration
;

modifiers
:   (
    annotation
|   PUBLIC
|   PROTECTED
|   PRIVATE
|   STATIC
|   ABSTRACT
|   FINAL
|   NATIVE
|   SYNCHRONIZED
|   TRANSIENT
|   VOLATILE
|   STRICTFP
)*
;

variableModifiers
:   (
    FINAL
|   annotation
)*
;

classDeclaration
:   normalClassDeclaration
|   enumDeclaration
;
```

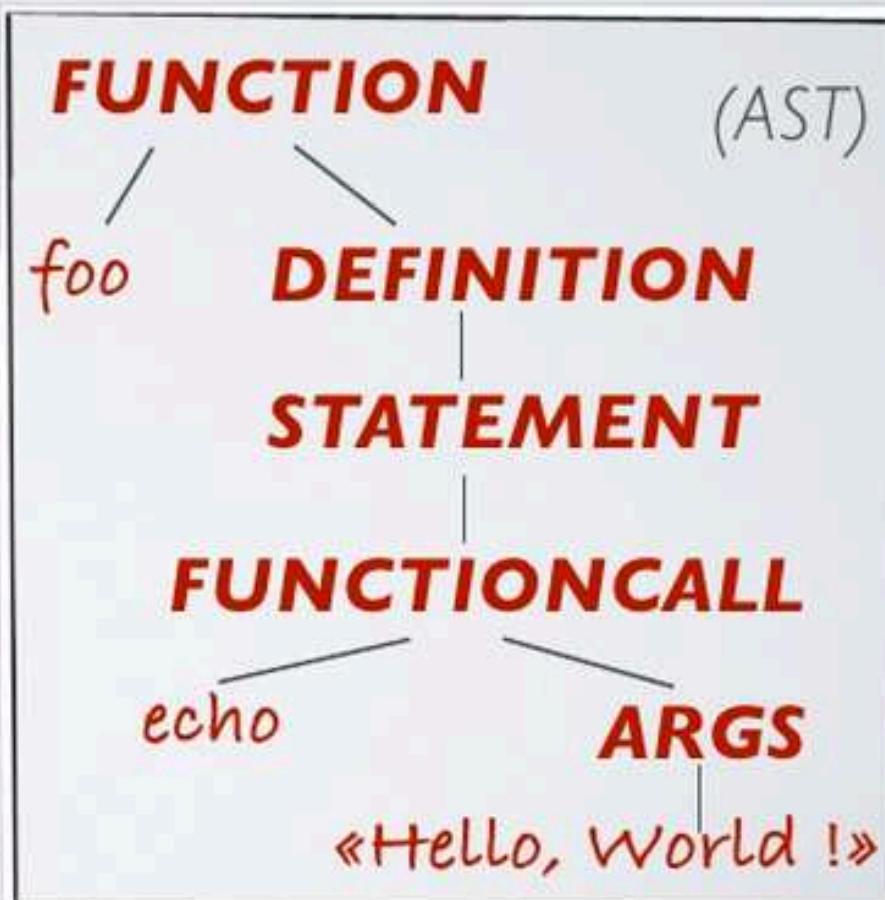
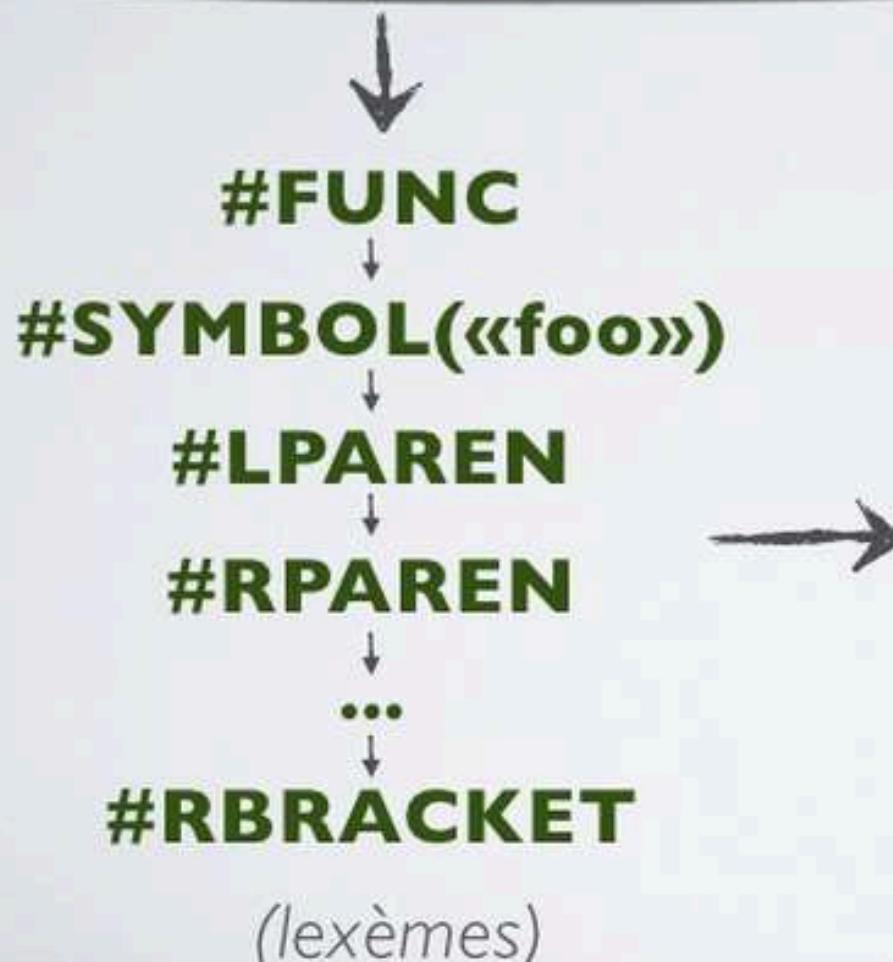


O'REILLY®

John R. Levine,  
Tony Mason & Doug Brown

# EXEMPLE

```
function foo() {  
    echo «Hello, World !»;  
}  
(Syntaxe concrète)
```



```

class StringInterp {
    val int = 42
    val dbl = Math.PI
    val str = "My hovercraft is full of eels"

    println(s"String: $str Double: $dbl Int: $int Int Expr: ${int * 1.0}")
}

```

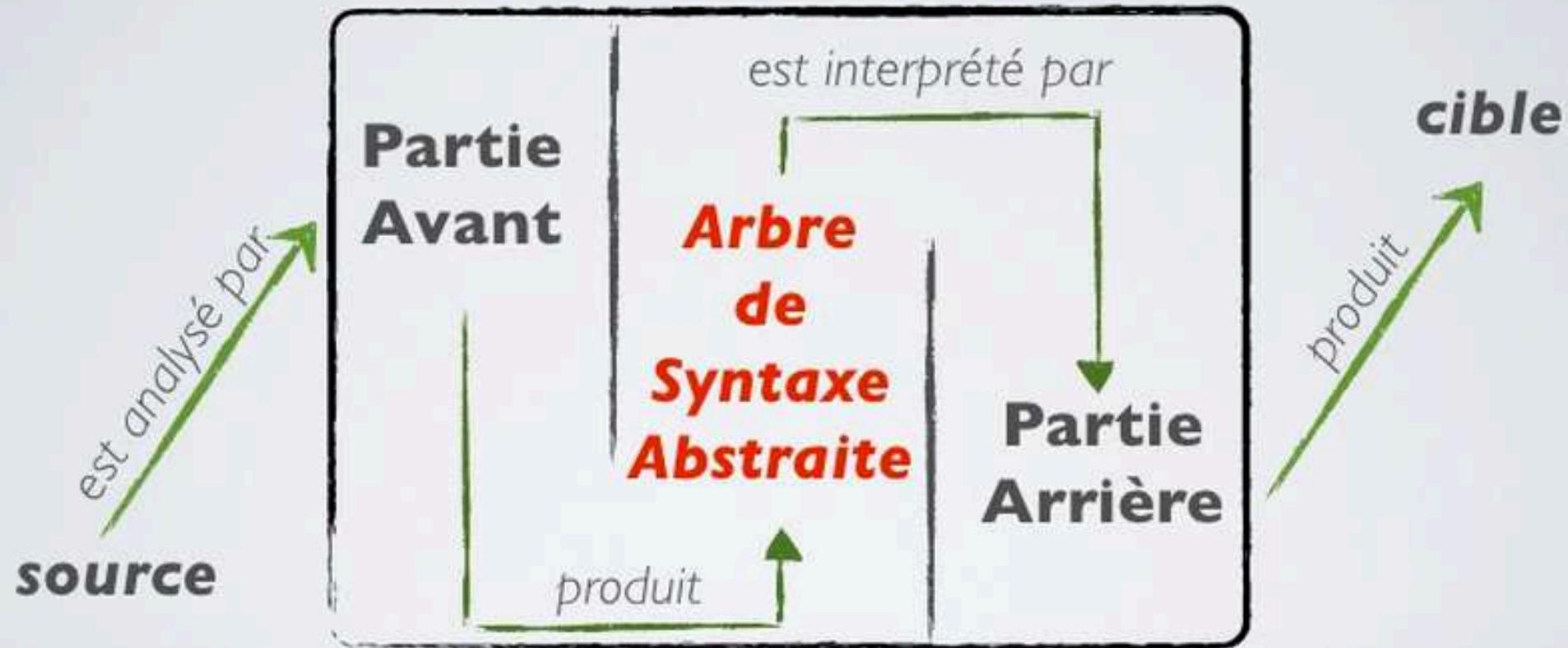
## Scala AST (example)

```

Block(
  List(
    ClassDef(Modifiers(), TypeName("StringInterp"), List(), Template(
      List(Ident(TypeName("AnyRef"))), noSelfType, List(DefDef(Modifiers(), termNames.CONSTRUCTOR,
      List(),
      List(List())),
      TypeTree(), Block(List(Apply(Select(Super(This(typeNames.EMPTY), typeNames.EMPTY),
      termNames.CONSTRUCTOR), List()))), Literal(Constant(()))), ValDef(Modifiers(), TermName("int"),
      TypeTree(), Literal(Constant(42))), ValDef(Modifiers(), TermName("dbl"), TypeTree(),
      Literal(Constant(3.141592653589793))), ValDef(Modifiers(), TermName("str"), TypeTree(),
      Literal(Constant("My hovercraft is full of eels"))), Apply(Select(Ident(scala.Predef),
      TermName("println")), List(Apply(Select(Apply(Select(Ident(scala.StringContext), TermName("apply")),
      List(Literal(Constant("String: ")), Literal(Constant(" Double: ")), Literal(Constant(" Int: ")),
      Literal(Constant(" Int Expr: ")), Literal(Constant(""))))), TermName("s")),
      List(Select(This(TypeName("StringInterp")), TermName("str")), Select(This(TypeName("StringInterp")),
      TermName("dbl")), Select(This(TypeName("StringInterp")), TermName("int")),
      Apply(Select(Select(This(TypeName("StringInterp")), TermName("int")), TermName("$times")),
      List(Literal(Constant(1.0))))))), TermName("s"))),
      List(Literal(Constant(())))))))

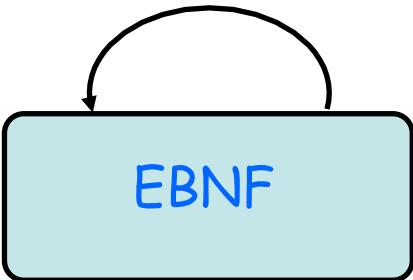
```

# Compilation (en français)

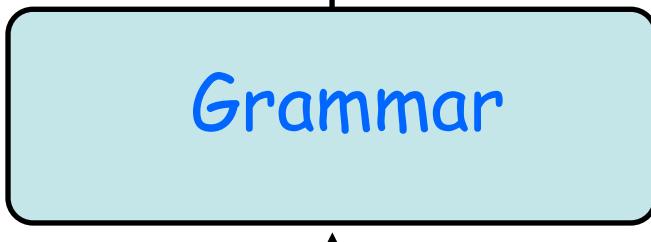


# DSL? The same!

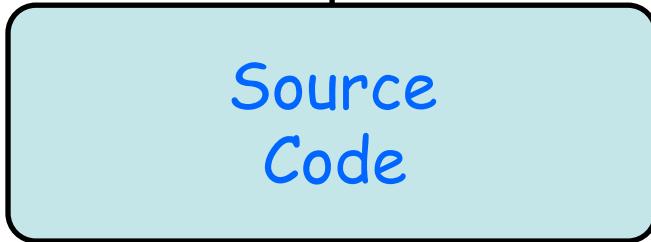
M<sup>3</sup>



M<sup>2</sup>



M<sup>1</sup>



DSL Grammar

DSL specification/  
program

*UNIX Programming Tools*



O'REILLY™

*John R. Levine,  
Tony Mason & Doug Brown*

Programmer's  
Reference

The Definitive  
**ANTLR**  
Reference

Building Domain-  
Specific Languages

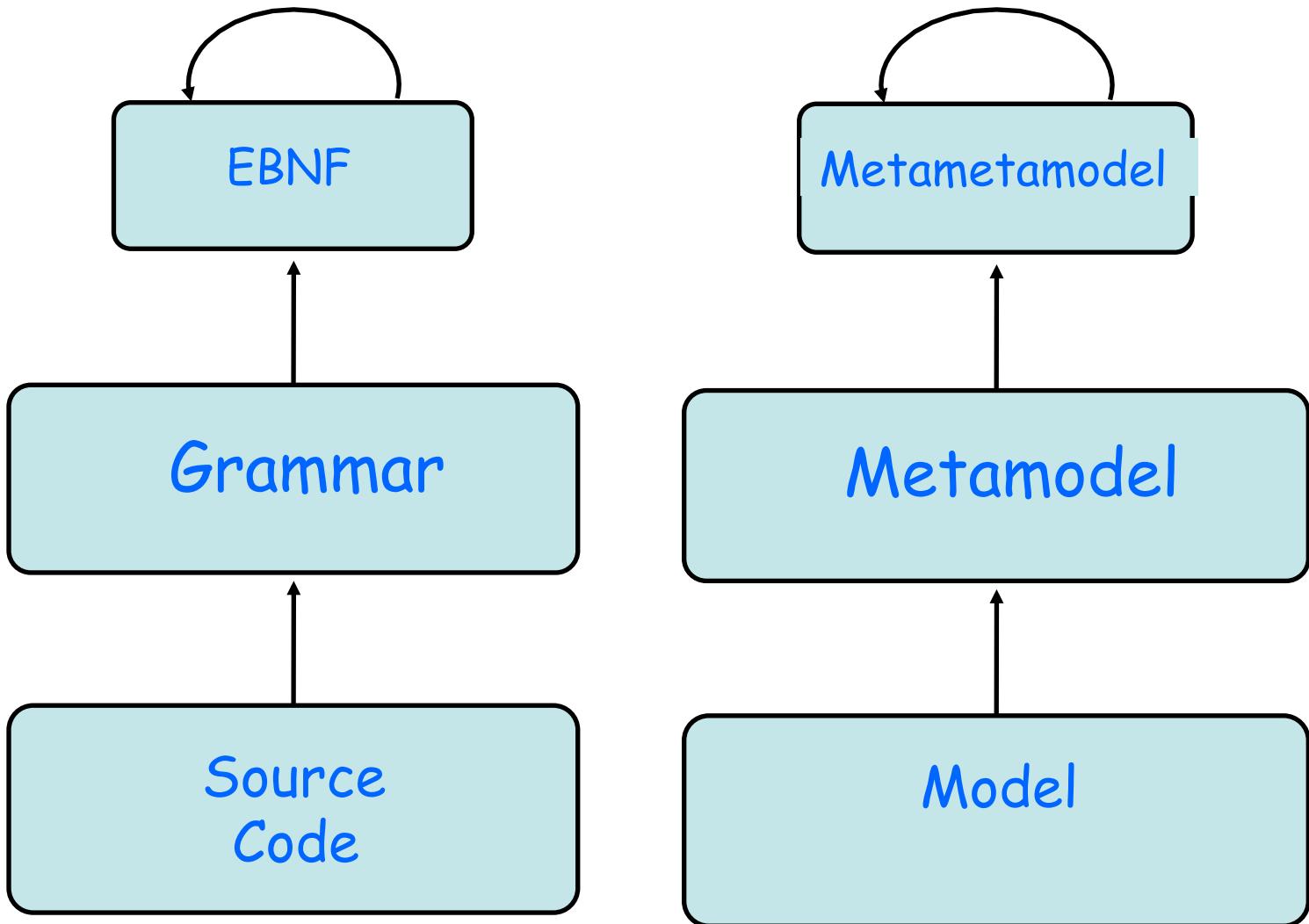


Terence Parr

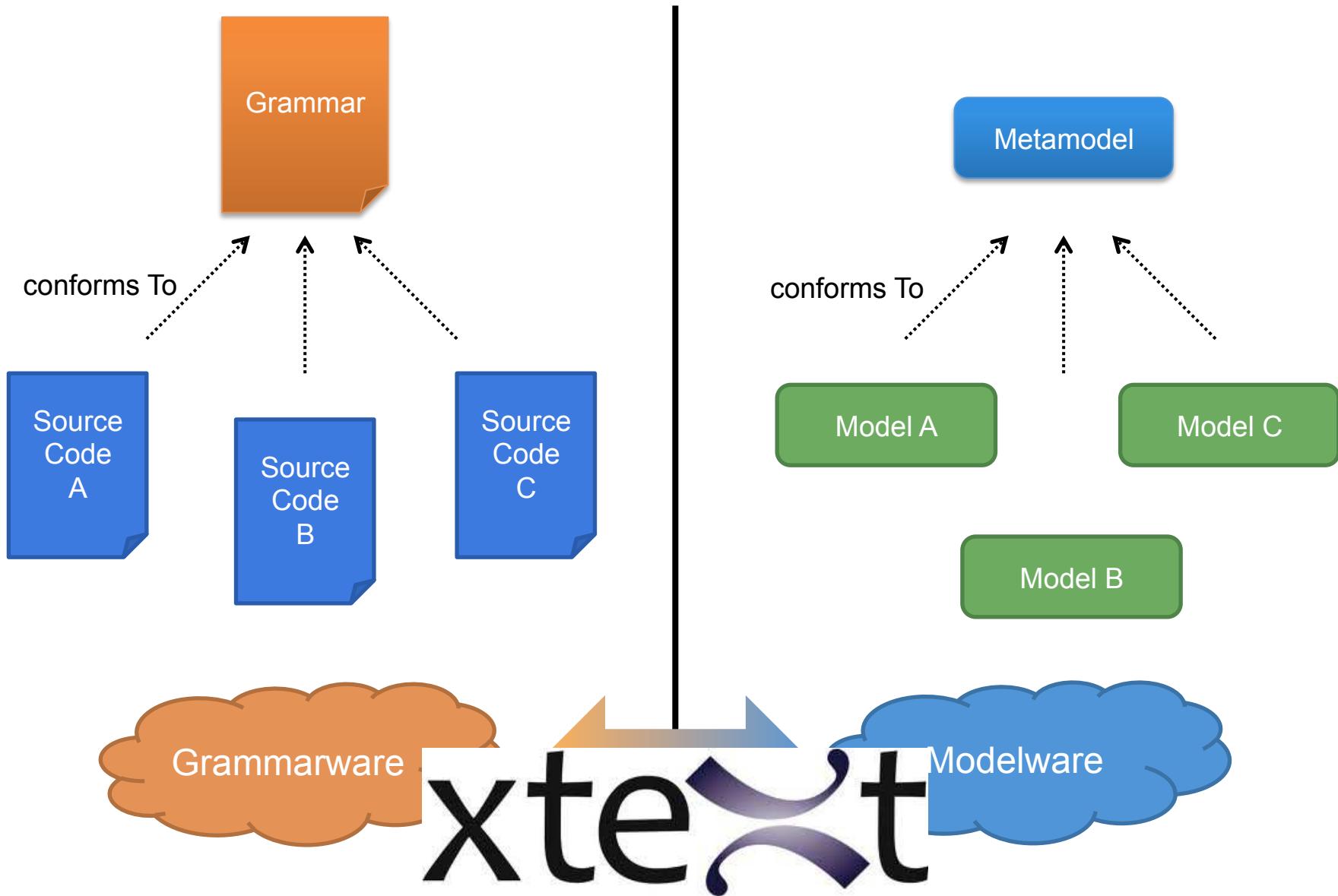
M<sup>3</sup>

M<sup>2</sup>

M<sup>1</sup>



# Language and MDE





Give me a **grammar**,

I'll give you (for free)

- \* a comprehensive editor (auto-completion, syntax highlighting, etc.) in Eclipse
- \* an Ecore metamodel and facilities to load/serialize/visit conformant models (Java ecosystem)
- \* extension to override/extend « default » facilities (e.g., checker)

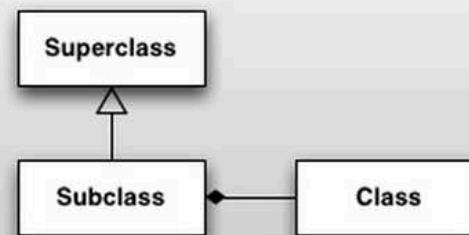
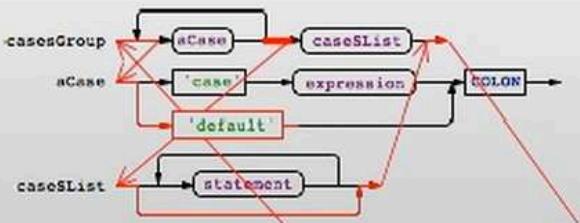
Model

Grammar

Xtext  
Generator



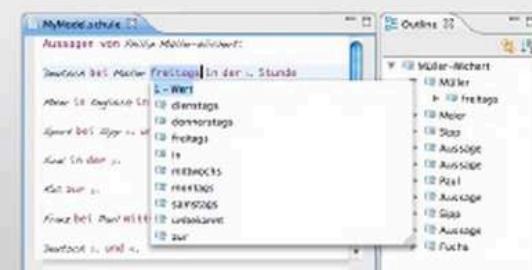
# Xtext Runtime



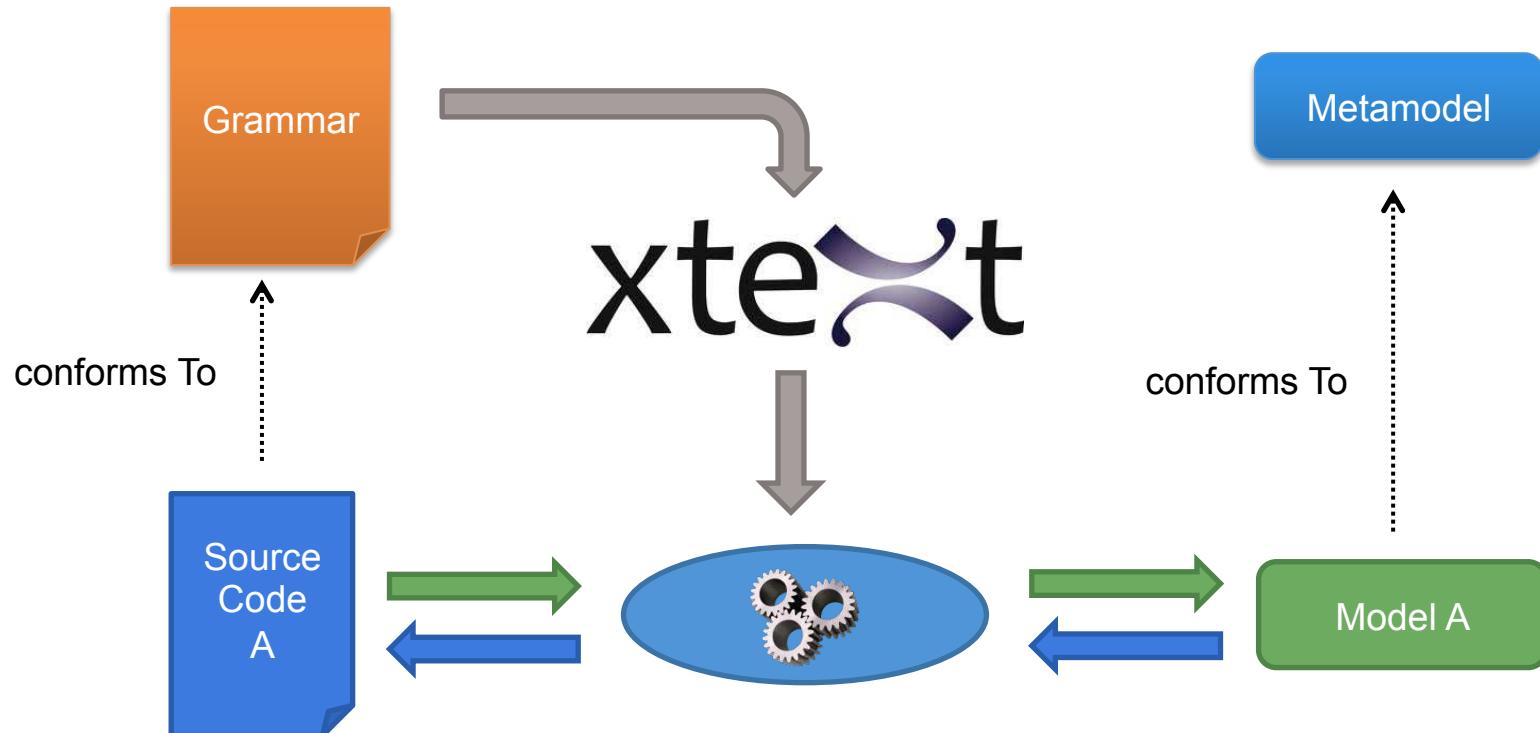
LL(\*) Parser

ecore meta model

editor



# Xtext, Grammar, Metamodel

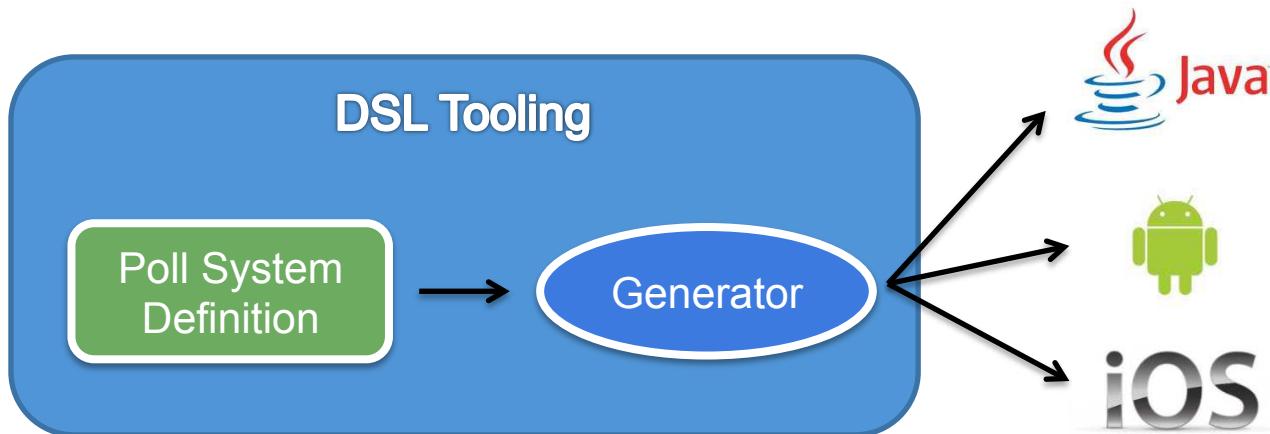


# The Grammar Language of Xtext

- Corner-stone of Xtext
- A... DSL to define textual languages
  - Describe the concrete syntax
  - Specify the mapping between concrete syntax and domain model
- From the grammar, it is generated:
  - The domain model
  - The parser
  - The tooling

# A first example

- Poll System application
  - Define a Poll with the corresponding questions
  - Each question has a text and a set of options
  - Each option has a text
- Generate the application in different platforms



# Something like...

## DSL Tooling

```
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"  
            }  
        }  
    }  
}
```



# Xtext Grammar

Grammar  
definition →

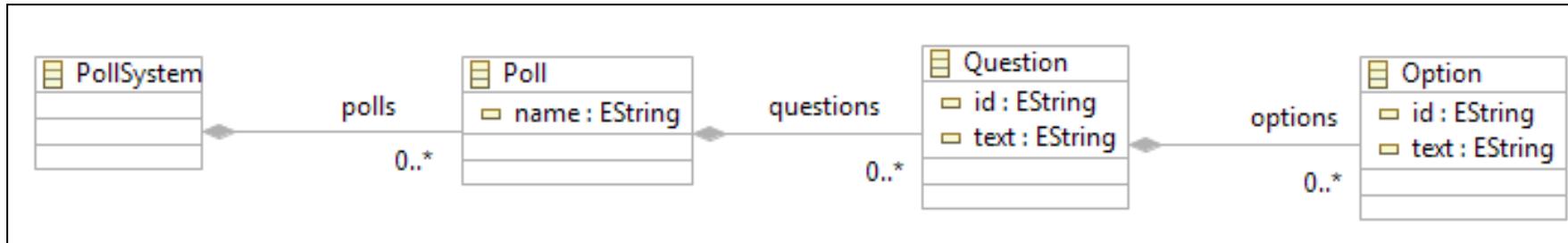
```
grammar fr.miage.xtext.Poll with org.eclipse.xtext.common.Terminals

generate poll "http://www.miage.fr/xtext/Poll"

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

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

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



# Xtext Grammar

Grammar  
reuse

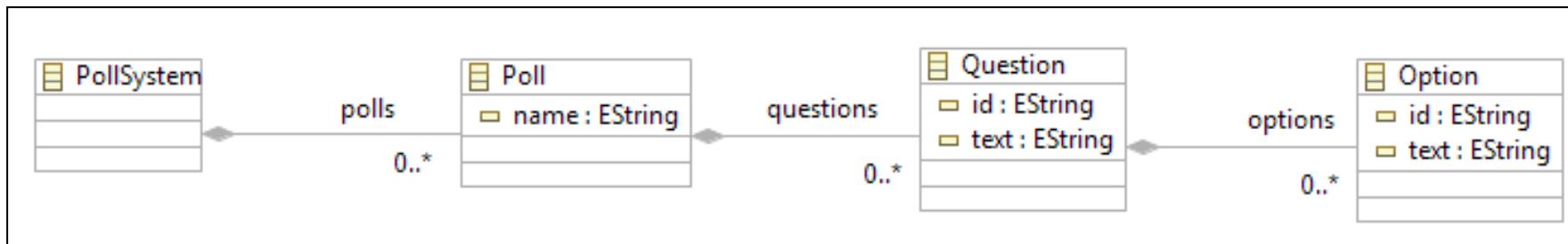
```
grammar fr.miage.xtext.Poll with org.eclipse.xtext.common.Terminals

generate poll "http://www.miage.fr/xtext/Poll"

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

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

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



# Xtext Grammar

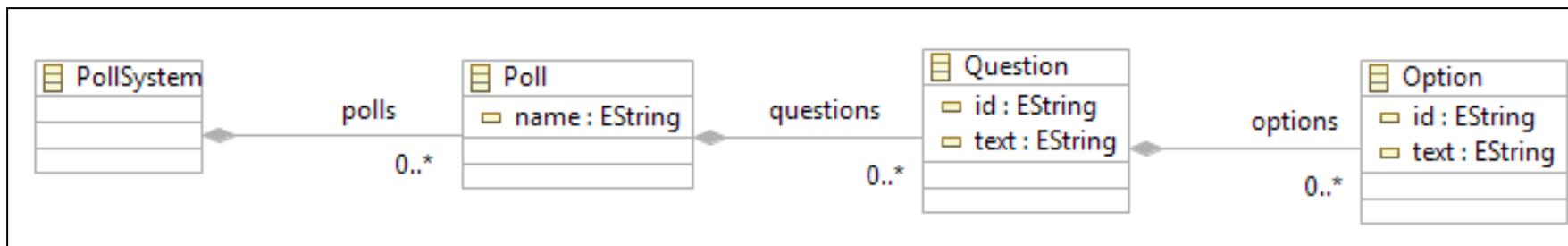
```
grammar fr.miage.xtext.Poll with org.eclipse.xtext.common.Terminals

generate poll "http://www.miage.fr/xtext/Poll"

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

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

Derived  
metamodel



# Xtext Grammar

## Parser Rules

```
grammar fr.miage.xtext.Poll with org.eclipse.xtext.common.Terminals

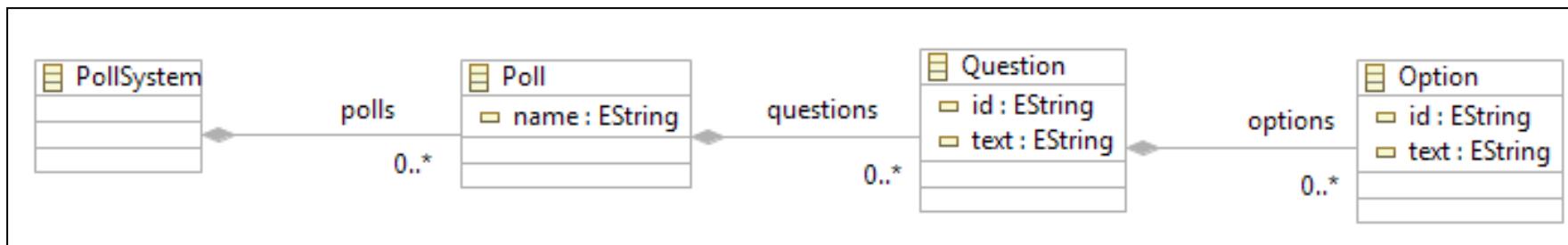
generate poll "http://www.miage.fr/xtext/Poll"

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

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

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

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



# Xtext Grammar

```
grammar fr.miage.xtext.Poll with org.eclipse.xtext.common.Terminals

generate poll "http://www.miage.fr/xtext/Poll"
```

PollSystem:

```
→ 'PollSystem' '{' polls+=Poll+ '}';

```

Poll:

```
→ 'Poll' name=ID '{' questions+=Question+ '}';

```

Question:

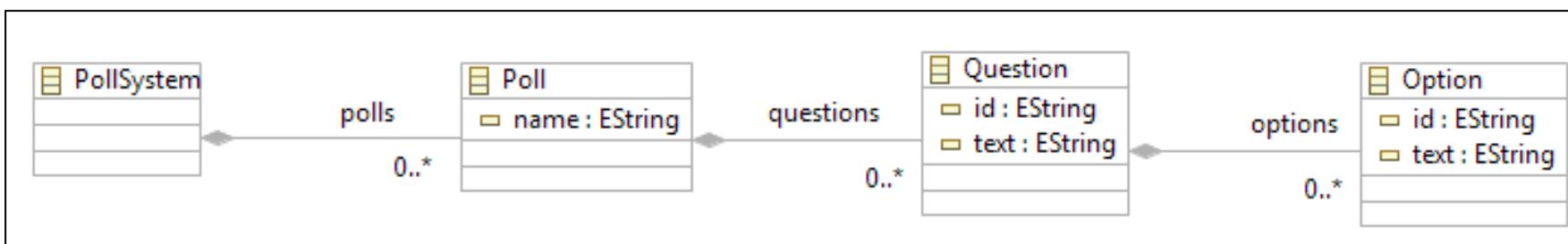
```
→ 'Question' id=ID '{' text=STRING 'options' '{' options+=Option+ '}'} '}';

```

Option:

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

Keywords



# Xtext Grammar

```
grammar fr.miage.xtext.Poll with org.eclipse.xtext.common.Terminals

generate poll "http://www.miage.fr/xtext/Poll"

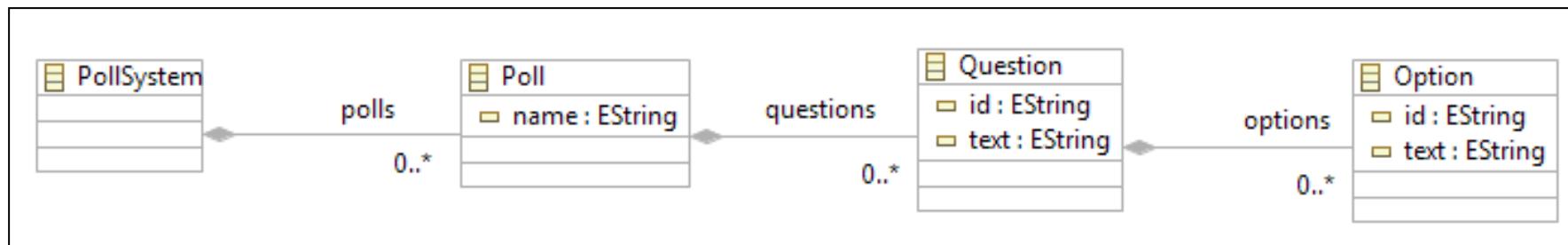
PollSystem:
    'PollSystem' '{' polls+=Poll+ '}';
    ^ Multivalue assignment

Poll:
    'Poll' name=ID '{' questions+=Question+ '}';
    ^ Simple assignment

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

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

(not here → **?= Boolean assignment**)



# Xtext Grammar

```
grammar fr.miage.xtext.Poll with org.eclipse.xtext.common.Terminals

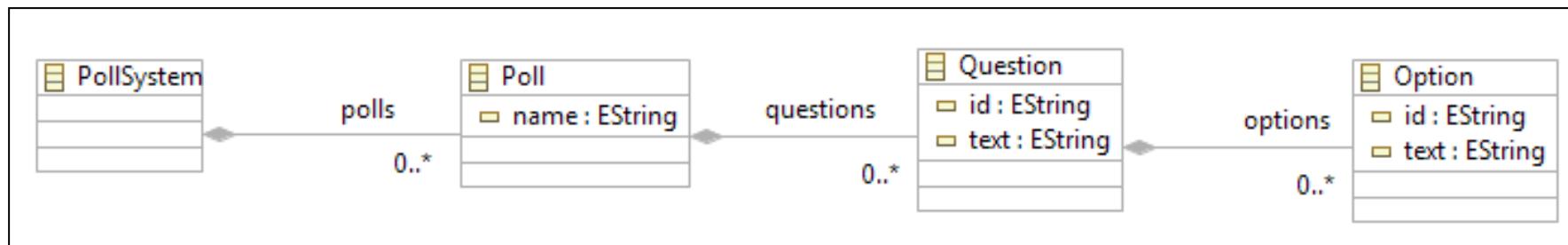
generate poll "http://www.miage.fr/xtext/Poll"

PollSystem:
    'PollSystem' '{' polls+=Poll+ '}';
    ^Cardinality (others: * ?)

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

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

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



# Xtext Grammar

```
grammar fr.miage.xtext.Poll with org.eclipse.xtext.common.Terminals
```

```
generate poll "http://www.miage.fr/xtext/Poll"
```

```
PollSystem:
```

```
    'PollSystem' '{' polls+=Poll+ '}';
    
```

```
Poll:
```

```
    'Poll' name=ID '{' questions+=Question+'}';
    
```

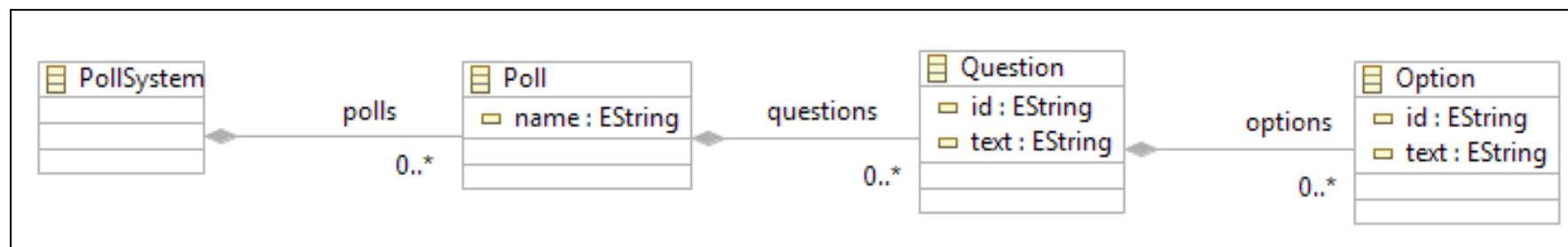
```
Question:
```

```
    'Question' id=ID '{' text=STRING 'options' '{' options+=Option+ '}';
    
```

```
Option:
```

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

Containment



# Grammar and Programs/Specifications/Models

```
grammar fr.miage.xtext.Poll with org.eclipse.xtext.common.Terminals

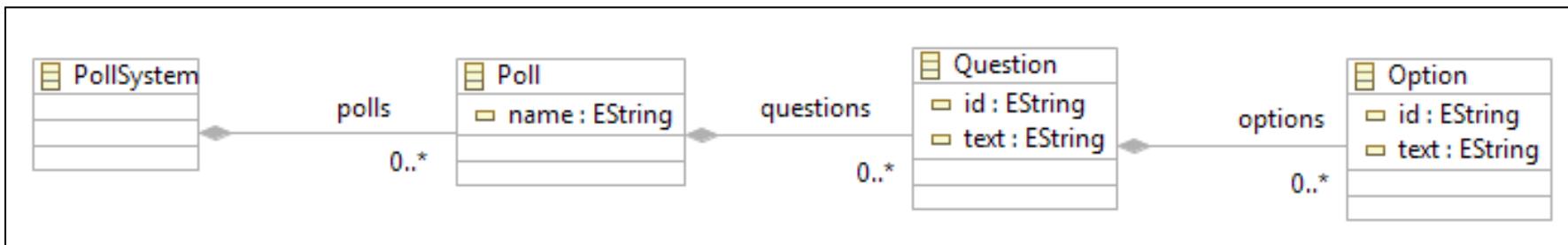
generate poll "http://www.miage.fr/xtext/Poll"

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

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

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

```
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"
            }
        }
    }
}
```



# Grammar and Programs/Specifications/Models

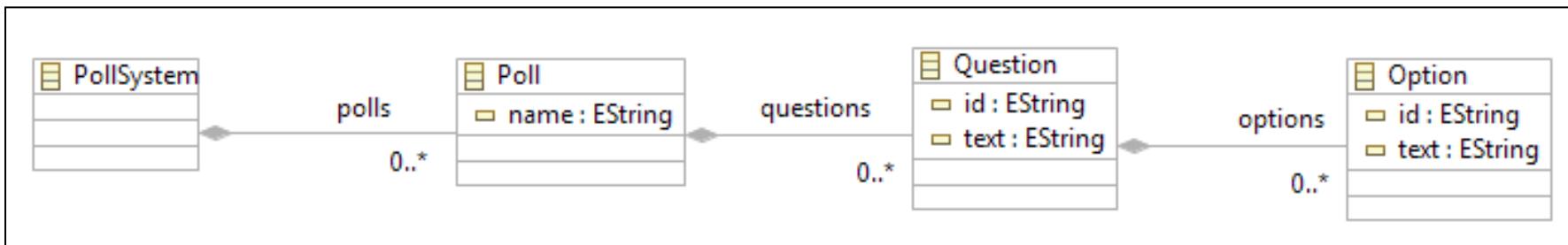
```
grammar fr.miage.xtext.Poll with org.eclipse.xtext.common.Terminals

generate poll "http://www.miage.fr/xtext/Poll"

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

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

```
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"
            }
        }
    }
}
```



# Grammar and Programs/Specifications/Models

```
grammar fr.miage.xtext.Poll with org.eclipse.xtext.common.Terminals

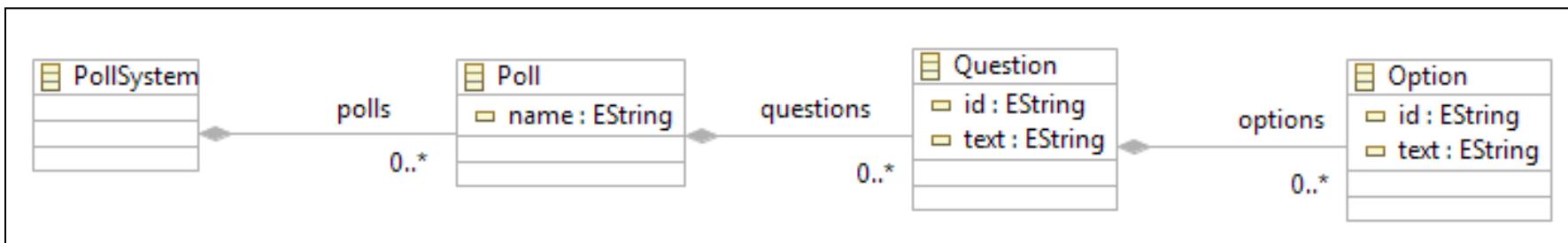
generate poll "http://www.miage.fr/xtext/Poll"

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

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

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

```
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"
            }
        }
    }
}
```



# Grammar and Programs/Specifications/Models

```
grammar fr.miage.xtext.Poll with org.eclipse.xtext.common.Terminals

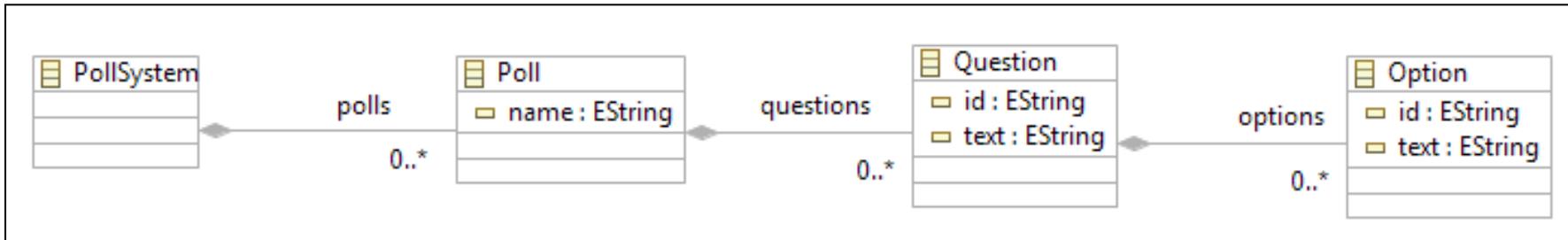
generate poll "http://www.miage.fr/xtext/Poll"

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

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

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

```
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"
            }
        }
    }
}
```



# Quizz Time

Quetionnaire.xtext

```
1 grammar org.xtext.example.mydsl.Quetionnaire with org.eclipse.xtext.common.Terminals
2
3 generate questionnaire "http://www.xtext.org/example/mydsl/Questionnaire"
4
5 @PollSystem:
6   'PollSystem' '{' polls+=Poll+ '}';
7
8 @Poll:
9   'Poll' name=ID '{' questions+=Question+ '}';
10
11 Question : 'Question' ID? '{' text=STRING 'options' options+=Option+ '}';
12
13 Option : id=ID ':' text=STRING ;
```

Est-ce que le fichier vide .q est correct vis-à-vis de la grammaire Xtext? Pourquoi?

# Quizz Time

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

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

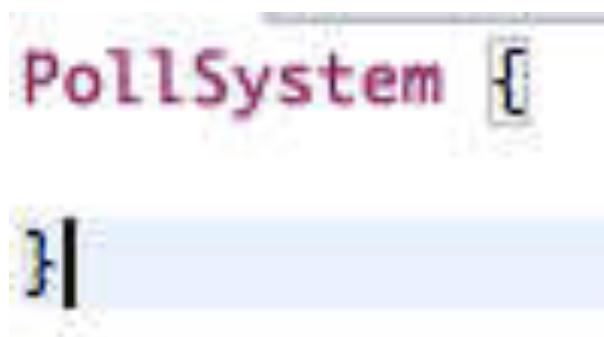
PollSystem:
    {PollSystem} 'PollSystem' '{' polls+=Poll* '}';

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

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

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

Est-ce que le fichier .q suivant est correct vis-à-vis de la grammaire Xtext?  
Pourquoi?



# Quizz Time

Quetionnaire.xtext

```
1 grammar org.xtext.example.mydsl.Quetionnaire with org.eclipse.xtext.common.Terminals
2
3 generate questionnaire "http://www.xtext.org/example/mydsl/Questionnaire"
4
5@PollSystem:
6     'PollSystem' '{' polls+=Poll+ '}';
7
8@Poll:
9     'Poll' name=ID '{' questions+=Question+ '}';
10
11 Question : 'Question' ID '{' text=STRING 'options' options+=Option+ '}';
12
13 Option : id=ID ':' text=STRING ;
```

Est-ce que le fichier .q suivant est correct vis-à-vis de la grammaire Xtext? Pourquoi?

```
PollSystem {
    Poll p1 {
        Question {
            "Q1"
            options o1 : "R1"
        }
    }
}
```

Xtext, your DSL in  
5' (incl. editors and  
serializers)

Live Demonstration

The screenshot shows the Eclipse IDE interface with two main panes. On the left is the 'Package Explorer' view, which displays the project structure of 'org.xtext.example.questionnaire'. The 'src' folder contains 'org.xtext.example.mydsl' with files 'GenerateQuestionnaire.mwe2' (selected) and 'Questionnaire.xtext'. Other visible files include 'src-gen', 'xtend-gen', 'JRE System Library [JavaSE-1.8]', 'Plug-in Dependencies', 'META-INF', 'build.properties', 'org.xtext.example.questionnaire.sdk', 'org.xtext.example.questionnaire.tests', and 'org.xtext.example.questionnaire.ui'. On the right is the 'Questionnaire.xtext' editor view, showing the Xtext grammar code:

```
1 grammar org.xtext.example.mydsl.Questionnaire with org.eclipse.xtext.common.Terminals
2
3 generate questionnaire "http://www.xtext.org/example/mydsl/Questionnaire"
4
5 @PollSystem:
6     'PollSystem' '{' polls+=Poll+ '}';
7
8 @Poll:
9     'Poll' name=ID '{' questions+=Question+ '}';
10
11 Question : 'Question' id=ID '{' text=STRING 'options' '{' options+=Option+ '}' '}';
12
13 Option : id=ID ':' text=STRING ;
```

Package Explorer    Questionnaire.xtext

org.xtext.example.questionnaire  
  src  
    org.xtext.example.mydsl  
      GenerateQuestionnaire.mwe2  
        Questionnaire.xtext

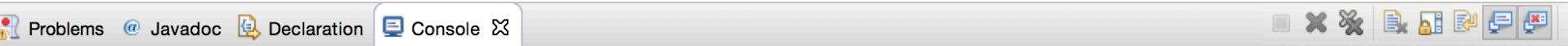
New ►  
Open F3  
Open With ►  
Show In ⌘W ►  
Copy ⌘C  
Copy Qualified Name  
Paste ⌘V  
Delete ⌘X  
Build Path ►  
Refactor ⌘T ►  
Import...  
Export...  
Refresh F5  
Assign Working Sets...

Validate  
Run As ►  
Debug As ►  
Replace With ►  
Team ►  
Compare With ►

Properties ⌘I

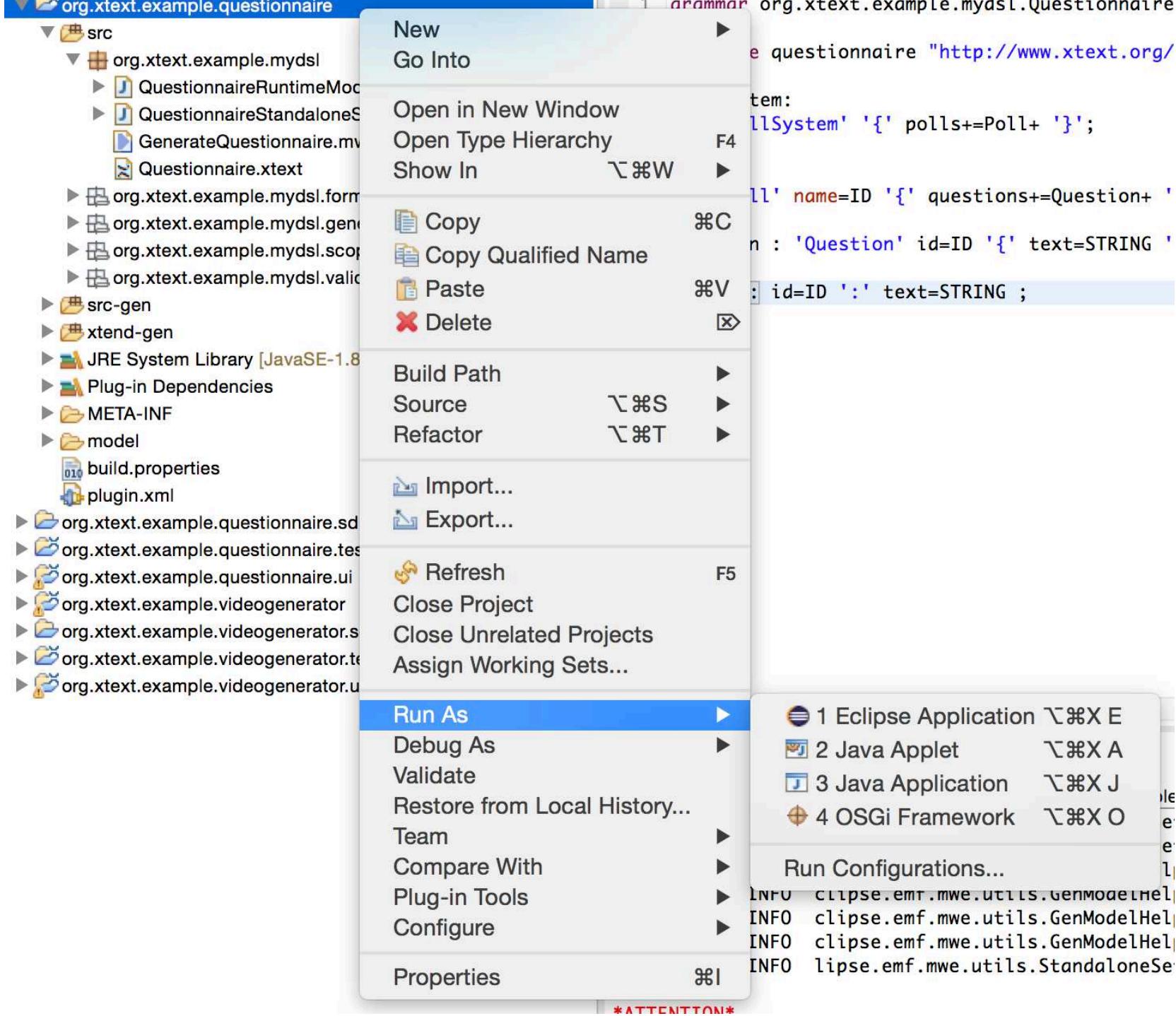
1 grammar org.xtext.example.mydsl.Questionnaire  
2 generate questionnaire "http://www.xtext.org"  
3 item '{' polls+=Poll+ '}';  
4 name=ID '{' questions+=Question '}' id=ID '{' text=STRING ;  
  ':' text=STRING ;

1 MWE2 Workflow  
Run Configurations...



```
<terminated> Generate Language Infrastructure (org.xtext.example.questionnaire) [Mwe2 Launch] /Library/Java/JavaVirtualMachines/jdk1.8.0_31.jdk/Contents/Home/bin/java (28 sept. 2014)
0 [main] INFO lipse.emf.mwe.utils.StandaloneSetup - Registering platform uri '/Users/macher1/Documents/workspaceIDM1516'
127 [main] INFO lipse.emf.mwe.utils.StandaloneSetup - Adding generated EPackage 'org.eclipse.xtext.Xbase.XbasePackage'
408 [main] INFO clipse.emf.mwe.utils.GenModelHelper - Registered GenModel 'http://www.eclipse.org/Xtext/Xbase/XAnnotations' from 'platform:/resource/Questionnaire/XAnnotations.genmodel'
413 [main] INFO clipse.emf.mwe.utils.GenModelHelper - Registered GenModel 'http://www.eclipse.org/xtext/xbase/Xtype' from 'platform:/resource/Questionnaire/Xtype.genmodel'
436 [main] INFO clipse.emf.mwe.utils.GenModelHelper - Registered GenModel 'http://www.eclipse.org/xtext/xbase/Xbase' from 'platform:/resource/Questionnaire/Xbase.genmodel'
436 [main] INFO clipse.emf.mwe.utils.GenModelHelper - Registered GenModel 'http://www.eclipse.org/xtext/common/JavaVMTypes' from 'platform:/resource/Questionnaire/JavaVMTypes.genmodel'
1005 [main] INFO lipse.emf.mwe.utils.StandaloneSetup - Adding generated EPackage 'org.eclipse.xtext.common.types.TypesPackage'

*ATTENTION*
It is recommended to use the ANTLR 3 parser generator (BSD licence - http://www.antlr.org/license.html).
Do you agree to download it (size 1MB) from 'http://download.itemis.com/antlr-generator-3.2.0-patch.jar'? (type 'y' or 'n' and hit enter)y
11812 [main] INFO erator.parser.antlr.AntlrToolFacade - downloading file from 'http://download.itemis.com/antlr-generator-3.2.0-patch.jar' ...
108842 [main] INFO erator.parser.antlr.AntlrToolFacade - finished downloading.
108848 [main] INFO ipse.emf.mwe.utils.DirectoryCleaner - Cleaning /Users/macher1/Documents/workspaceIDM1516/org.xtext.example.questionnaire
108849 [main] INFO ipse.emf.mwe.utils.DirectoryCleaner - Cleaning /Users/macher1/Documents/workspaceIDM1516/org.xtext.example.questionnaire
108849 [main] INFO ipse.emf.mwe.utils.DirectoryCleaner - Cleaning /Users/macher1/Documents/workspaceIDM1516/org.xtext.example.questionnaire
110353 [main] INFO clipse.emf.mwe.utils.GenModelHelper - Registered GenModel 'http://www.xtext.org/example/mydsl/Questionnaire' from 'platform:/resource/Questionnaire/Questionnaire.genmodel'
113410 [main] INFO text.generator.junit.Junit4Fragment - generating Junit4 Test support classes
113428 [main] INFO text.generator.junit.Junit4Fragment - generating Compare Framework infrastructure
113584 [main] INFO .emf.mwe2.runtime.workflow.Workflow - Done.
```



**File**

Create a new file resource.



Enter or select the parent folder:

FooQuestionnaire



FooQuestionnaire

VideoGen1

File name:

Advanced >>



Cancel

Finish

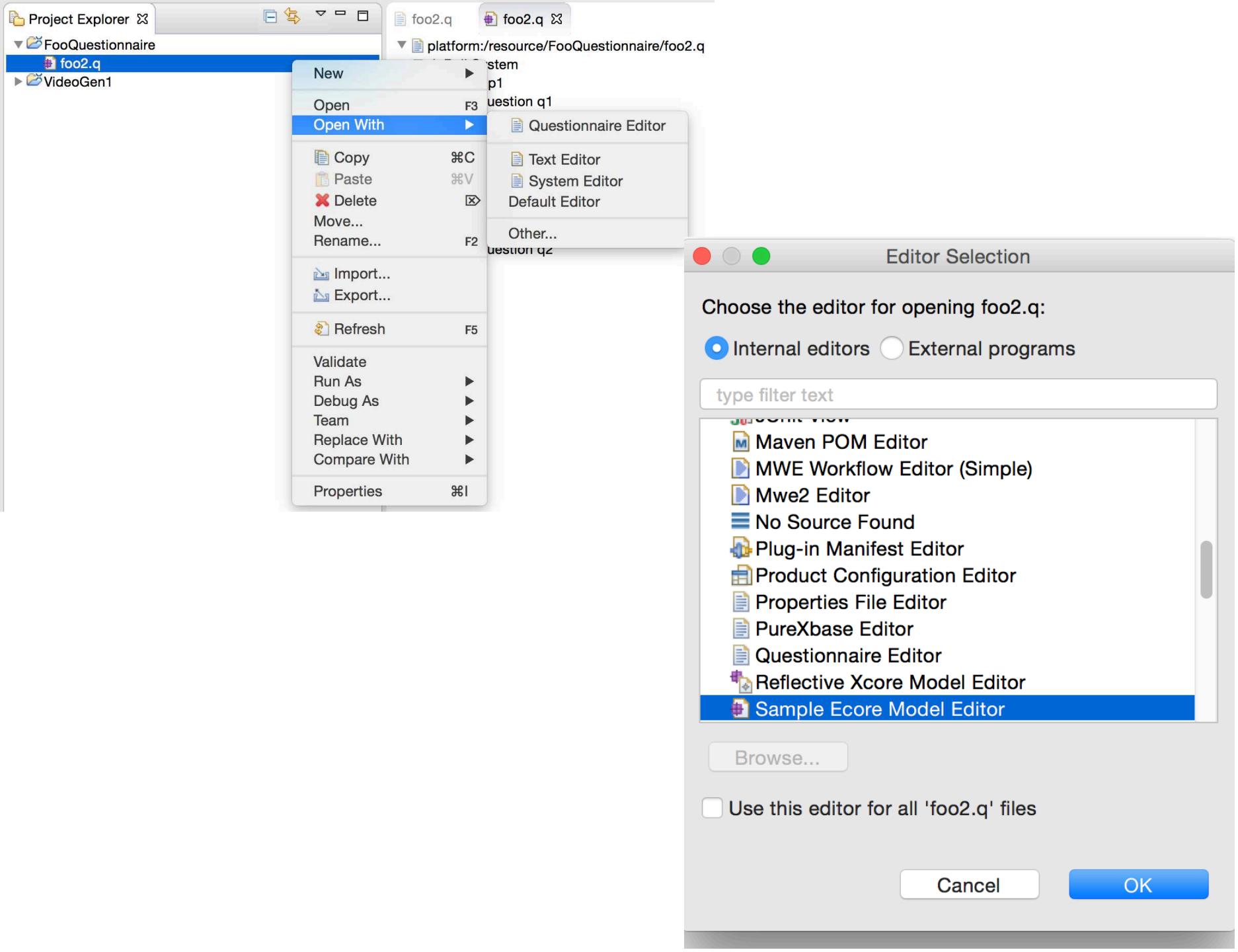
```
PollSystem {

    Poll p1 {
        Question q1 {
            "What is the best JavaScript framework for testing?"
            options [
                A1: "PhantomJS"
                A2: "Jasmine"
                A3: "Mocha"
                A4: "I prefer to develop my own framework"
            ]
        }

        Question q2 {
            "What is the best CSS preprocessor?"
            options [
                A1: "Less.js"
                A2: "Sass"
                A3: "Stylus"
                A4: "I don't care about preprocessing CSS"
            ]
        }
    }

    Poll p2 {
        Question q1 {
            "What is the best Java framework for testing?"
            options [
                A1: "JUnit"
                A2: "Jasmine"
                A3: "I prefer to develop my own framework"
            ]
        }

        Question q2 {
            "What is the best Java library for logging?"
            options [
                A1: "Log4J"
                A2: "java.util.logging"
                A3: "I don't care about logging"
            ]
        }
    }
}
```



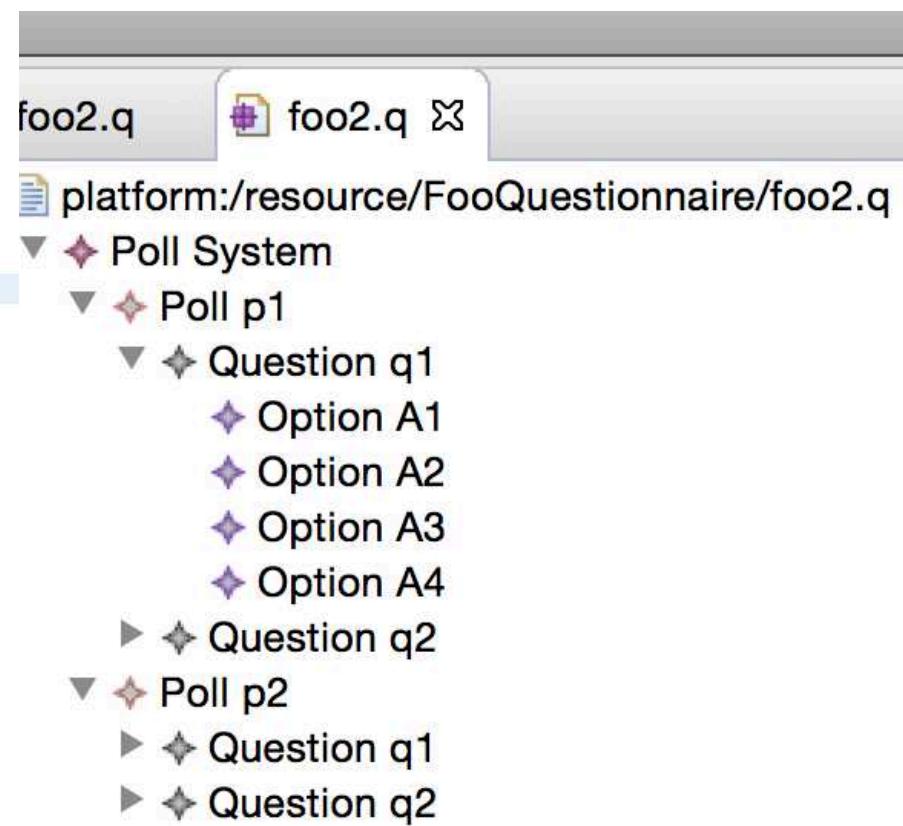
```
2.q ✎
ollSystem {

Poll p1 {
    Question q1 {
        "What is the best JavaScript framework for testing?"
        options [
            A1: "PhantomJS"
            A2: "Jasmine"
            A3: "Mocha"
            A4: "I prefer to develop my own framework"
        ]
    }

    Question q2 {
        "What is the best CSS preprocessor?"
        options {
            A1: "Less.js"
            A2: "Sass"
            A3: "Stylus"
            A4: "I don't care about preprocessing CSS"
        }
    }

Poll p2 {
    Question q1 {
        "What is the best Java framework for testing?"
        options {
            A1: "JUnit"
            A2: "Jasmine"
            A3: "I prefer to develop my own framework"
        }
    }

    Question q2 {
        "What is the best Java library for logging?"
        options {
            A1: "Log4J"
            A2: "java.util.logging"
            A3: "I don't care about logging"
        }
    }
}
}
```



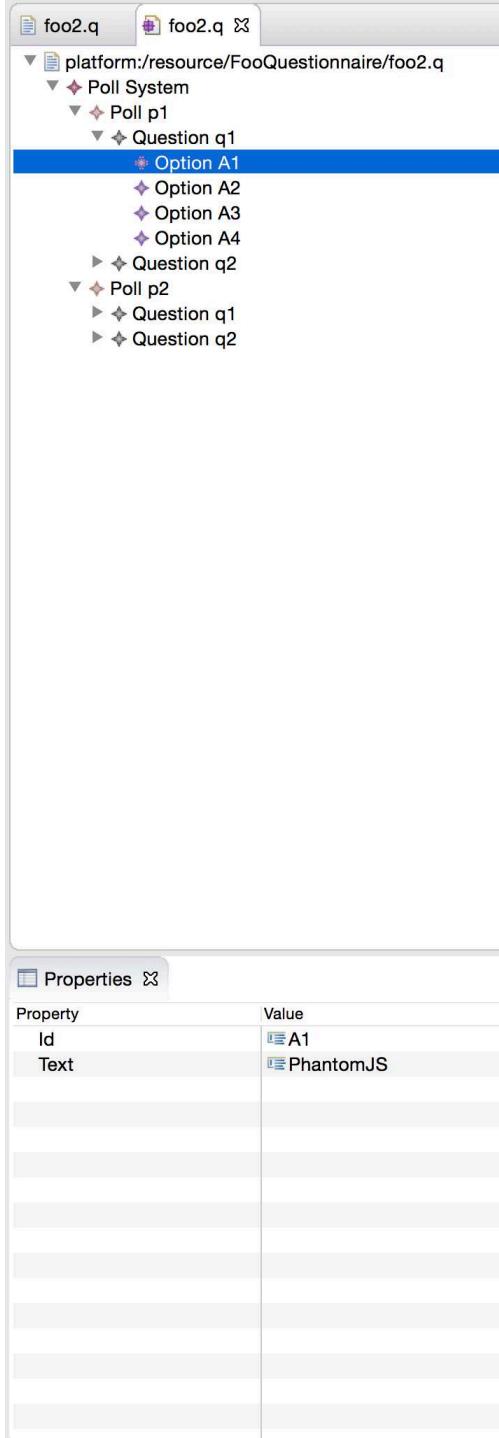
```
2.q ✎
ollSystem {

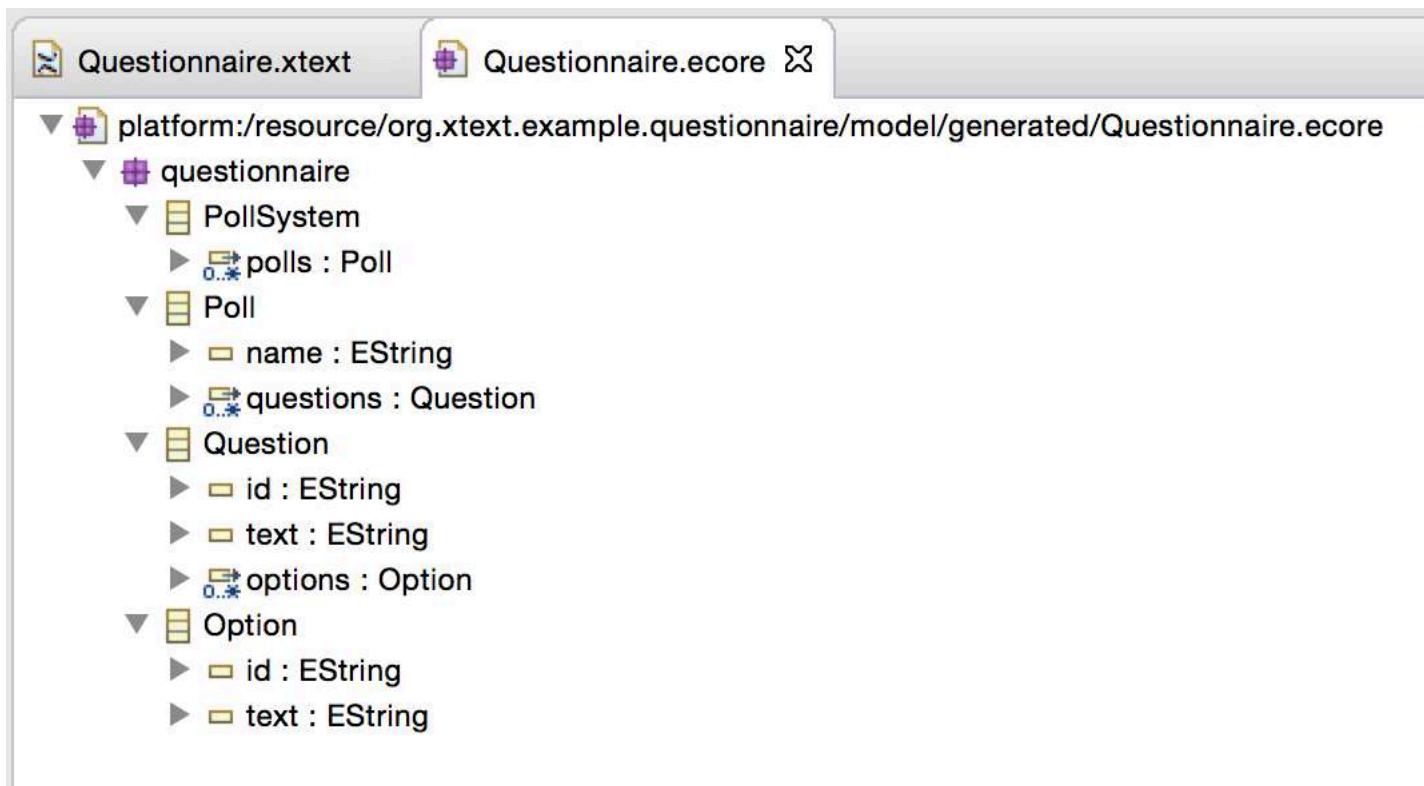
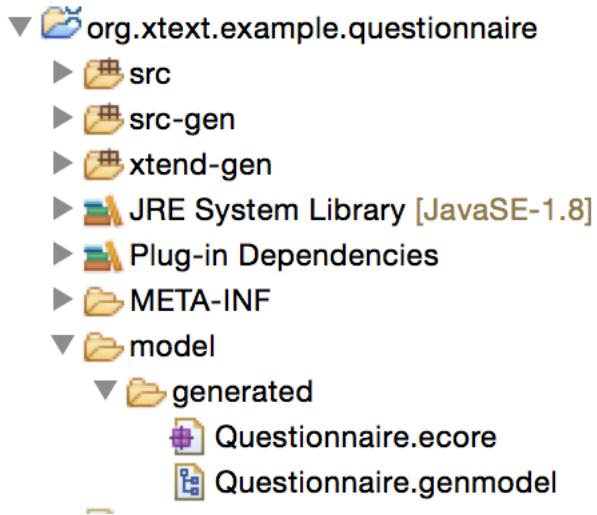
    Poll p1 {
        Question q1 {
            "What is the best JavaScript framework for testing?"
            options [
                A1: "PhantomJS"
                A2: "Jasmine"
                A3: "Mocha"
                A4: "I prefer to develop my own framework"
            ]
        }

        Question q2 {
            "What is the best CSS preprocessor?"
            options [
                A1: "Less.js"
                A2: "Sass"
                A3: "Stylus"
                A4: "I don't care about preprocessing CSS"
            ]
        }
    }

    Poll p2 {
        Question q1 {
            "What is the best Java framework for testing?"
            options [
                A1: "JUnit"
                A2: "Jasmine"
                A3: "I prefer to develop my own framework"
            ]
        }

        Question q2 {
            "What is the best Java library for logging?"
            options [
                A1: "Log4J"
                A2: "java.util.logging"
                A3: "I don't care about logging"
            ]
        }
    }
}
```





Another example:

Chess

**“Queen to c7.  
Check.”**



**“Rd2-c2,  
rook at d2 moves to c2”**

# Moves in Chess:

Rook at a1 moves to a5.

Piece      Square      Action      Destination

Bishop at c8 captures knight at h3.

Piece      Square      Action      Destination

N b1 x c3

Piece      Square      Action      Destination

g2 - g4

Square      Action      Destination

Bishop at c8 captures knight at h3

$\mathbb{B} \ c8 \times h3$



P e2 – e4

p g7 – g5

Knight at b2 moves to c3

pawn at f7 moves to f5

Q d1 – h5

# 1-0

**Concrete Syntax**

**Constraints !!!**

**Abstract Syntax**



# Chess Example - Grammar

**Game:**

```
"White:" whitePlayer=STRING  
"Black:" blackPlayer=STRING  
(moves+=Move)+;
```

**Move:**

```
AlgebraicMove | SpokenMove;
```

**AlgebraicMove:**

```
(piece=Piece) ? source=Square (captures?='x' | '-') dest=Square;
```

**SpokenMove:**

```
piece=Piece 'at' source=Square  
(captures?='captures' capturedPiece=Piece 'at' | 'moves to')  
dest=Square;
```

**terminal Square:**

```
('a'..'h')('1'..'8');
```

**enum Piece:**

```
pawn    = 'P' | pawn = 'pawn' |  
knight  = 'N' | knight = 'knight' |  
bishop   = 'B' | bishop = 'bishop' |  
rook    = 'R' | rook = 'rook' |  
queen   = 'Q' | queen = 'queen' |  
king    = 'K' | king = 'king';
```

# Chess Example - Model

White: "Mayfield"

Black: "Trinks"

pawn at e2 moves to e4

pawn at f7 moves to g5

K b1 - c3

f7 - f5

queen at d1 moves to h5

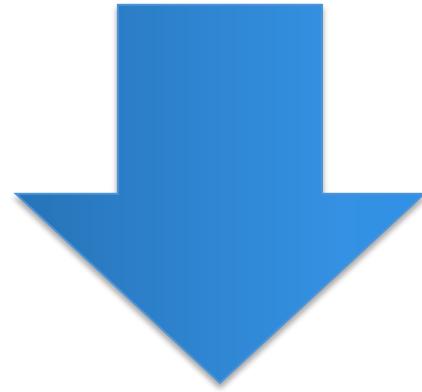
// 1-0



```
foo1.videogen ✘

mandatory videotseq v1 "https://www.youtube.com/watch?v=PjNi1uYhV5w"
optional videotseq v2 "v2folder/v2.mp4"
alternatives v3 {
    videotseq v31 "v3/seq1.mp4"
    videotseq v32 "v3/seq1.mp4"
    videotseq v33 "v3/seq1.mp4"
}

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



 FFmpeg

## foo1.videoogen

```
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 ✘

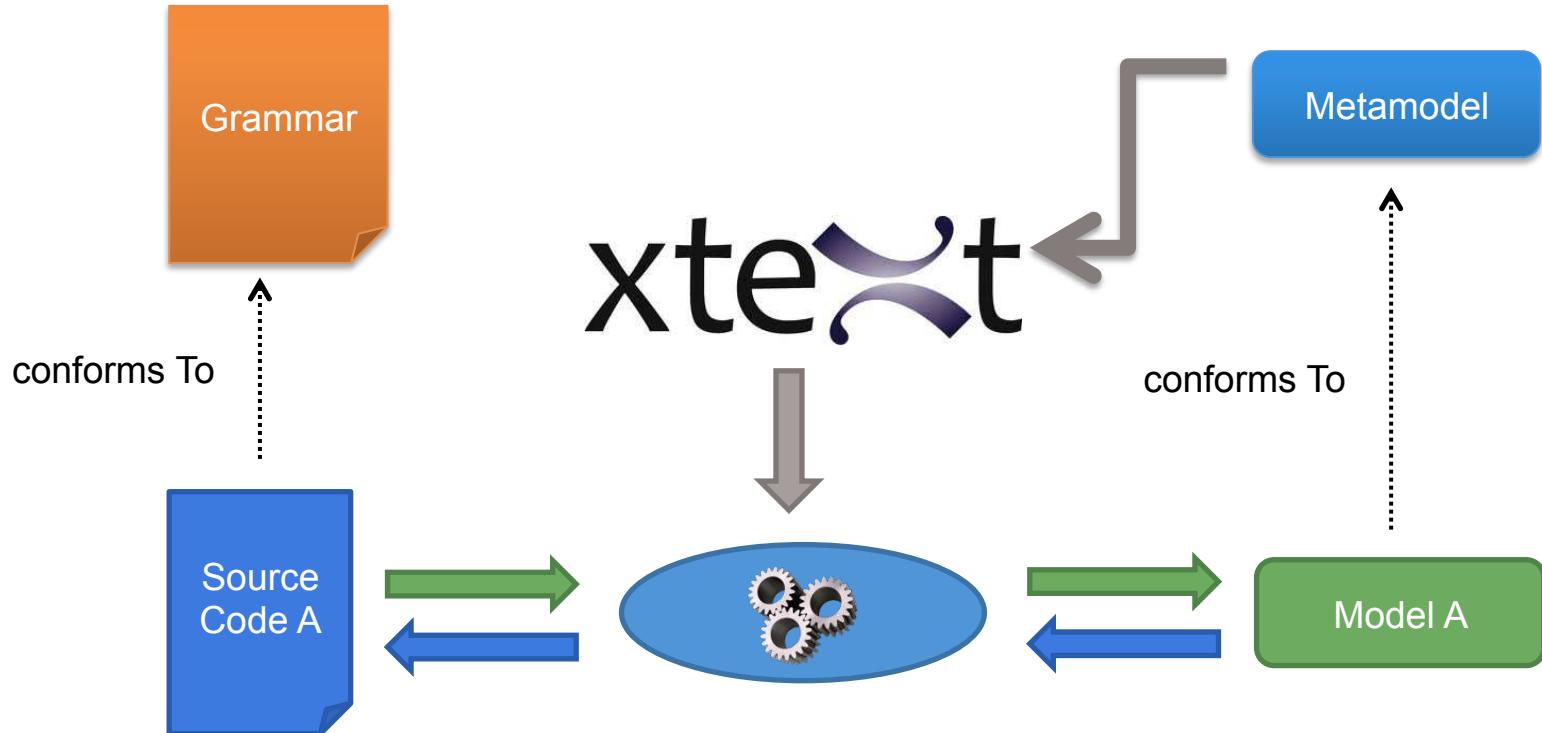
mandatory videoseq v1 "https://www.youtube.com/watch?v=PJNi1uYhV5w"
optional videoseq v2 "v2folder/v2.mp4"
@alternatives v3 {
    videoseq v31 "v3/seq1.mp4"
    videoseq v32 "v3/seq1.mp4"
    videoseq v33 "v3/seq1.mp4"
}
@alternatives v4 {
    videoseq v41 "v4/seq1.mp4"
    videoseq v42 "v4/seq1.mp4"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

From Metamodel

To

Grammar (other side)

# From Metamodel to Grammar





Give me a **metamodel**,

I'll give you (for free)

- \* a comprehensive editor (auto-completion, syntax highlighting, etc.) in Eclipse
- \* a grammar and facilities to load/serialize/visit conformant models (Java ecosystem)
- \* extension to override/extend « default » facilities (e.g., checker)



Give me a **metamodel**,

The grammar can be « weird » (i.e., not as concise and as comprehensible than if you made it manually)

[Same observation actually applies to the other side: generated metamodels (from grammar) can be weird as well, but you have at least some control in Xtext-based grammar]  
[We will experiment in the lab sessions]

Live

Demonstration

New

Select a wizard

Create an Xtext project from existing Ecore models

Wizards:

Xtext

- ▼ Xtext
  - Xtext Project
  - Xtext Project From Existing Ecore Models**
- ▼ Continuous Integration
  - Build Xtext with Buckminster
- ▼ Examples
  - Xtext Domain-Model Example
  - Xtext Home Automation Example
  - Xtext Simple Arithmetics Example
  - Xtext State-Machine Example
- ▼ Examples
  - Xtext Examples
    - Xtext Domain-Model Example
    - Xtext Home Automation Example

?

< Back    Next >    Cancel    Finish

New Xtext Project From Ecore

Select EPackages

Select the EPackages to generate an Xtext grammar for.

EPackages:

org.xtext.example.mydsl.questionnaire.QuestionnairePackage (default package)

Add...    Set Default    Remove

Entry rule:

PollSystem - questionnaire

?

< Back    Next >    Cancel    Finish

```
// automatically generated by Xtext
grammar org.xtext.example.mydsl.Questionnaire2 with org.eclipse.xtext.common.Terminal

import "http://www.xtext.org/example/mydsl/Questionnaire"
import "http://www.eclipse.org/emf/2002/Ecore" as ecore

@PollSystem returns PollSystem:
    {PollSystem}
    'PollSystem'
    '{'
        ('polls' '{' polls+=Poll ( "," polls+=Poll)* '}' )?
    '}';
    13
    14
    15
    16

@Poll returns Poll:
    {Poll}
    'Poll'
    name=EString
    '{'
        ('questions' '{' questions+=Question ( "," questions+=Question)* '}' )?
    '}';
    24

@EString returns ecore::EString:
    STRING | ID;
    27

@Question returns Question:
    {Question}
    'Question'
    '{'
        ('id' id=EString)?
        ('text' text=EString)?
        ('options' '{' options+=Option ( "," options+=Option)* '}' )?
    '}';
    36

@Option returns Option:
    {Option}
    'Option'
    '{'
        ('id' id=EString)?
        ('text' text=EString)?
    '}';
    44
```

# **Quizz Time**

**Explain (roughly) the  
« algorithm » of Xtext to  
generate a grammar from an  
ecore Metamodel**

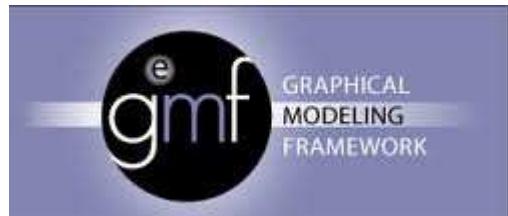
# My 3 take away messages

- #1 DSLs are important (as intuited for a long time - it will become more and more apparent)
- #2 DSL technology is here (no excuse)
- #3 MDE meets language engineering

# Questions ?

(see also resources and  
lab sessions)

[http://martinfowler.com/bliki/  
DomainSpecificLanguage.html](http://martinfowler.com/bliki/DomainSpecificLanguage.html)

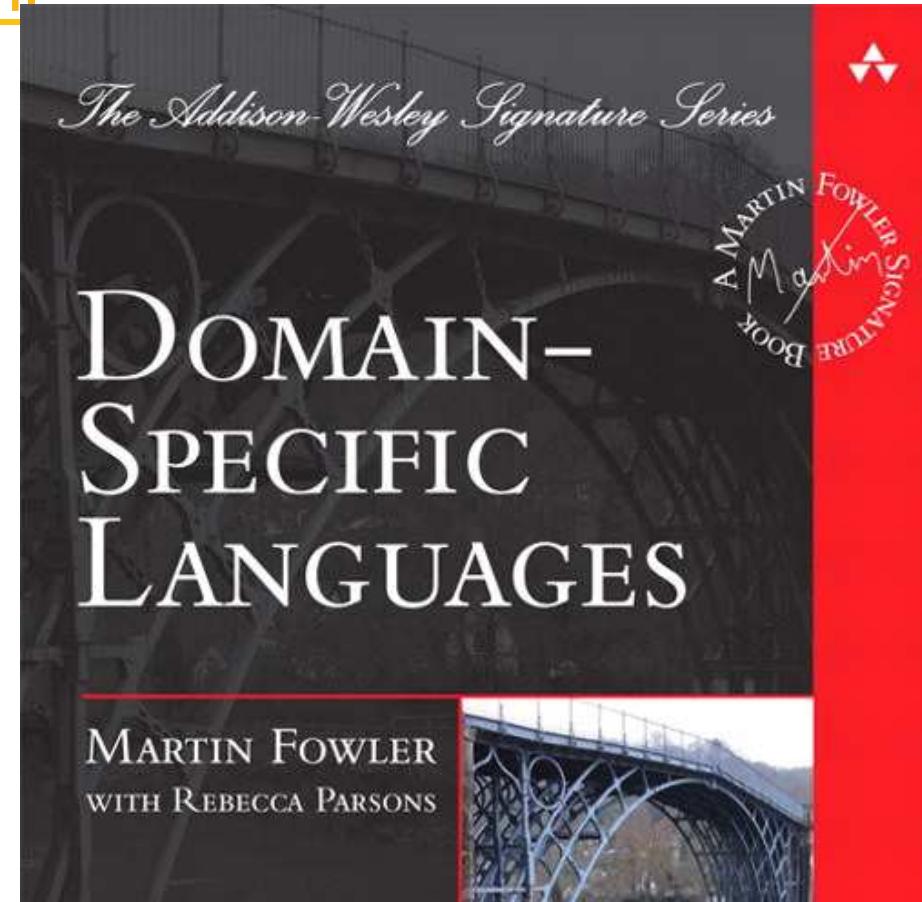


## Empirical Assessment of MDE in Industry

Jon Hutchinson, Jon Whittle, Mark Rouncefield  
School of Computing and Communications  
Lancaster University, UK  
+44 1524 510492

{j.hutchinson, j.n.whittle,  
m.rouncefield}@lancaster.ac.uk

Steinar Kristoffersen  
Østfold University College and Møreforskning Molde AS  
NO-1757 Halden  
Norway  
+47 6921 5000  
steinar.kristoffersen@hiof.no



# References

- Martin Fowler. Domain Specific Languages. Addison-Wesley Professional, 2010.
- Markus Voelter et al. “DSL Engineering: Designing, Implementing and Using Domain-Specific Languages.” dslbook.org, 2013.
- 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
- Steven Kelly and Risto Pohjonen. Worst practices for domain-specific modeling. IEEE Software, 26(4):22–29, 2009.
- Lennart C.L. Kats and Eelco Visser. The spoofax language workbench: Rules for declarative specification of languages and ides OOPSLA’10

# References

- Sebastian Erdweg, Tijs van der Storm, Markus Völter, Meinte Boersma, Remi Bosman, William R. Cook, Albert Gerritsen, Angelo Hulshout, Steven Kelly, Alex Loh, Gabriël D. P. Konat, Pedro J. Molina, Martin Palatnik, Risto Pohjonen, Eugen Schindler, Klemens Schindler, Riccardo Solmi, Vlad A. Vergu, Eelco Visser, Kevin van der Vlist, Guido Wachsmuth, and Jimi van der Woning. The state of the art in language workbenches conclusions from the language workbench challenge. SLE'13
- Steven Kelly, Kalle Lyytinen, Matti Rossi, and Juha-Pekka Tolvanen. Metaedit+ at the age of 20. In Seminal Contributions to Information Systems Engineering, pages 131–137. Springer, 2013.
- Sebastian Erdweg, Tillmann Rendel, Christian Kästner, and Klaus Ostermann. Sugarj: Library-based syntactic language extensibility. OOPSLA'11