Domain-Specific Languages

Mathieu Acher

Maître de Conférences

mathieu.acher@irisa.fr

Material

http://mathieuacher.com/teaching/MDE/MRI1516/

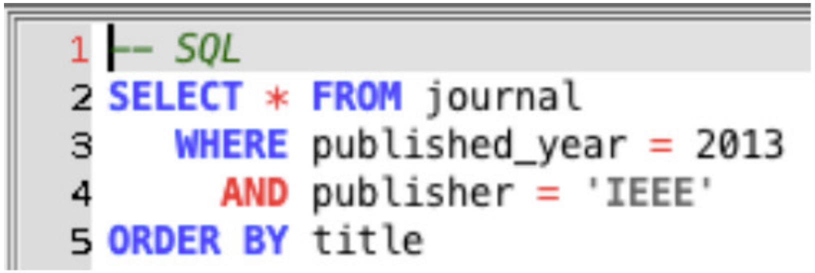
Homework

- Deadline: 19th november
  - email: mathieu.acher@irisa.fr
- Choose a DSL that is both external and internal (but not present in the Github repository below).
- The exercice is to develop a program in the DSL in three equivalent variants:
  - Two variants with an internal shape of the DSL, in two different GPLs
  - One variant with the external shape of the DSL
  - The three variants should have the same behavior
- Source code and instructions on how to execute the programs on the repository (by pull request):
  - https://github.com/acherm/metamorphicDSL-IDM1516

SQL

Plain SQL
(external DSL)

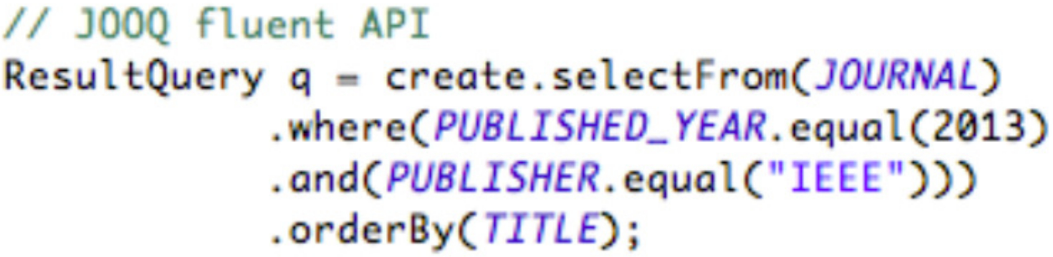
shape
#1



```sql
-- SQL
SELECT * FROM journal
   WHERE published_year = 2013
     AND publisher = 'IEEE'
ORDER BY title
```

Java
(internal DSL)

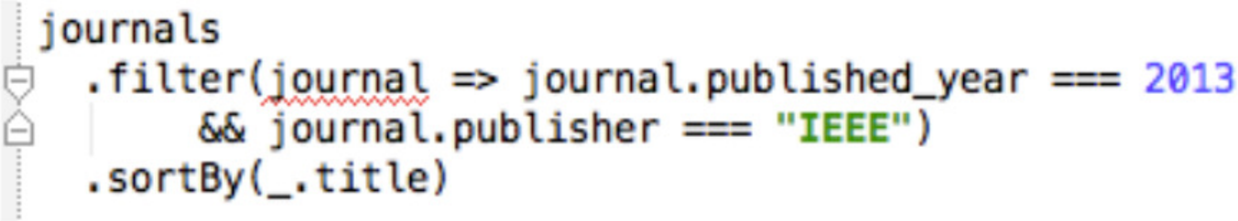
shape
#2



```java
// JOOQ fluent API
ResultQuery q = create.selectFrom(JOURNAL)
                      .where(PUBLISHED_YEAR.equal(2013)
                      .and(PUBLISHER.equal("IEEE")))
                      .orderBy(TITLE);
```

Scala
(internal DSL)

shape
#3



```scala
journals
  .filter(journal => journal.published_year === 2013
        && journal.publisher === "IEEE")
  .sortBy(_.title)
```

Plan

- Domain-Specific Languages (DSLs)
  - Languages and abstraction gap
  - Examples and rationale
  - DSLs vs General purpose languages, taxonomy
- External DSLs
  - Grammar and parsing
  - Language workbenches, Xtext
- DSLs, DSMLs, and (meta-)modeling

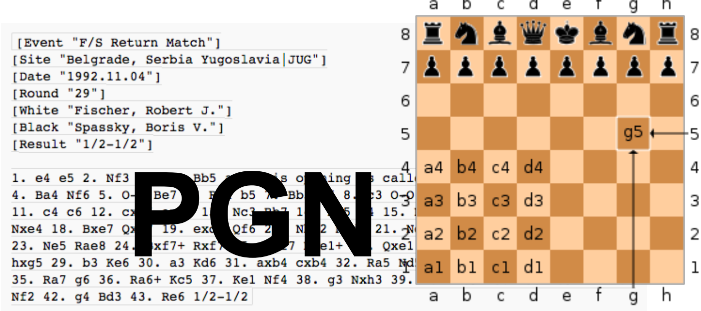
Contract

- 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)

- Models and Languages
  - Perhaps a more concrete way to see models, metamodels and MDE (IDM in french)

BibTeX

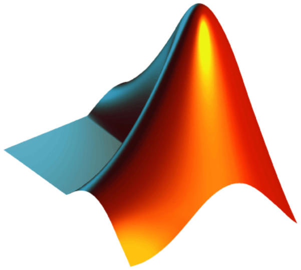

```
[Event "F/S Return Match"]
[Site "Belgrade, Serbia Yugoslavia|JUG"]
[Date "1992.11.04"]
[Round "29"]
[White "Fischer, Robert J."]
[Black "Spassky, Boris V."]
[Result "1/2-1/2"]

1. e4 e5 2. Nf3 Bb5 is opening s call
4. Ba4 Nf6 5. O- Be7 b5 7. Bb 8. c3 O-O
11. c4 c6 12. cx 1 Nc3 Bb7 1 5 4 15.
Nxe4 18. Bxe7 Qx 19. exc Qf6 2 N 2 21. N
23. Ne5 Rae8 24. xf7+ Rxf7 27 e1+ Qxe1
hxg5 29. b3 Ke6 30. a3 Kd6 31. axb4 cxb4 32. Ra5 Nd
35. Ra7 g6 36. Ra6+ Kc5 37. Ke1 Nf4 38. g3 Nxh3 39.
Nf2 42. g4 Bd3 43. Re6 1/2-1/2
```

PGN

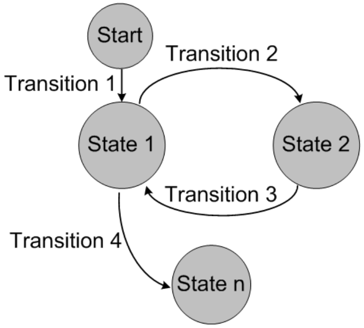
Graphviz

HTML

Make



Matlab



Finite State Machine

SQL

CSS

Domain-Specific Languages (DSLs)

DSL =

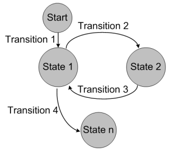
Syntax + Services

Specialized notation:

Textual or Graphical
Specific Vocabulary
Idiomatic constructs

Specialized tools/IDE:

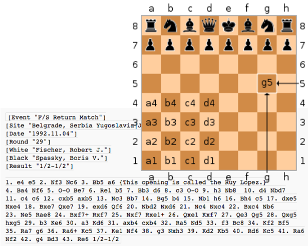
Editor with auto-completion, syntax highlighting, etc.
Compiler
Interpreter
Debugger
Profiler
Syntax/Type Checker
…



BIBTEX

SQL

CSS 3



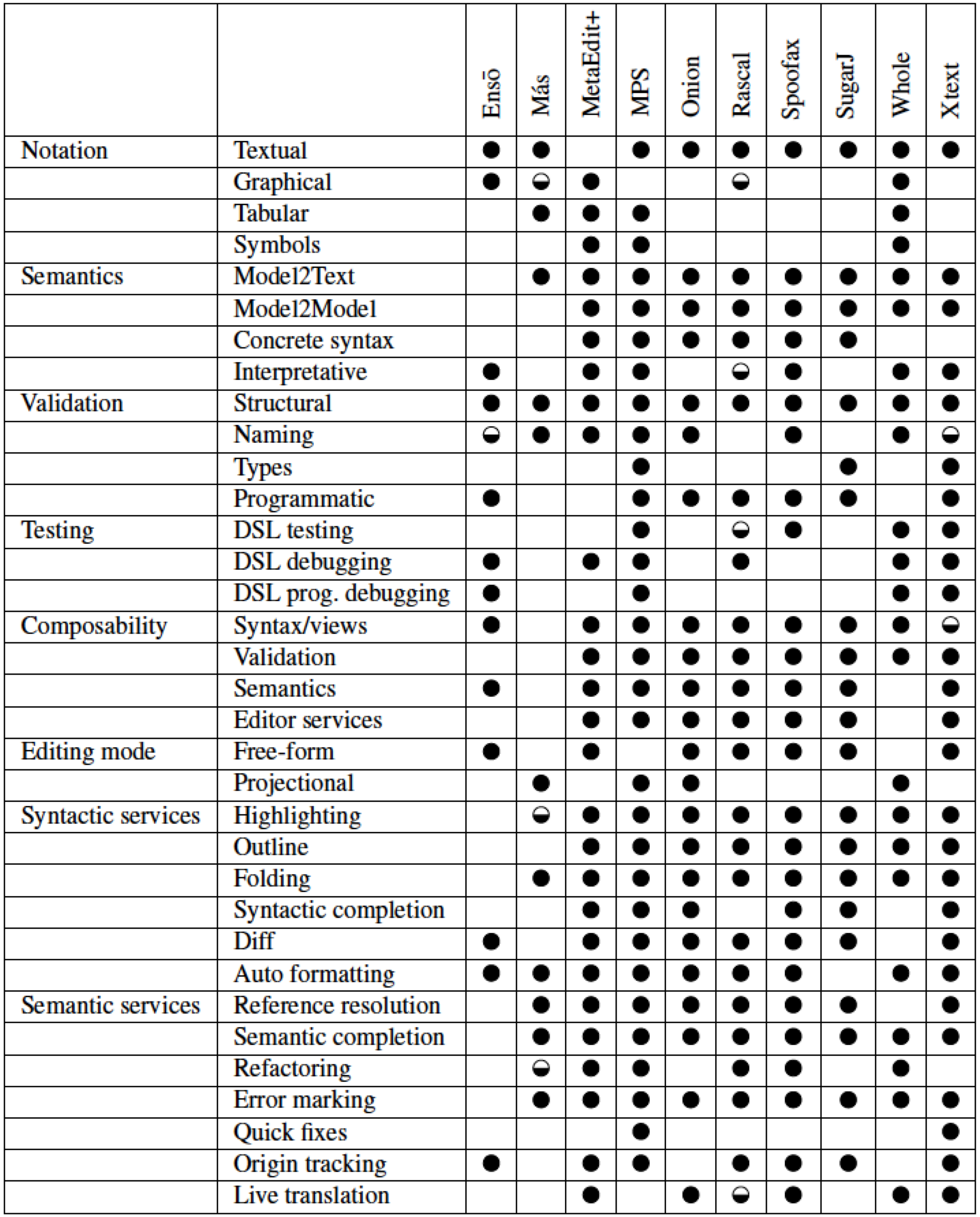
```
[Event "F/S Return Match"]
[Site "Belgrade, Serbia Yugoslavia|J
[Date "1992.11.04"]
[Round "29"]
[White "Fischer, Robert J."]
[Black "Spassky, Boris V."]
[Result "1/2-1/2"]

1. e4 e5 2. Nf3 Nc6 3. Bb5 a6 {This opening is called the Ruy Lopez.}
4. Ba4 Nf6 5. O-O Be7 6. Re1 b5 7. Bb3 d6 8. c3 O-O 9. h3 Nb8  10. d4 Nbd7
11. c4 c6 12. cxb5 axb5 13. Nc3 Bb7 14. Bg5 b4 15. Nb1 h6 16. Bh4 c5 17. dxe5
Nxe4 18. Bxe7 Qxe7 19. exd6 Qf6 20. Nbd2 Nxd6 21. Nc4 Nxc4 22. Bxc4 Nb6
23. Ne5 Rae8 24. Bxf7+ Rxf7 25. Nxf7 Rxe1+ 26. Qxe1 Kxf7 27. Qe3 Qg5 28. Qxg5
hxg5 29. b3 Ke6 30. a3 Kd6 31. axb4 cxb4 32. Ra5 Nd5 33. f3 Bc8 34. Kf2 Bf5
35. Ra7 g6 36. Ra6+ Kc5 37. Ke1 Nf4 38. g3 Nxh3 39. Kd2 Kb5 40. Rd6 Kc5 41. Ra6
Nf2 42. g4 Bd3 43. Re6 1/2-1/2
```

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

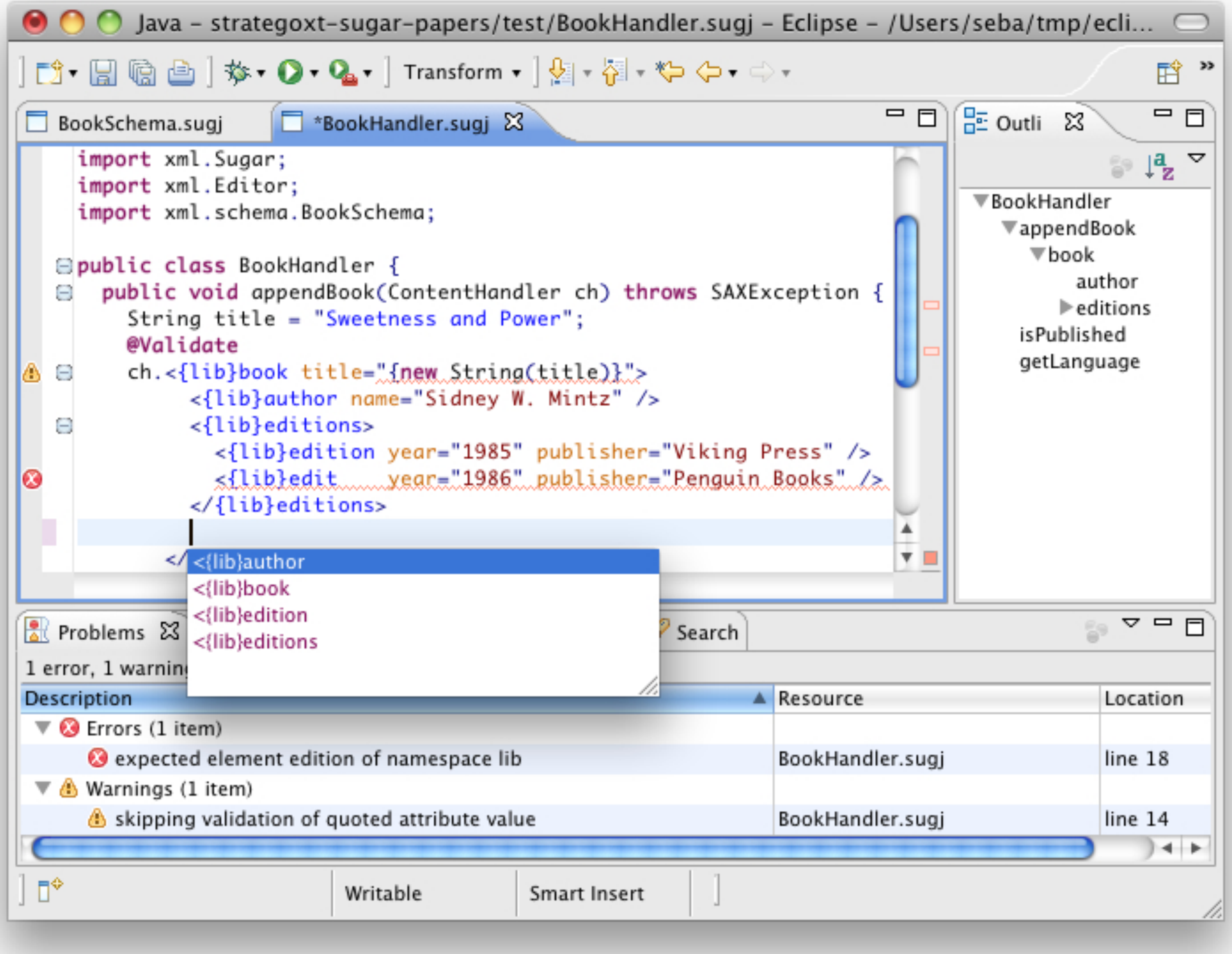
Language Workbenches

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)



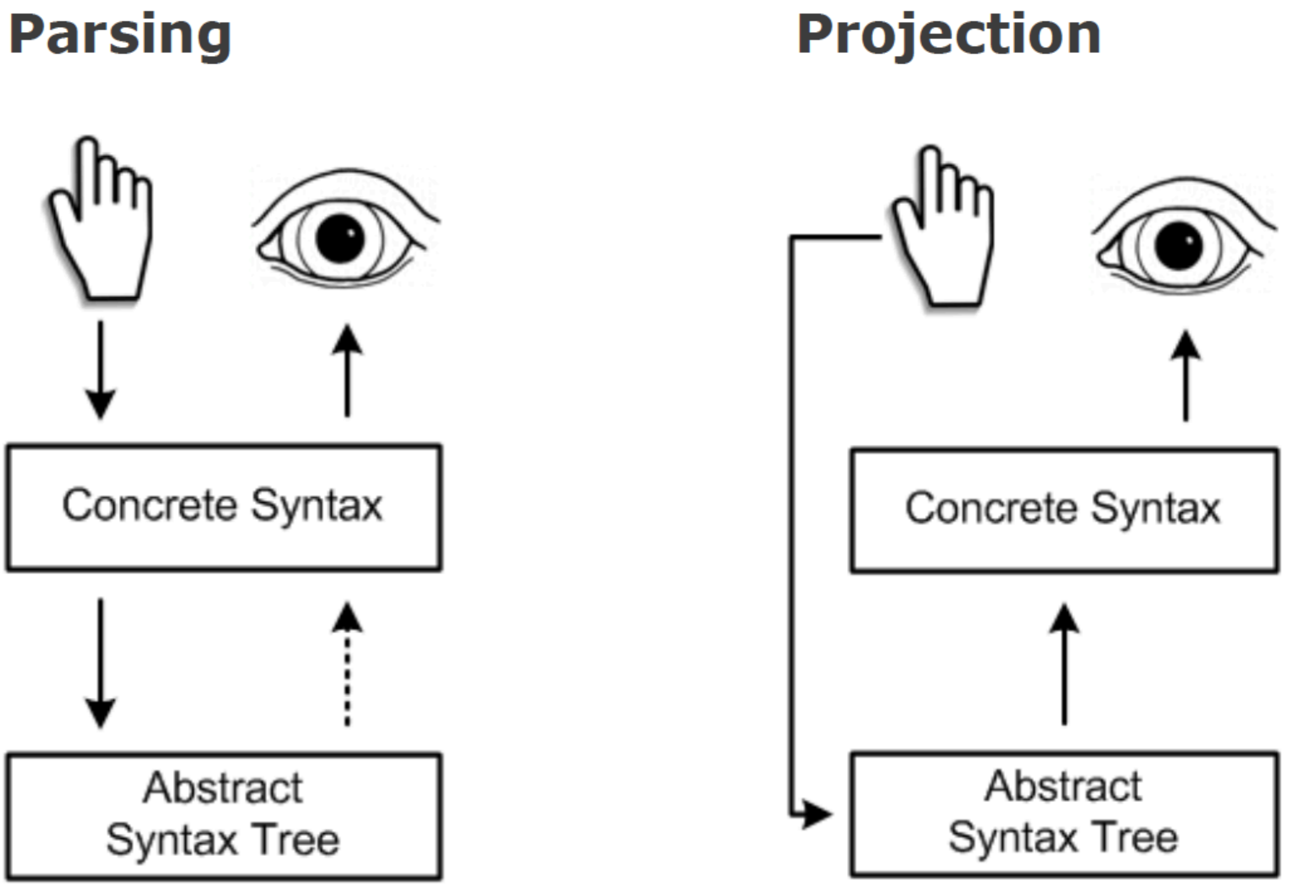
```
import xml.Sugar;
import xml.Editor;
import xml.schema.BookSchema;

public class BookHandler {
  public void appendBook(ContentHandler ch) throws SAXException {
    String title = "Sweetness and Power";
    @Validate
    ch.<{lib}book title="{new String(title)}">
         <{lib}author name="Sidney W. Mintz" />
         <{lib}editions>
           <{lib}edition year="1985" publisher="Viking Press" />
           <{lib}edit    year="1986" publisher="Penguin Books" />
         </{lib}editions>

       </
```

1 error, 1 warnin

Description	Resource	Location
Errors (1 item)		
expected element edition of namespace lib	BookHandler.sugj	line 18
Warnings (1 item)		
skipping validation of quoted attribute value	BookHandler.sugj	line 14

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

Projectional editing



Parsing

Concrete Syntax

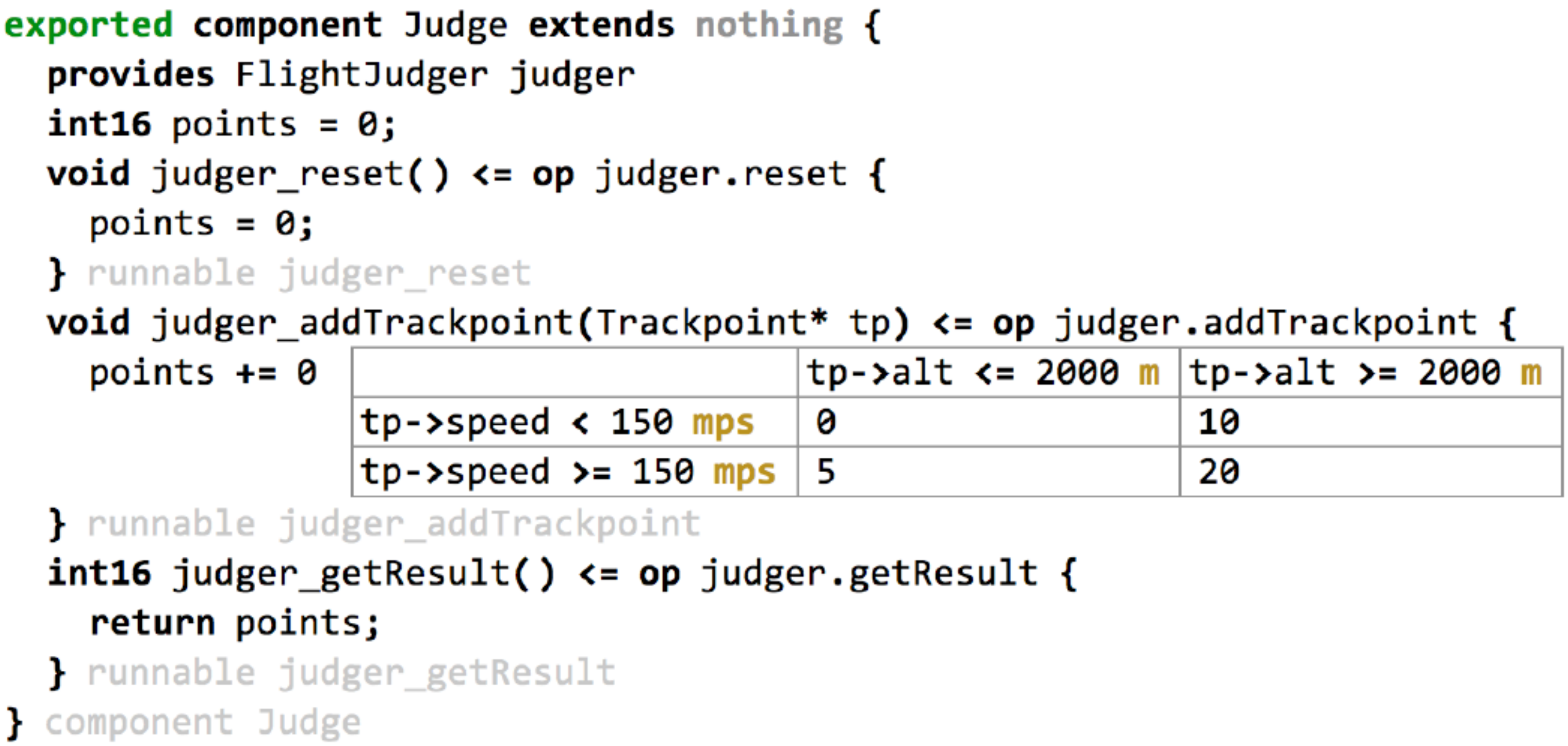
Abstract Syntax Tree

Projection

Concrete Syntax

Abstract Syntax Tree

Projectional editing

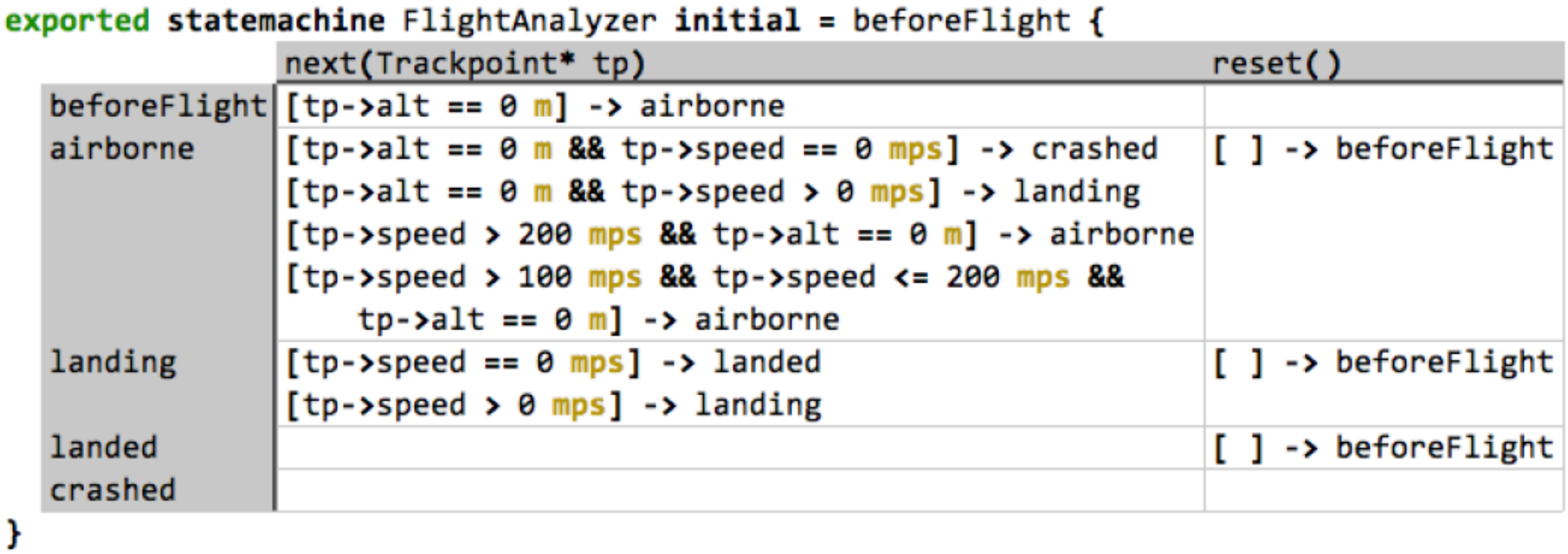


```
exported component Judge extends nothing {
  provides FlightJudger judger
  int16 points = 0;
  void judger_reset() <= op judger.reset {
    points = 0;
  } runnable judger_reset
  void judger_addTrackpoint(Trackpoint* tp) <= op judger.addTrackpoint {
    points += 0
```

	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
```

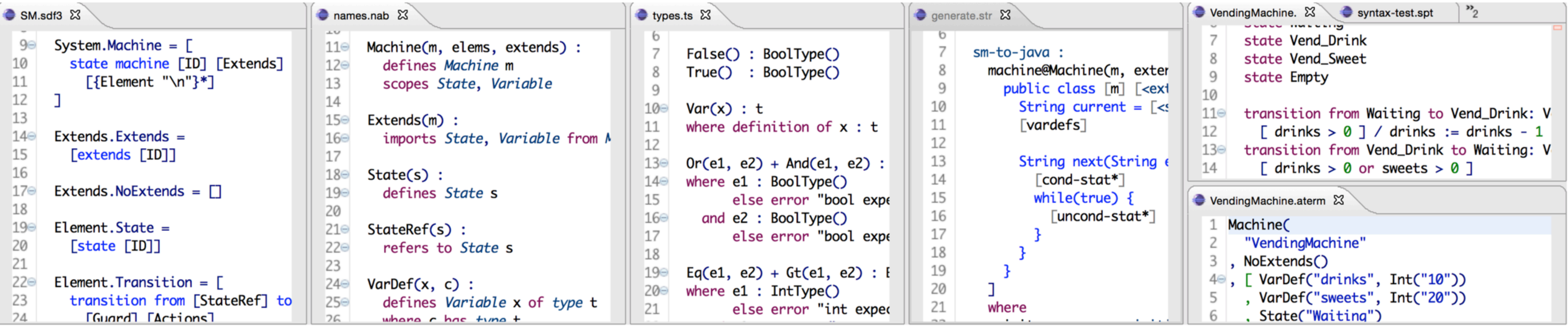
Projectional Editing



```
exported statemachine FlightAnalyzer initial = beforeFlight {
```

	next(Trackpoint* tp)	reset()
beforeFlight	[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		

```
}
```



```
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]
```

```
Machine(m, elems, extends) :
  defines Machine m
  scopes State, Variable

Extends(m) :
  imports State, Variable from M

State(s) :
  defines State s

StateRef(s) :
  refers to State s

VarDef(x, c) :
  defines Variable x of type t
  where c has type t
```

```
False() : BoolType()
True()  : BoolType()

Var(x) : t
where definition of x : t

Or(e1, e2) + And(e1, e2) :
where e1 : BoolType()
        else error "bool expe
  and e2 : BoolType()
        else error "bool expe

Eq(e1, e2) + Gt(e1, e2) : B
where e1 : IntType()
        else error "int expec
```

```
sm-to-java :
  machine@Machine(m, exter
    public class [m] [<ext
      String current = [<s
      [vardefs]

      String next(String e
        [cond-stat*]
        while(true) {
          [uncond-stat*]
        }
      }
    }
  ]
  where
```

```
state Vend_Drink
state Vend_Sweet
state Empty

transition from Waiting to Vend_Drink: V
  [ drinks > 0 ] / drinks := drinks - 1
transition from Vend_Drink to Waiting: V
  [ drinks > 0 or sweets > 0 ]
```

```
Machine(
  "VendingMachine"
, NoExtends()
, [ VarDef("drinks", Int("10"))
  , VarDef("sweets", Int("20"))
  , 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

http://metaborg.org/spoofax/#meta-languages

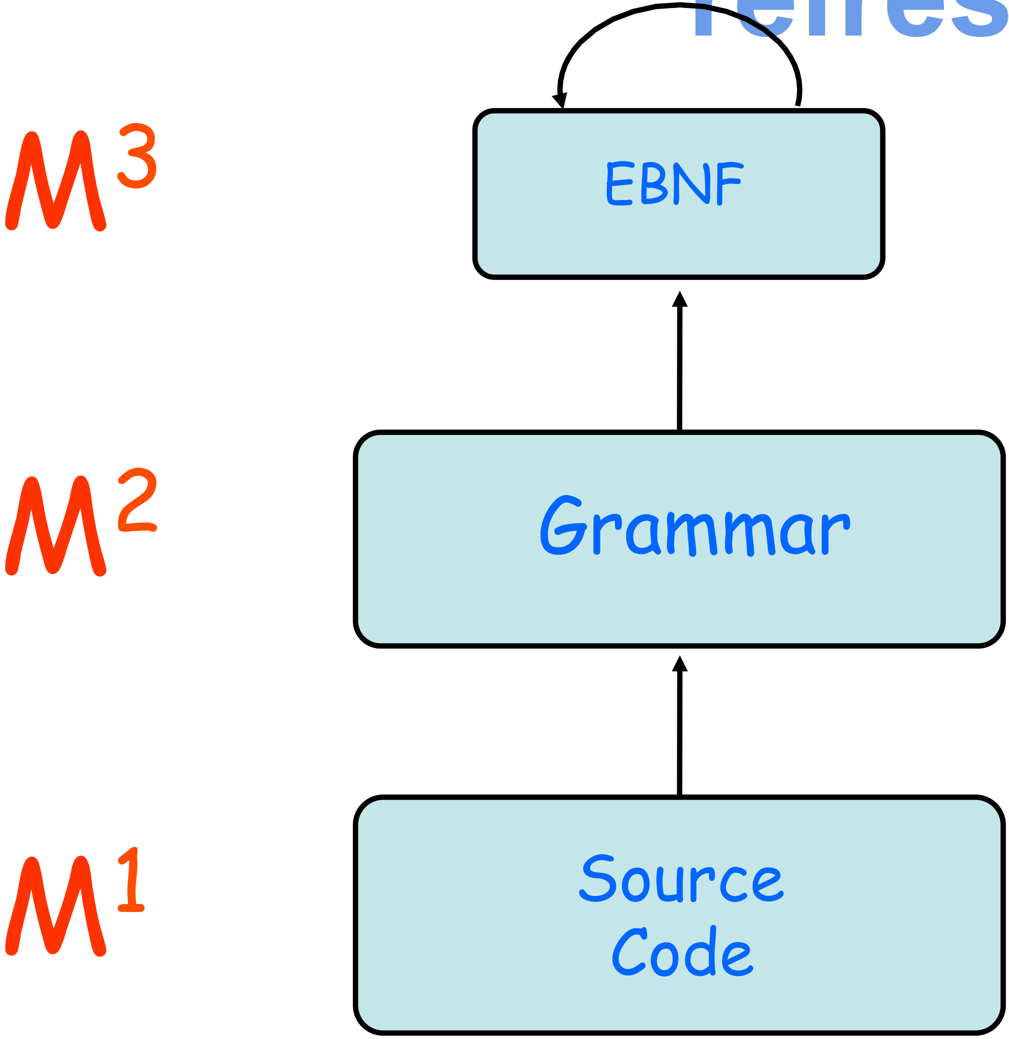
Xtext, a popular, easy-to-use model-based tool for developping DSLs

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

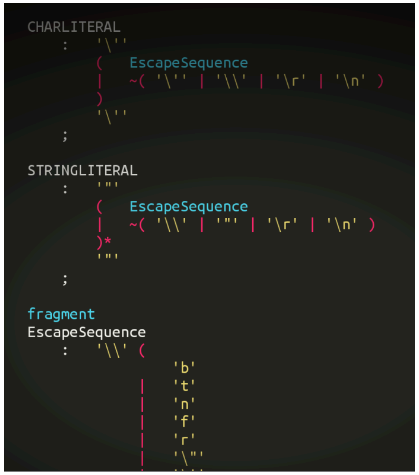
Your DSL in 5'

Short Demonstration

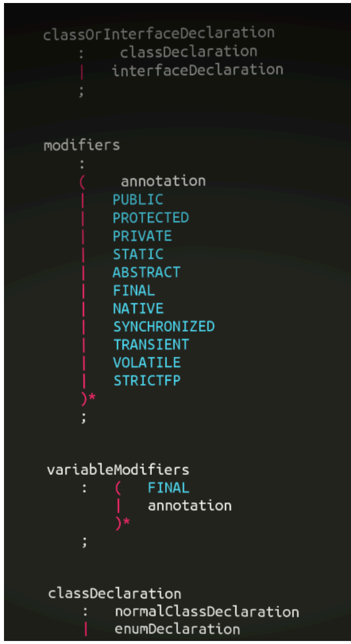
Foundations (or some course refresh)



Java Grammar

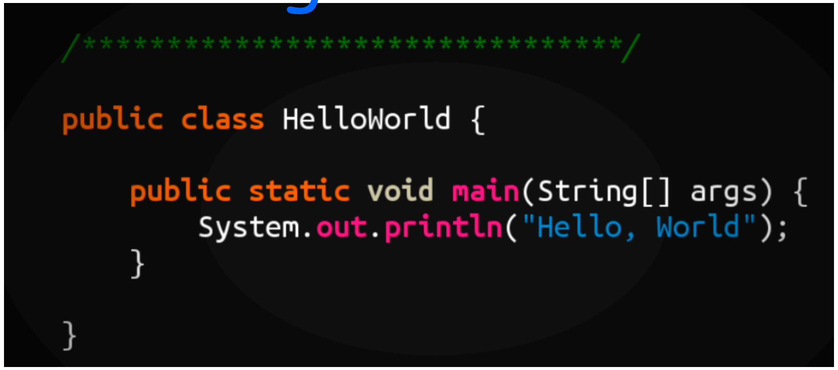


```
CHARLITERAL
    :   '\''
        (   EscapeSequence
        |   ~( '\'' | '\\' | '\r' | '\n' )
        )
        '\''
    ;

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
```

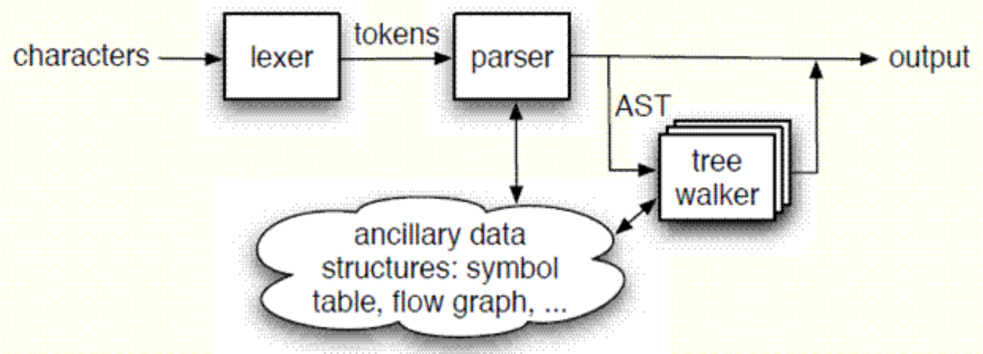
Java Program

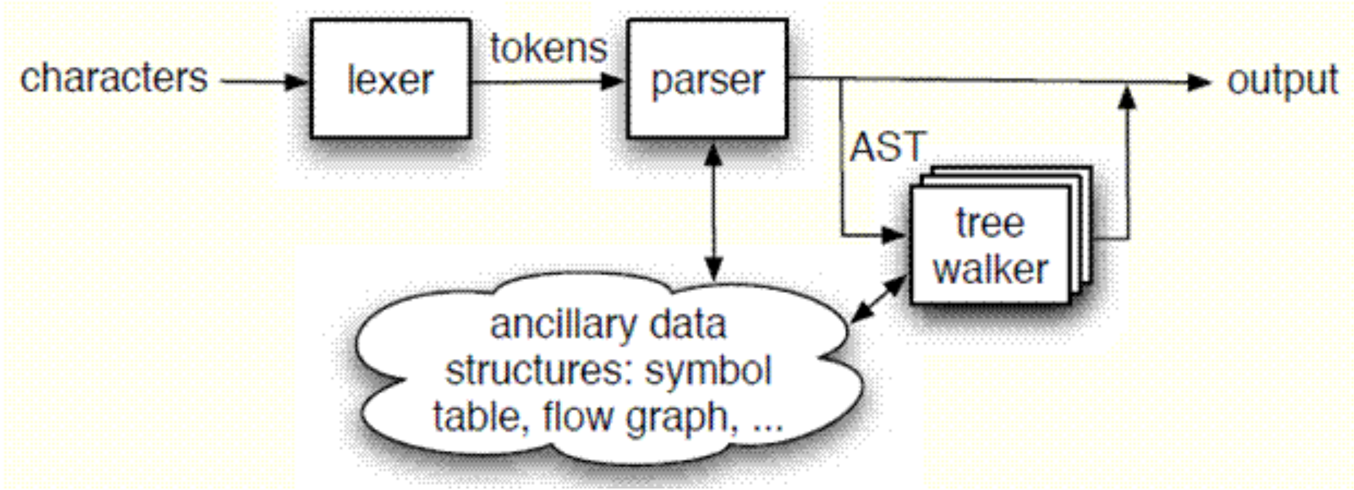


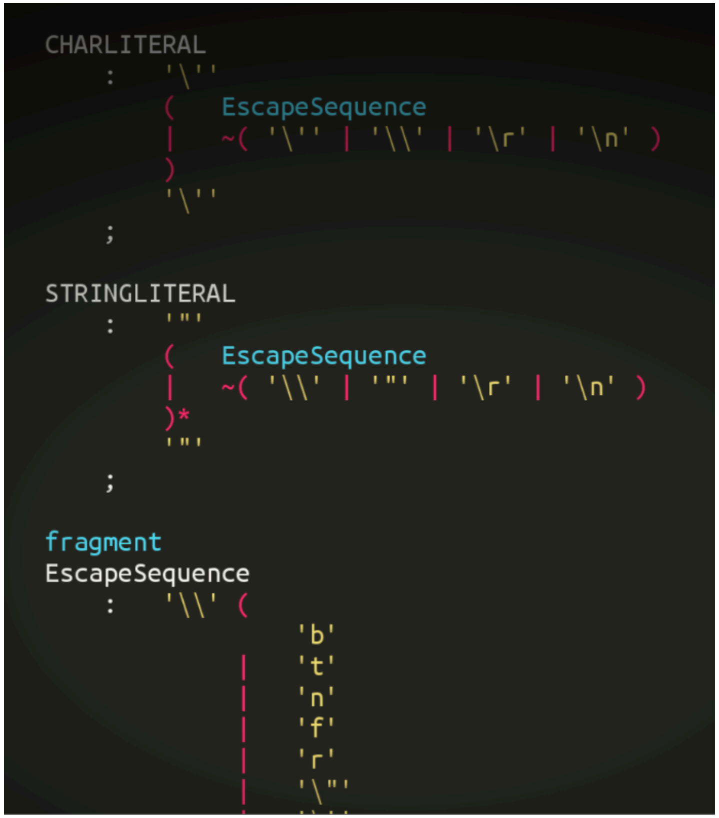
```
/*******************************/

public class HelloWorld {

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

}
```

Compilation Process

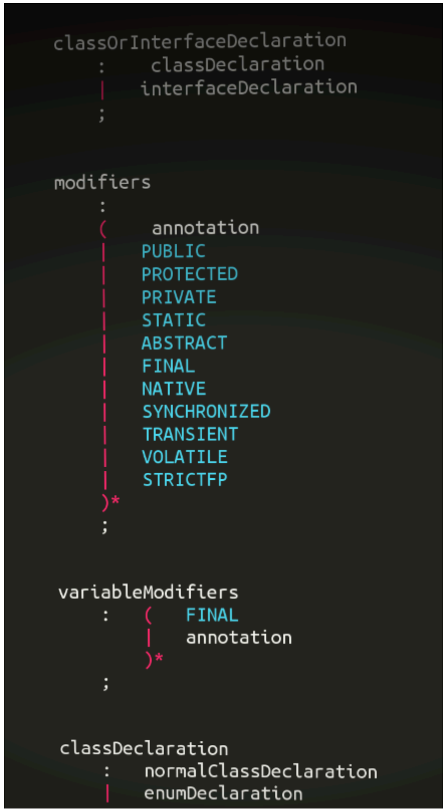
- Source code
  - Concrete syntax used for specifying a program
  - Conformant to a grammar
- Lexical analysis
  - Conveting a sequence of characters into a sequence of **tokens**
- Parsing (Syntactical analysis)
  - Abtsract Syntax Tree (AST)





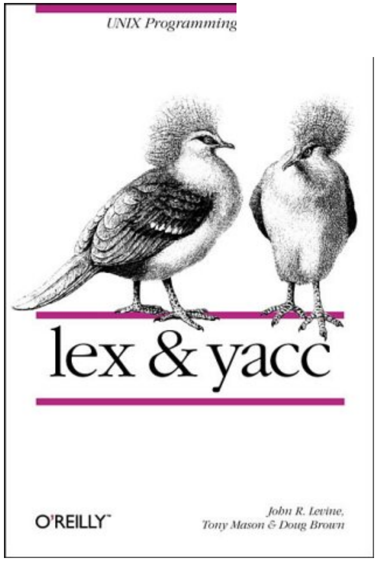


```
CHARLITERAL
    :   '\''
        (   EscapeSequence
        |   ~( '\'' | '\\' | '\r' | '\n' )
        )
        '\''
    ;

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
```

The Pragmatic Programmers
The Definitive ANTLR Reference
Building Domain-Specific Languages
Terence Parr



EXEMPLE

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

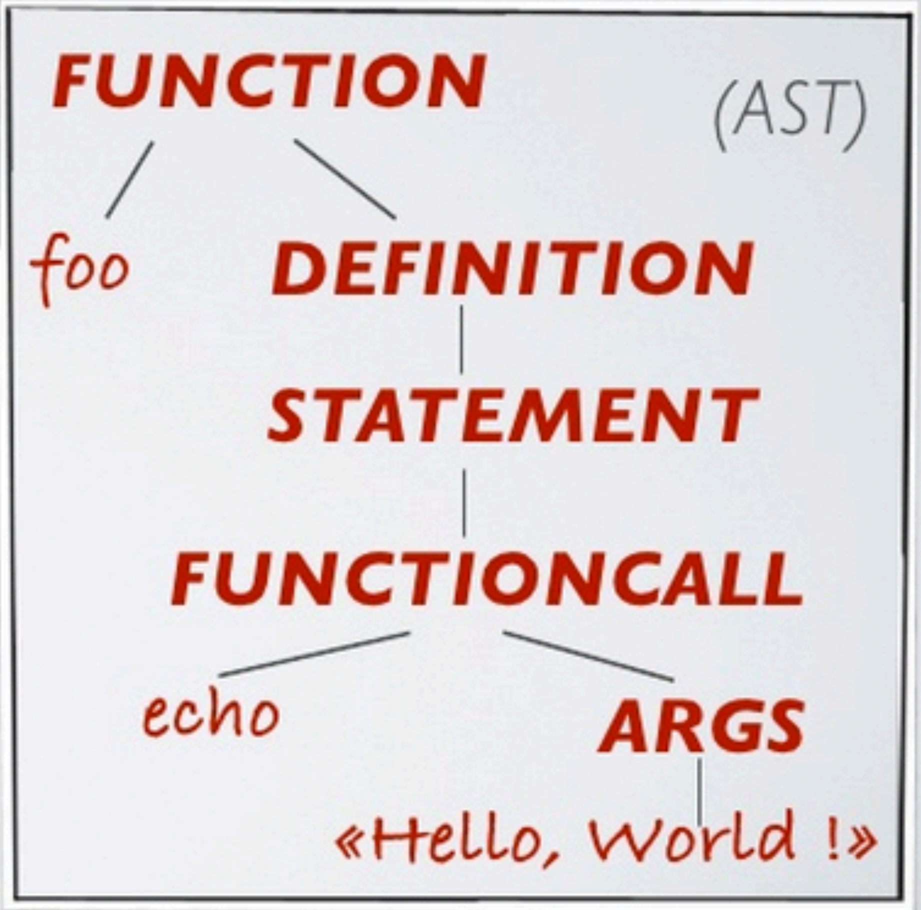
(Syntaxe concrète)

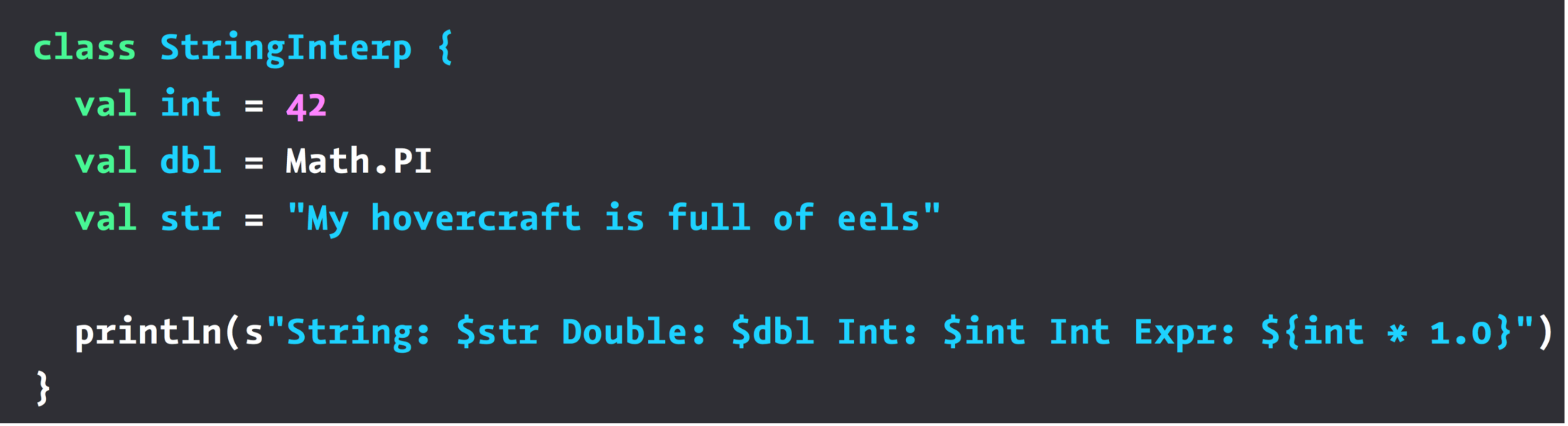
↓

#FUNC
↓
#SYMBOL(«foo»)
↓
#LPAREN
↓
#RPAREN
↓
…
↓
#RBRACKET

(lexèmes)

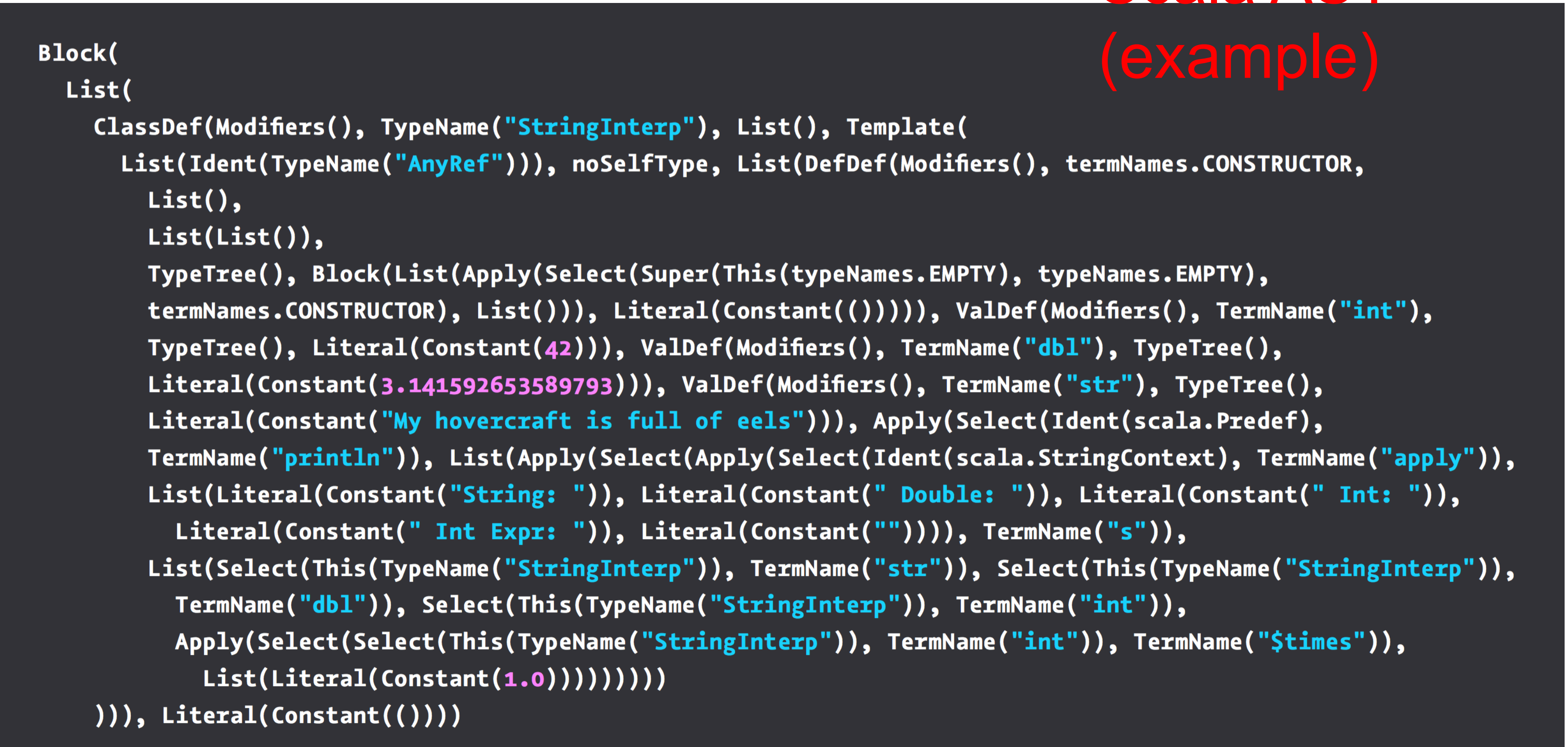
→





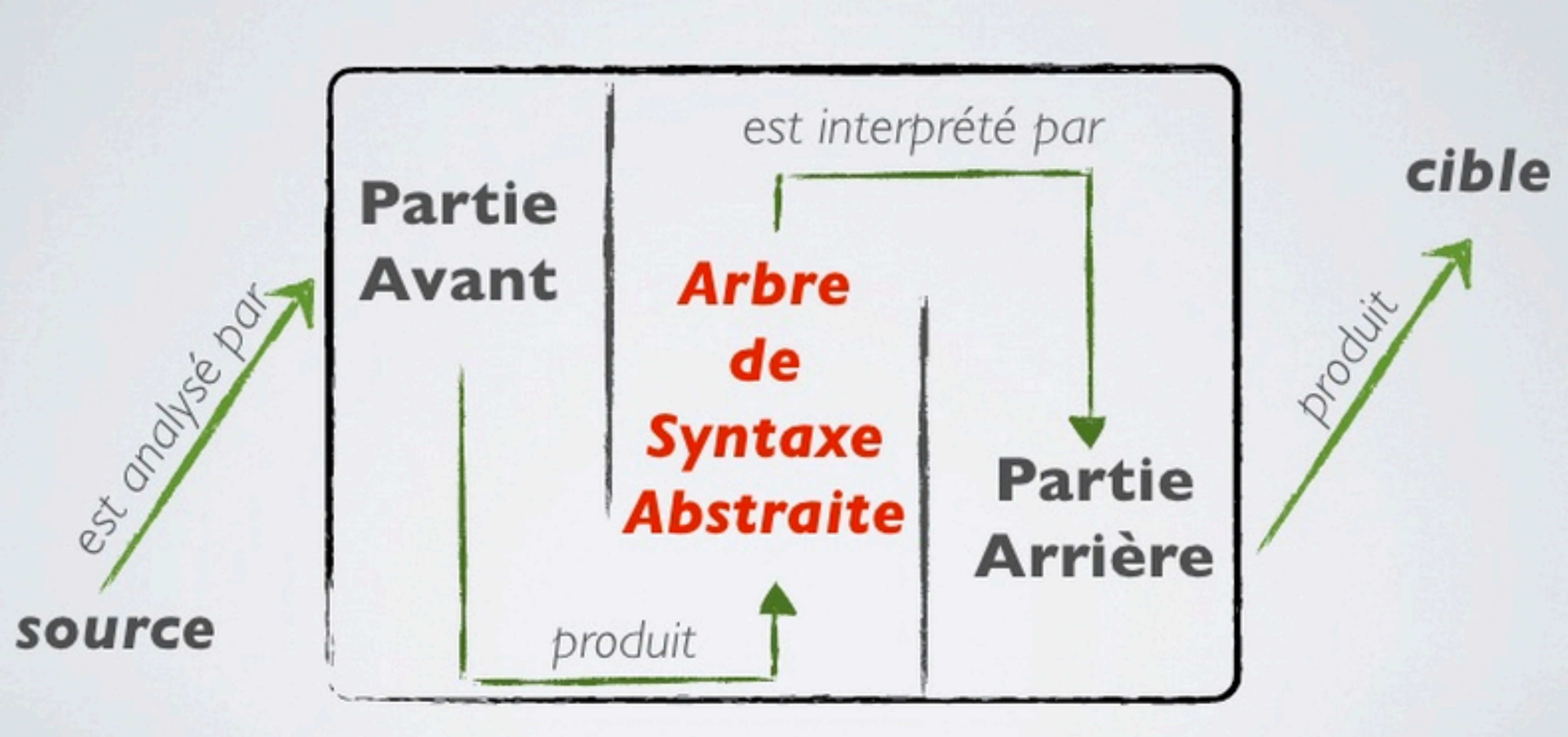
```scala
class StringInterp {
  val int = 42
  val dbl = Math.PI
  val str = "My hovercraft is full of eels"

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

Scala AST (example)

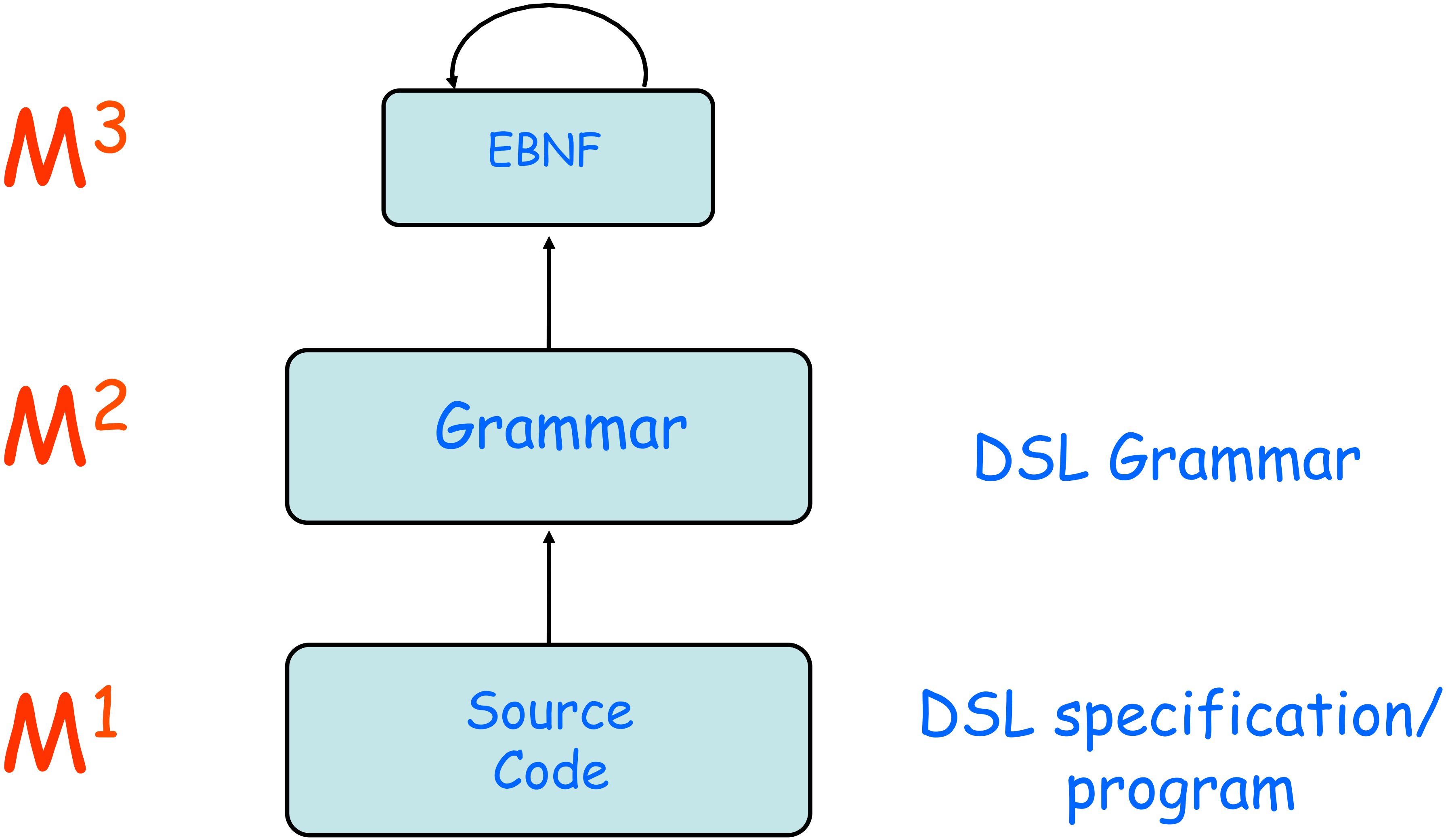


```
Block(
  List(
    ClassDef(Modifiers(), TypeName("StringInterp"), List(), Template(
      List(Ident(TypeName("AnyRef"))), noSelfType, List(DefDef(Modifiers(), termNames.CONSTRUCTOR,
        List(),
        List(List()),
        TypeTree(), Block(List(Apply(Select(Super(This(typeNames.EMPTY), typeNames.EMPTY),
        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)))))))))
    ))), Literal(Constant(())))
```

Compilation (en français)



DSL? The same!



M^3

EBNF

M^2

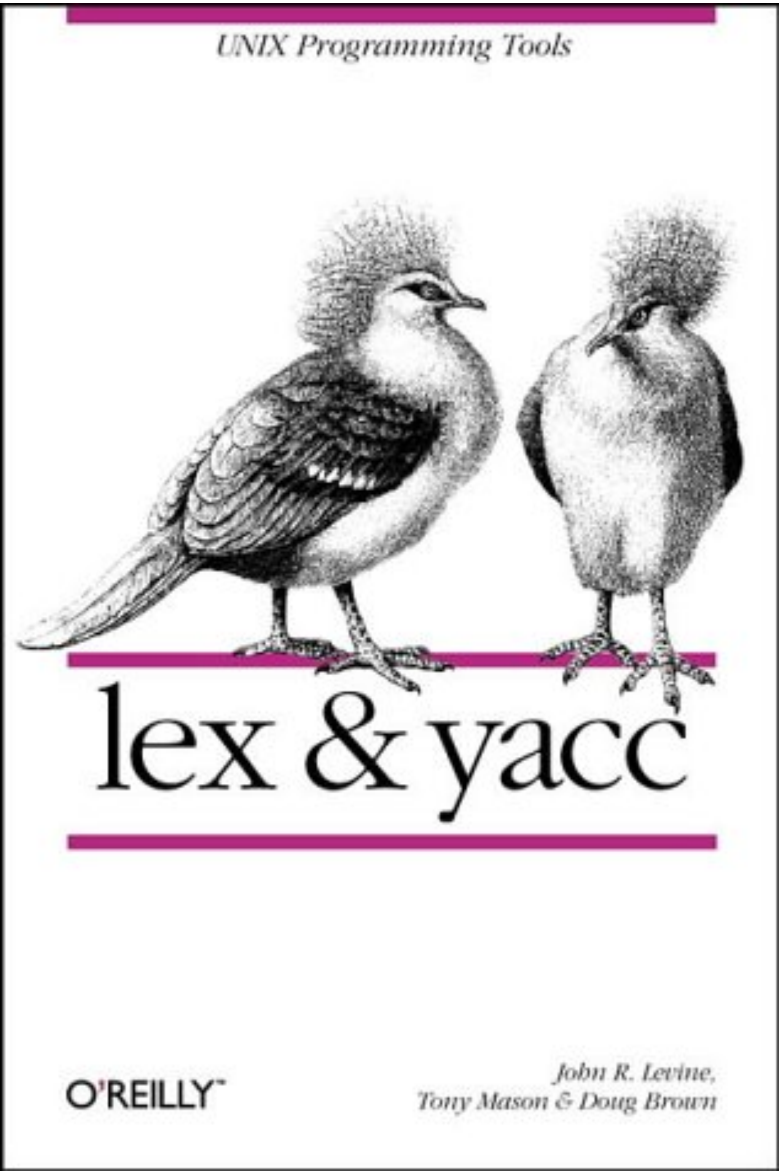
Grammar

DSL Grammar

M^1

Source Code

DSL specification/ program

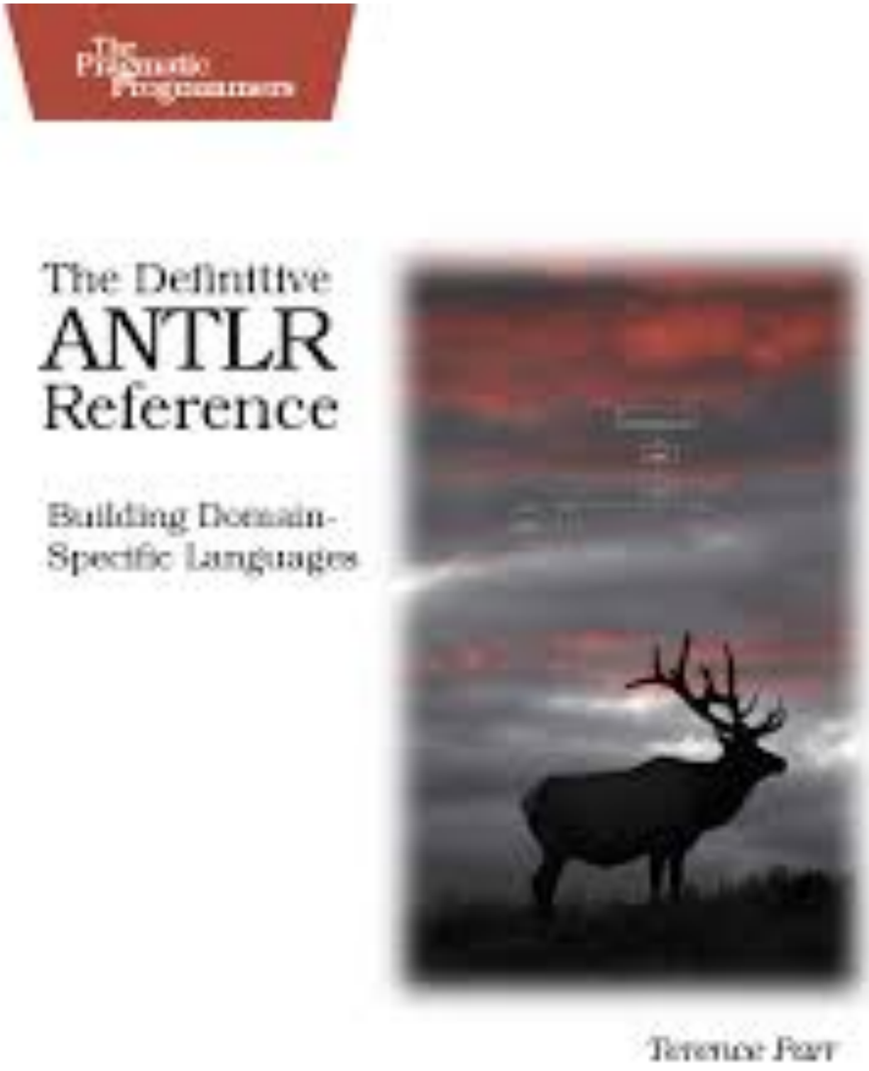


UNIX Programming Tools

lex & yacc

O'REILLY™

John R. Levine,
Tony Mason & Doug Brown

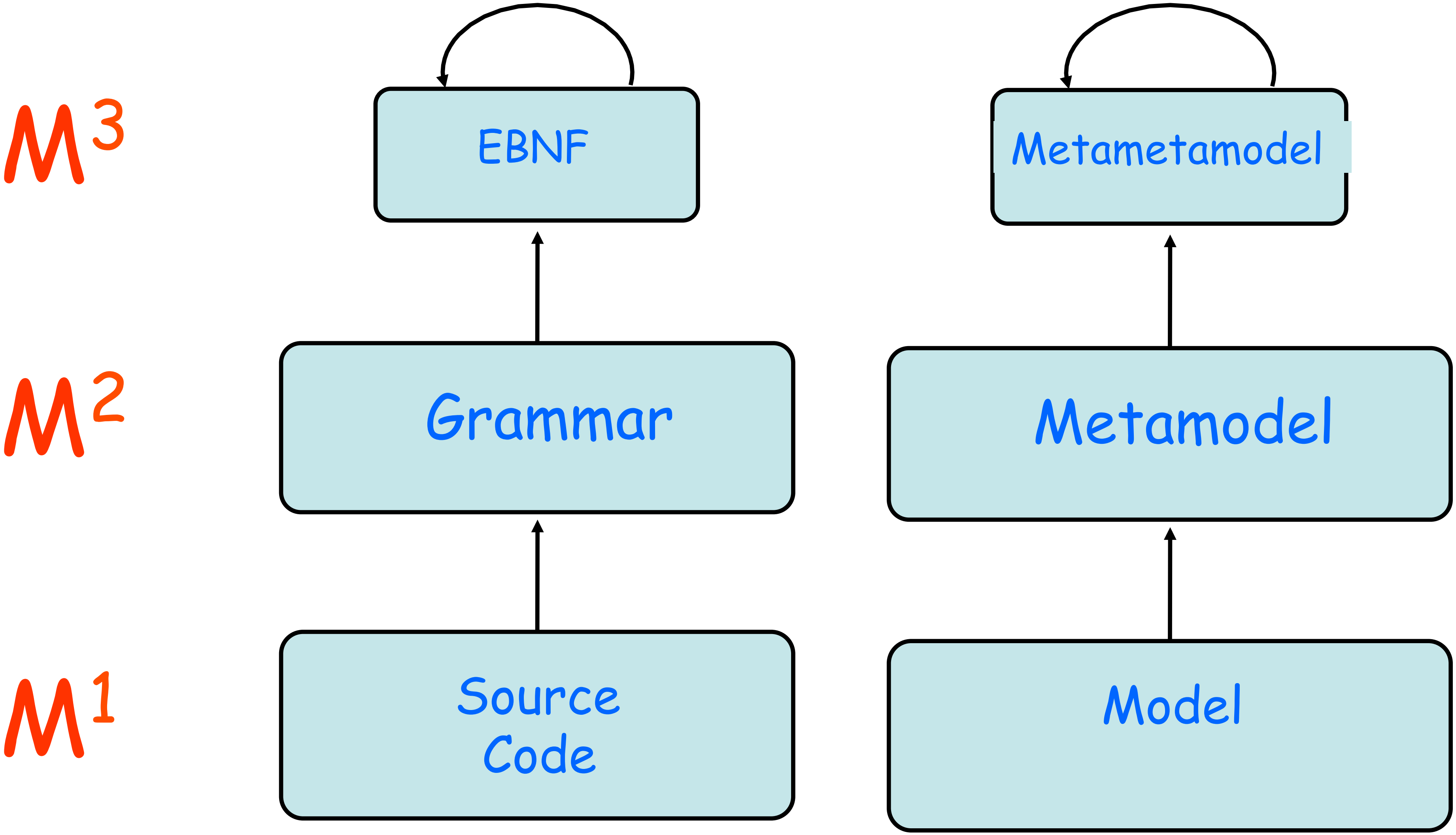


The Pragmatic Programmers

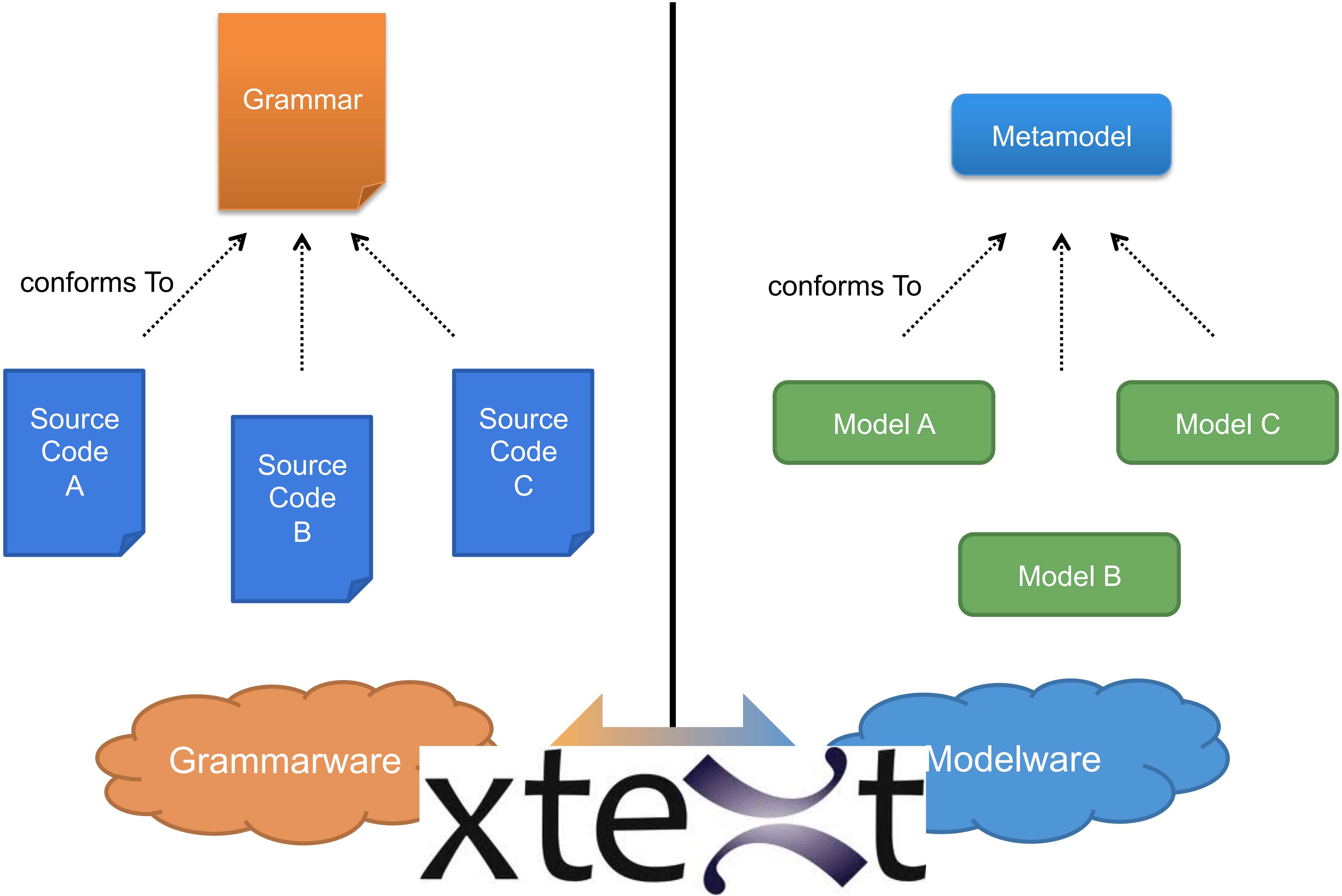
The Definitive
ANTLR
Reference

Building Domain-
Specific Languages

Terence Parr


M3
M2
M1
EBNF
Grammar
Source Code
Metametamodel
Metamodel
Model

Language and MDE



xtext

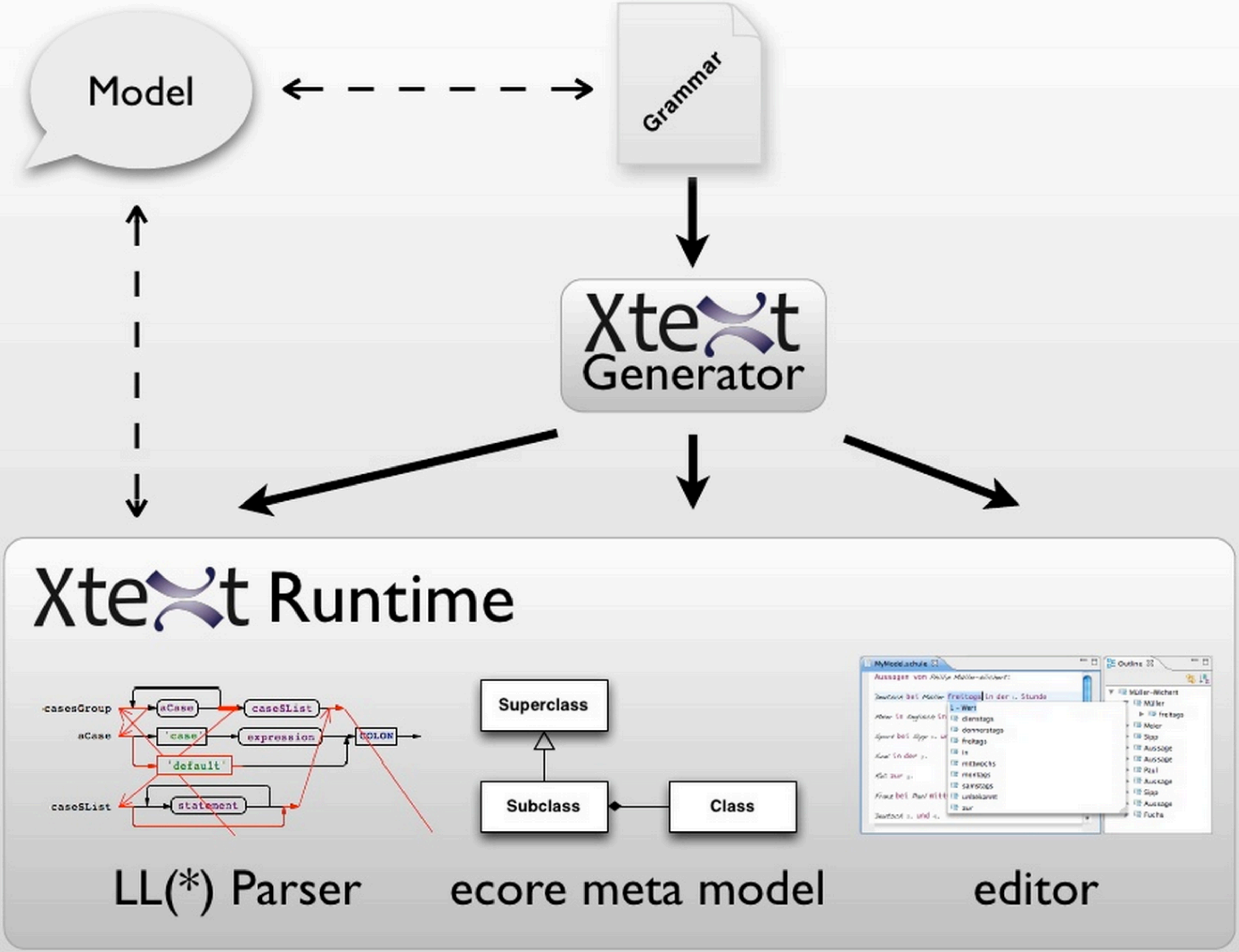
Give me a **grammar**,

I’ll give you (for free)

 * a comprehensive editor (auto-completion, syntax highlitening, 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

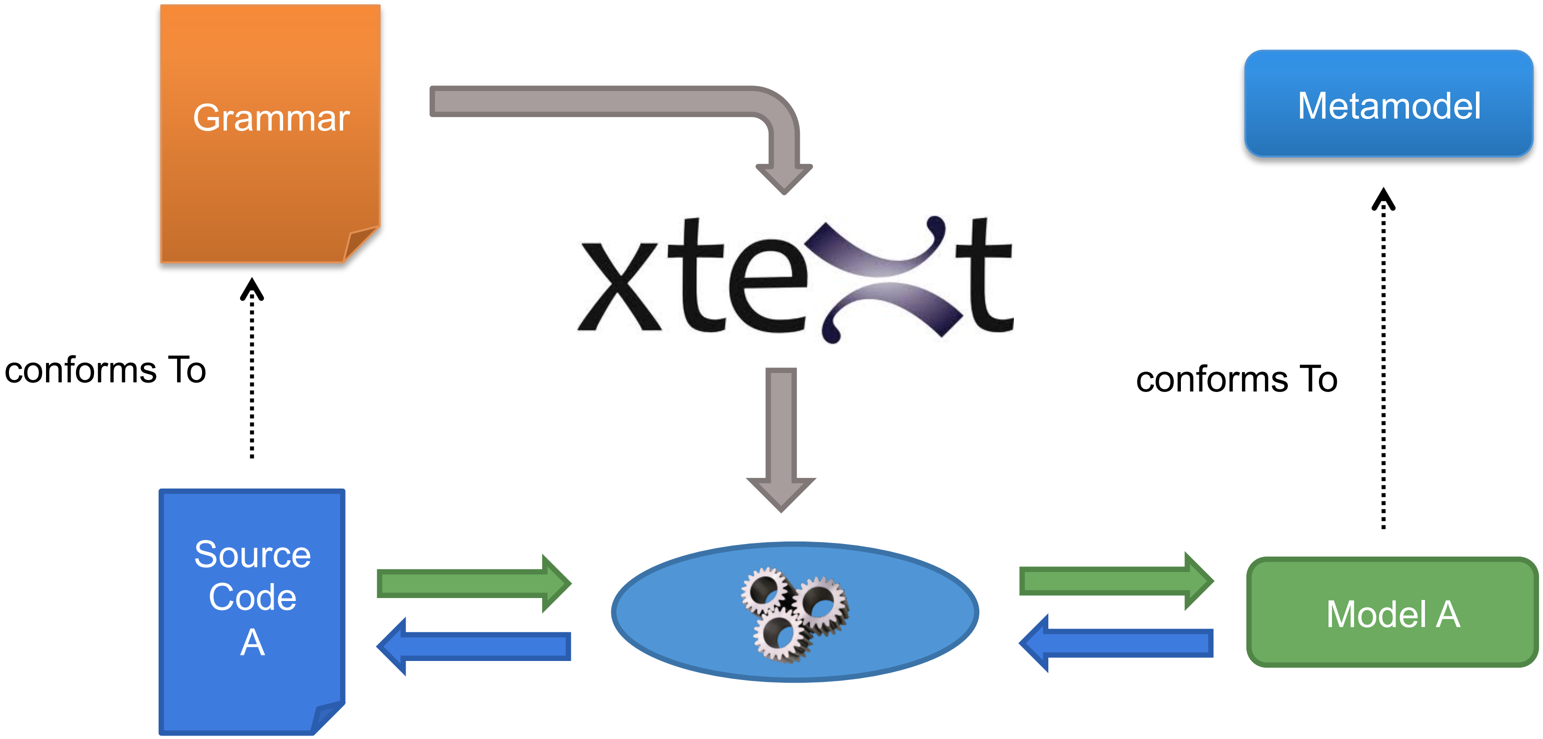
Xtext Runtime

LL(*) Parser

ecore meta model

editor

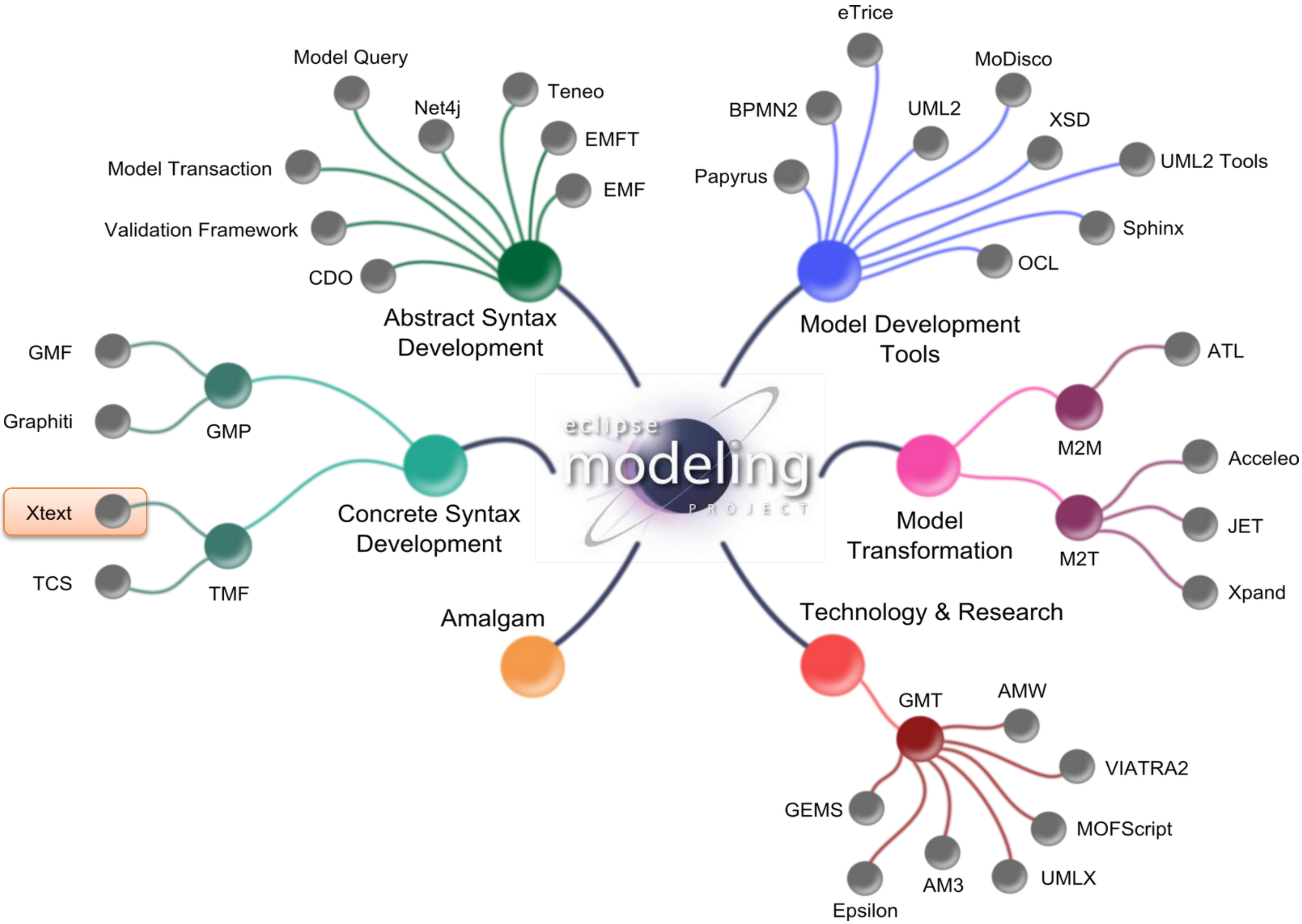
Xtext, Grammar, Metamodel



Xtext Project

- Eclipse Project
  - Part of Eclipse Modeling
  - Part of Open Architecture Ware
- Model-driven development of Textual DSLs
- Part of a family of languages
  - **Xtext**
  - Xtend
  - Xbase
  - Xpand
  - Xcore

Eclipse Modeling Project



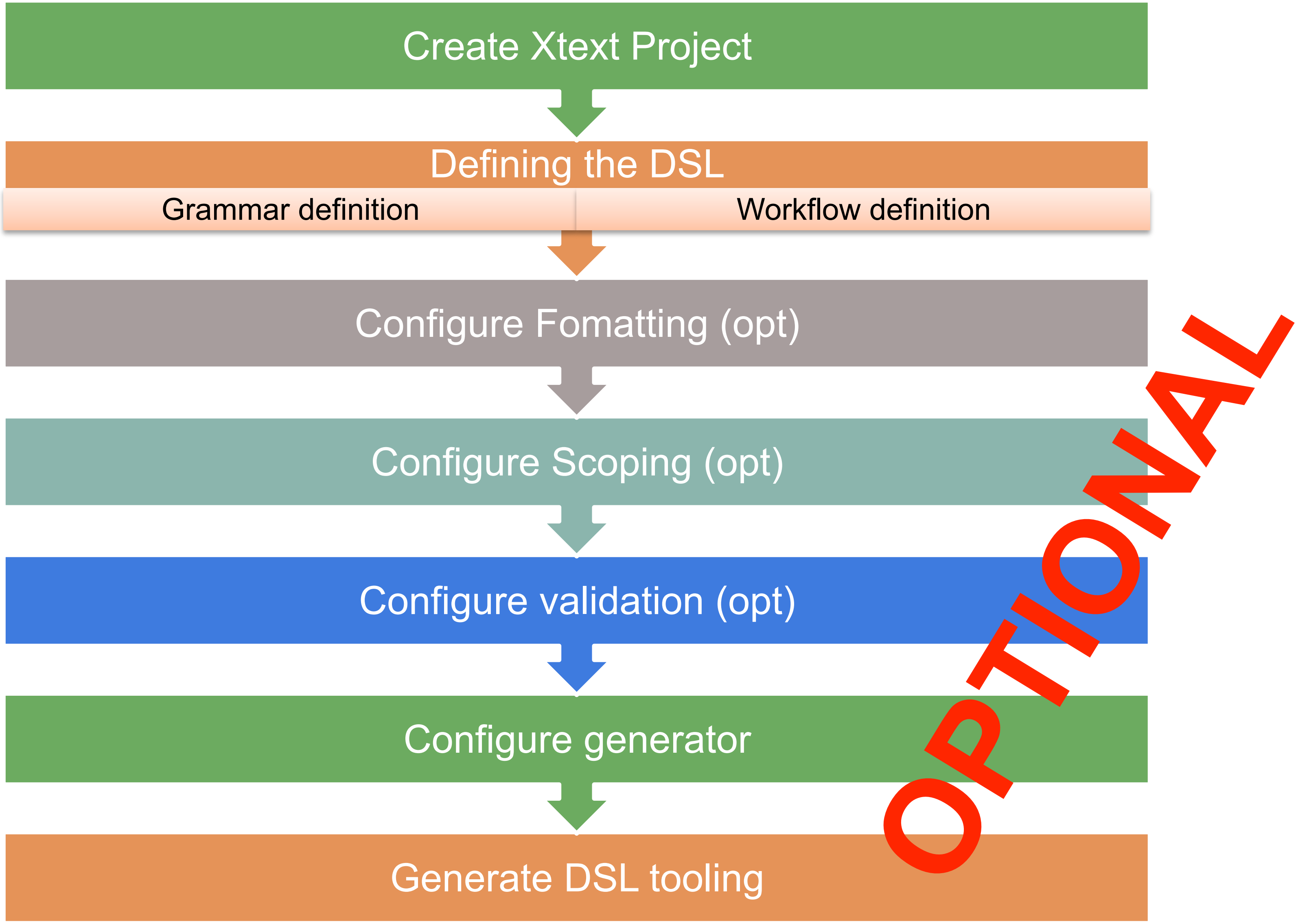
The Grammar Language of Xtext

- 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

Main Advantages

- Consistent look and feel
- Textual DSLs are a resource in Eclipse
- Open editors can be extended
- Complete framework to develop DSLs
- Easy to connect to any Java-based language

Development Process



Create Xtext Project

Defining the DSL

Grammar definition

Workflow definition

Configure Fomatting (opt)

Configure Scoping (opt)

Configure validation (opt)

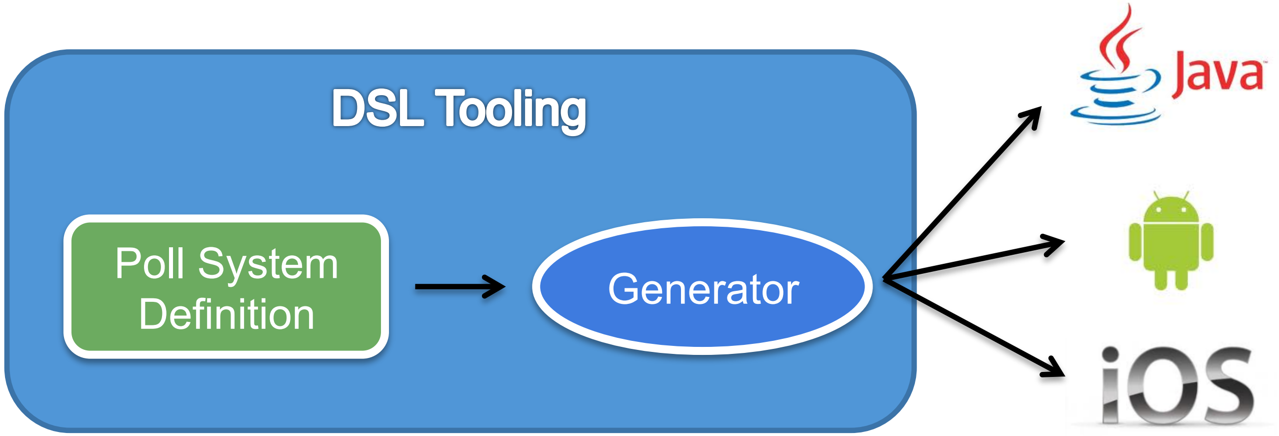
Configure generator

Generate DSL tooling

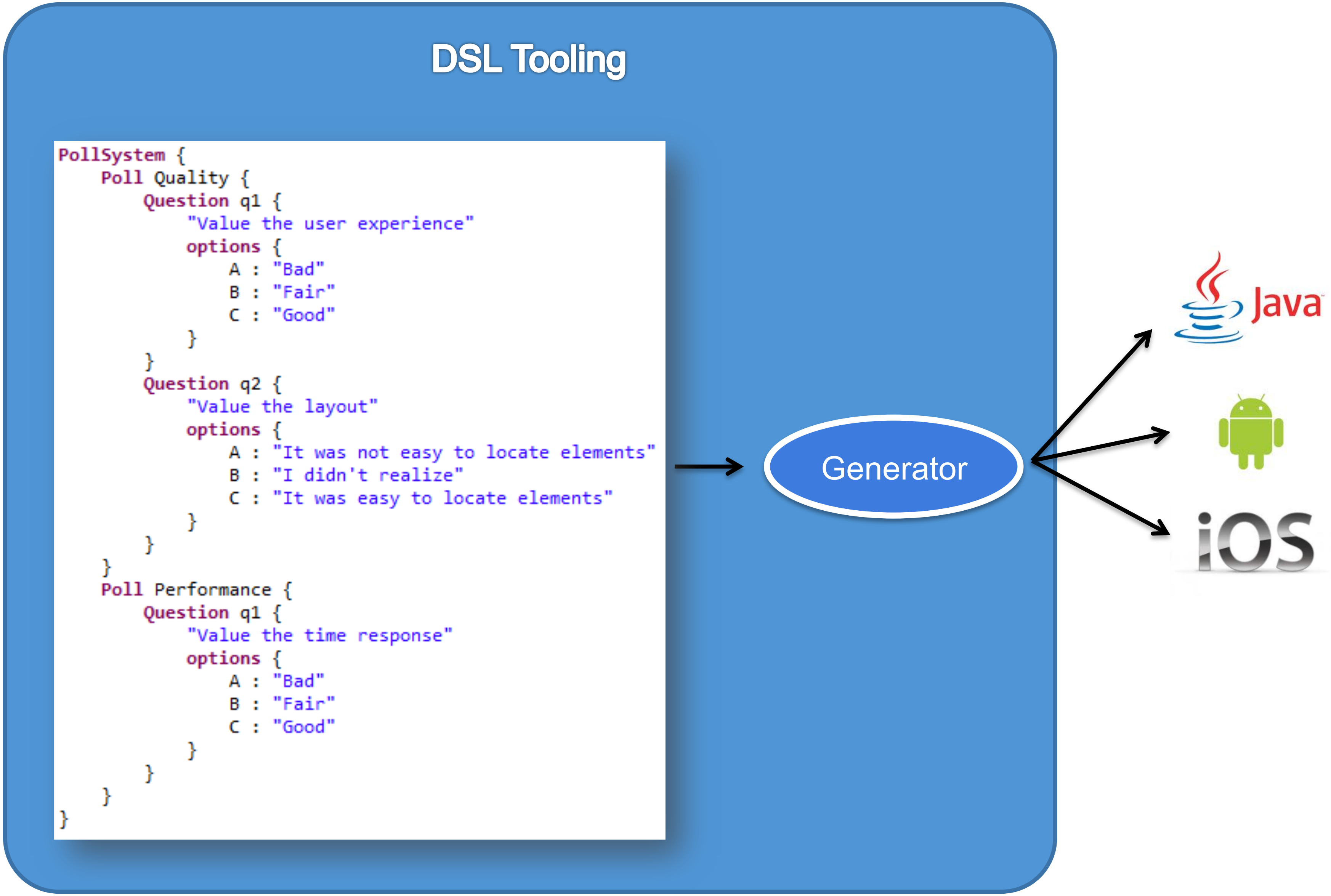
OPTIONAL

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



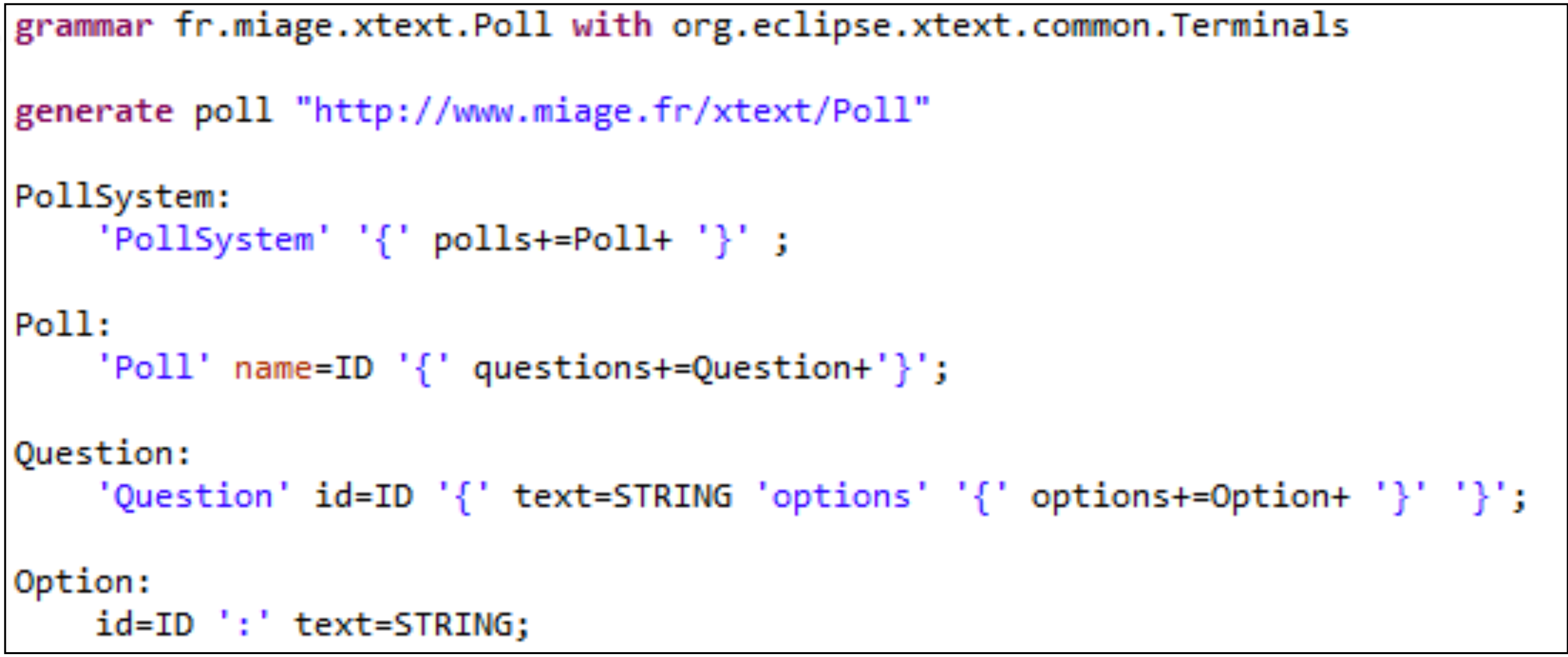
Something like…



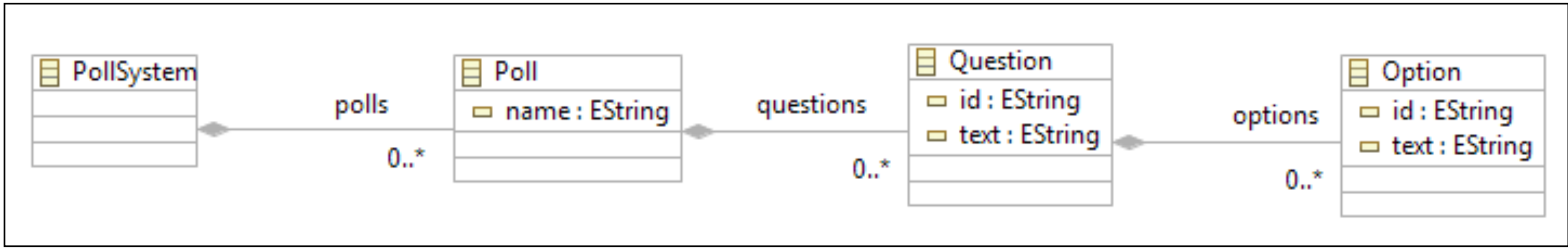
```
PollSystem {
    Poll Quality {
        Question q1 {
            "Value the user experience"
            options {
                A : "Bad"
                B : "Fair"
                C : "Good"
            }
        }
        Question q2 {
            "Value the layout"
            options {
                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

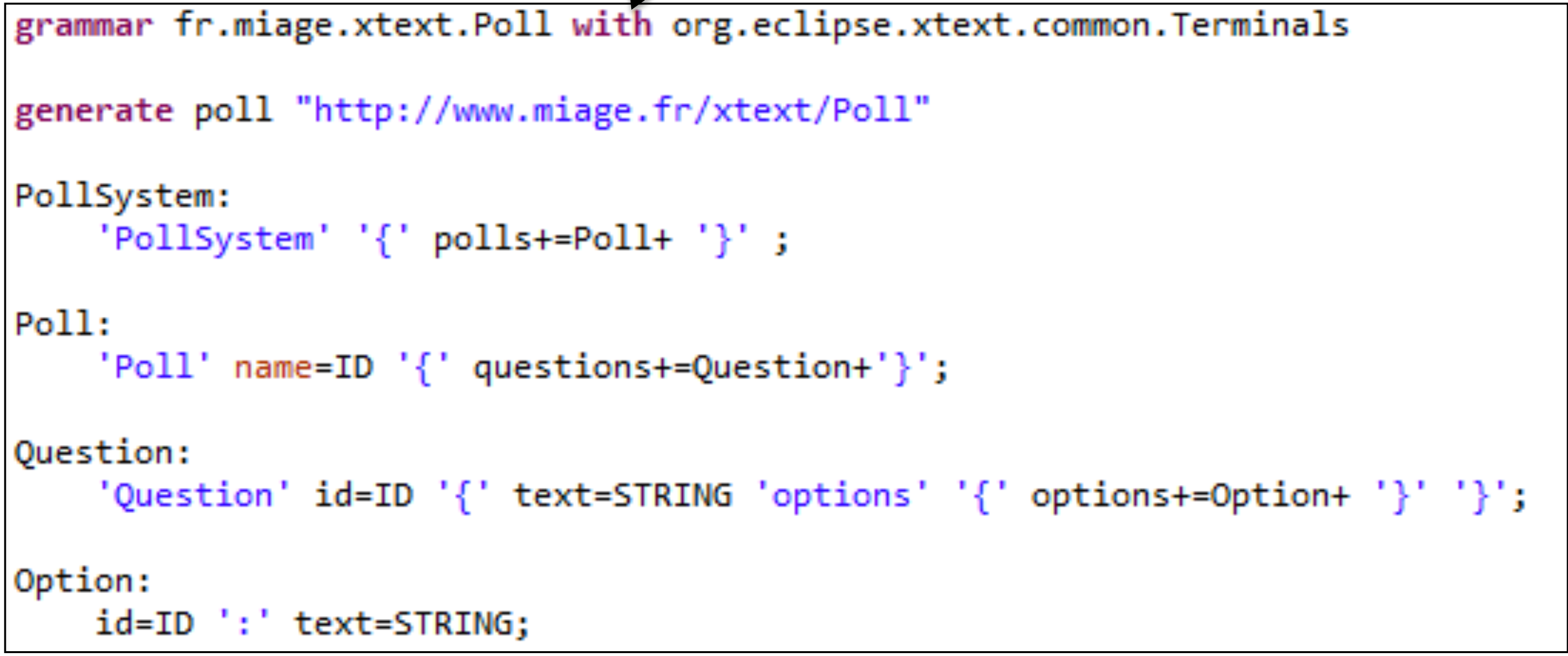


```
grammar fr.miage.xtext.Poll with org.eclipse.xtext.common.Terminals

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;
```

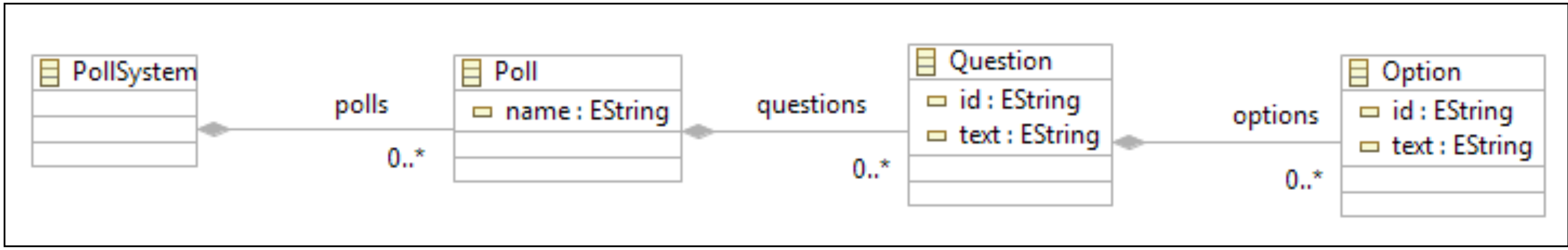


Xtext Grammar

Grammar
reuse

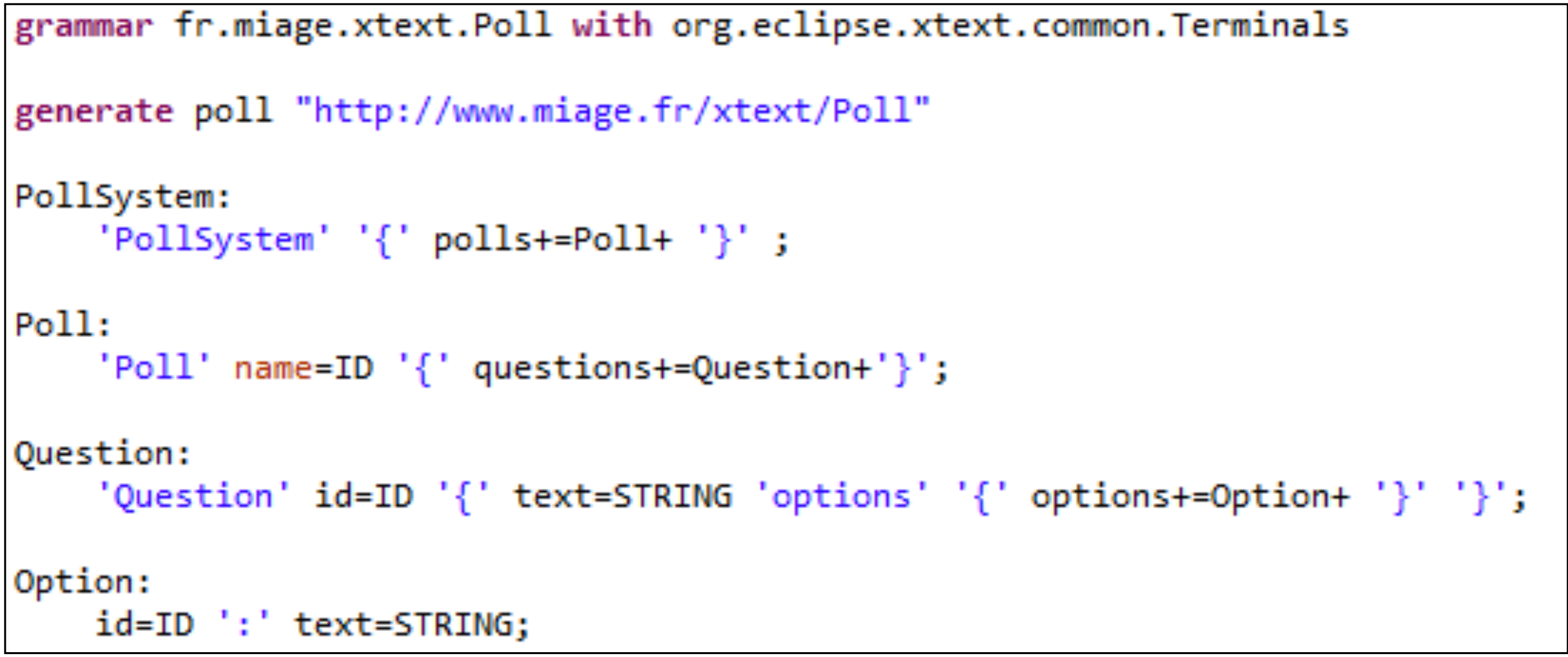


```
grammar fr.miage.xtext.Poll with org.eclipse.xtext.common.Terminals

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;
```

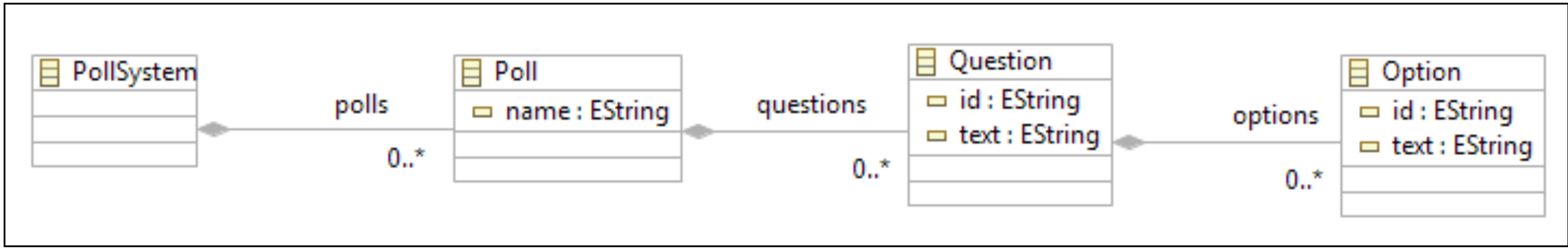


Xtext Grammar

Derived metamodel

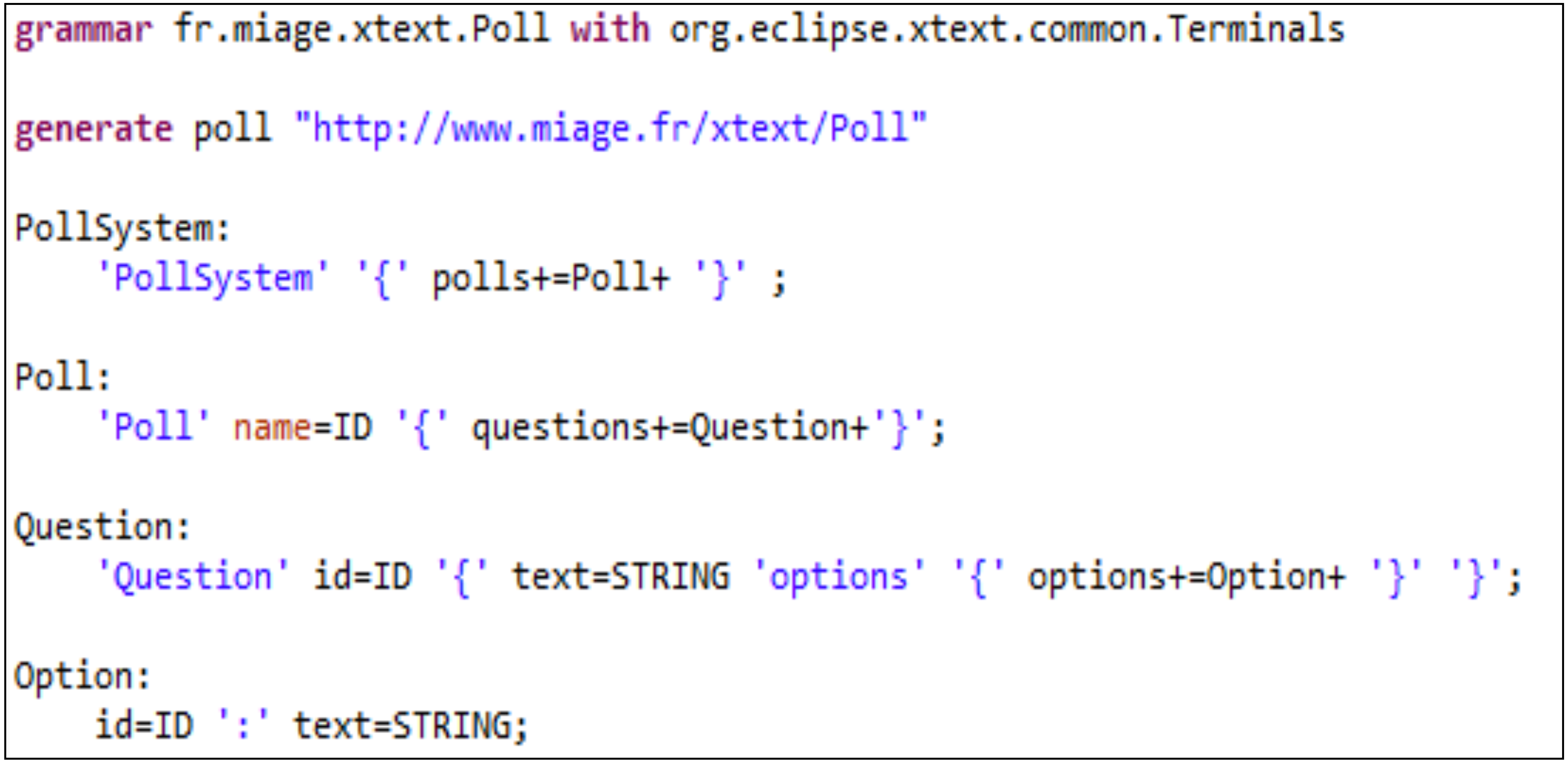


```
grammar fr.miage.xtext.Poll with org.eclipse.xtext.common.Terminals

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;
```

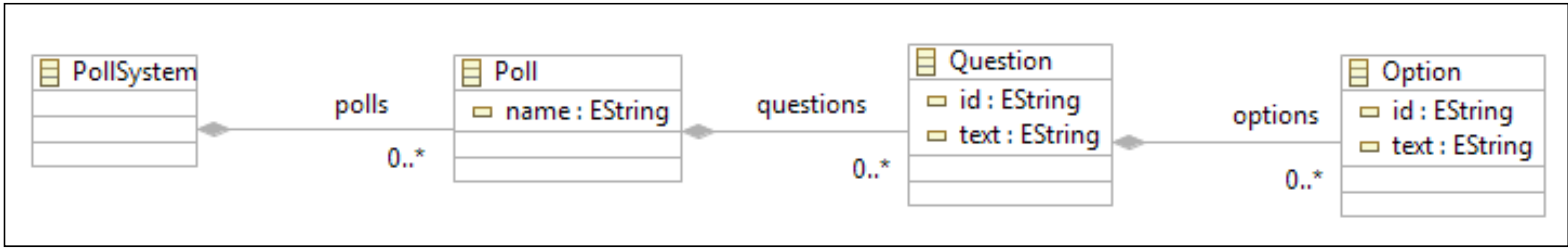


Xtext Grammar

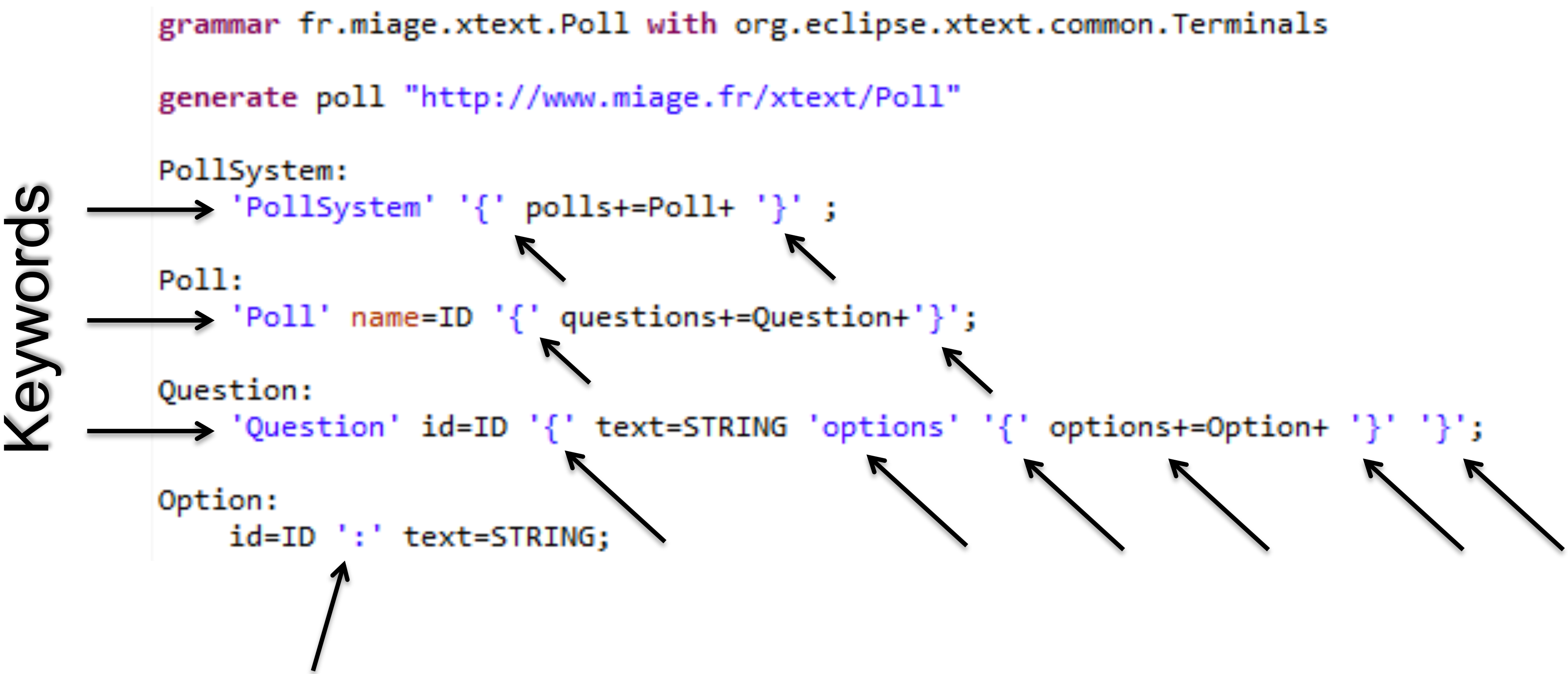
Parser Rules



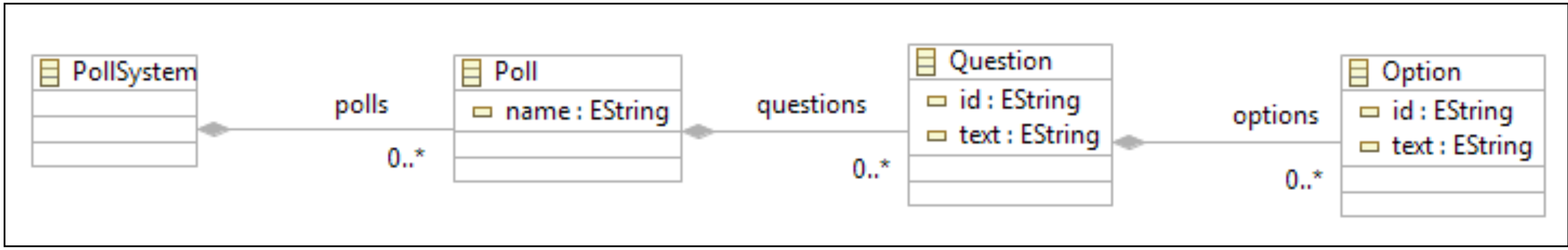
```
grammar fr.miage.xtext.Poll with org.eclipse.xtext.common.Terminals

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;
```



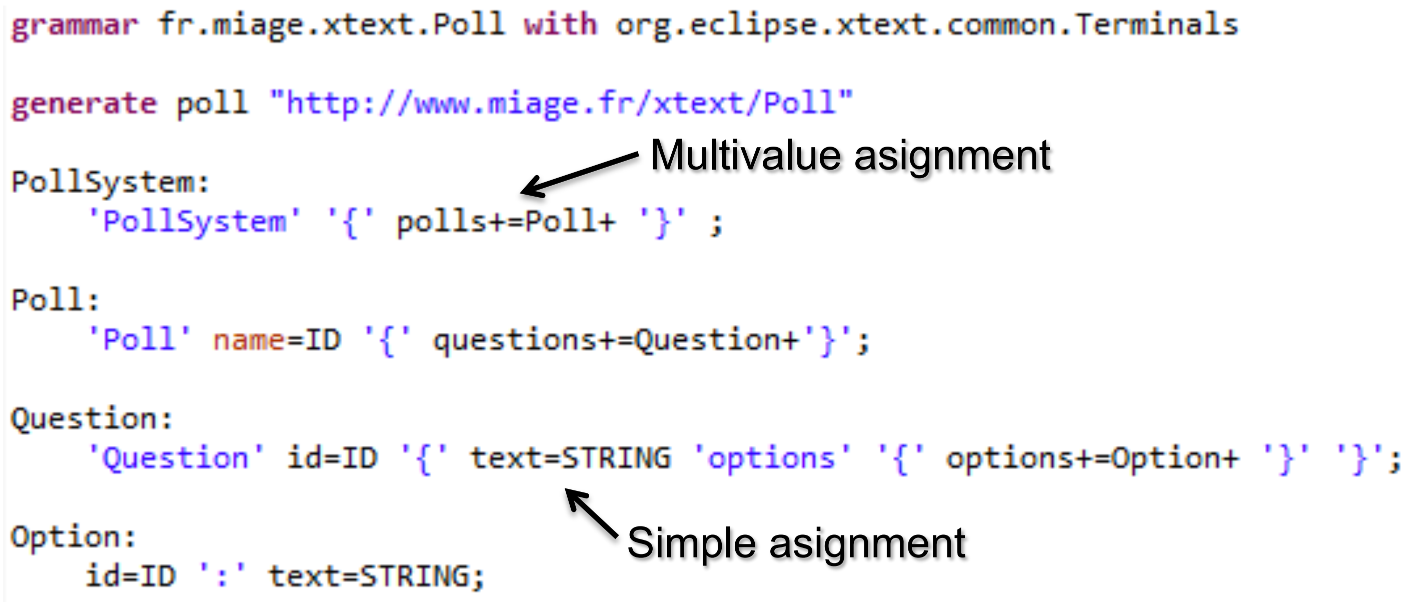
Xtext Grammar



```
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;
```



Xtext Grammar

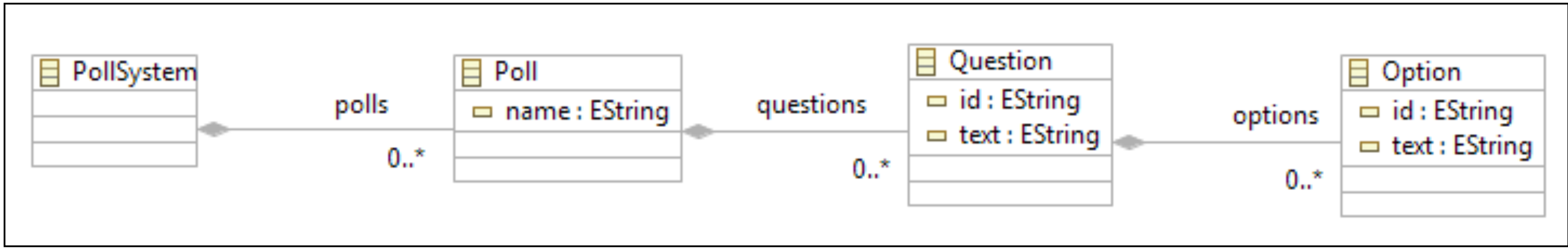


```
grammar fr.miage.xtext.Poll with org.eclipse.xtext.common.Terminals

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;
```

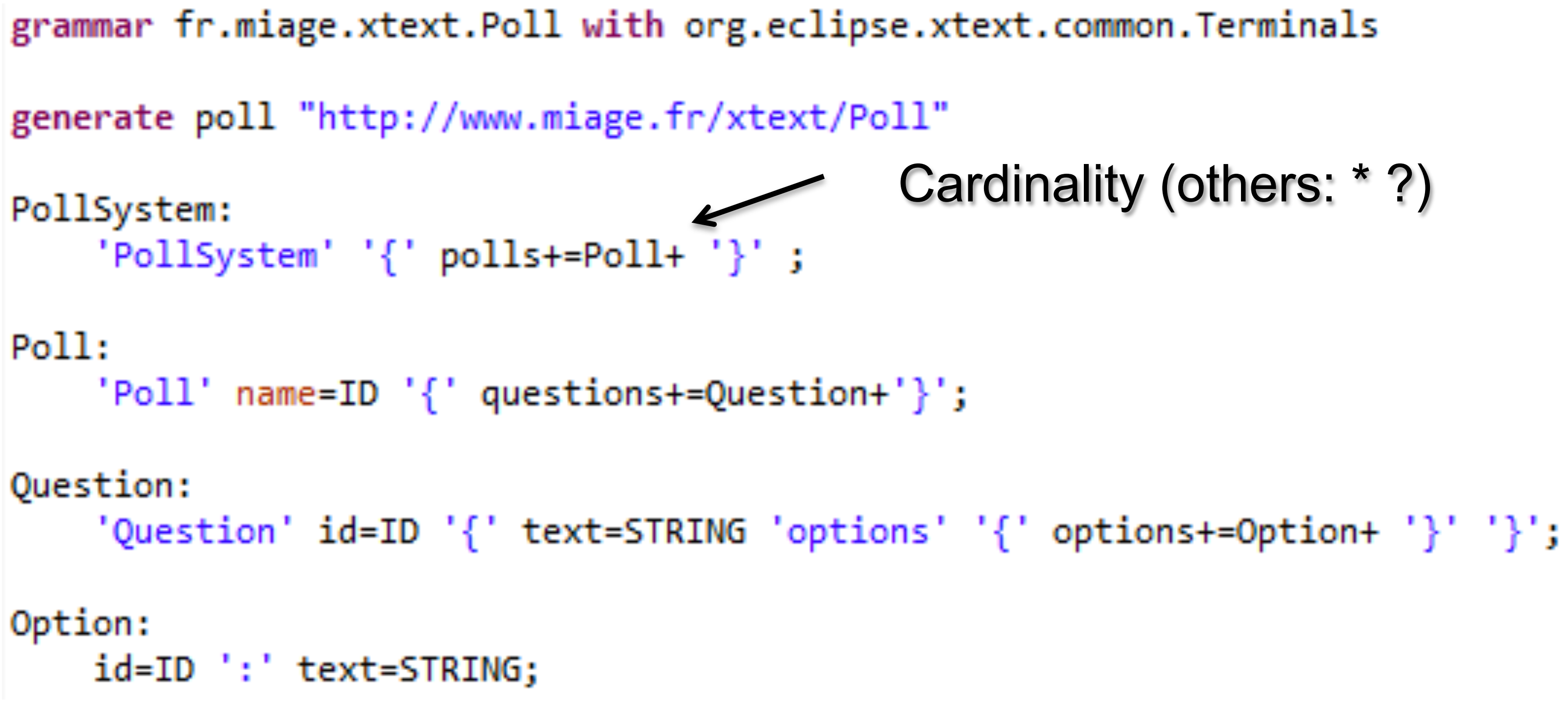
Multivalue asignment

Simple asignment

(not here → ?= Boolean asignment)

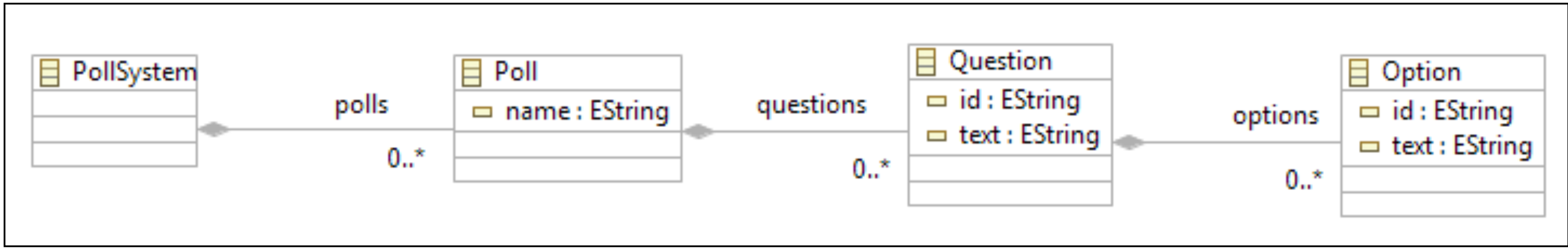


Xtext Grammar

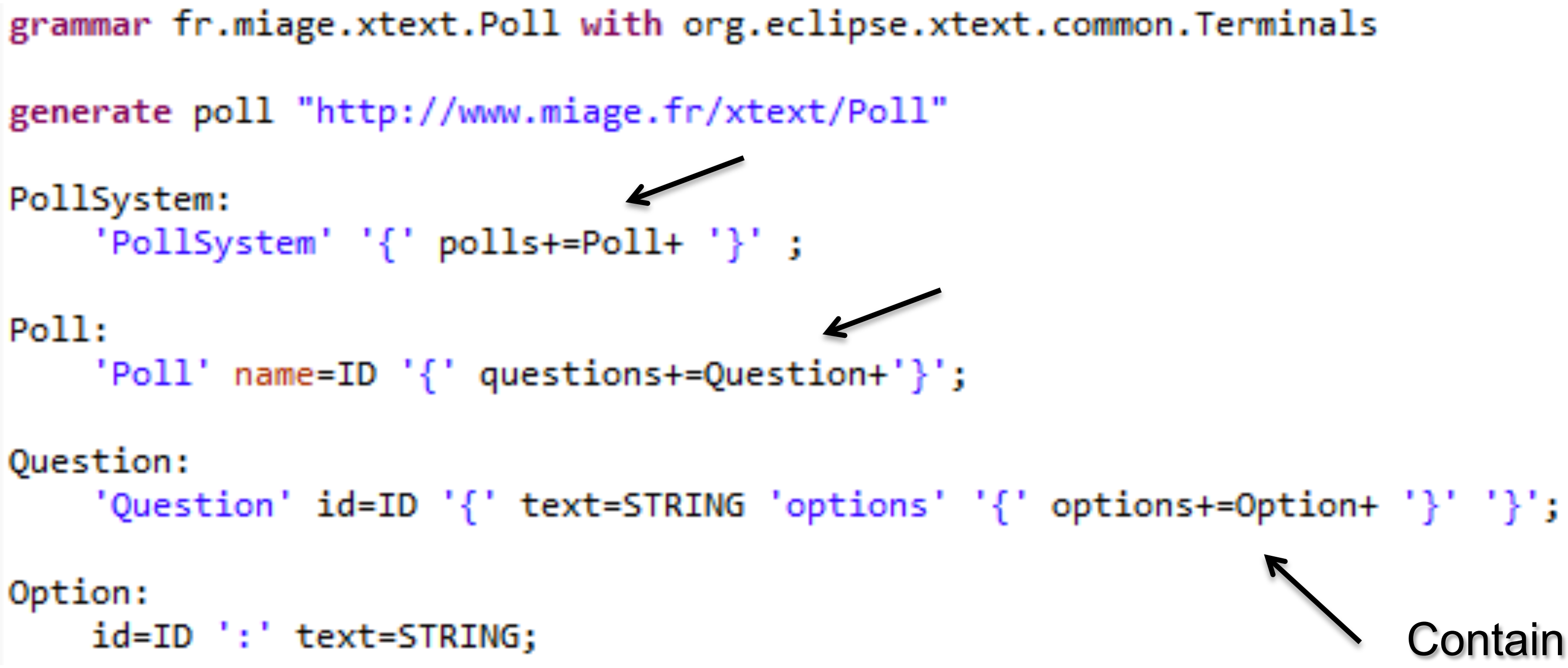


```
grammar fr.miage.xtext.Poll with org.eclipse.xtext.common.Terminals

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;
```

Cardinality (others: * ?)

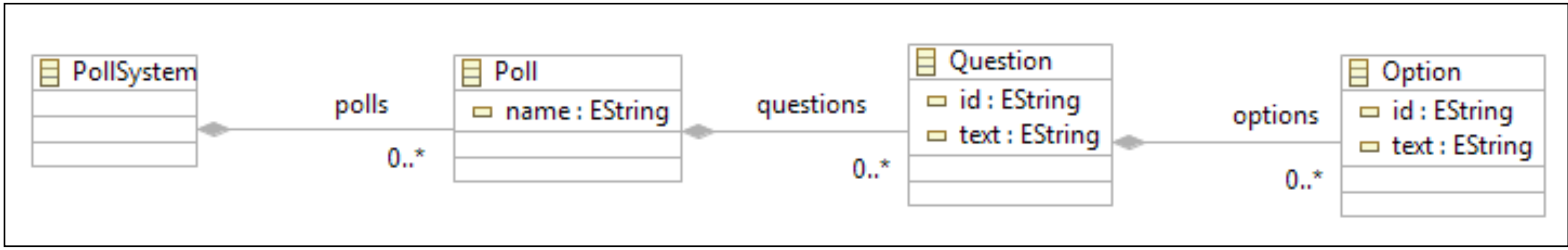


Xtext Grammar

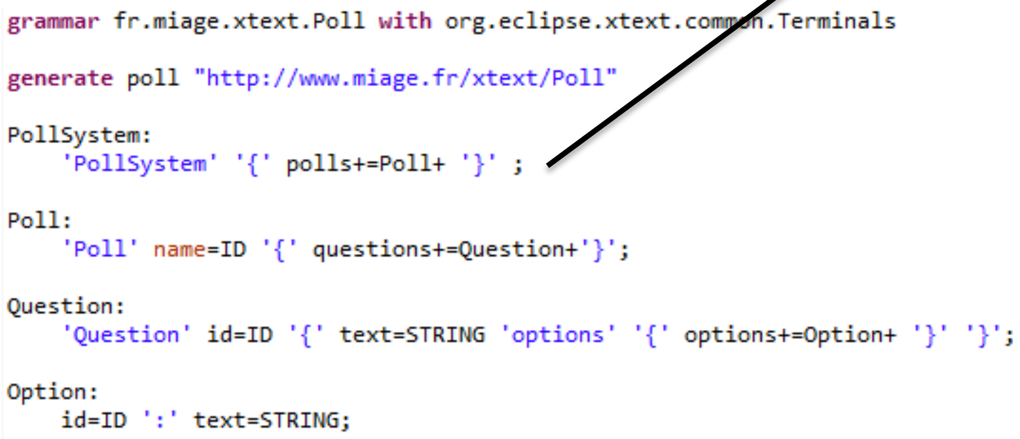


```
grammar fr.miage.xtext.Poll with org.eclipse.xtext.common.Terminals

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;
```

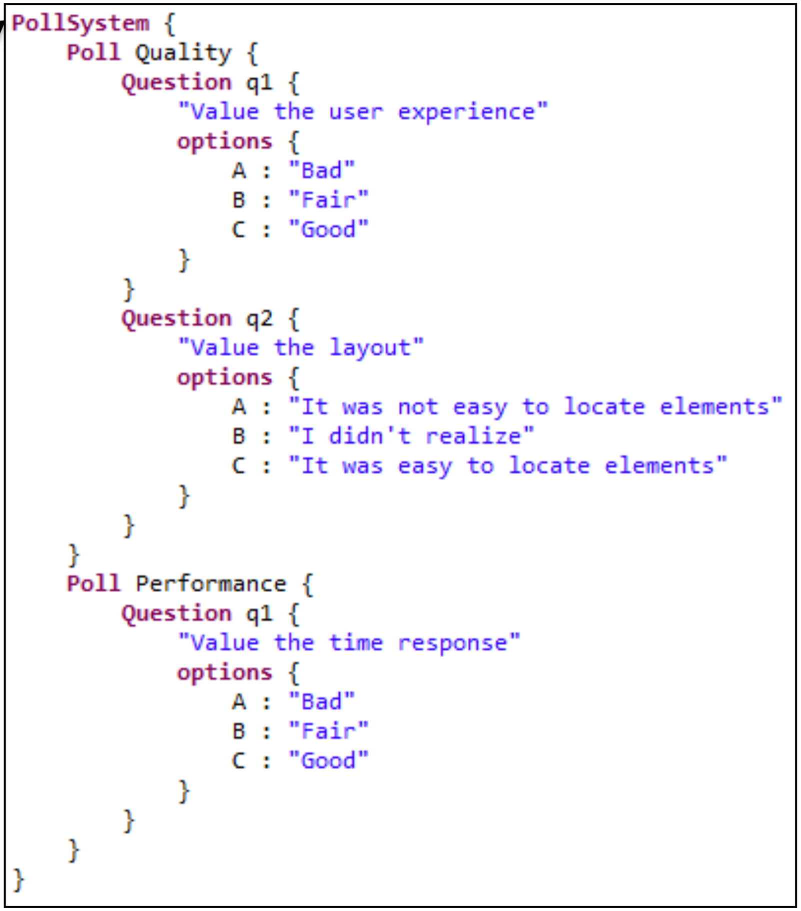
Containment



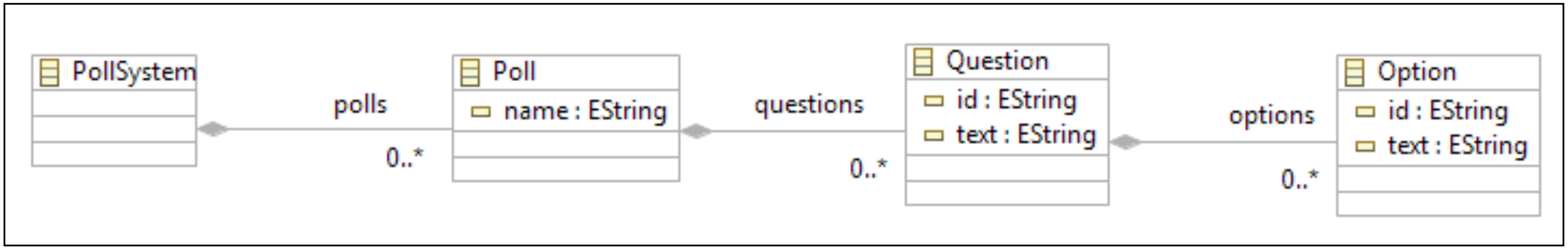
Grammar and Programs/Specifications/Models



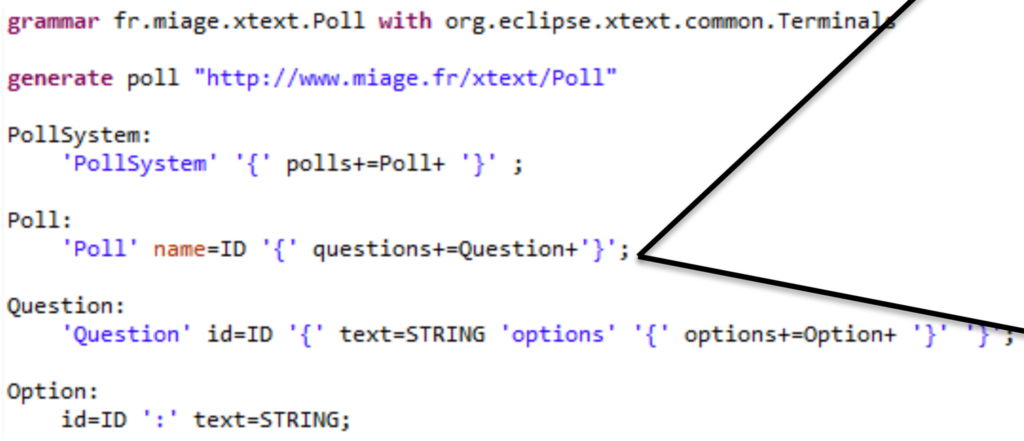
```
grammar fr.miage.xtext.Poll with org.eclipse.xtext.common.Terminals

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;
```



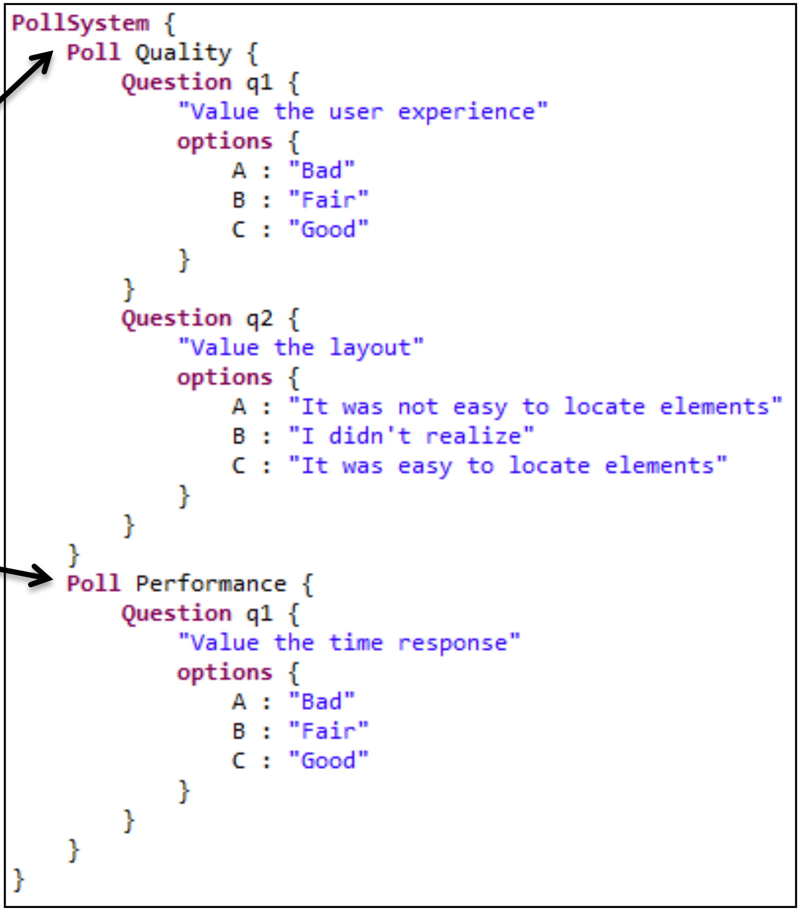
```
PollSystem {
    Poll Quality {
        Question q1 {
            "Value the user experience"
            options {
                A : "Bad"
                B : "Fair"
                C : "Good"
            }
        }
        Question q2 {
            "Value the layout"
            options {
                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"
            }
        }
    }
}
```



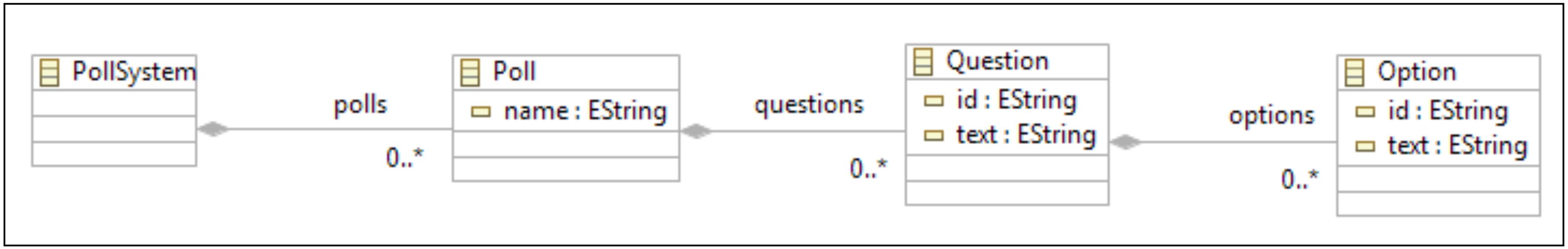
Grammar and Programs/Specifications/Models



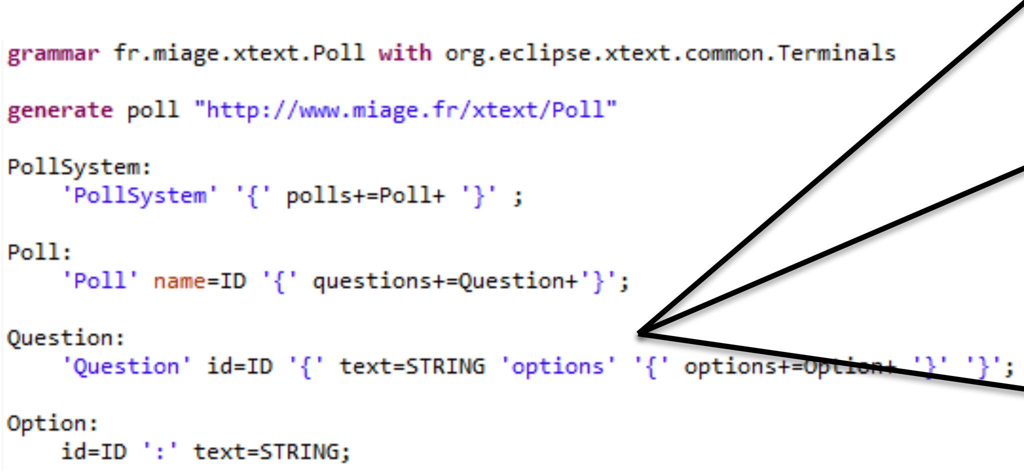
```
grammar fr.miage.xtext.Poll with org.eclipse.xtext.common.Terminals

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;
```



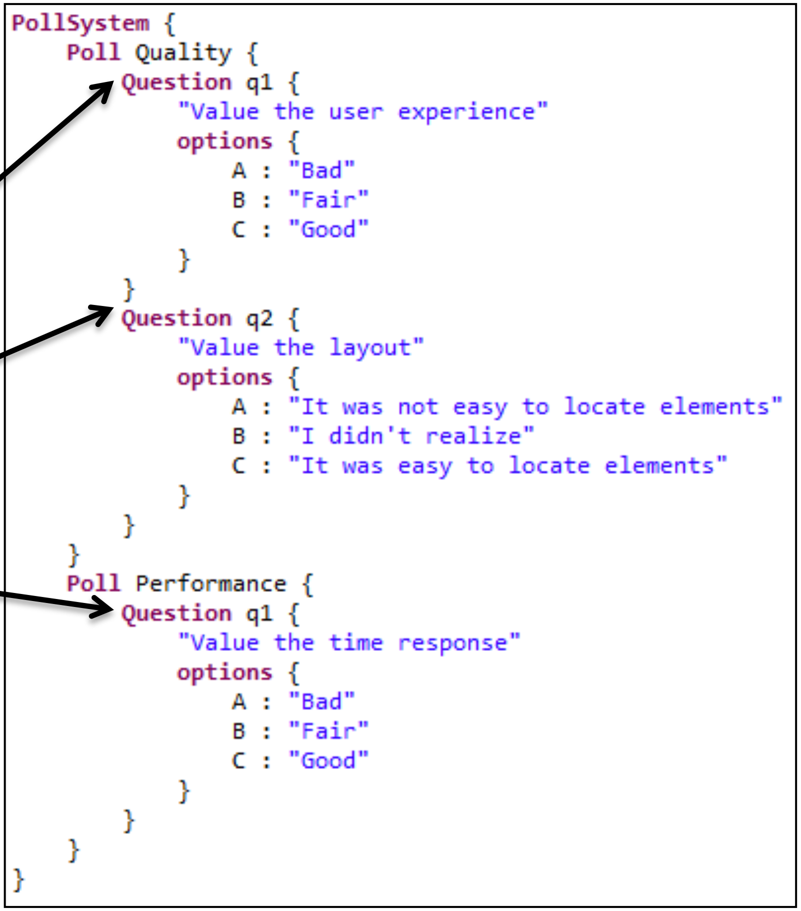
```
PollSystem {
    Poll Quality {
        Question q1 {
            "Value the user experience"
            options {
                A : "Bad"
                B : "Fair"
                C : "Good"
            }
        }
        Question q2 {
            "Value the layout"
            options {
                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"
            }
        }
    }
}
```



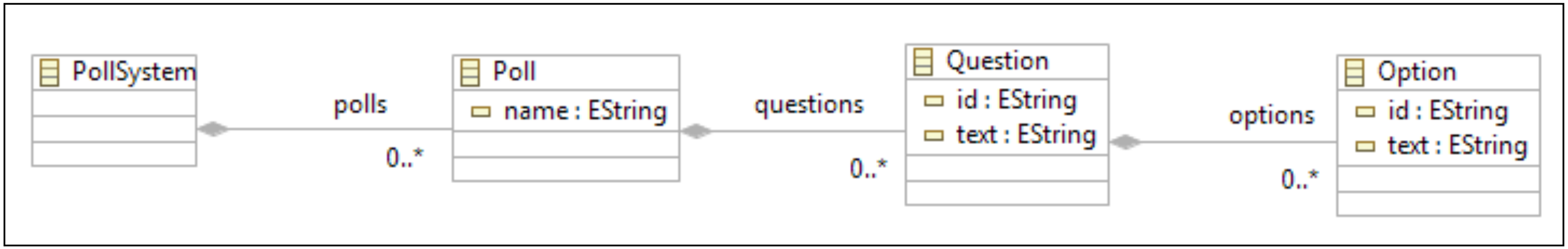
Grammar and Programs/Specifications/Models



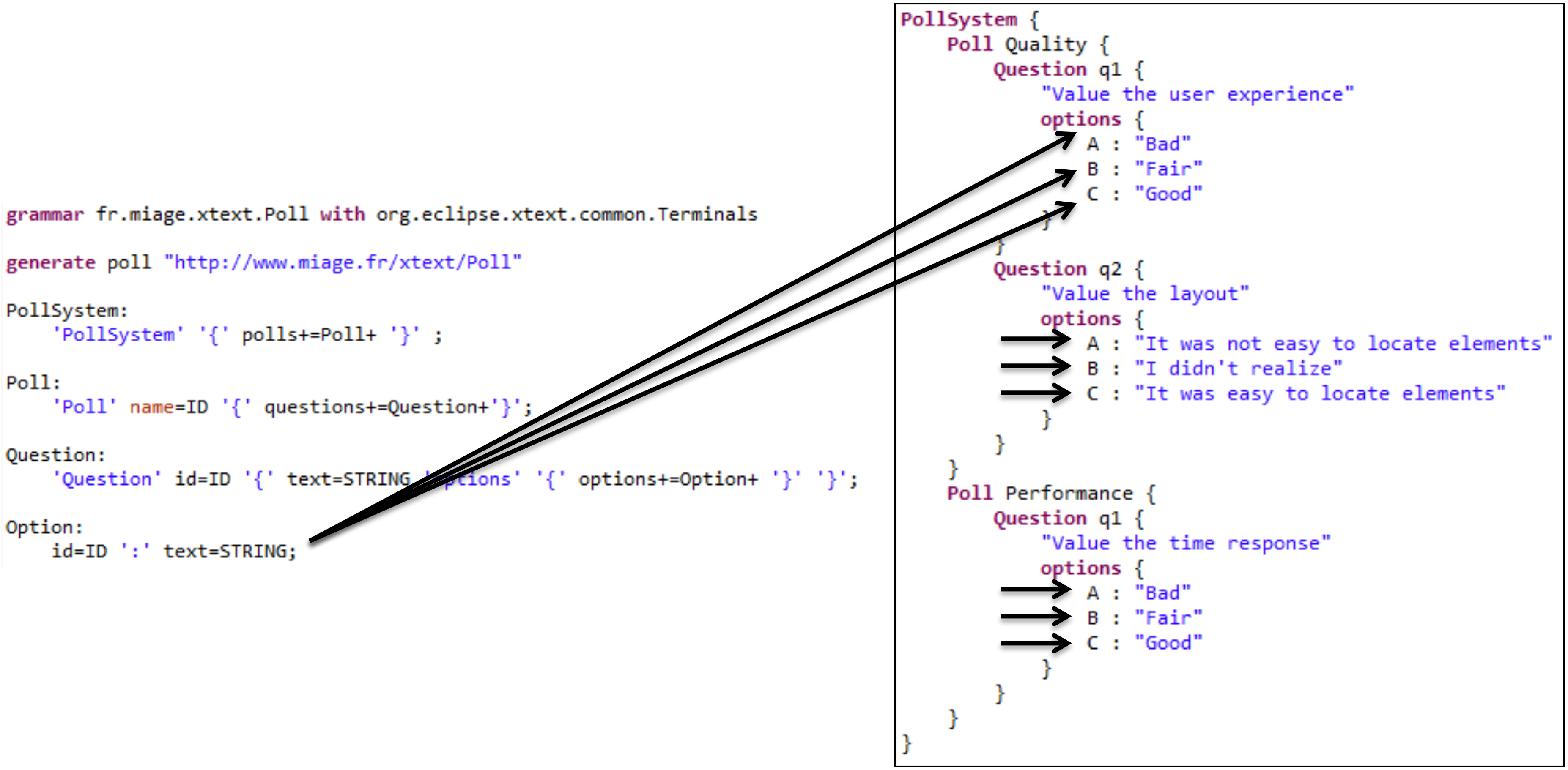
```
grammar fr.miage.xtext.Poll with org.eclipse.xtext.common.Terminals

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"
            }
        }
    }
}
```

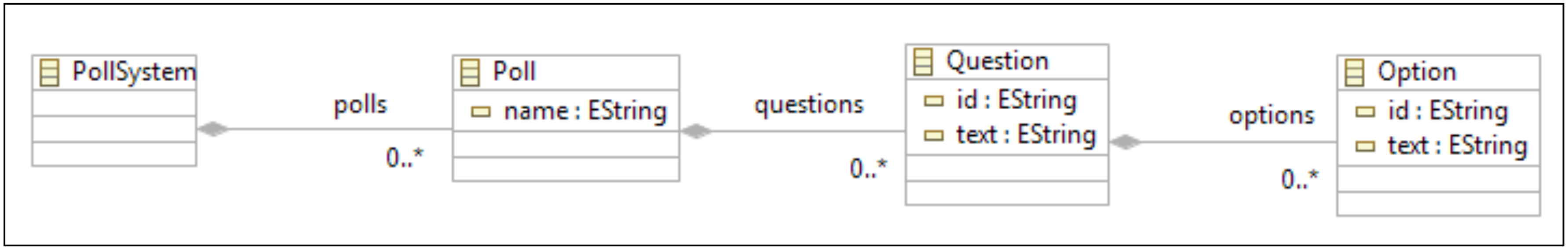


Grammar and Programs/Specifications/Models



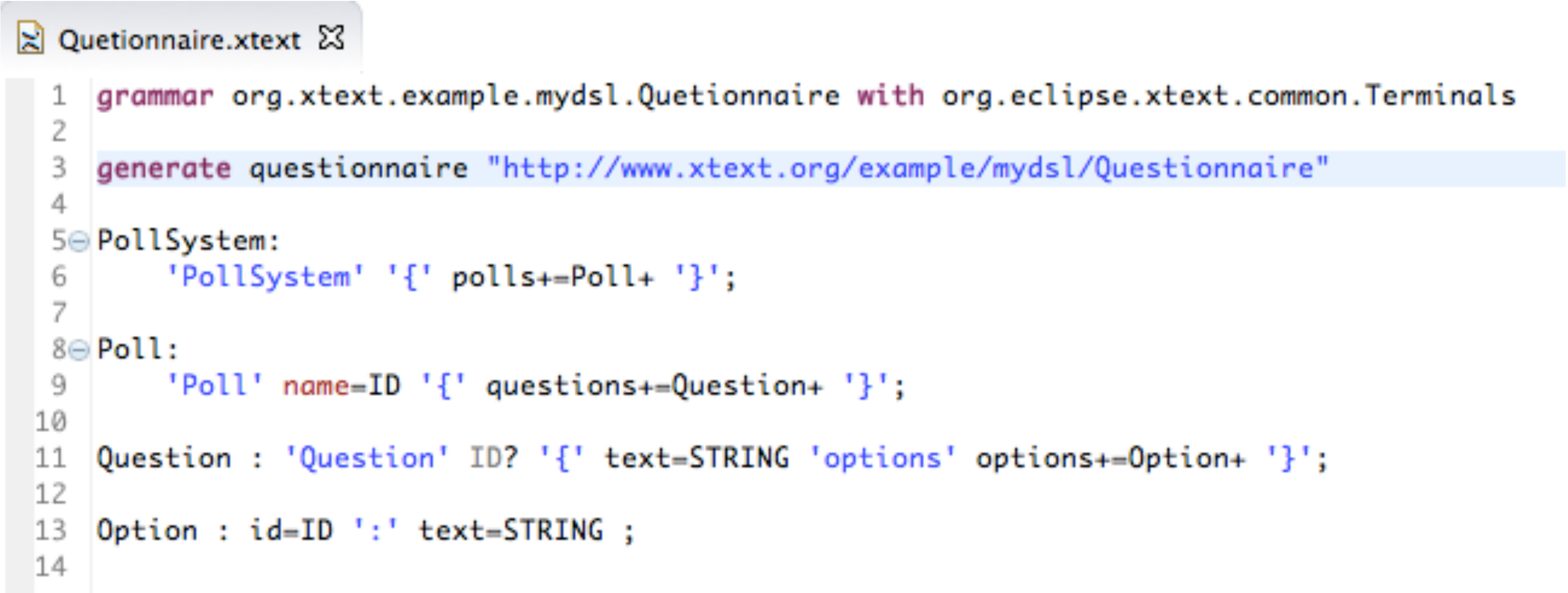
```
grammar fr.miage.xtext.Poll with org.eclipse.xtext.common.Terminals

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

#4 e9a8d603

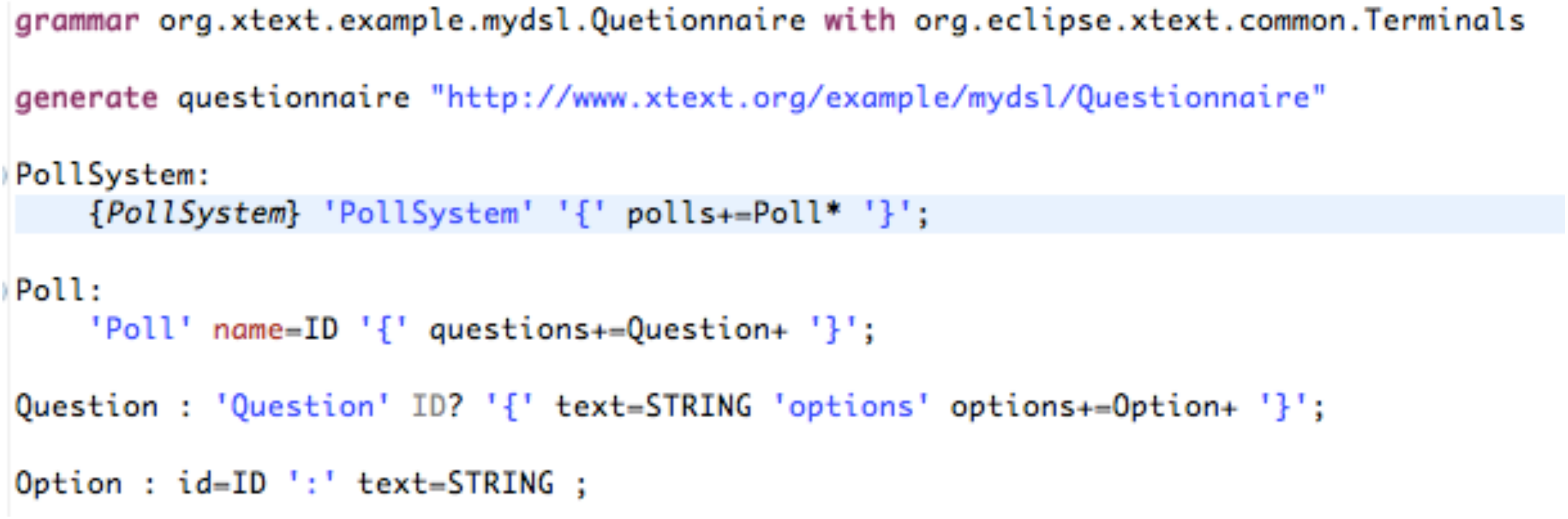


```
grammar org.xtext.example.mydsl.Quetionnaire with org.eclipse.xtext.common.Terminals

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 ;

```

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

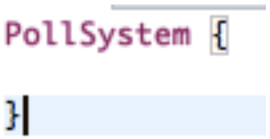
Quizz Time

#5 e9a8d603



```
grammar org.xtext.example.mydsl.Quetionnaire 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 ;
```

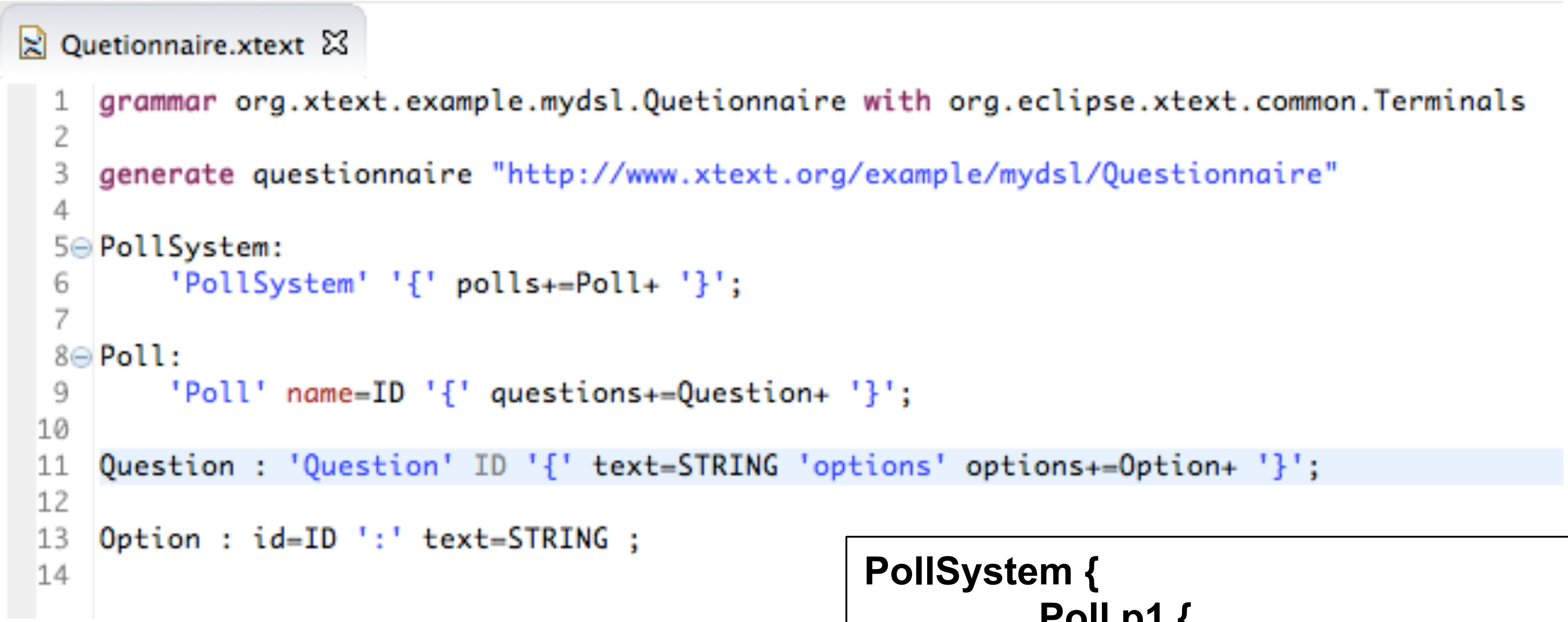
Est-ce que le fichier.q suivant est correct vis-à-vis de la grammaire Xtext? Pourquoi?



```
PollSystem {

}
```

Quizz Time

#6 e9a8d603



Quetionnaire.xtext

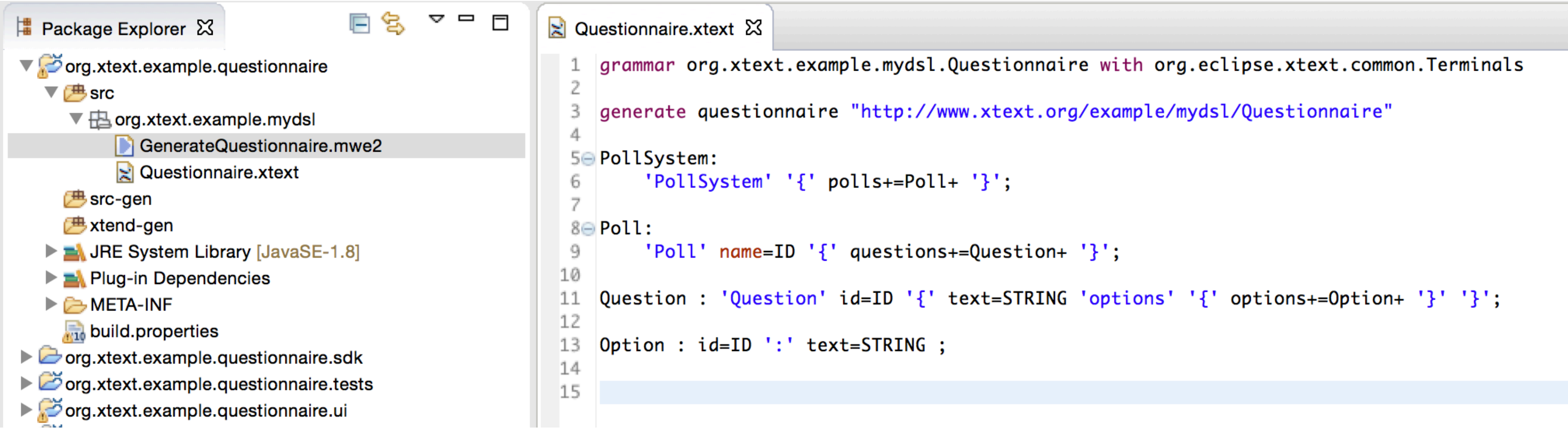
```
grammar org.xtext.example.mydsl.Quetionnaire with org.eclipse.xtext.common.Terminals

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 ;

```

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

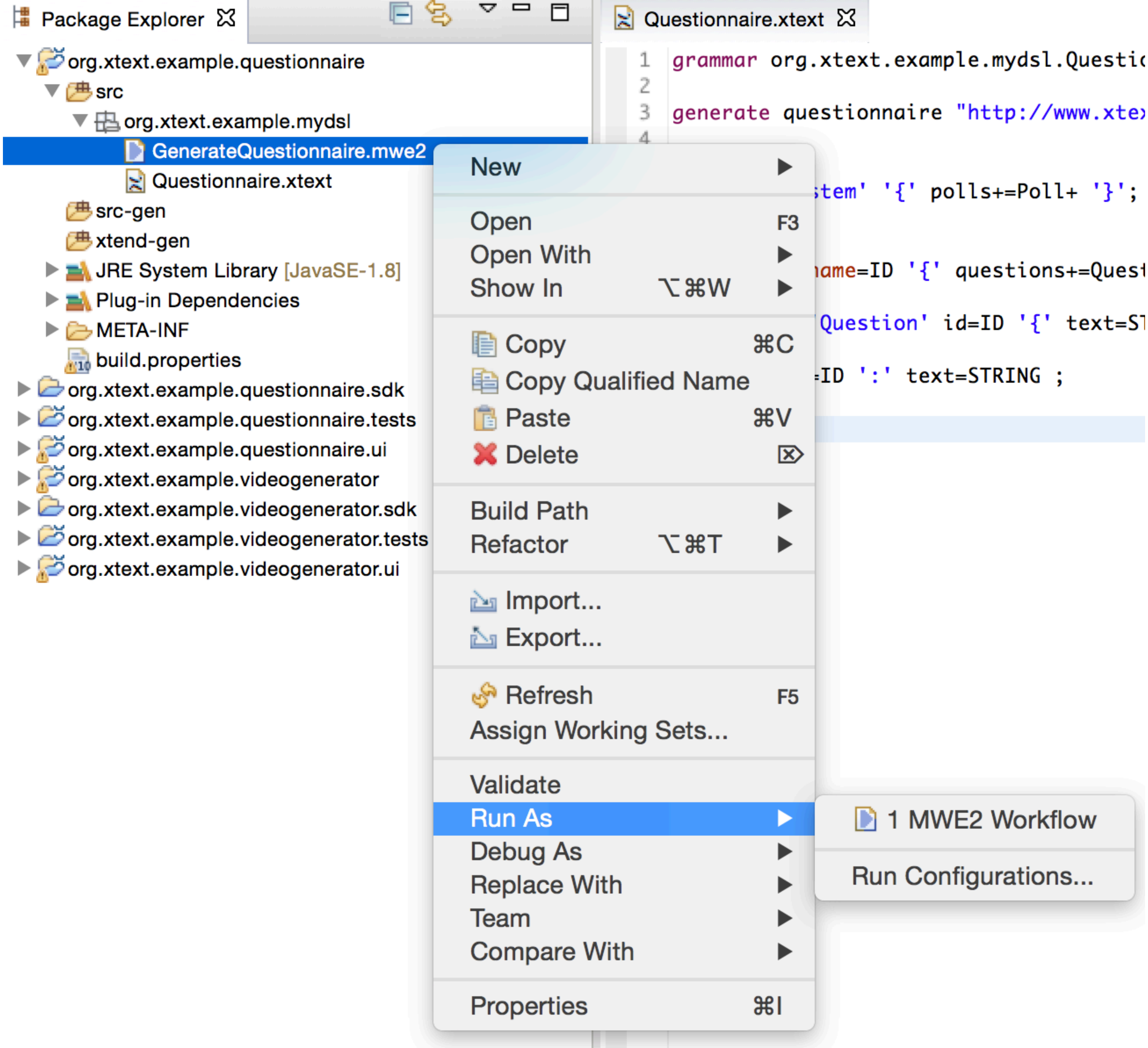
```
PollSystem {
        Poll p1 {
                Question {
                        "Q1"
                        options o1 : "R1"
                }
        }
}
```

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

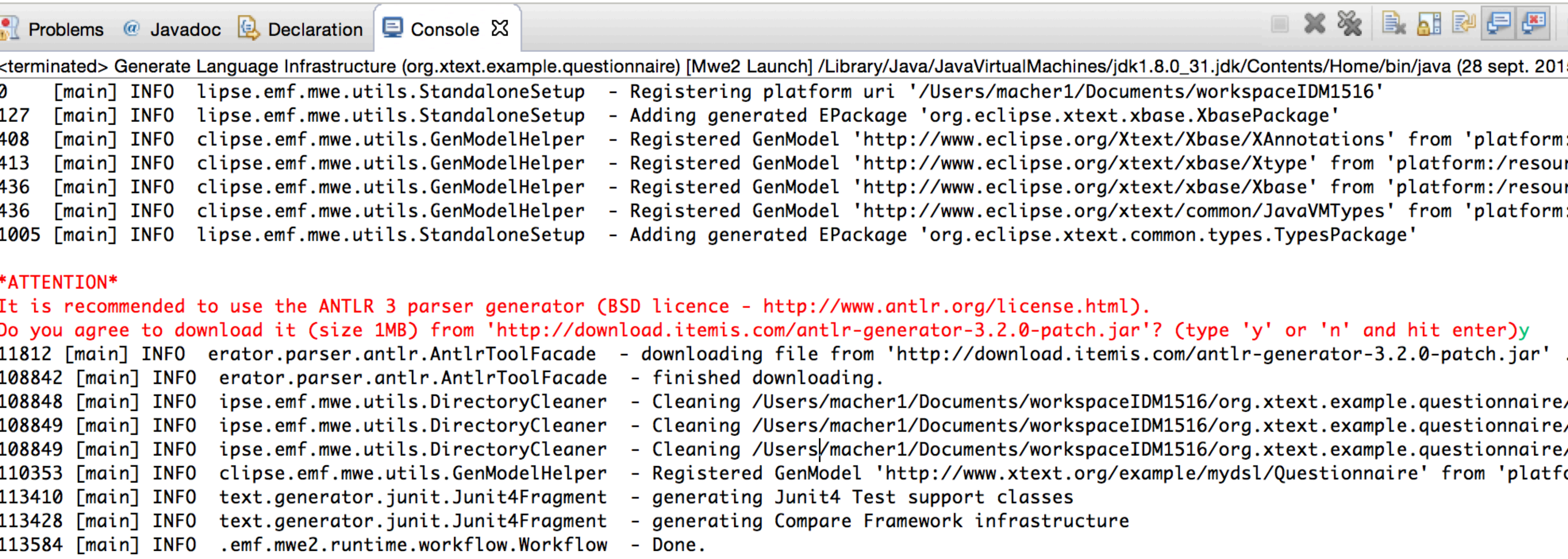
Live Demonstration


Package Explorer
org.xtext.example.questionnaire
src
org.xtext.example.mydsl
GenerateQuestionnaire.mwe2
Questionnaire.xtext
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
org.xtext.example.questionnaire.ui
Questionnaire.xtext

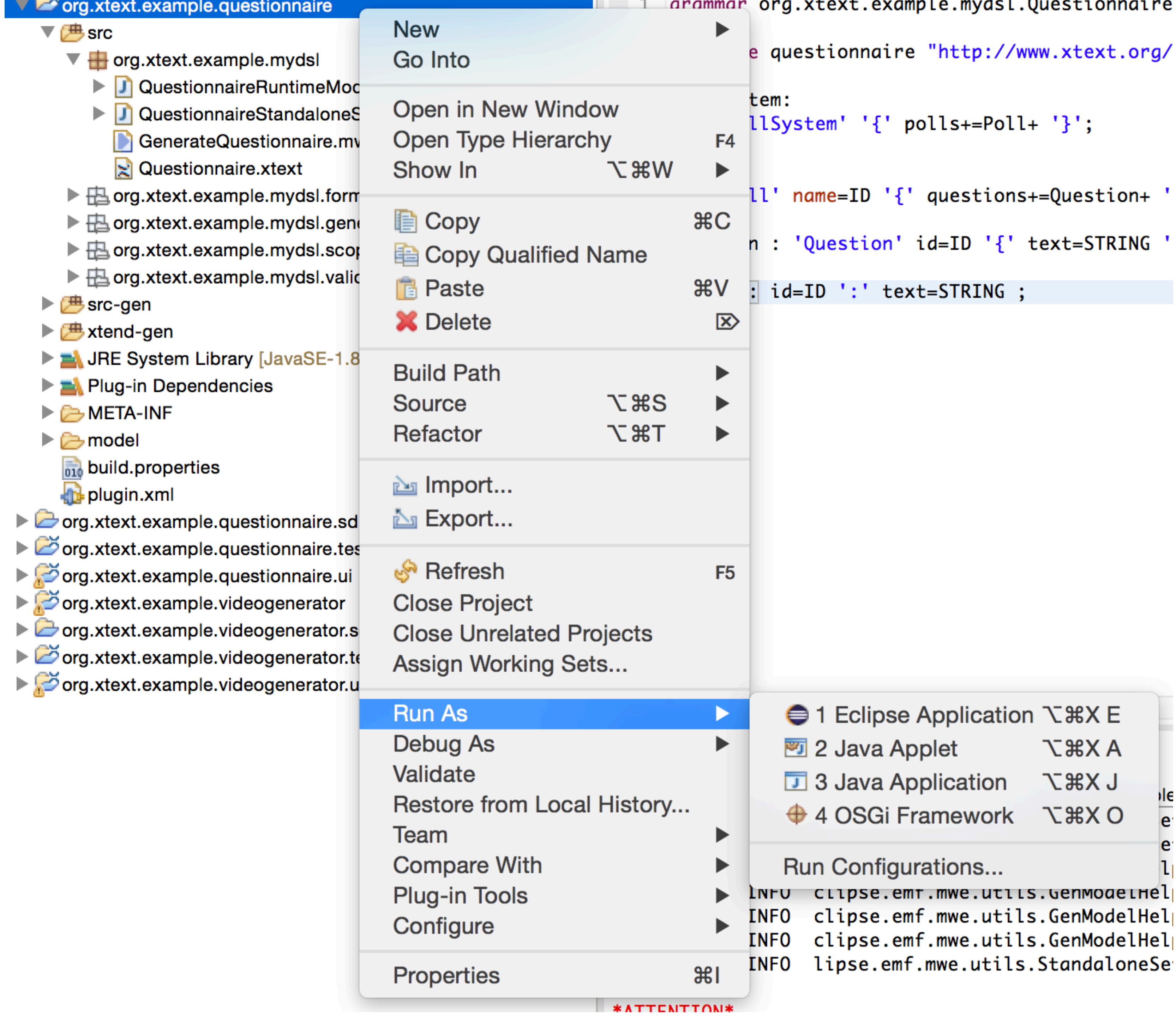
```
grammar org.xtext.example.mydsl.Questionnaire with org.eclipse.xtext.common.Terminals

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

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

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

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

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

```



```
grammar org.xtext.example.mydsl.Questic

generate questionnaire "http://www.xte

stem' '{' polls+=Poll+ '}';

ame=ID '{' questions+=Quest

'Question' id=ID '{' text=S

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

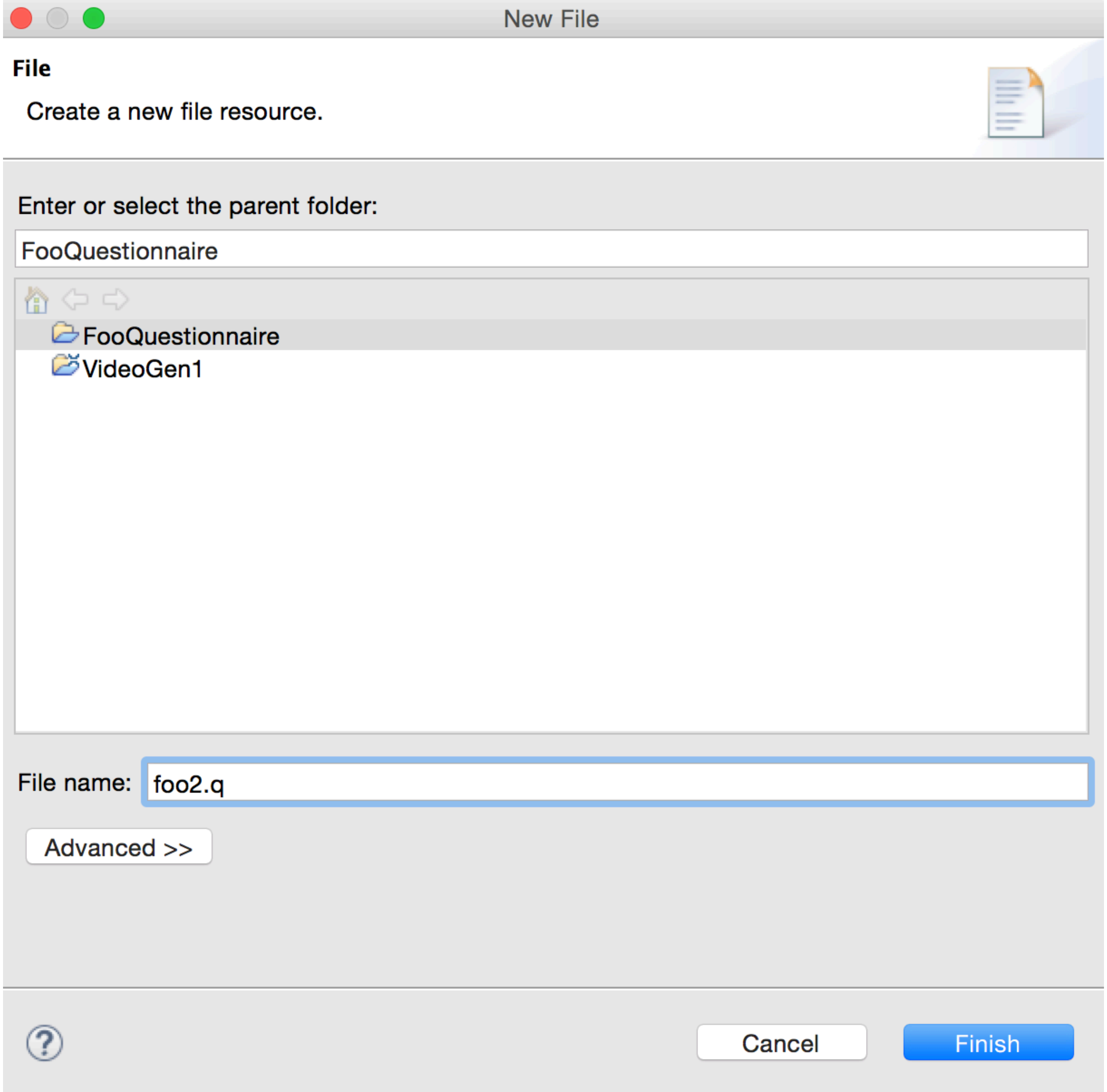


```
<terminated> Generate Language Infrastructure (org.xtext.example.questionnaire) [Mwe2 Launch] /Library/Java/JavaVirtualMachines/jdk1.8.0_31.jdk/Contents/Home/bin/java (28 sept. 201
0    [main] INFO  lipse.emf.mwe.utils.StandaloneSetup  - Registering platform uri '/Users/macher1/Documents/workspaceIDM1516'
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:
413  [main] INFO  clipse.emf.mwe.utils.GenModelHelper  - Registered GenModel 'http://www.eclipse.org/xtext/xbase/Xtype' from 'platform:/resour
436  [main] INFO  clipse.emf.mwe.utils.GenModelHelper  - Registered GenModel 'http://www.eclipse.org/xtext/xbase/Xbase' from 'platform:/resour
436  [main] INFO  clipse.emf.mwe.utils.GenModelHelper  - Registered GenModel 'http://www.eclipse.org/xtext/common/JavaVMTypes' from 'platform:
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 'platfo
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.
```



```
grammar org.xtext.example.mydsl.Questionnaire
e questionnaire "http://www.xtext.org/
tem:
llSystem' '{' polls+=Poll+ '}';

ll' name=ID '{' questions+=Question+ '
n : 'Question' id=ID '{' text=STRING '
: id=ID ':' text=STRING ;
```

```
INFO  clipse.emf.mwe.utils.GenModelHel
INFO  clipse.emf.mwe.utils.GenModelHel
INFO  clipse.emf.mwe.utils.GenModelHel
INFO  lipse.emf.mwe.utils.StandaloneSe
*ATTENTION*
```



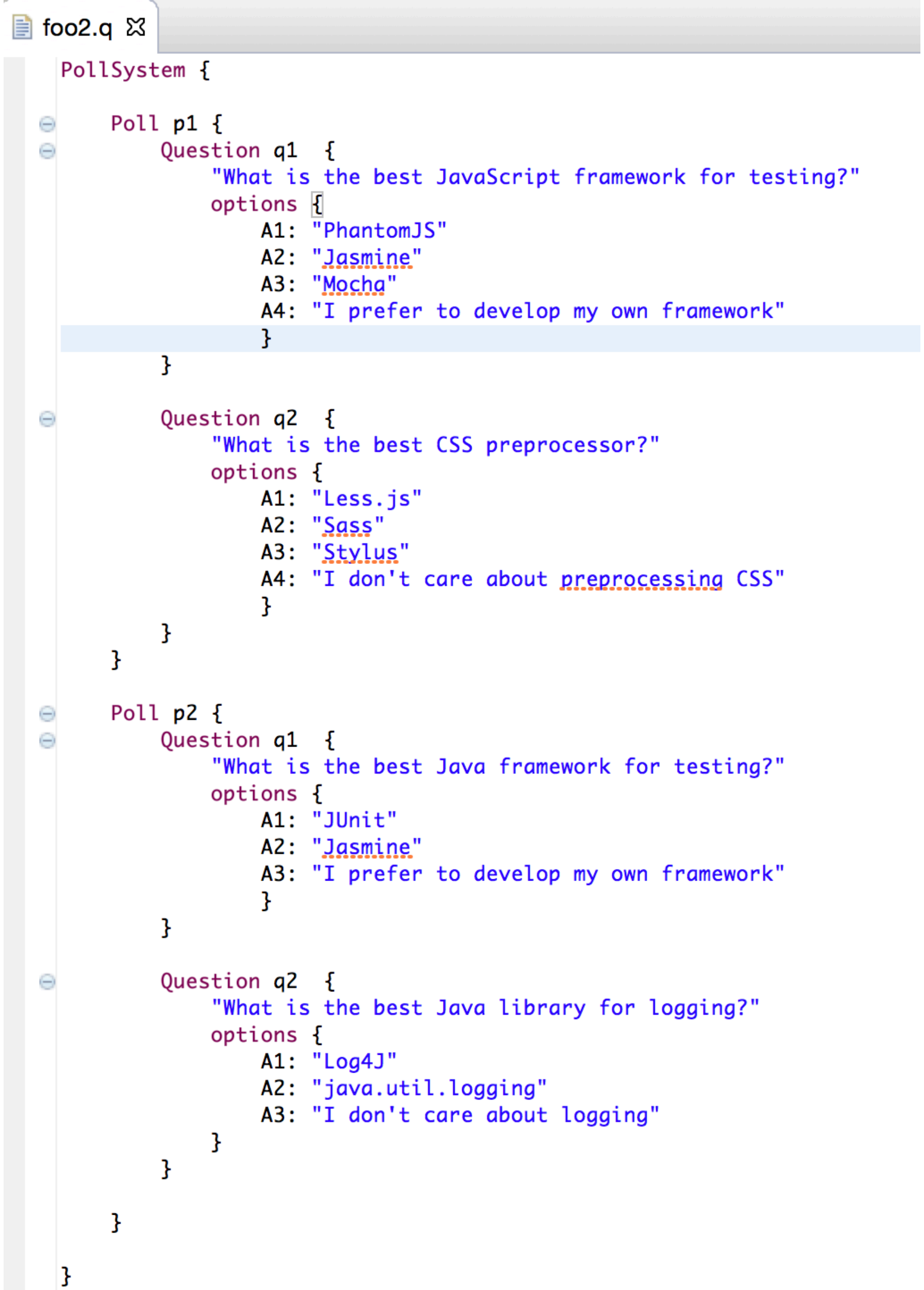
New File

File

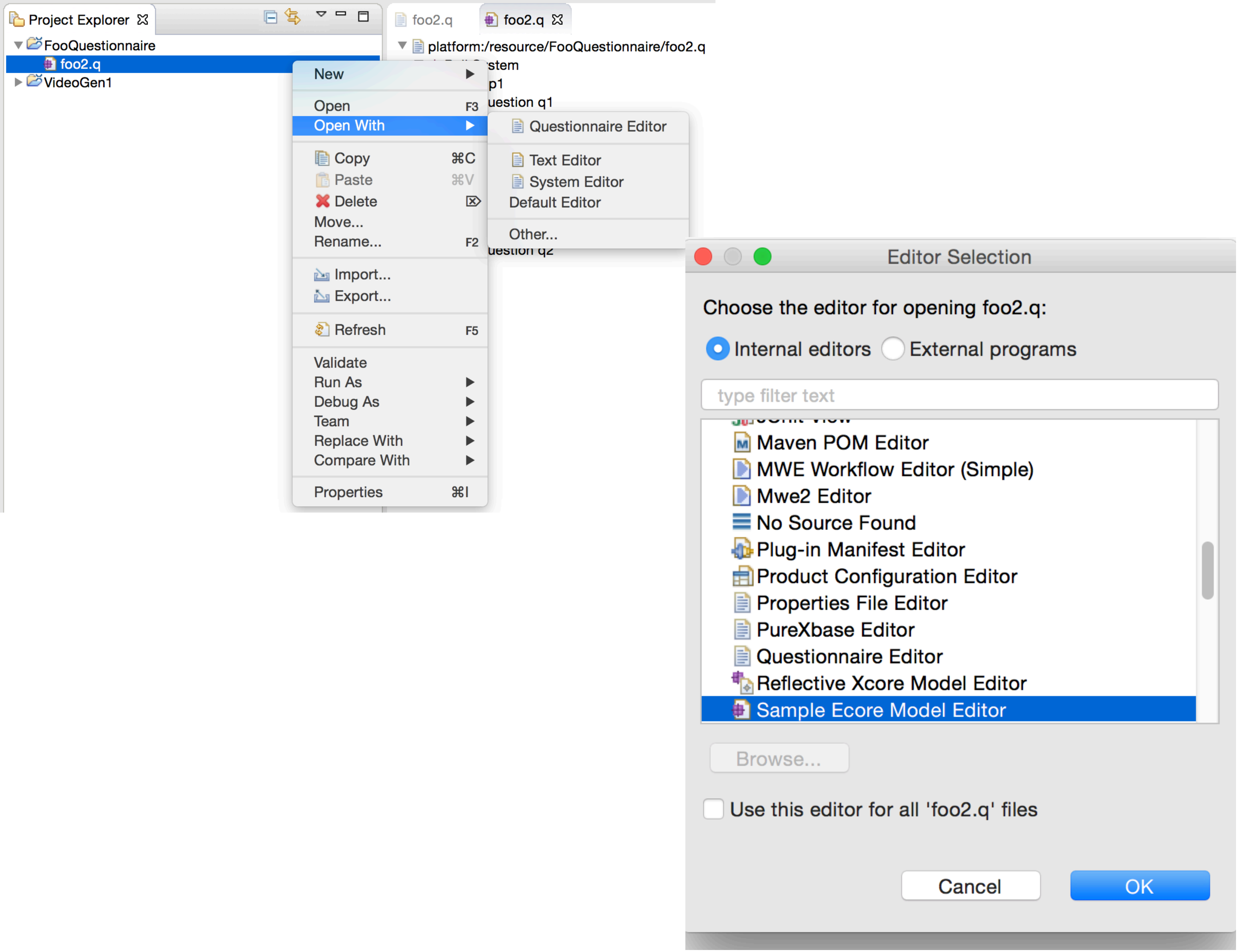
Create a new file resource.

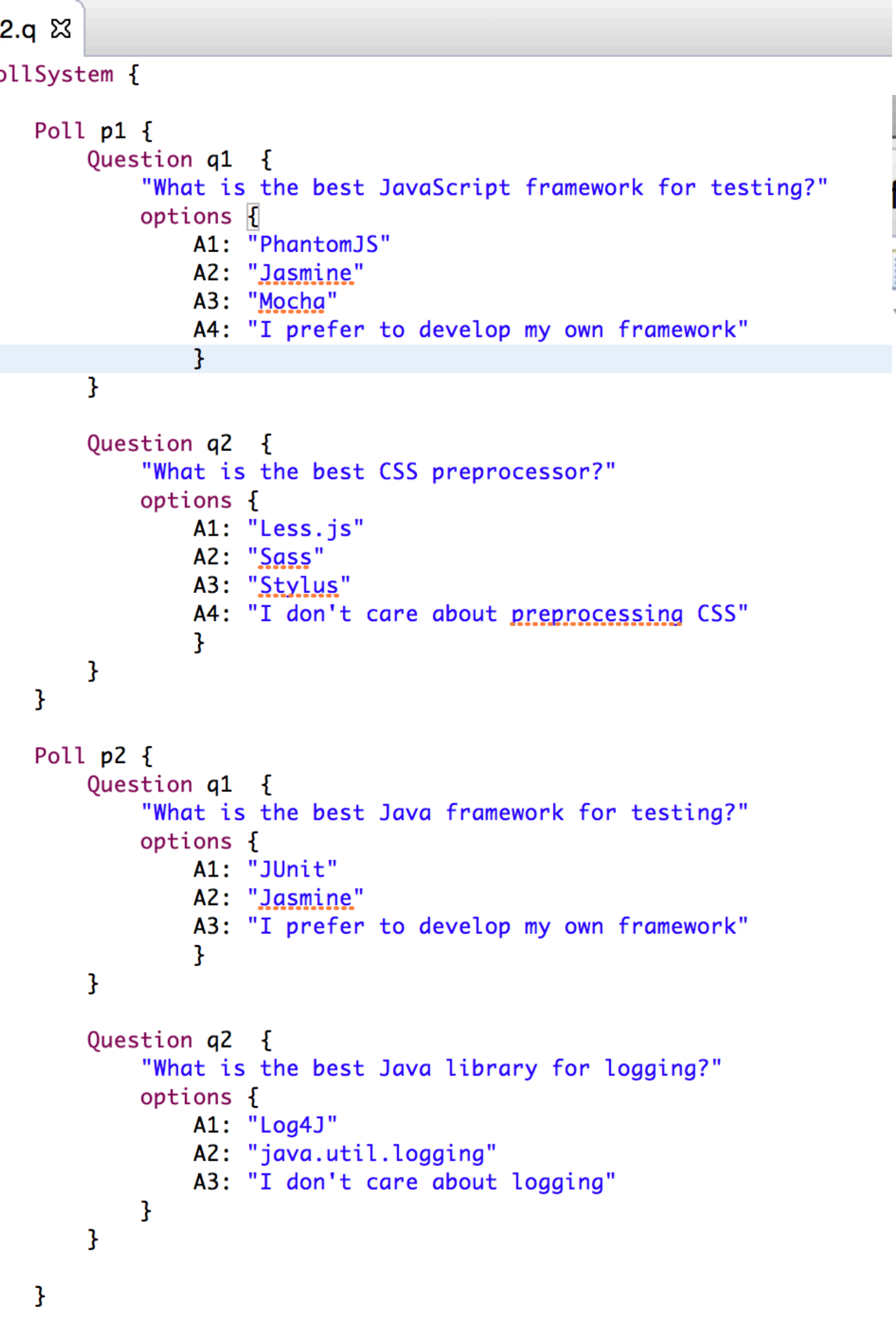
Enter or select the parent folder: FooQuestionnaire

File name: foo2.q

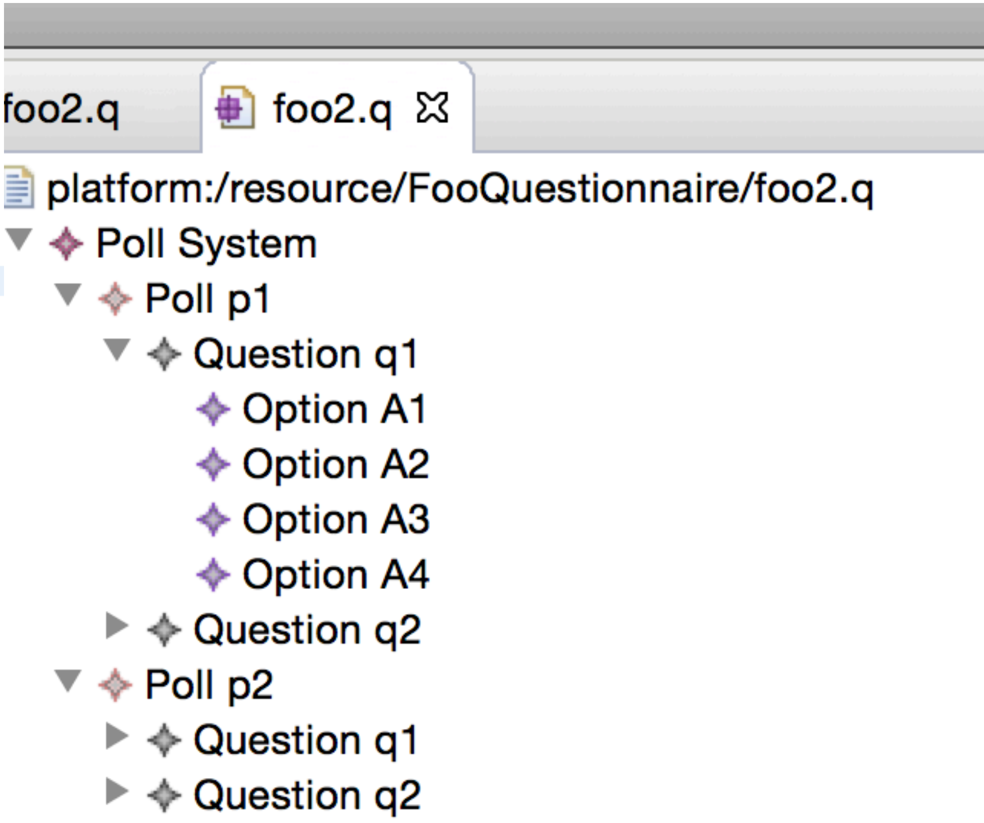


```
PollSystem {

    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"
            }
        }

    }

}
```

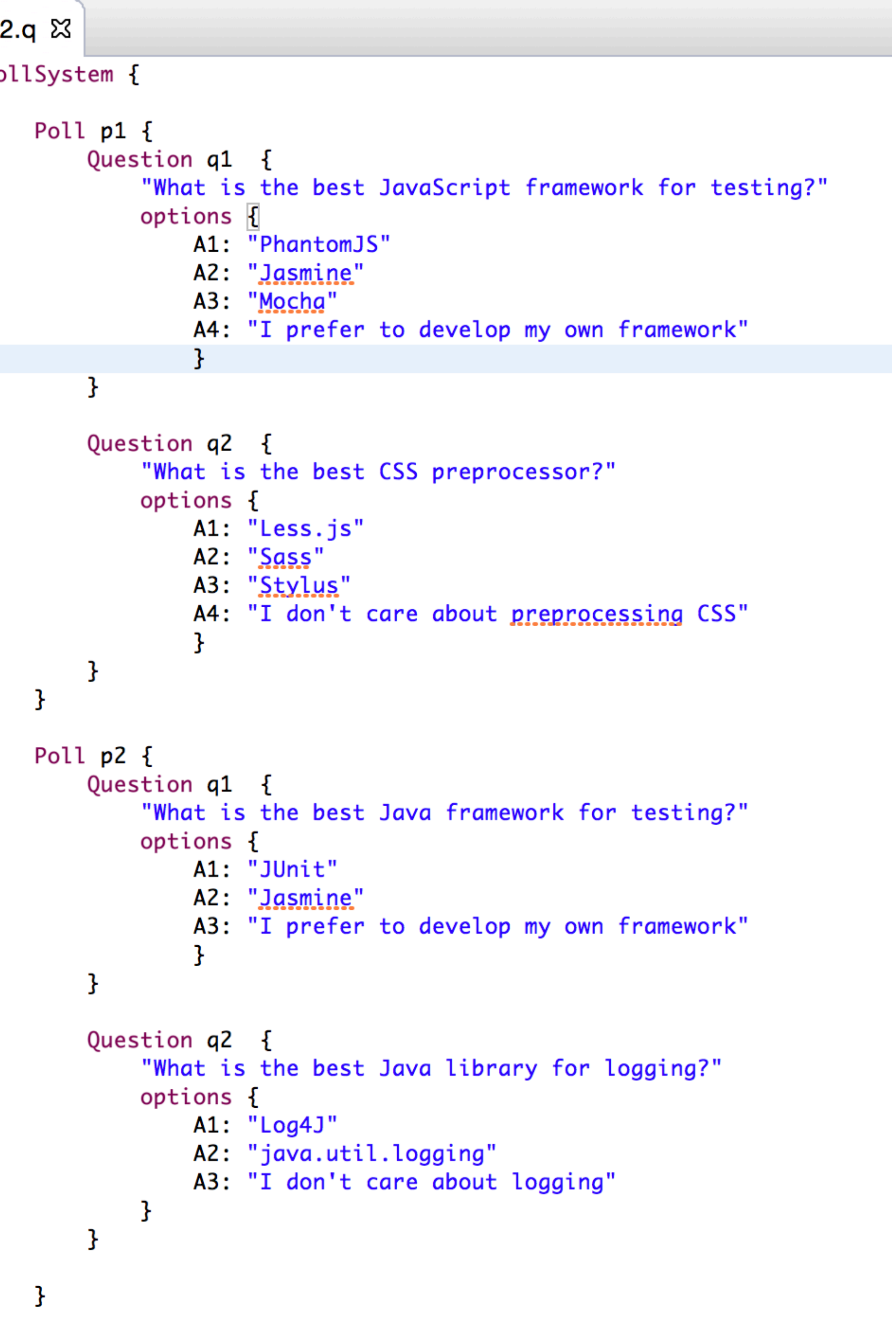

Project Explorer
FooQuestionnaire
foo2.q
VideoGen1
foo2.q
foo2.q
platform:/resource/FooQuestionnaire/foo2.q
stem
p1
uestion q1
uestion q2
New
Open F3
Open With
Copy ⌘C
Paste ⌘V
Delete
Move...
Rename... F2
Import...
Export...
Refresh F5
Validate
Run As
Debug As
Team
Replace With
Compare With
Properties ⌘I
Questionnaire Editor
Text Editor
System Editor
Default Editor
Other...
Editor Selection
Choose the editor for opening foo2.q:
● Internal editors
○ External programs
type filter text
Maven POM Editor
MWE Workflow Editor (Simple)
Mwe2 Editor
No Source Found
Plug-in Manifest Editor
Product Configuration Editor
Properties File Editor
PureXbase Editor
Questionnaire Editor
Reflective Xcore Model Editor
Sample Ecore Model Editor
Browse...
□ Use this editor for all 'foo2.q' files
Cancel
OK



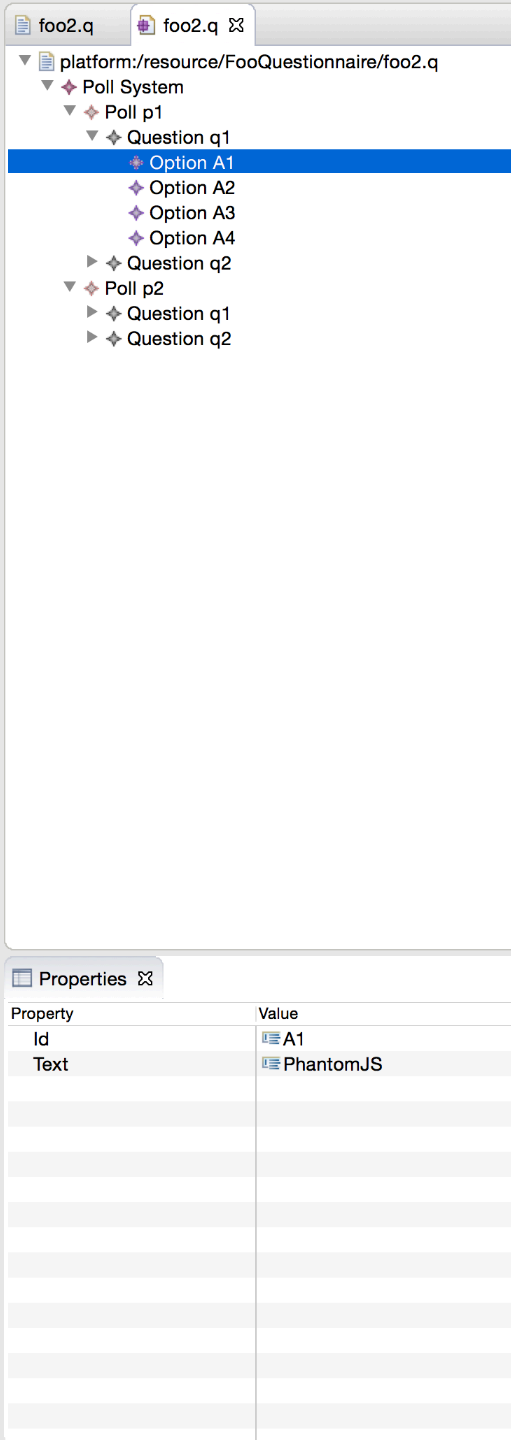
```
ollSystem {

  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"
          }
      }

  }
```



```
platform:/resource/FooQuestionnaire/foo2.q
▼ Poll System
  ▼ Poll p1
    ▼ Question q1
        Option A1
        Option A2
        Option A3
        Option A4
    ▶ Question q2
  ▼ Poll p2
    ▶ Question q1
    ▶ Question q2
```

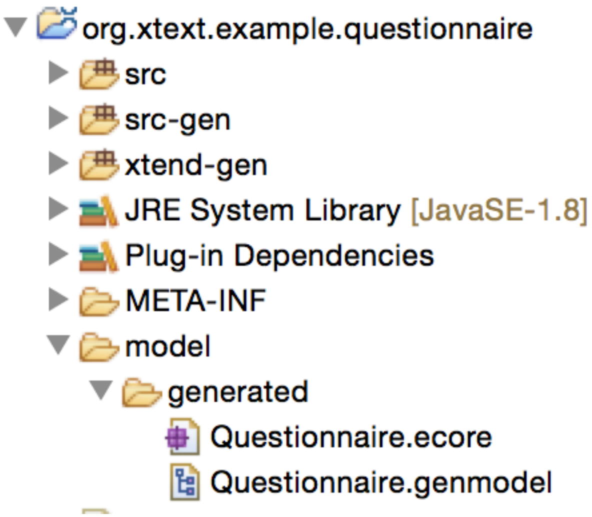


```
2.q ☒
ollSystem {

    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"
            }
        }

    }
```

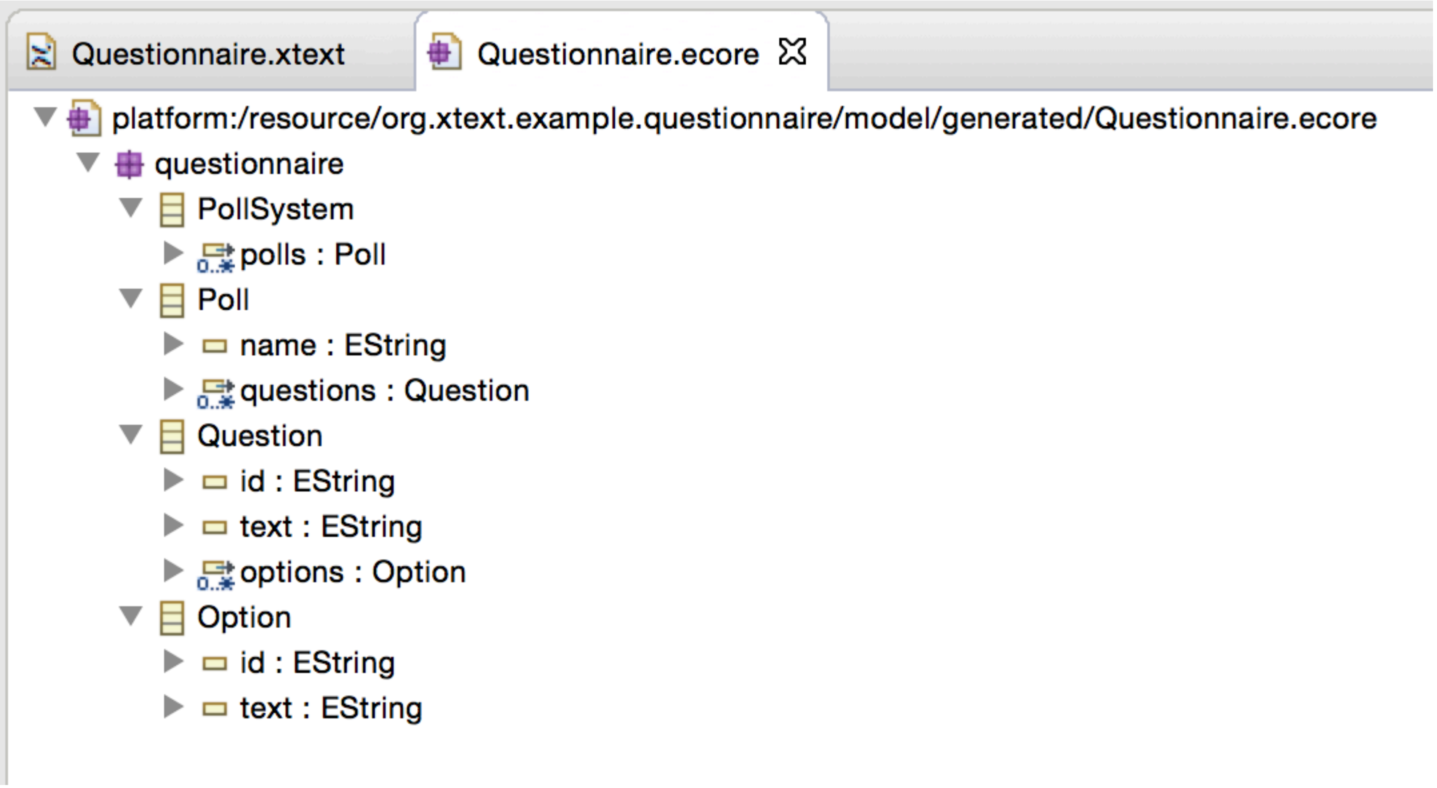


```
▼ platform:/resource/FooQuestionnaire/foo2.q
  ▼ Poll System
    ▼ Poll p1
      ▼ Question q1
          Option A1
          Option A2
          Option A3
          Option A4
      ▶ Question q2
    ▼ Poll p2
      ▶ Question q1
      ▶ Question q2
```

Property	Value
Id	A1
Text	PhantomJS



```
▼ org.xtext.example.questionnaire
    ▶ src
    ▶ src-gen
    ▶ xtend-gen
    ▶ JRE System Library [JavaSE-1.8]
    ▶ Plug-in Dependencies
    ▶ META-INF
    ▼ model
        ▼ generated
            Questionnaire.ecore
            Questionnaire.genmodel
```



Questionnaire.xtext | Questionnaire.ecore ☒

```
▼ platform:/resource/org.xtext.example.questionnaire/model/generated/Questionnaire.ecore
    ▼ questionnaire
        ▼ PollSystem
            ▶ polls : Poll
        ▼ Poll
            ▶ name : EString
            ▶ questions : Question
        ▼ Question
            ▶ id : EString
            ▶ text : EString
            ▶ options : Option
        ▼ Option
            ▶ id : EString
            ▶ text : EString
```

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

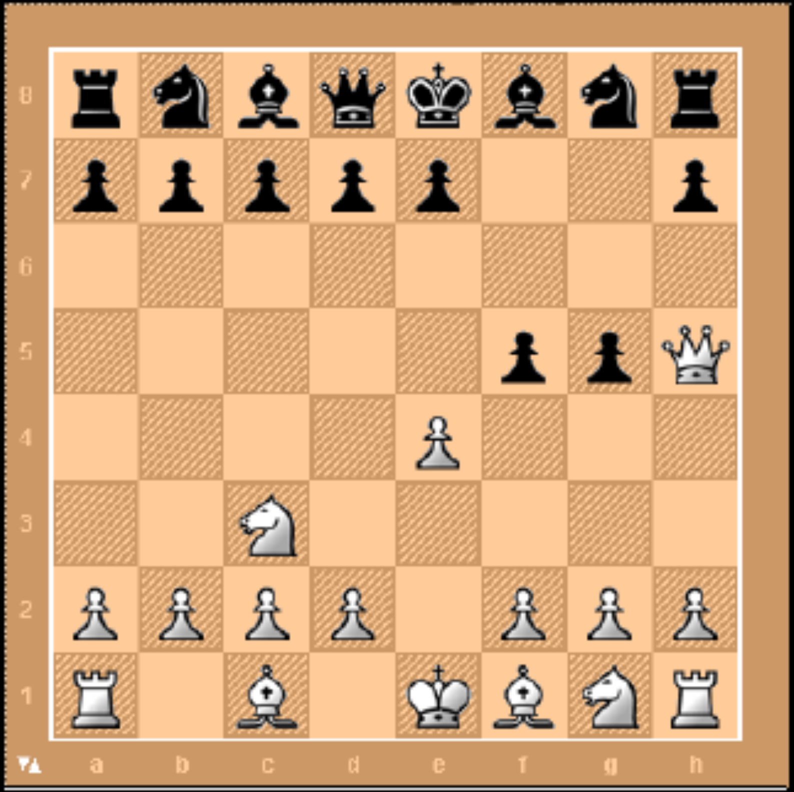
Piece Square Action Destination

g2 - g4

Square Action Destination

Bishop at c8 captures knight at h3

B c8 x h3



P e2 – e4

p g7 – g5

Knight at b2 moves to c3

pawn at f7 moves to f5

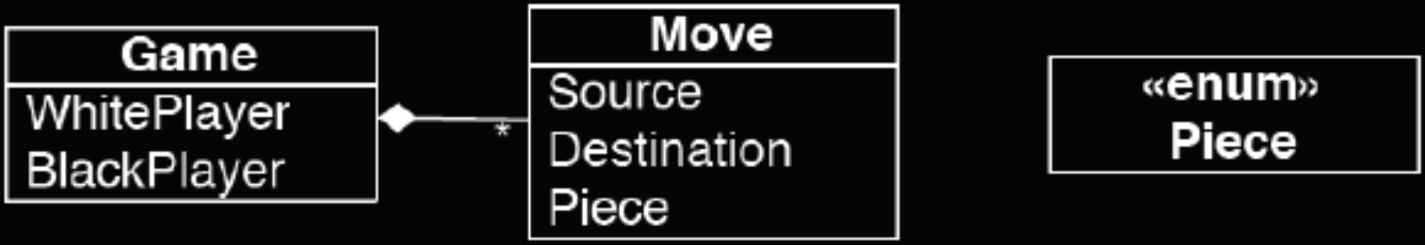
Q d1 – h5

1-0

Concrete Syntax

Constraints !!!

Abstract Syntax



Game

WhitePlayer
BlackPlayer

Move

Source
Destination
Piece

«enum»
Piece

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
```

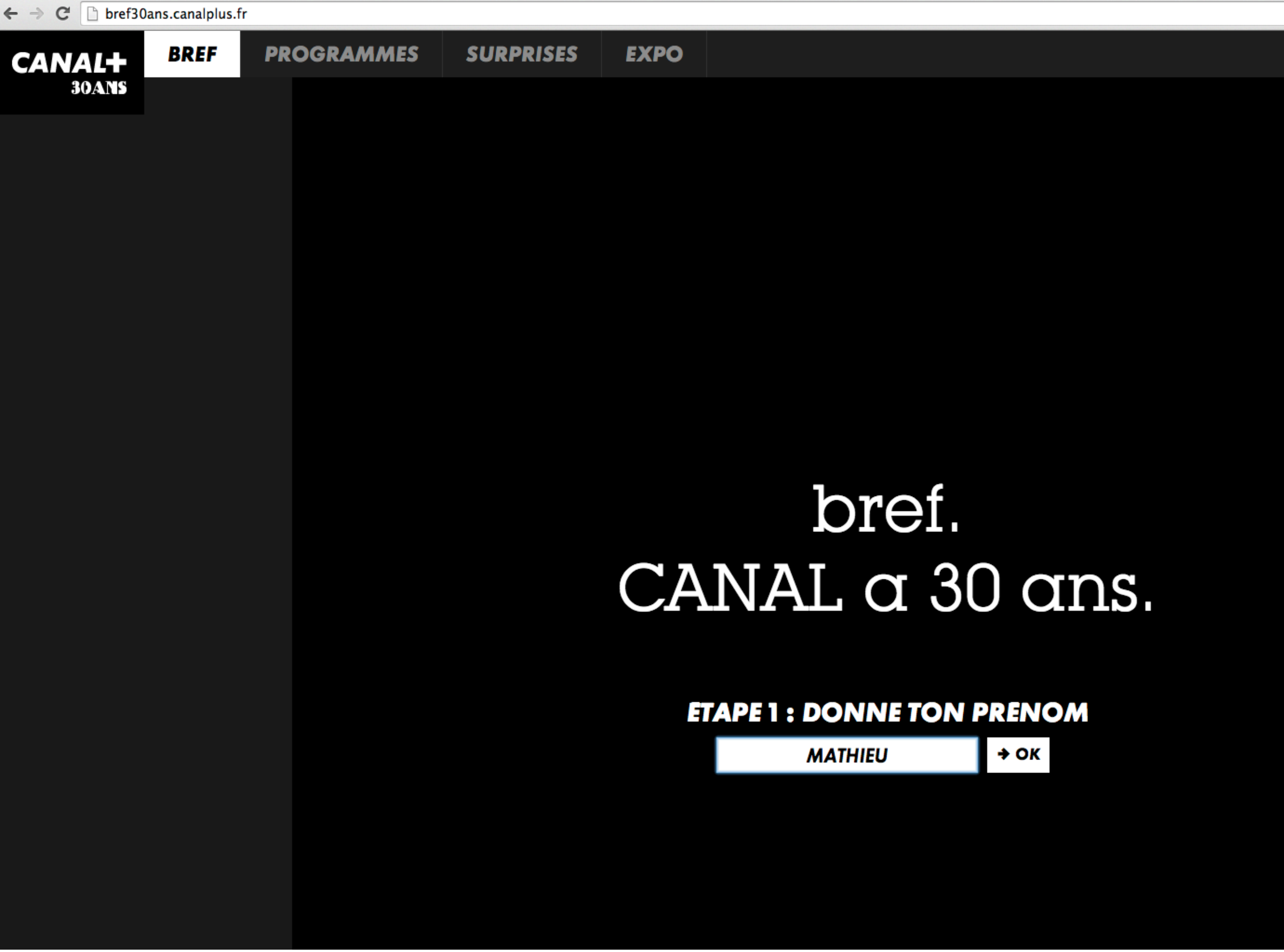
Running example

Models

Languages

Transformation

Variability



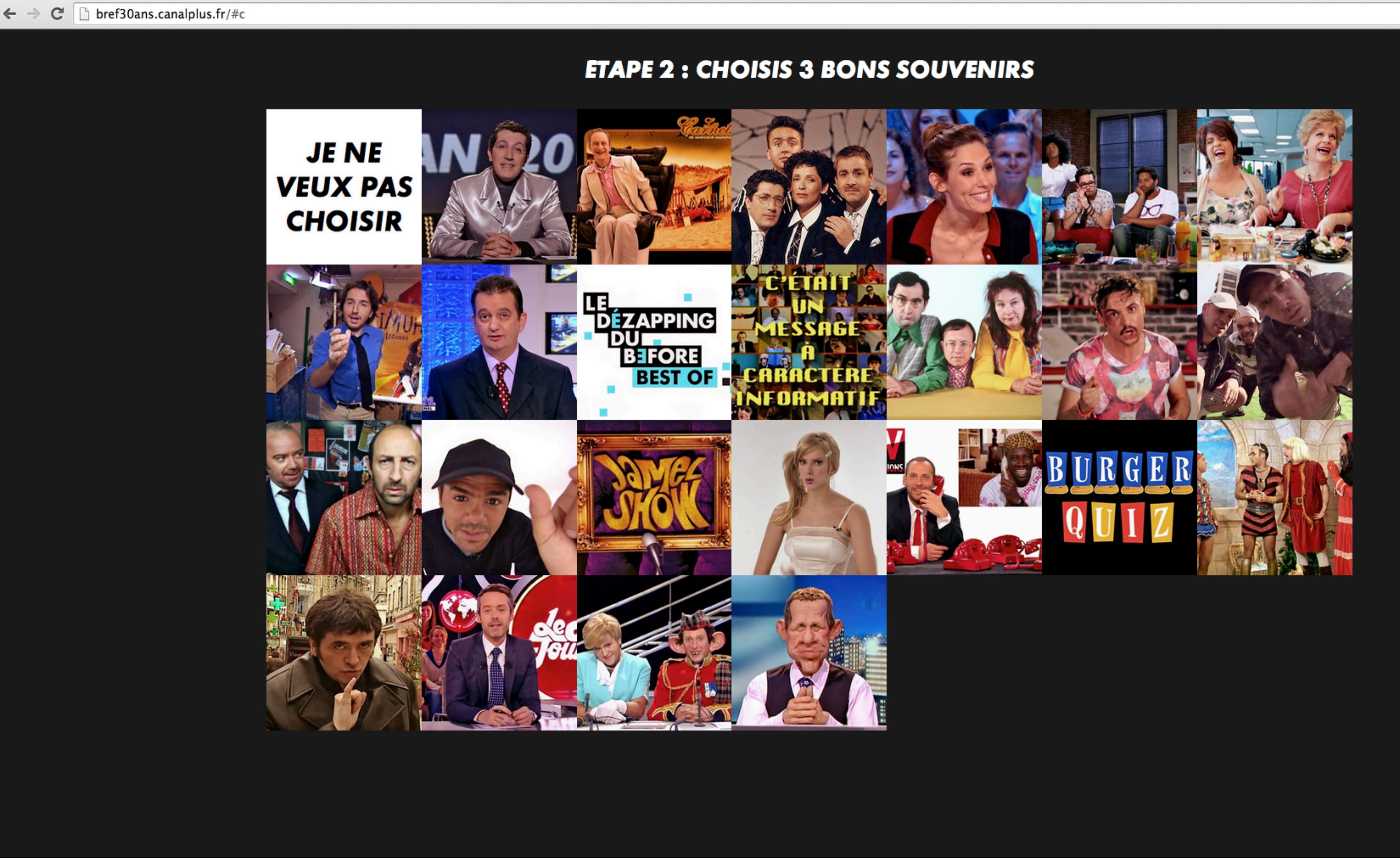
bref.

CANAL a 30 ans.

ETAPE 1 : DONNE TON PRENOM

MATHIEU

Online Generator

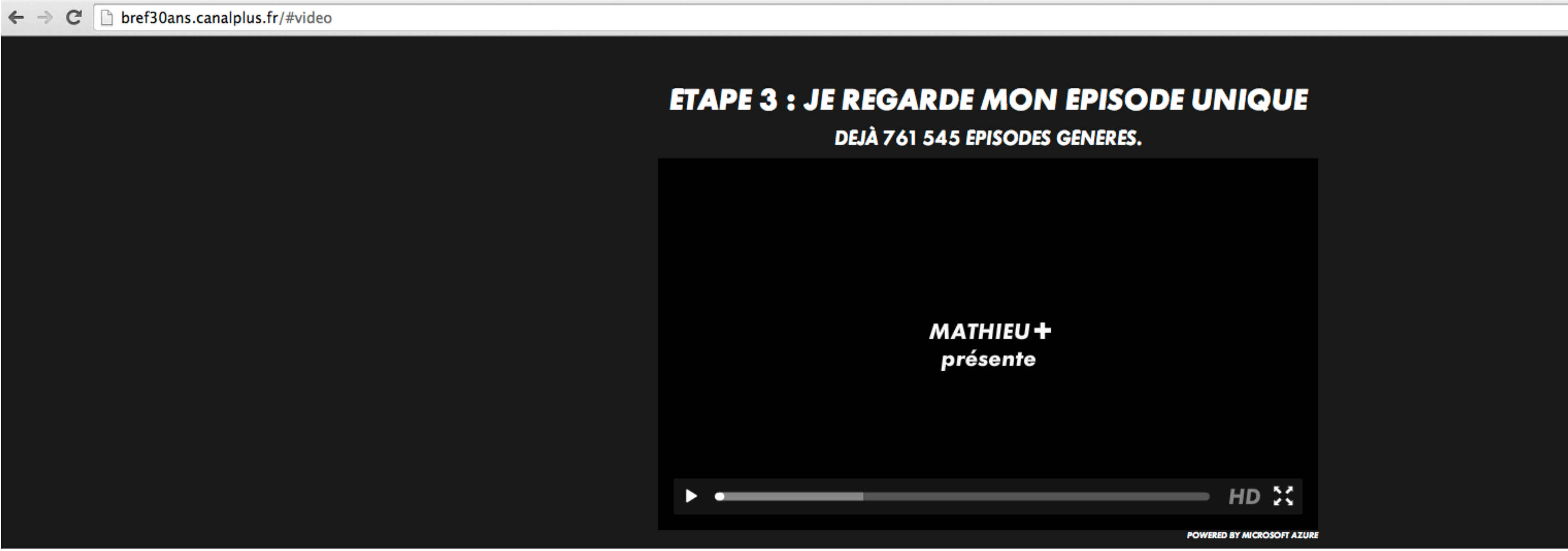


bref30ans.canalplus.fr/#c

ETAPE 2 : CHOISIS 3 BONS SOUVENIRS

JE NE VEUX PAS CHOISIR
LE DÉZAPPING DU BEFORE BEST OF
C'ÉTAIT UN MESSAGE À CARACTÈRE INFORMATIF
Jamel Show
BURGER QUIZ

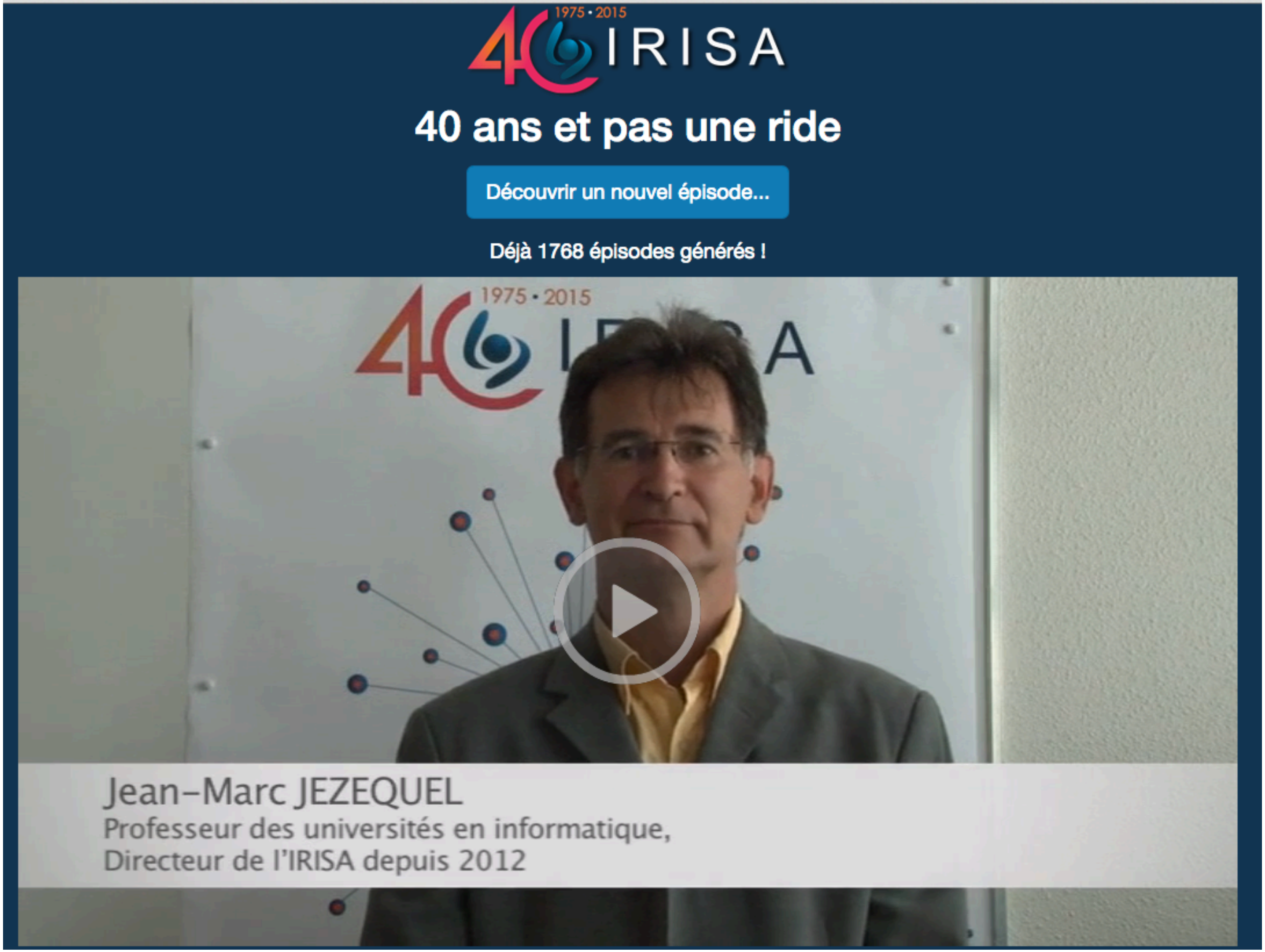
Variant



ETAPE 3 : JE REGARDE MON EPISODE UNIQUE

DEJÀ 761 545 EPISODES GENERES.

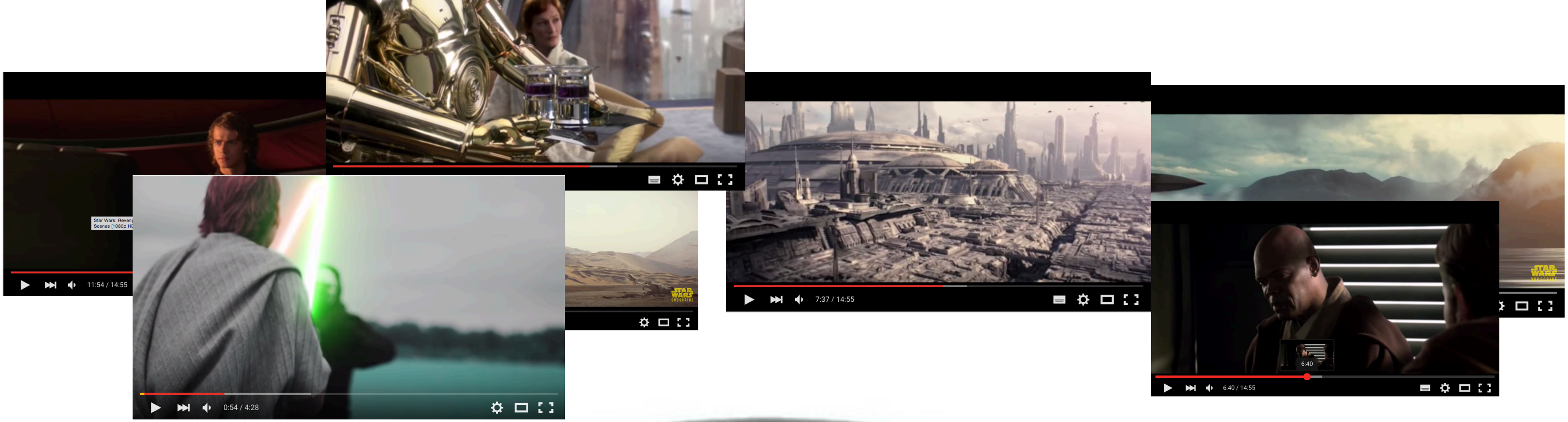
Guillaume Bécan, Mathieu Acher, Jean-Marc Jézéquel, and Thomas Menguy. On the Variability Secrets of an Online Video Generator (2015). In VaMoS'15



40 ans et pas une ride

Déjà 1768 épisodes générés !

Jean-Marc JEZEQUEL
Professeur des universités en informatique,
Directeur de l'IRISA depuis 2012



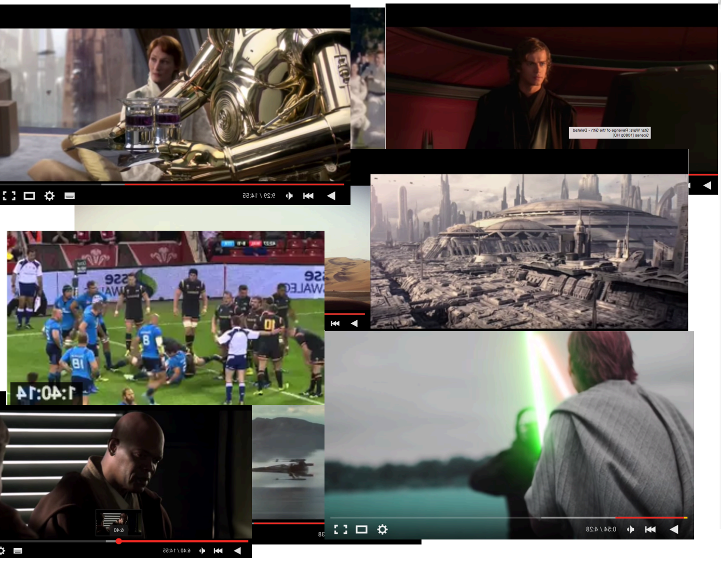
Generator

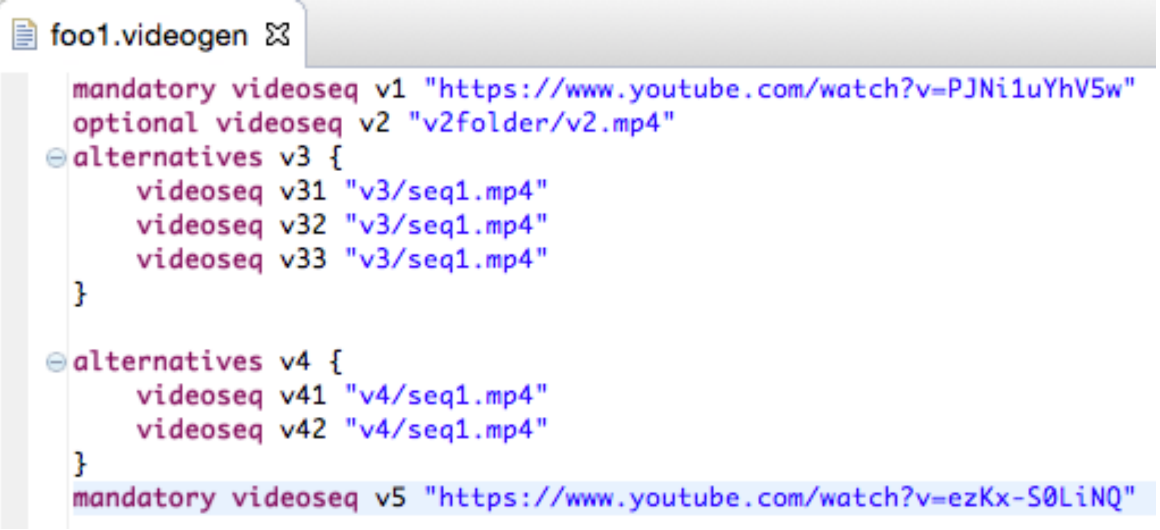
~ composition of video sequences

video variants

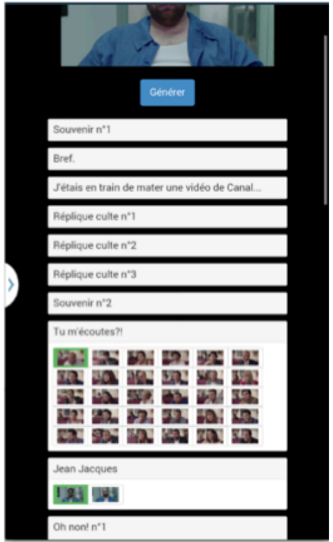
Generator
~ composition of
video sequences

video
variants



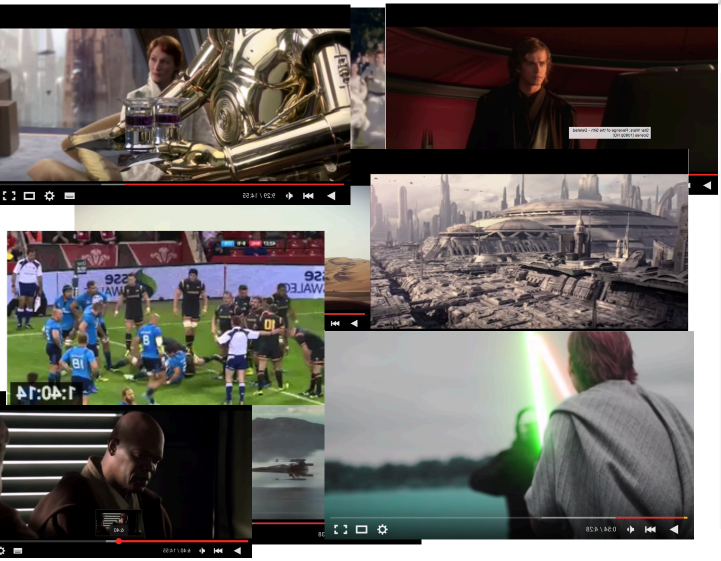


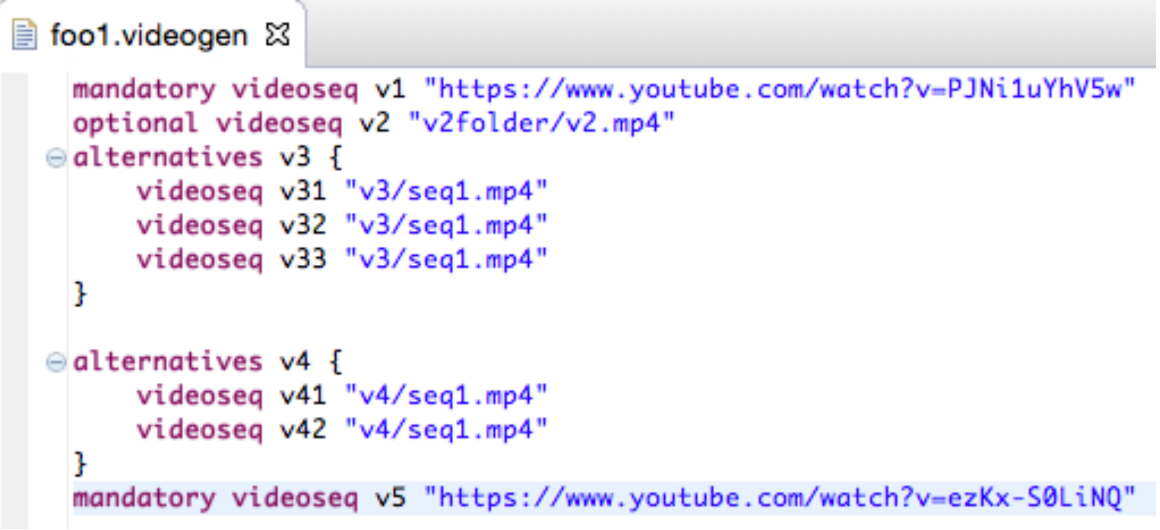
```
mandatory videoseq v1 "https://www.youtube.com/watch?v=PJNi1uYhV5w"
optional videoseq v2 "v2folder/v2.mp4"
alternatives v3 {
    videoseq v31 "v3/seq1.mp4"
    videoseq v32 "v3/seq1.mp4"
    videoseq v33 "v3/seq1.mp4"
}

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



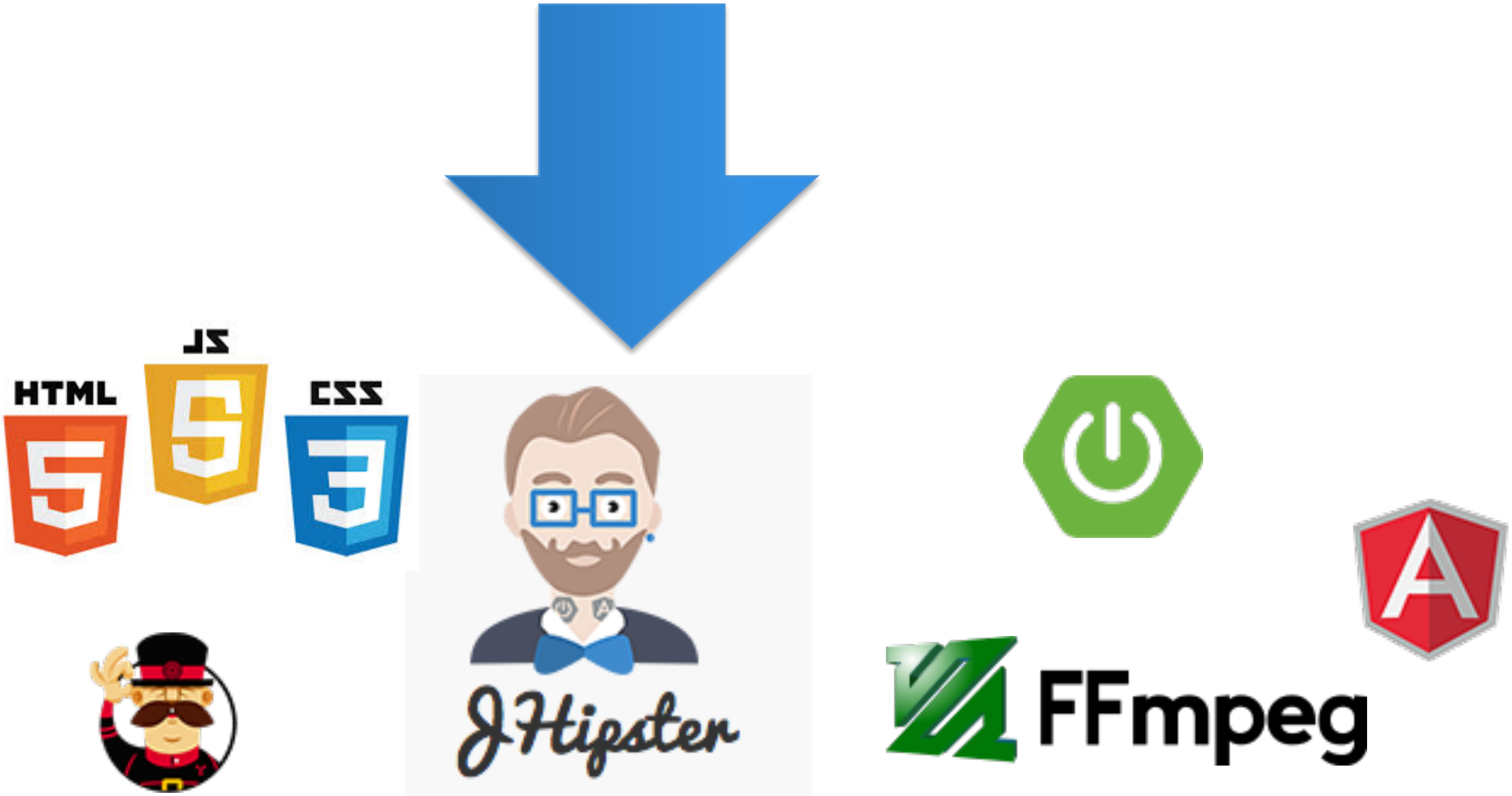
Website/online

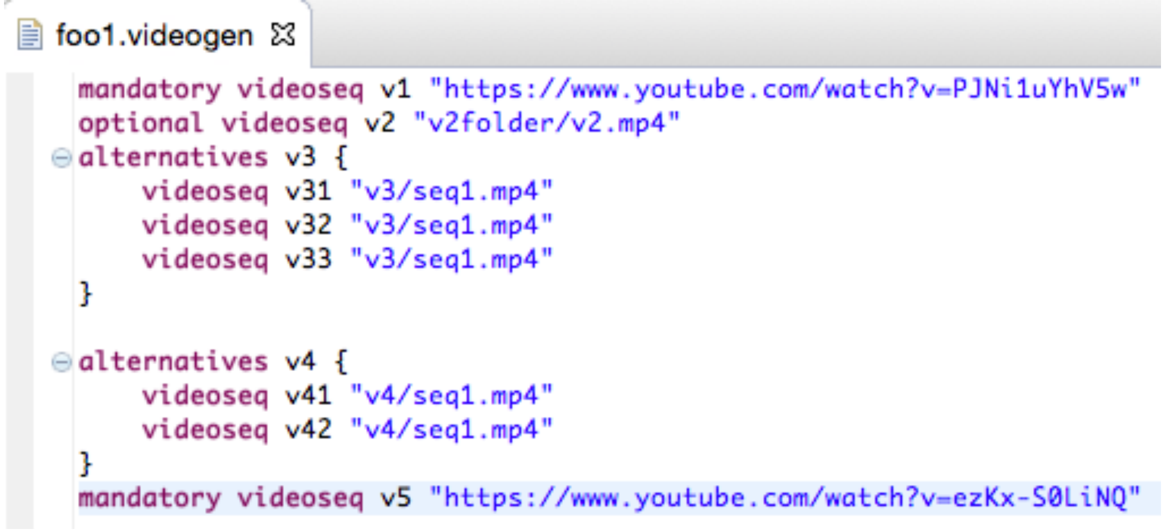
- Random generation
- Configurator
- Game
- …





```
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"
```





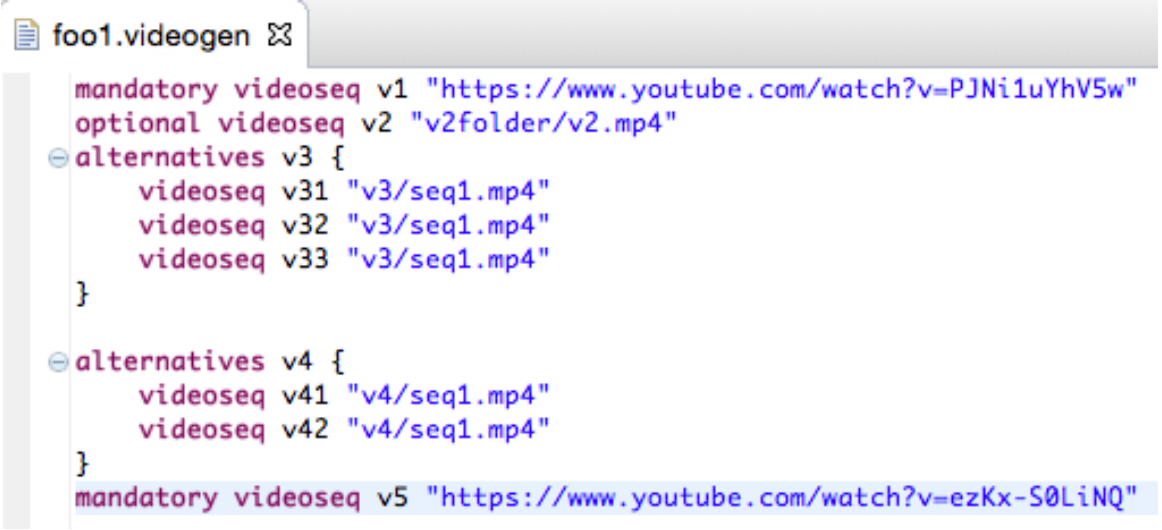
```
mandatory videoseq v1 "https://www.youtube.com/watch?v=PJNi1uYhV5w"
optional videoseq v2 "v2folder/v2.mp4"
alternatives v3 {
     videoseq v31 "v3/seq1.mp4"
     videoseq v32 "v3/seq1.mp4"
     videoseq v33 "v3/seq1.mp4"
}

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

#1 How to design, create, and support dedicated <u>languages</u> (DSLs)?

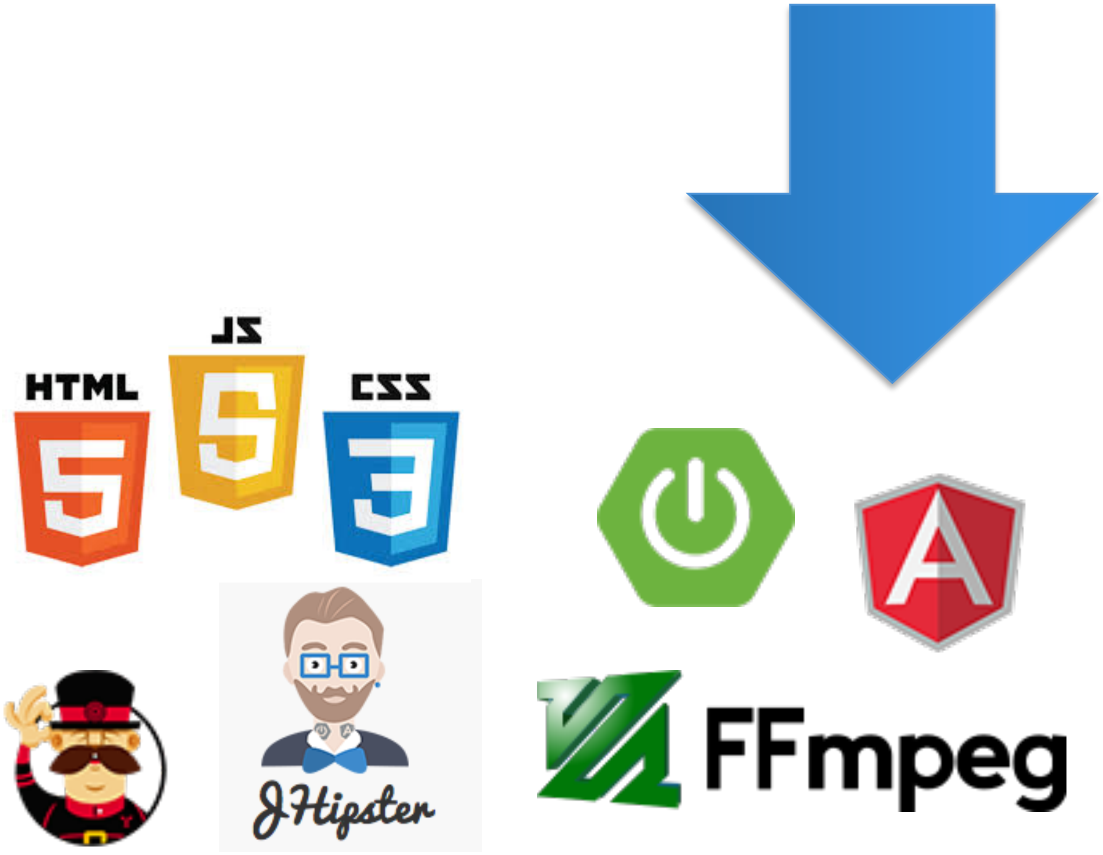
#2 How to <u>transform</u> models/programs?

#3 How to manage variability/variants?

#4 How do frameworks internally work?



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

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

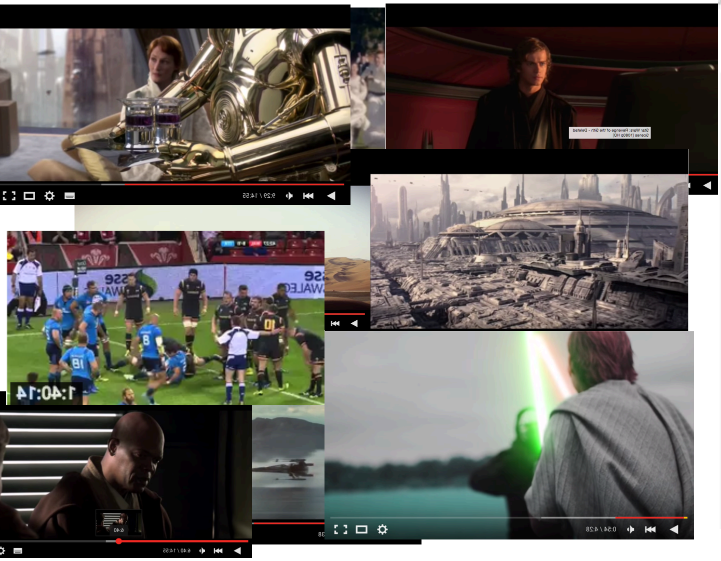


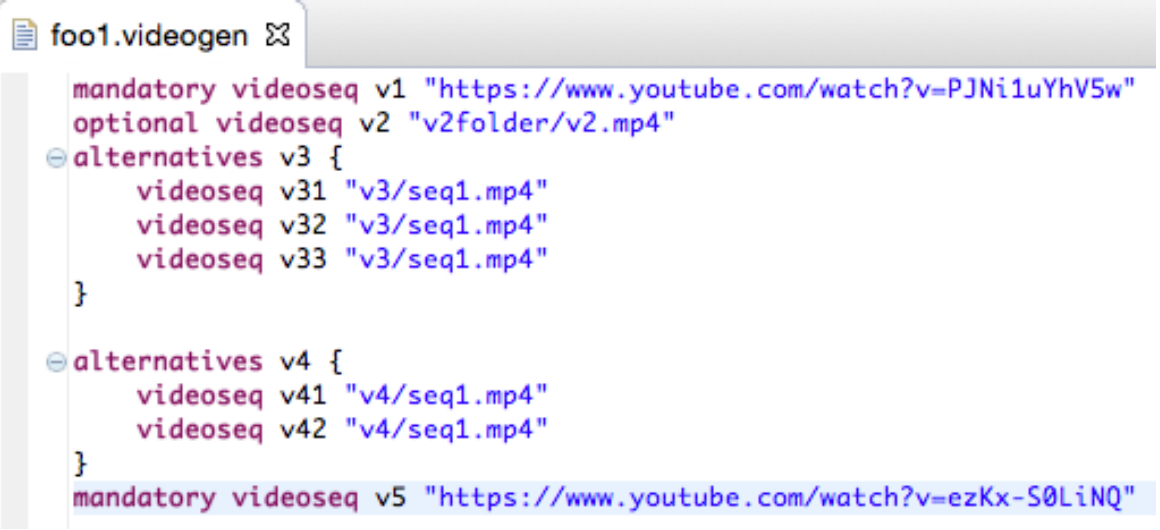
#1 How to design, create, and support dedicated languages (DSLs)?

#2 How to transform models/programs?

#3 How to manage variability/variants?

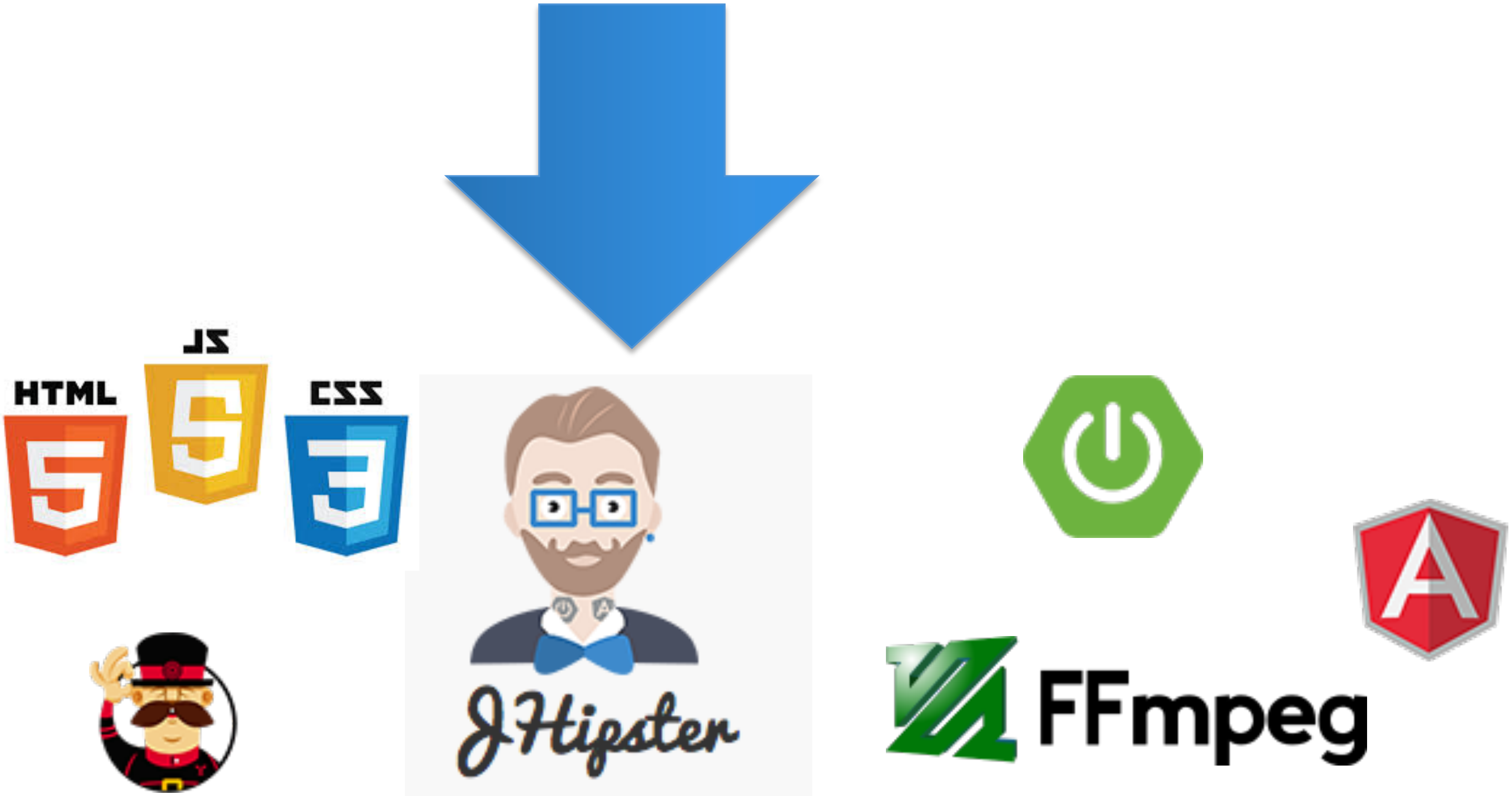
#4 How do frameworks internally work?

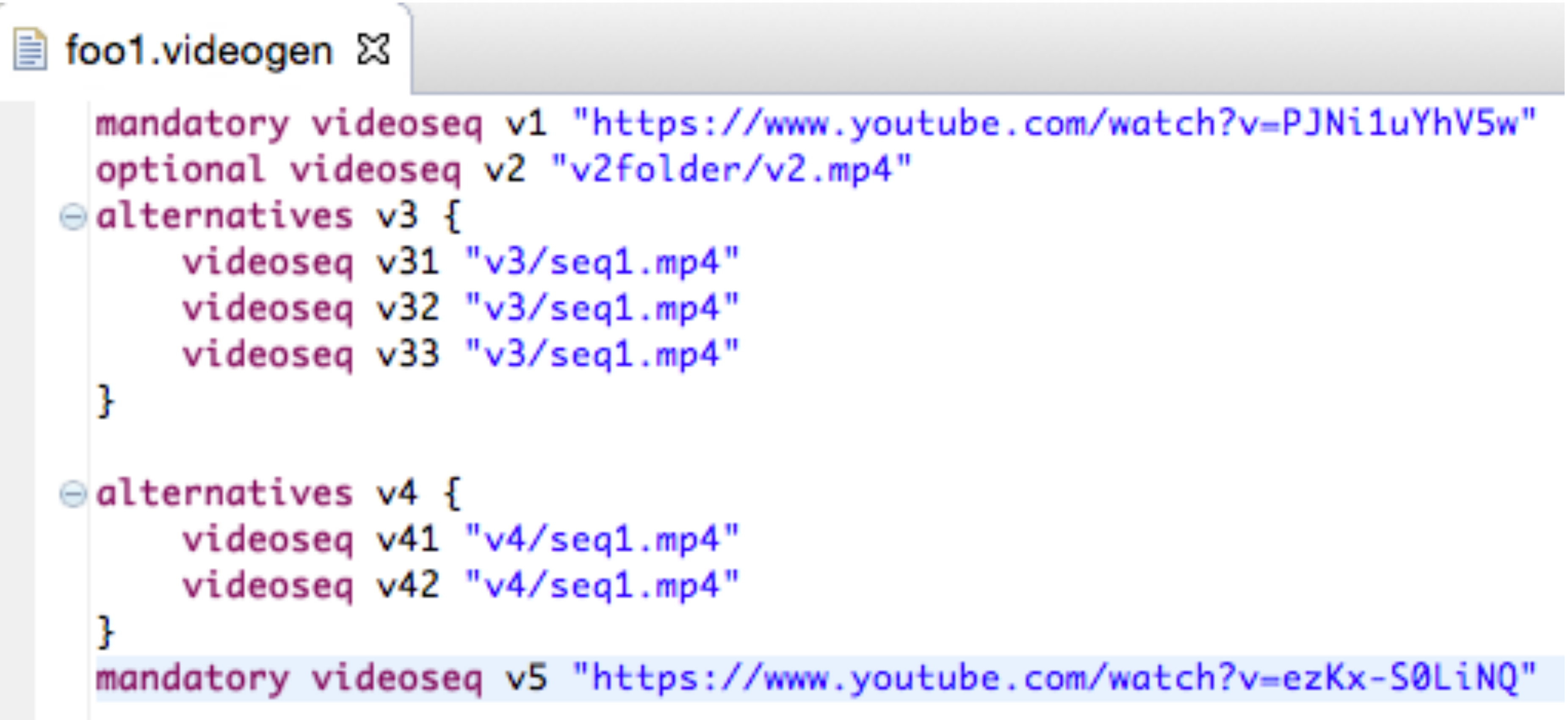




foo1.videogen

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

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





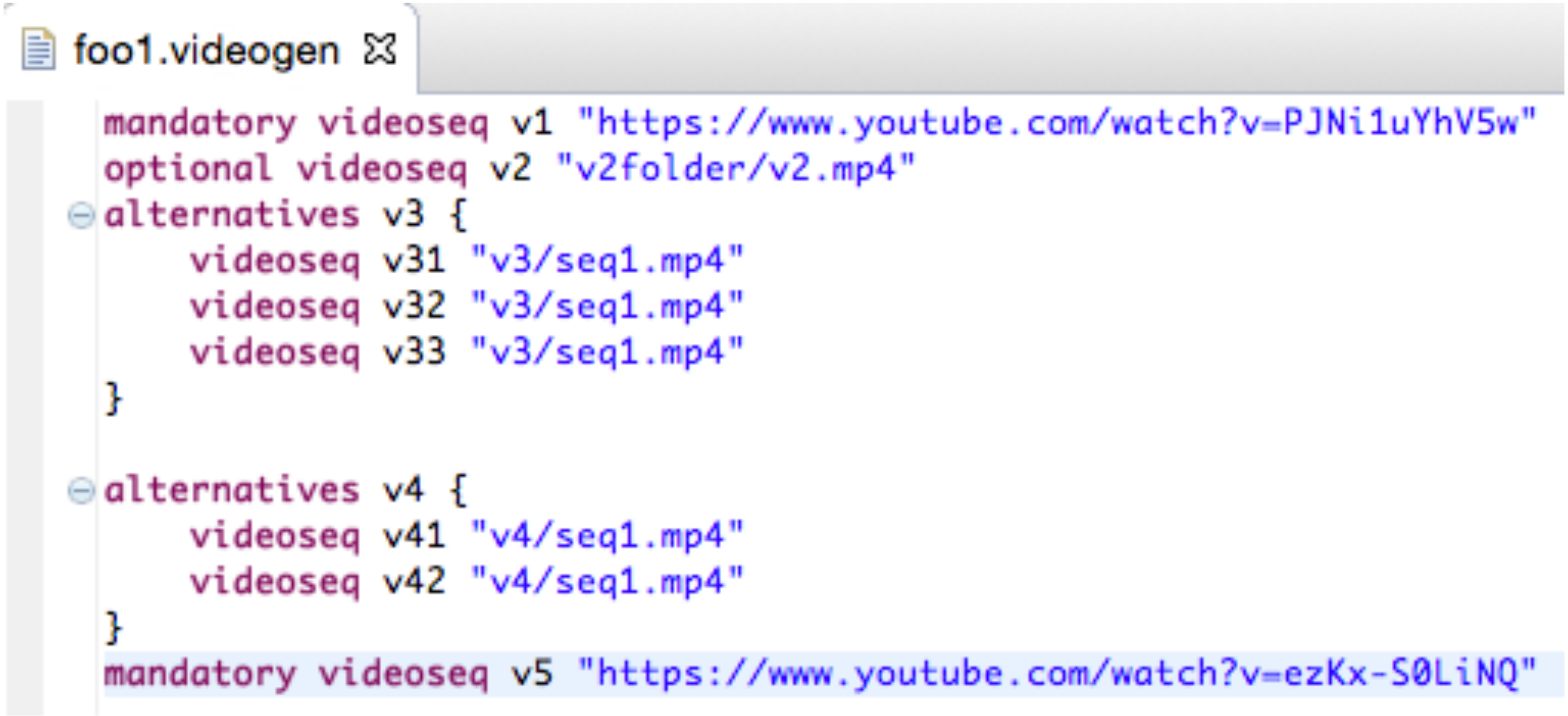
```
mandatory videoseq v1 "https://www.youtube.com/watch?v=PJNi1uYhV5w"
optional videoseq v2 "v2folder/v2.mp4"
alternatives v3 {
    videoseq v31 "v3/seq1.mp4"
    videoseq v32 "v3/seq1.mp4"
    videoseq v33 "v3/seq1.mp4"
}

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

#7

e9a8d603

Write a Xtext grammar so that the specification below is conformant



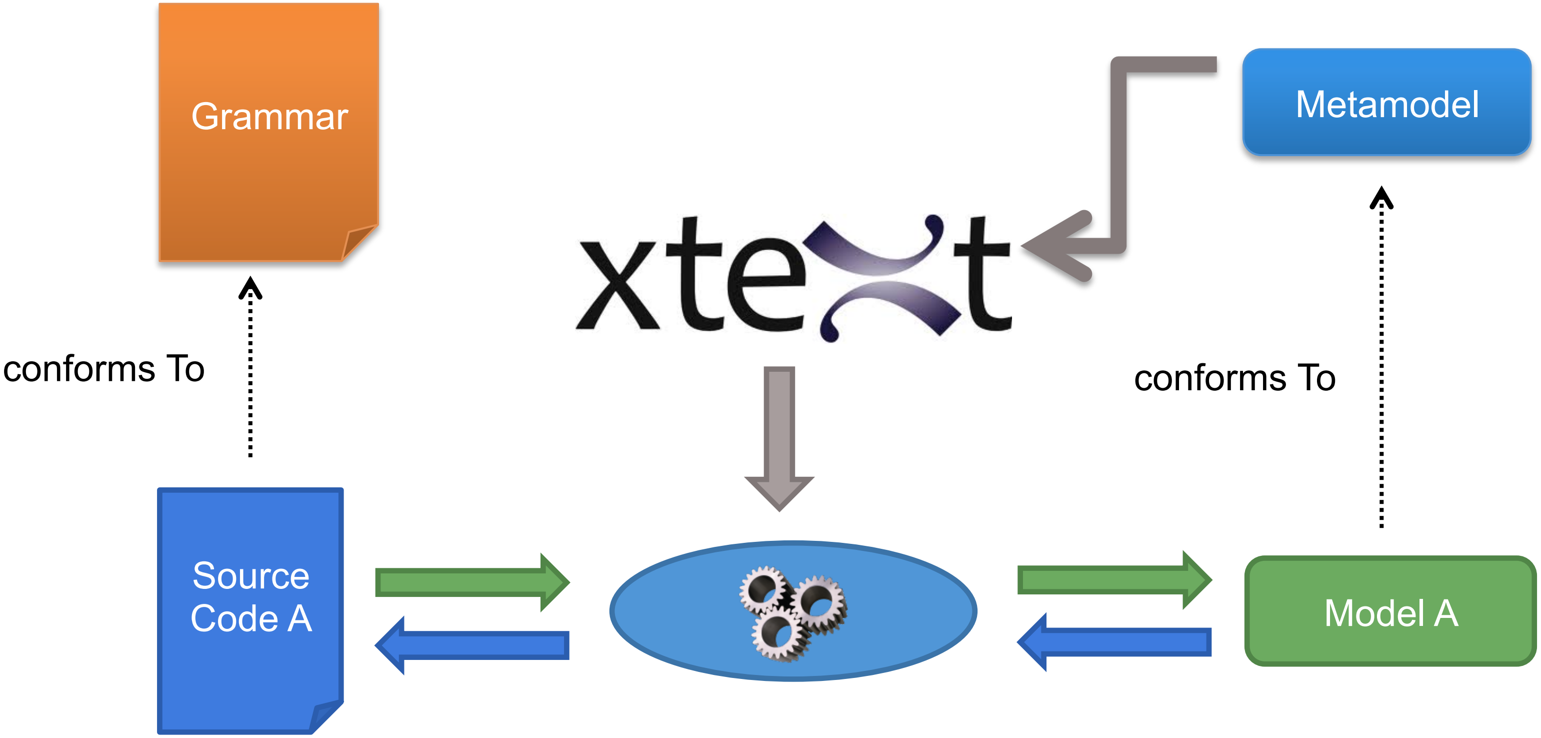
```
mandatory videoseq v1 "https://www.youtube.com/watch?v=PJNi1uYhV5w"
optional videoseq v2 "v2folder/v2.mp4"
alternatives v3 {
    videoseq v31 "v3/seq1.mp4"
    videoseq v32 "v3/seq1.mp4"
    videoseq v33 "v3/seq1.mp4"
}

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



xtext

Give me a **metamodel**,

I’ll give you (for free)
 * a comprehensive editor (auto-completion, syntax highlitening, etc.) in Eclipse
 * a grammar and facilities to load/serialize/visit conformant models (Java ecosystem)
 * extension to override/extend « default » facilities (e.g., checker)

xtext

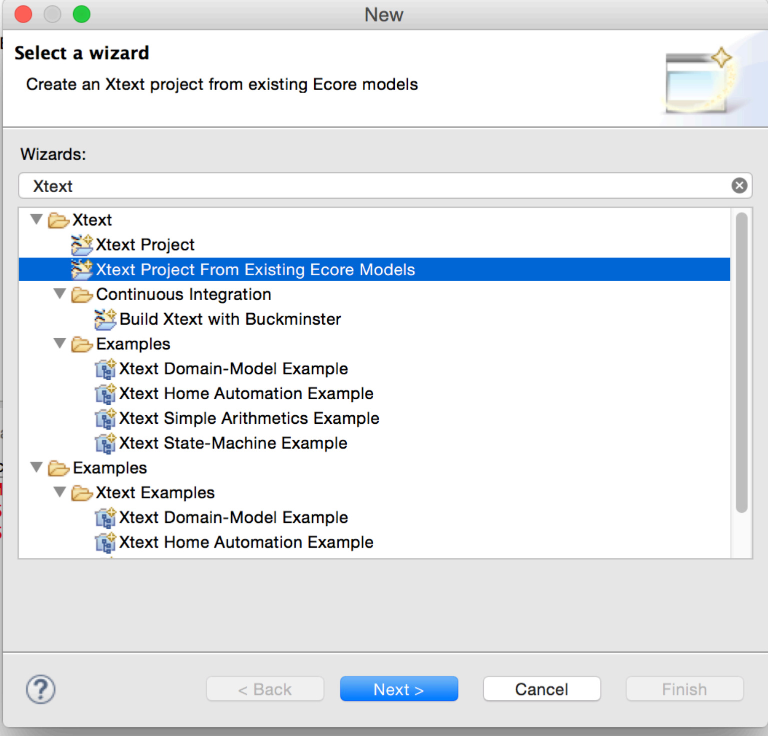
Give me a **metamodel**,

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

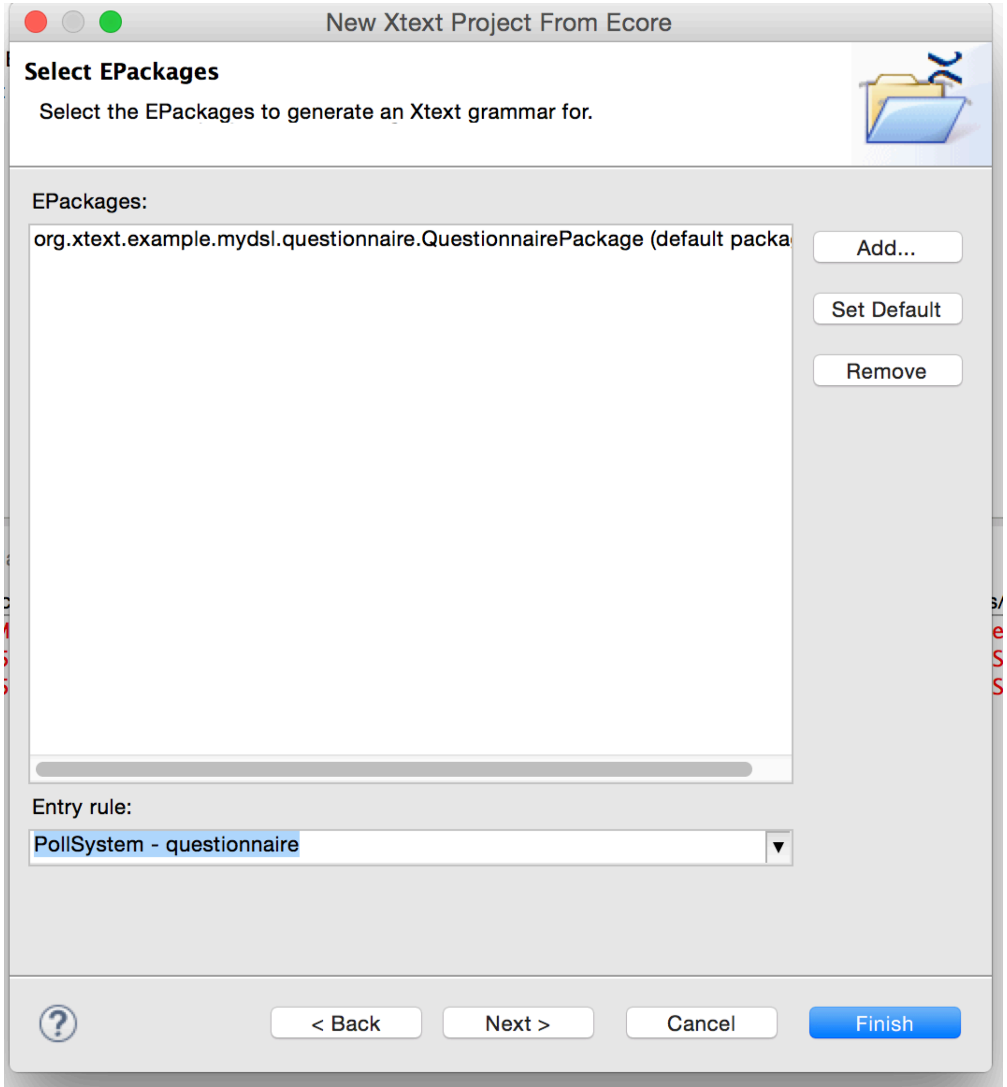


Select a wizard

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



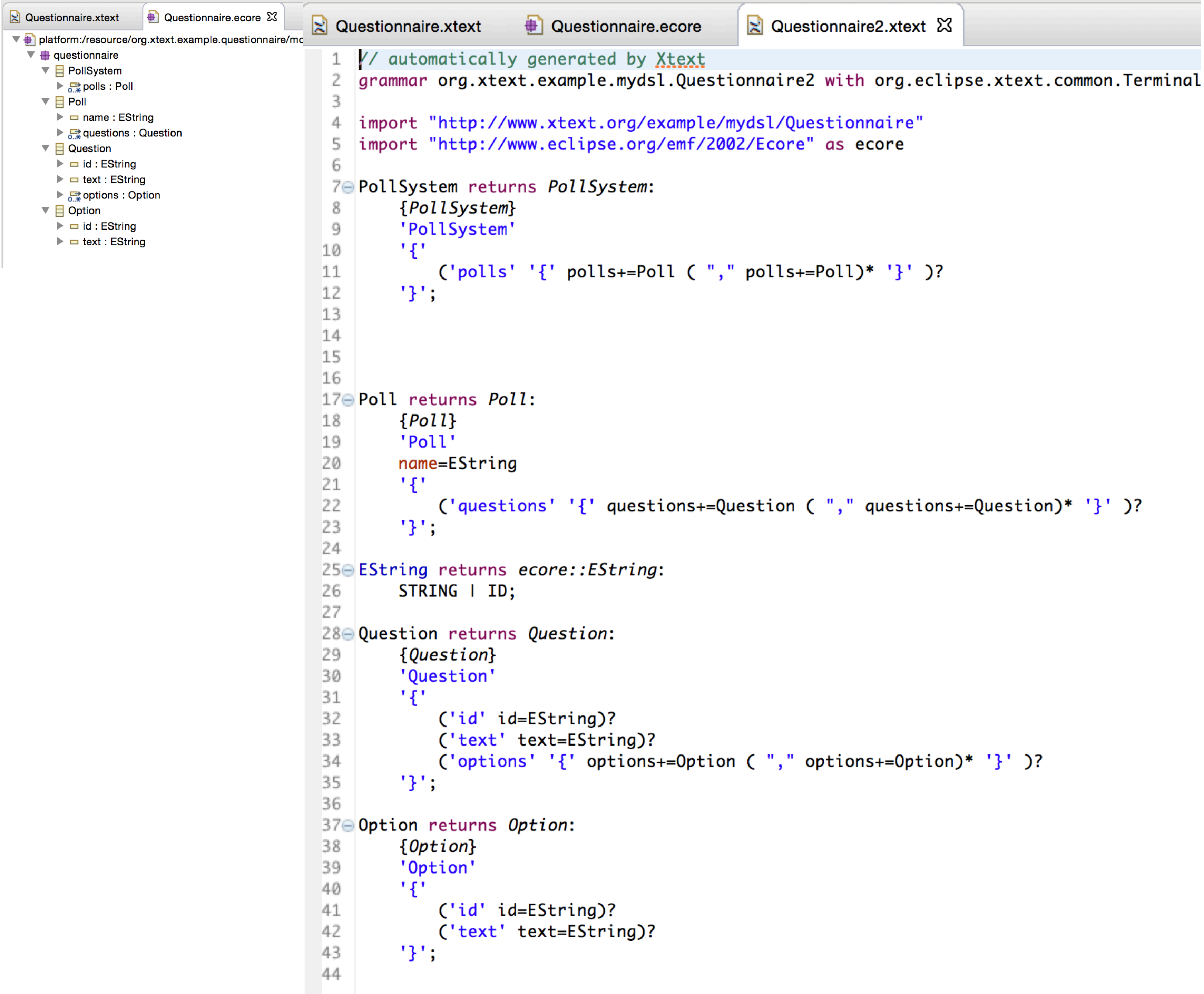
Select EPackages

Select the EPackages to generate an Xtext grammar for.

EPackages:

org.xtext.example.mydsl.questionnaire.QuestionnairePackage (default packa

Entry rule: PollSystem - questionnaire



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

```
// 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)* '}' )?
    '}';




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

EString returns ecore::EString:
    STRING | ID;

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

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

```

Quizz Time

#8

e9a8d603

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

Graphical DSL

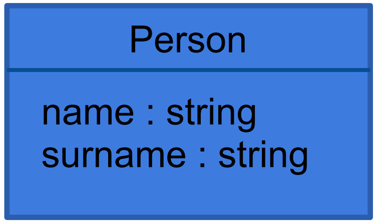
(vs Textual DSL)

Graphical vs Textual DSLs

- Success depends on how the notation fits the domain

```
class Person {
  private String name;
  private String name;
}
```

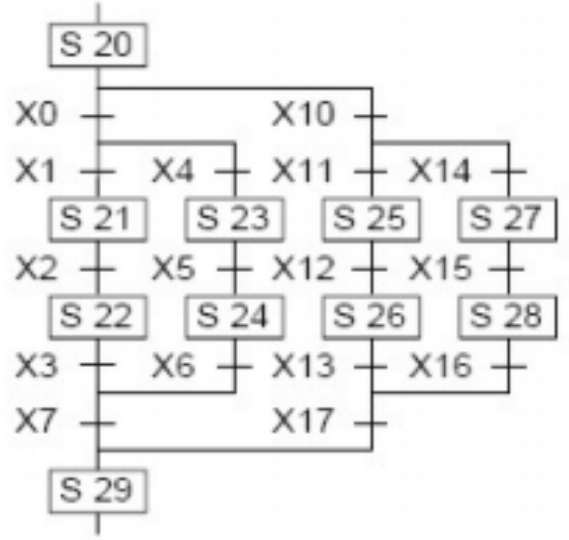
```
Person has (name, surname)
```

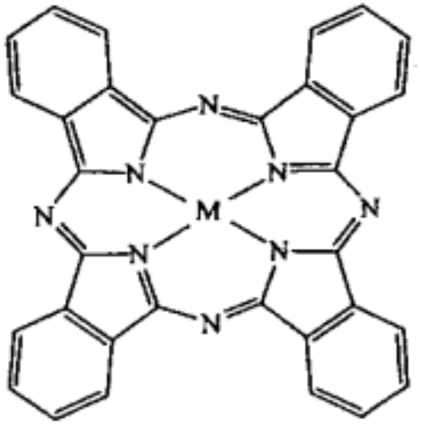


Person

name : string
surname : string

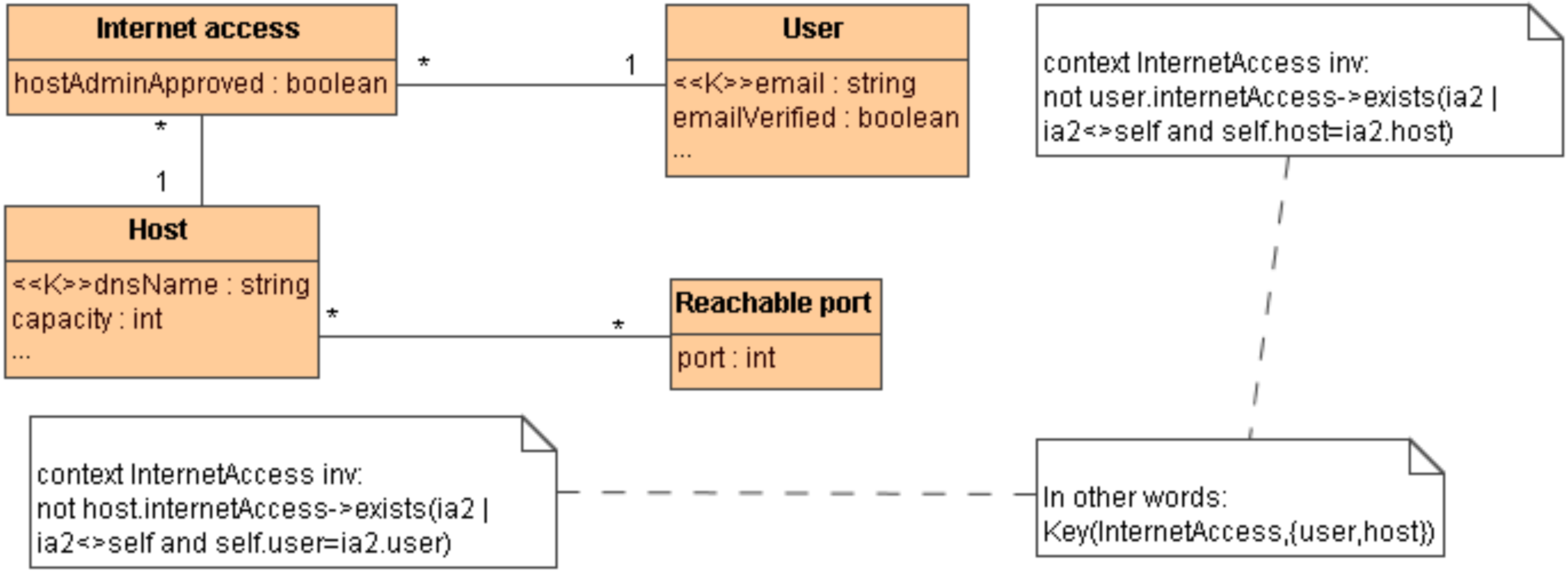
- Graphical DSLs are not always easier to understand





phthalocyanine

A language can be graphical and textual



Internet access

hostAdminApproved : boolean

User

<<K>>email : string
emailVerified : boolean
...

Host

<<K>>dnsName : string
capacity : int
...

Reachable port

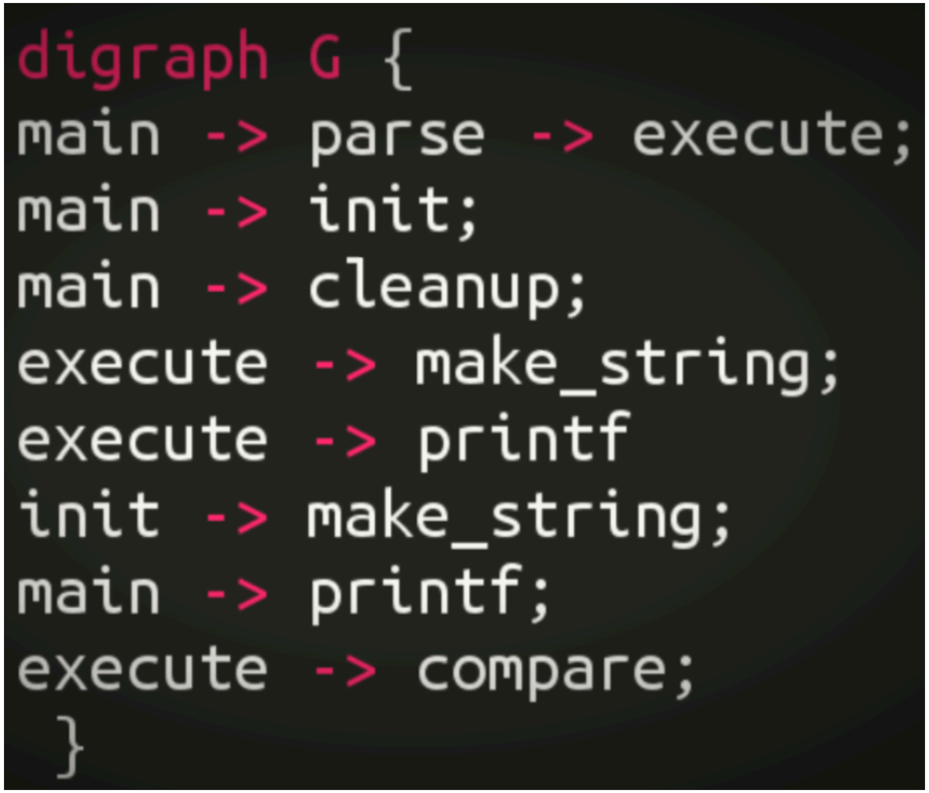
port : int

```
context InternetAccess inv:
not user.internetAccess->exists(ia2 |
ia2<>self and self.host=ia2.host)
```

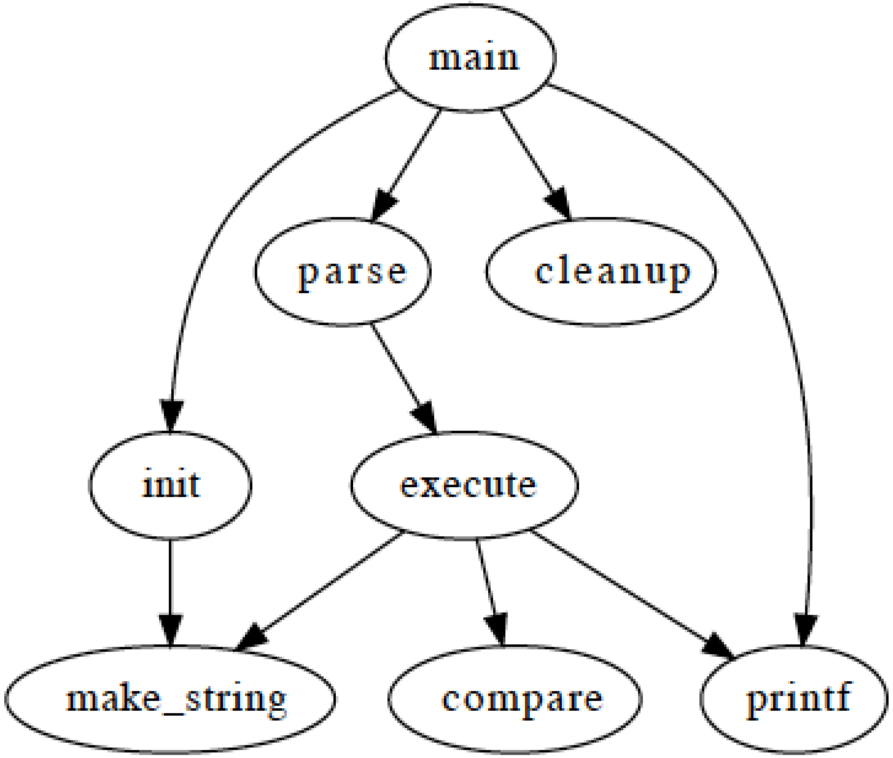
```
context InternetAccess inv:
not host.internetAccess->exists(ia2 |
ia2<>self and self.user=ia2.user)
```

In other words:
Key(InternetAccess,{user,host})

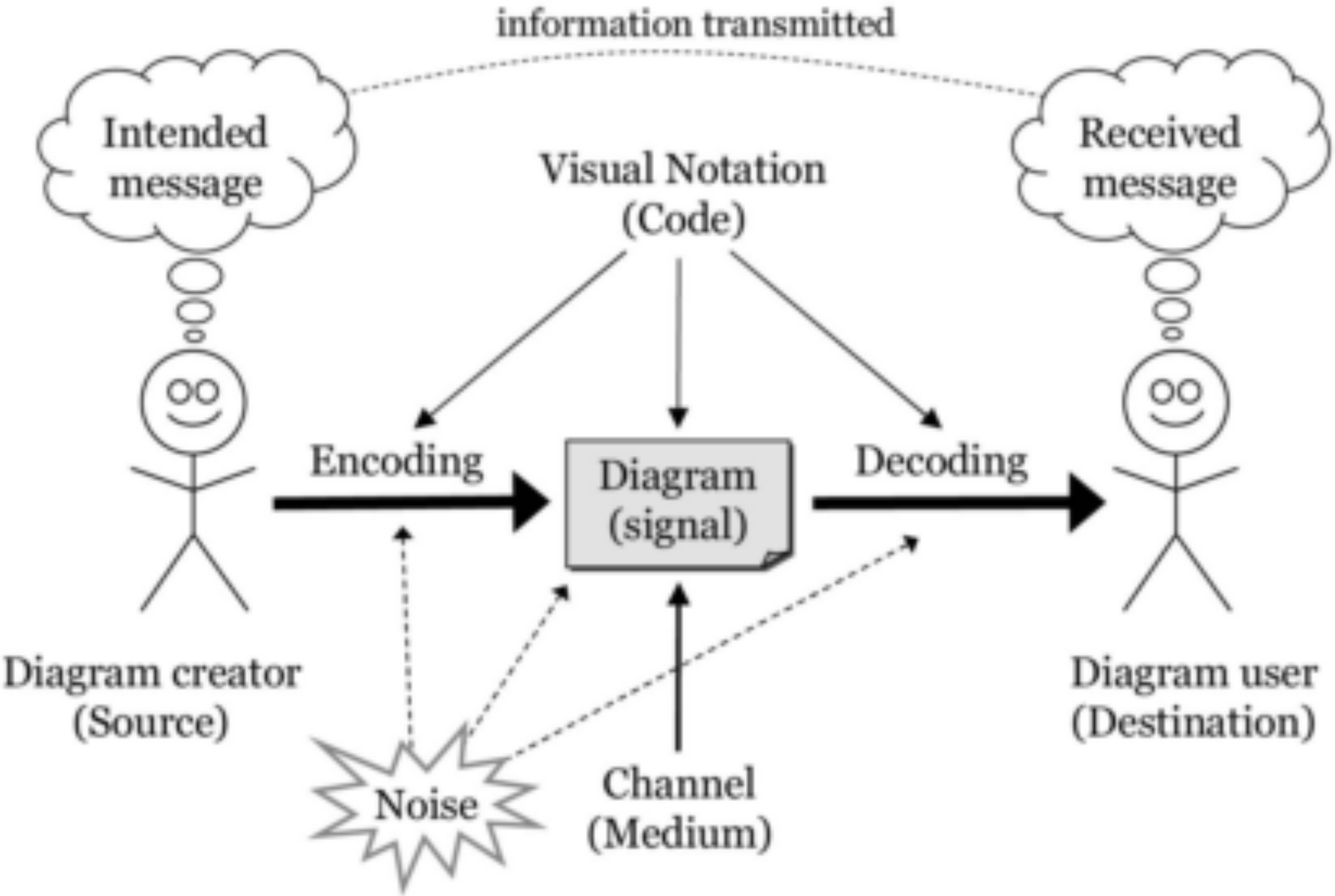
Alternative representation



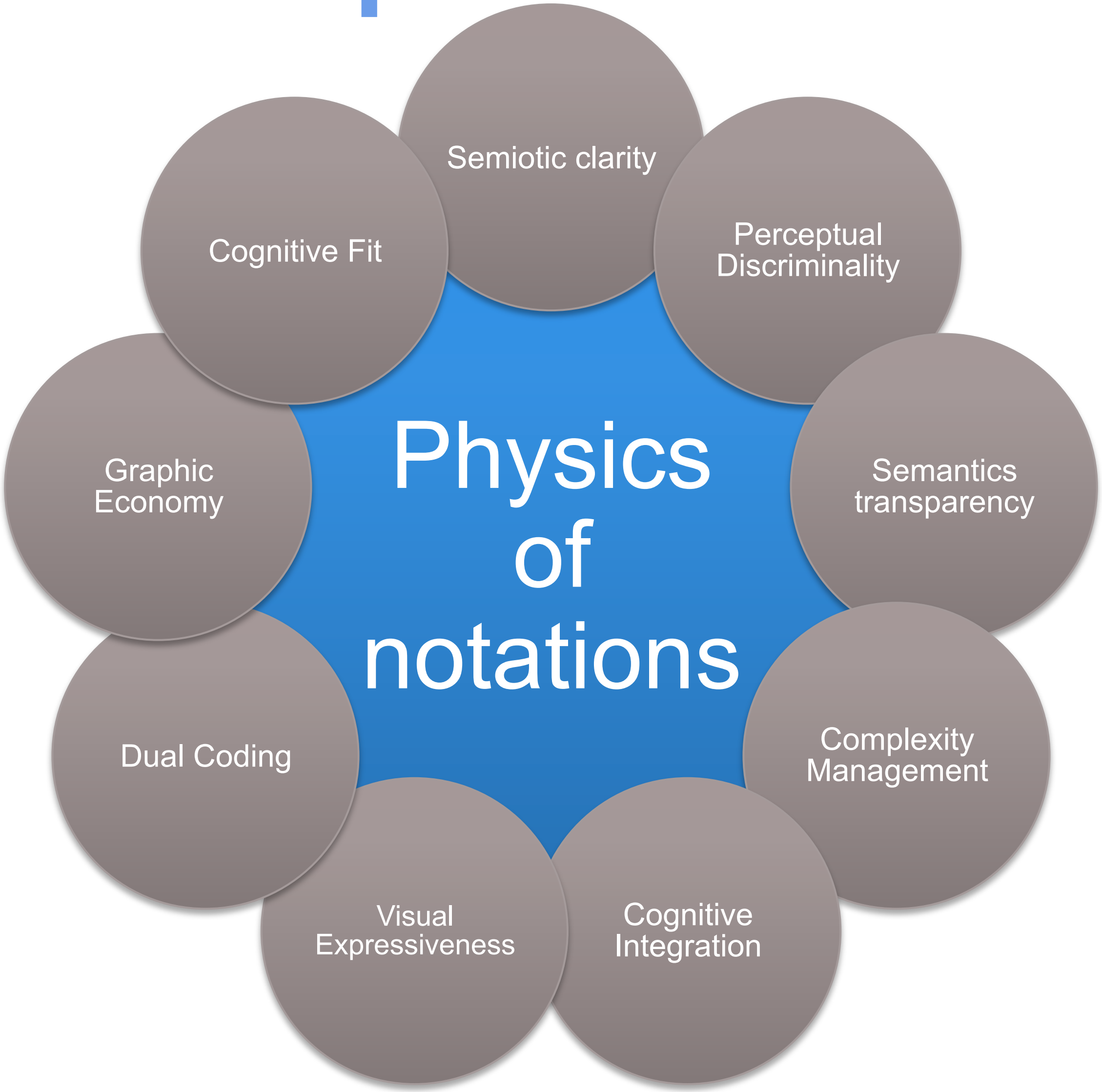
```
digraph G {
main -> parse -> execute;
main -> init;
main -> cleanup;
execute -> make_string;
execute -> printf
init -> make_string;
main -> printf;
execute -> compare;
 }
```



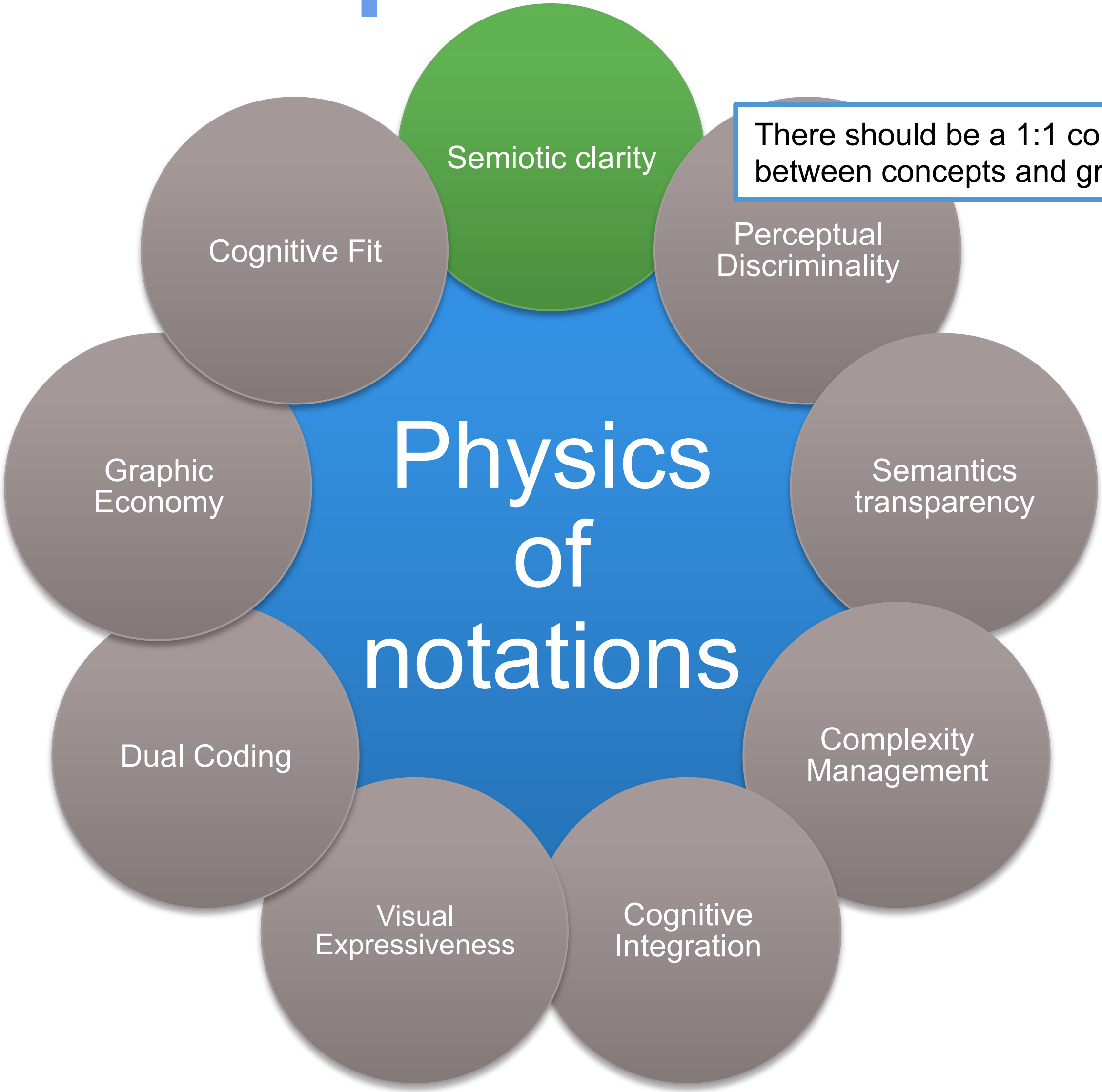
Recommendations for Graphical DSLs



Recommendations for Graphical DSLs

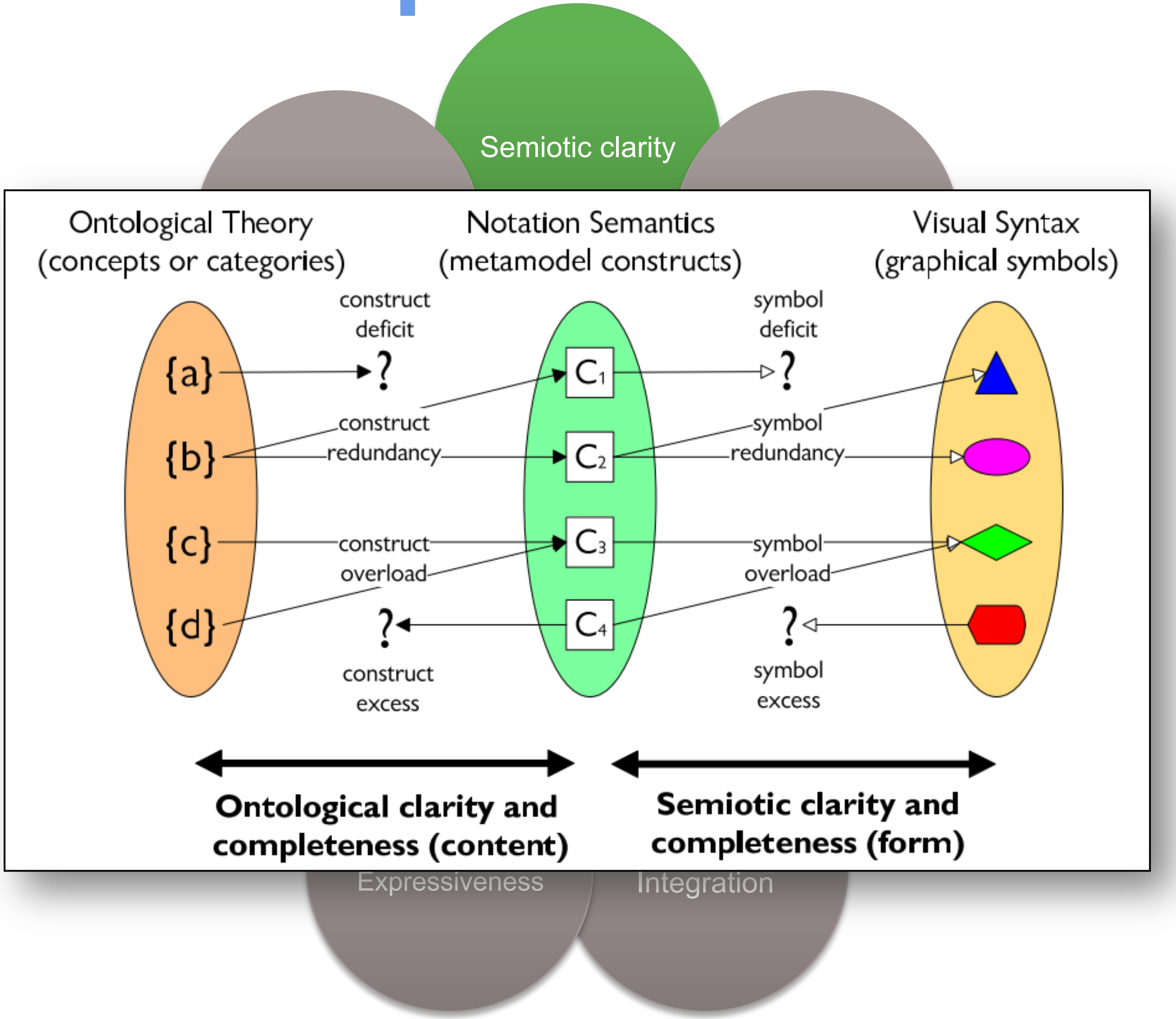


Recommendations for Graphical DSLs



There should be a 1:1 correspondence between concepts and graphical symbols

Recommendations for Graphical DSLs

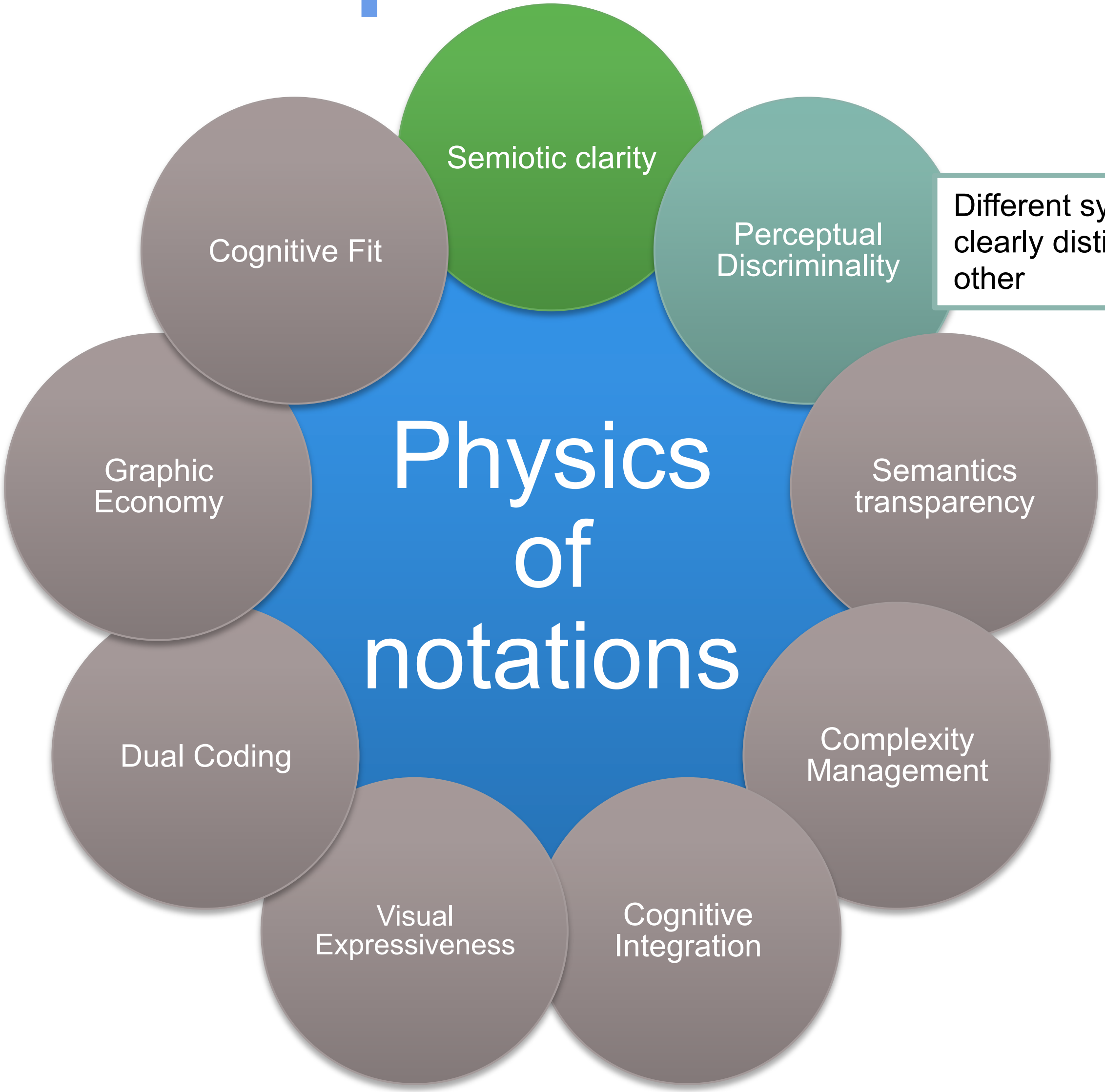


Semiotic clarity

Expressiveness

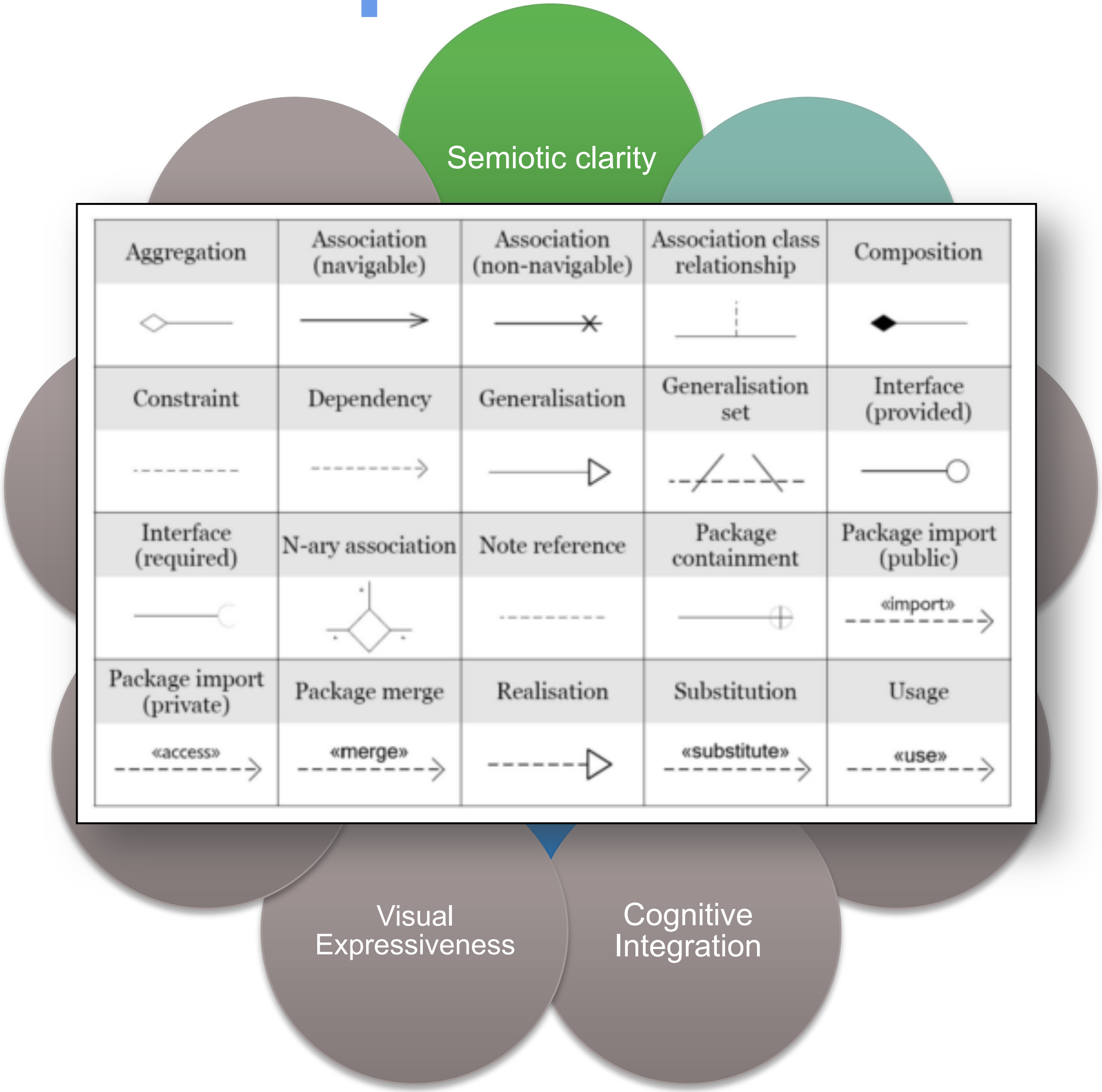
Integration

Recommendations for Graphical DSLs



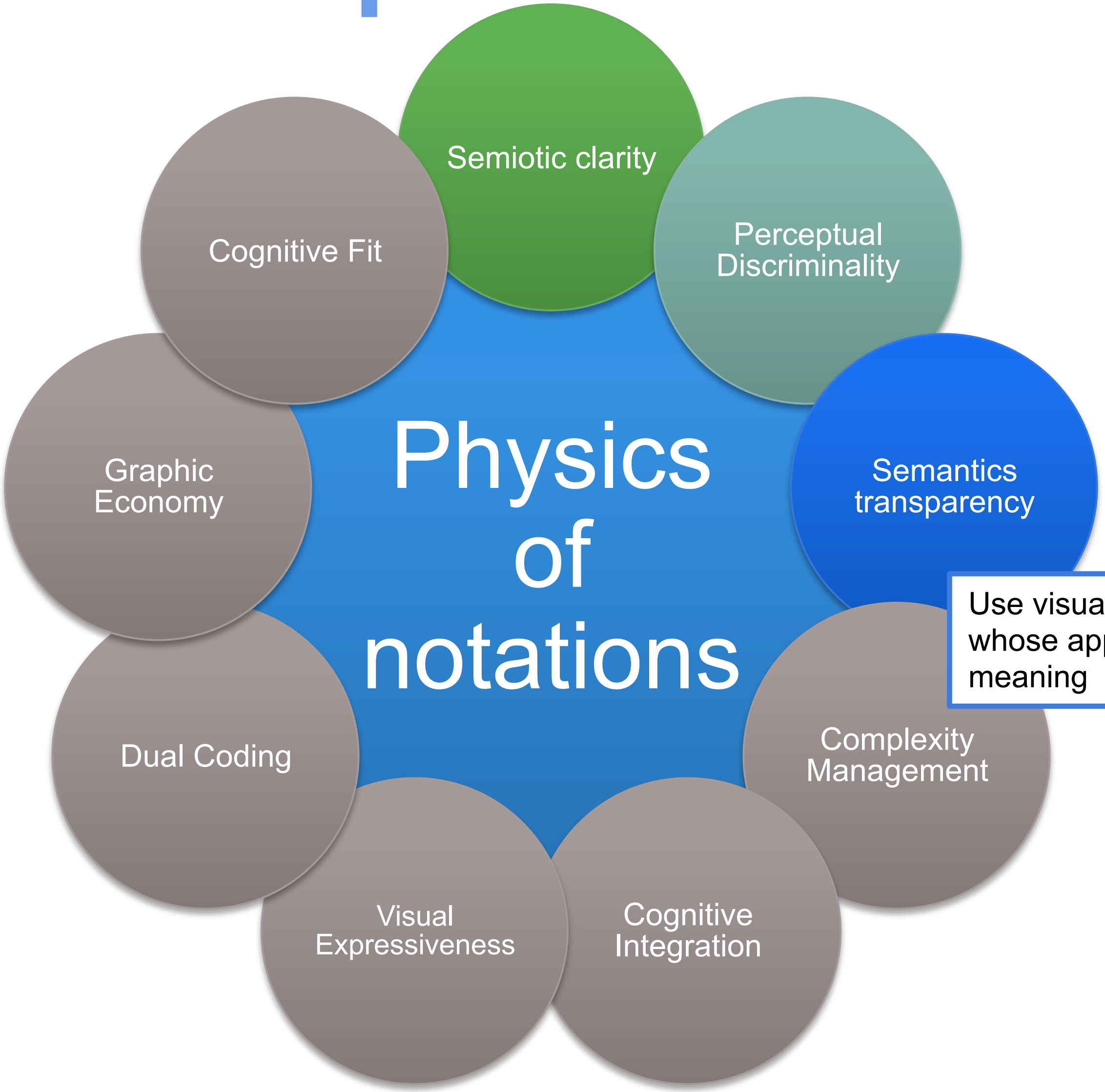
Different symbols should be clearly distinguishable from each other

Recommendations for Graphical DSLs



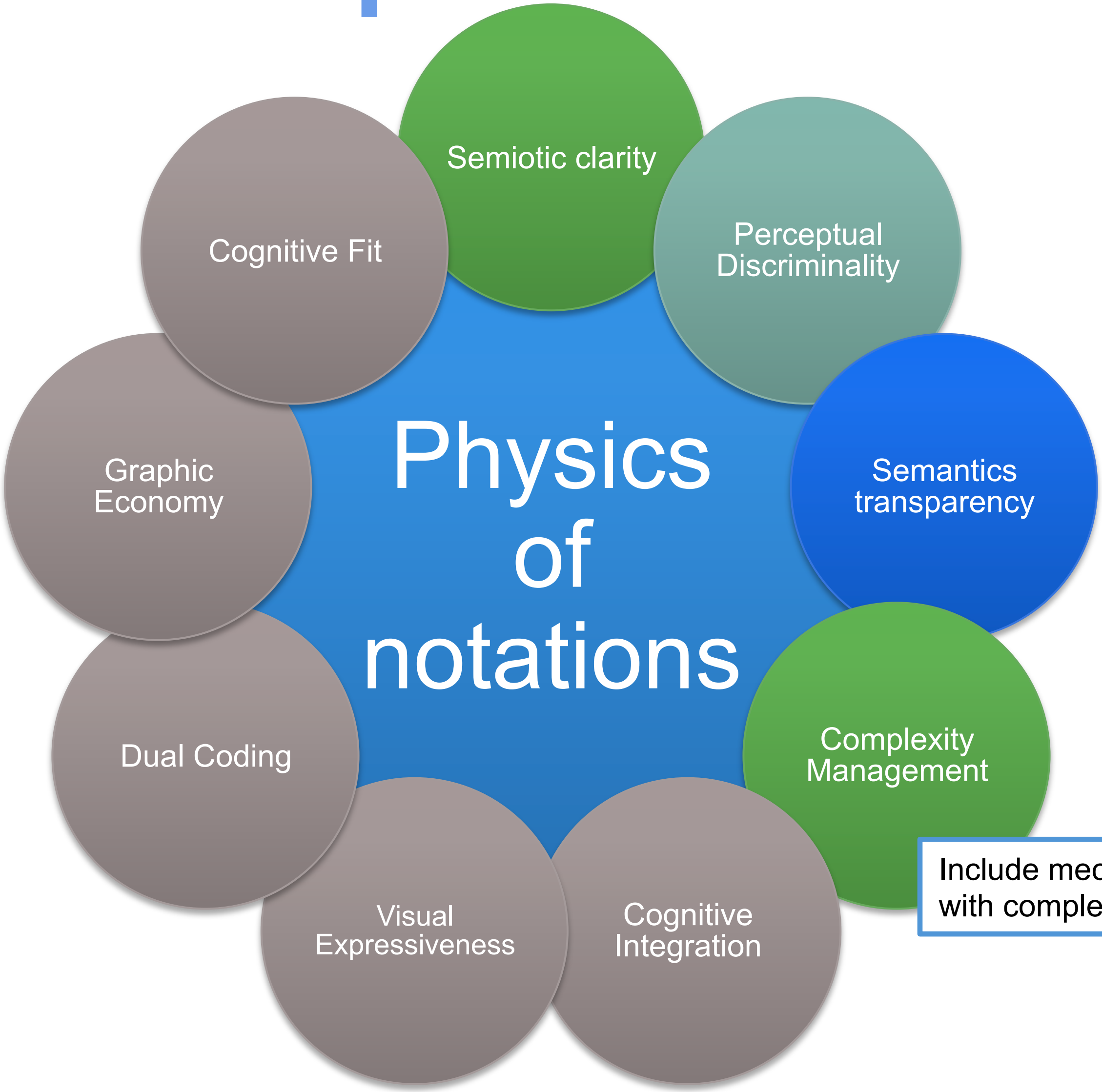
Aggregation	Association (navigable)	Association (non-navigable)	Association class relationship	Composition
Constraint	Dependency	Generalisation	Generalisation set	Interface (provided)
Interface (required)	N-ary association	Note reference	Package containment	Package import (public)
				«import»
Package import (private)	Package merge	Realisation	Substitution	Usage
«access»	«merge»		«substitute»	«use»

Recommendations for Graphical DSLs



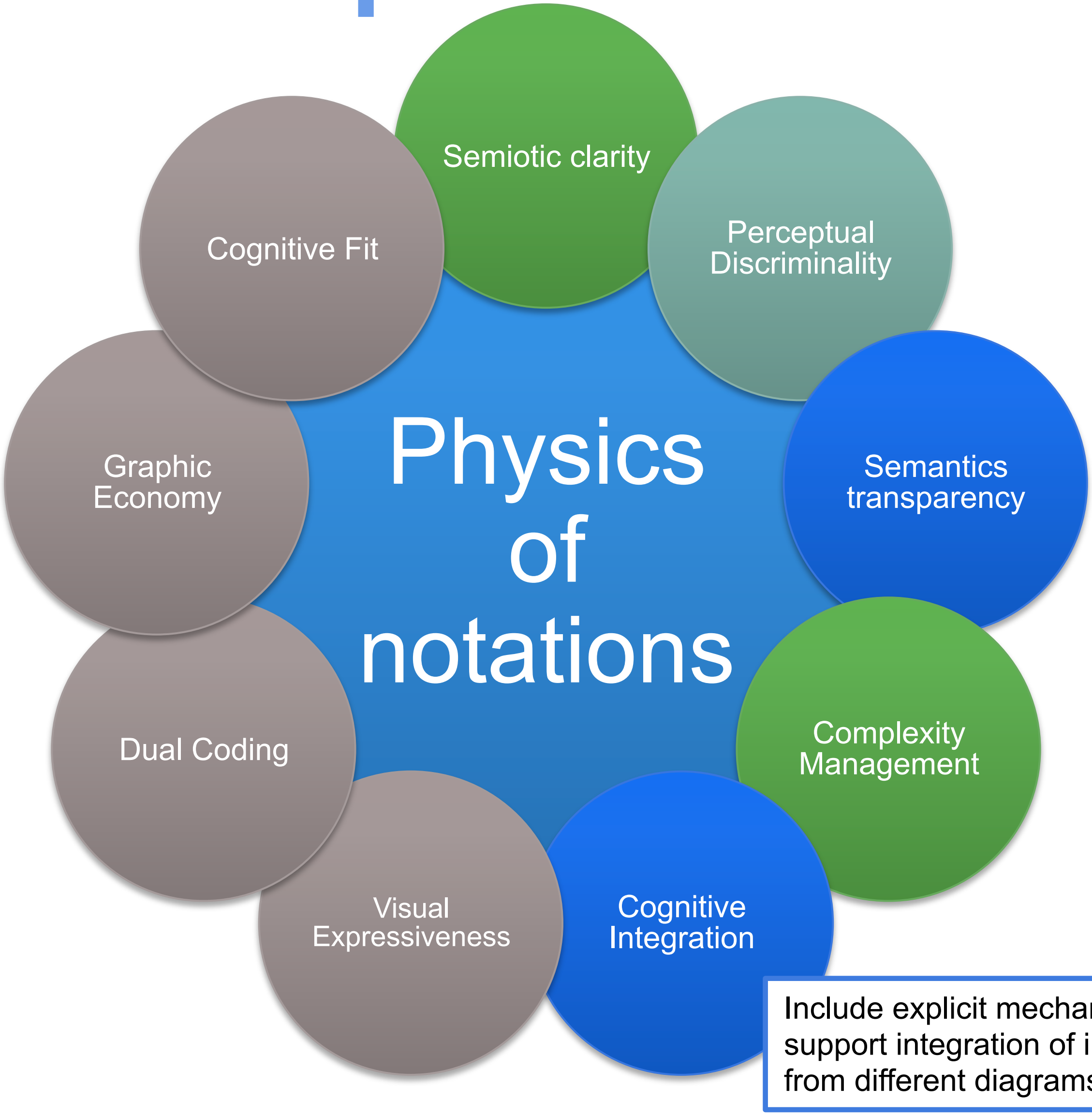
Use visual representations whose apprarance suggests their meaning

Recommendations for Graphical DSLs



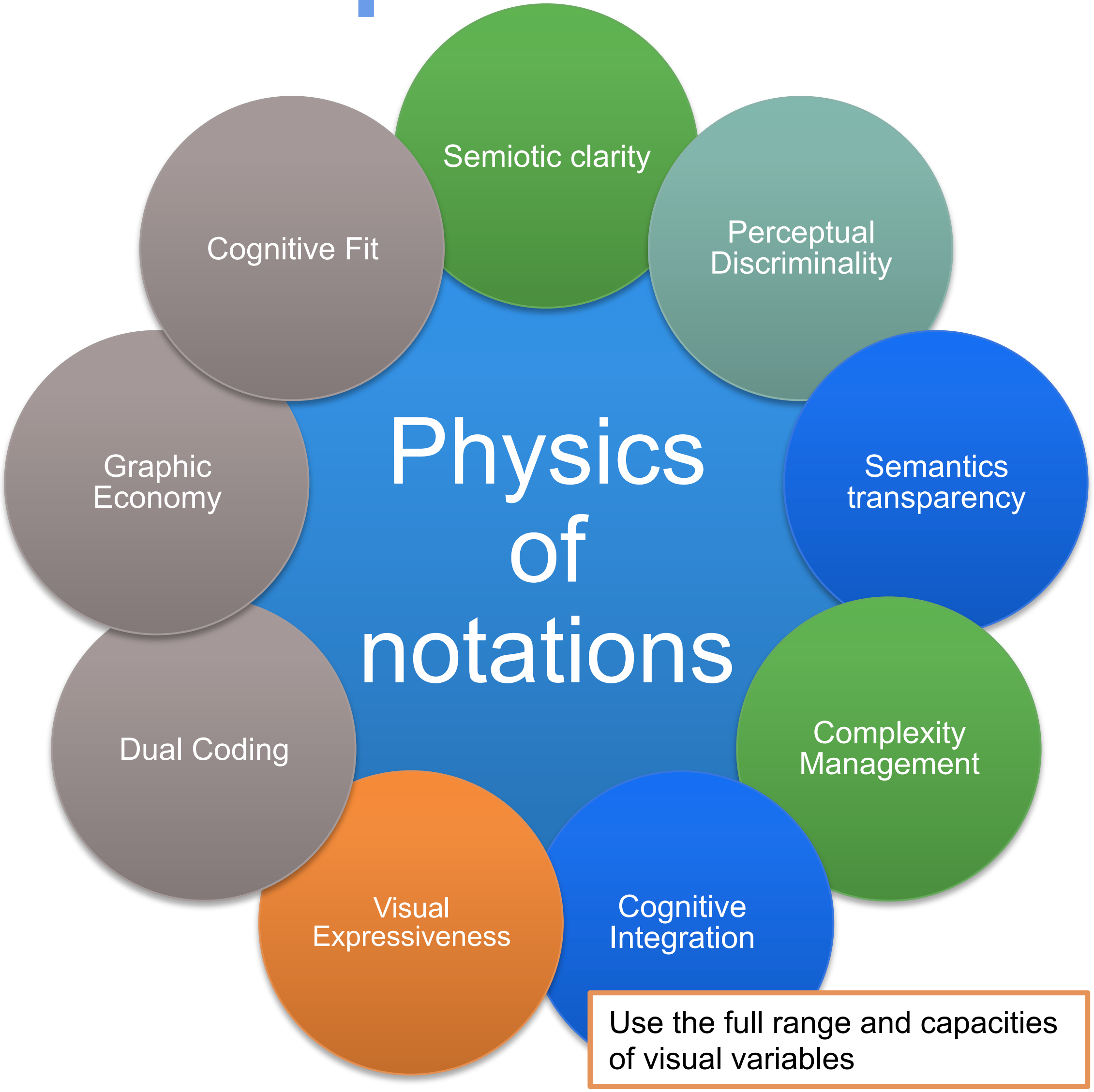
Include mechanisms for dealing with complexity

Recommendations for Graphical DSLs



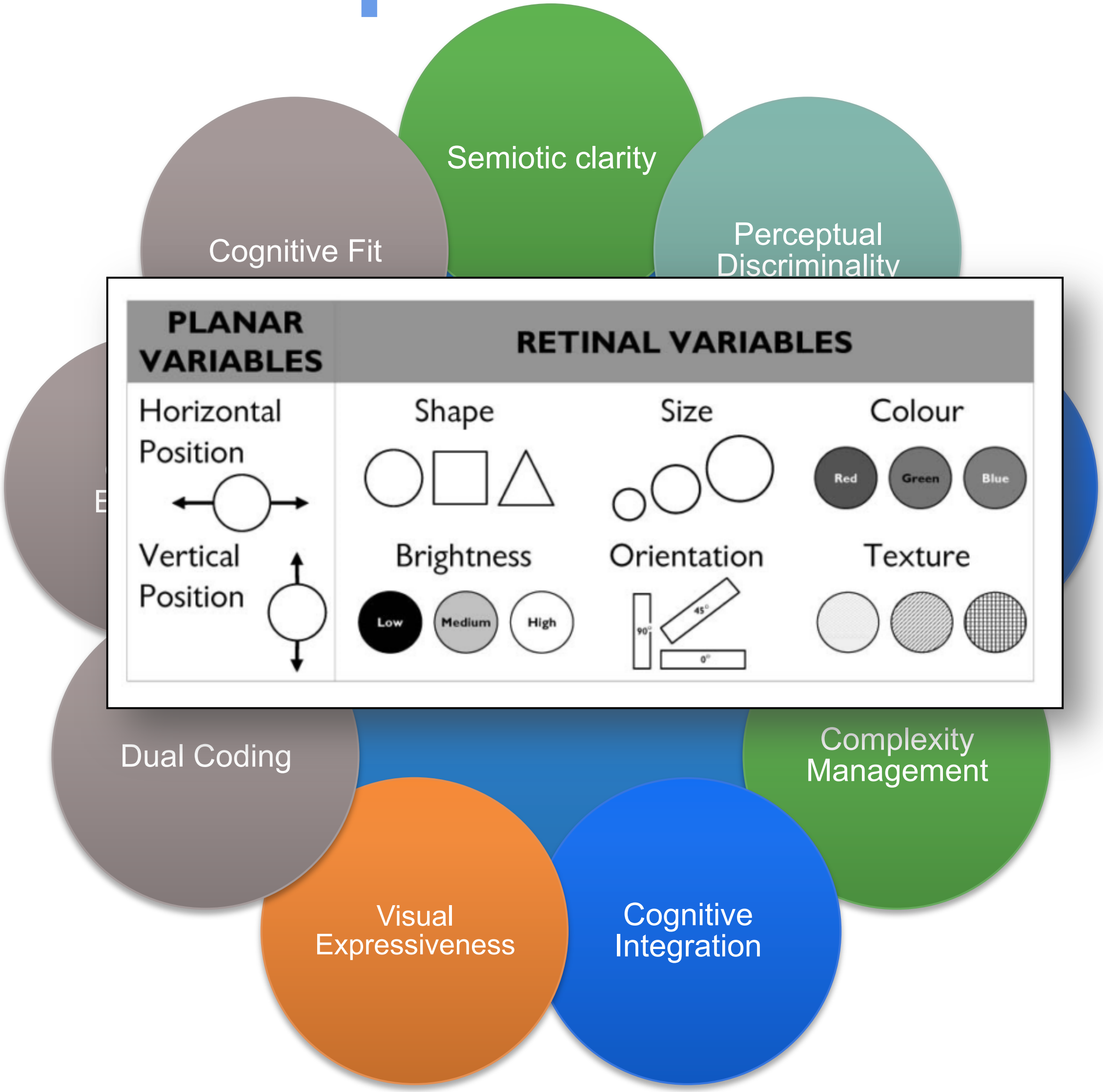
Include explicit mechanisms to support integration of information from different diagrams

Recommendations for Graphical DSLs



Use the full range and capacities of visual variables

Recommendations for Graphical DSLs



Semiotic clarity

Cognitive Fit

Perceptual Discriminality

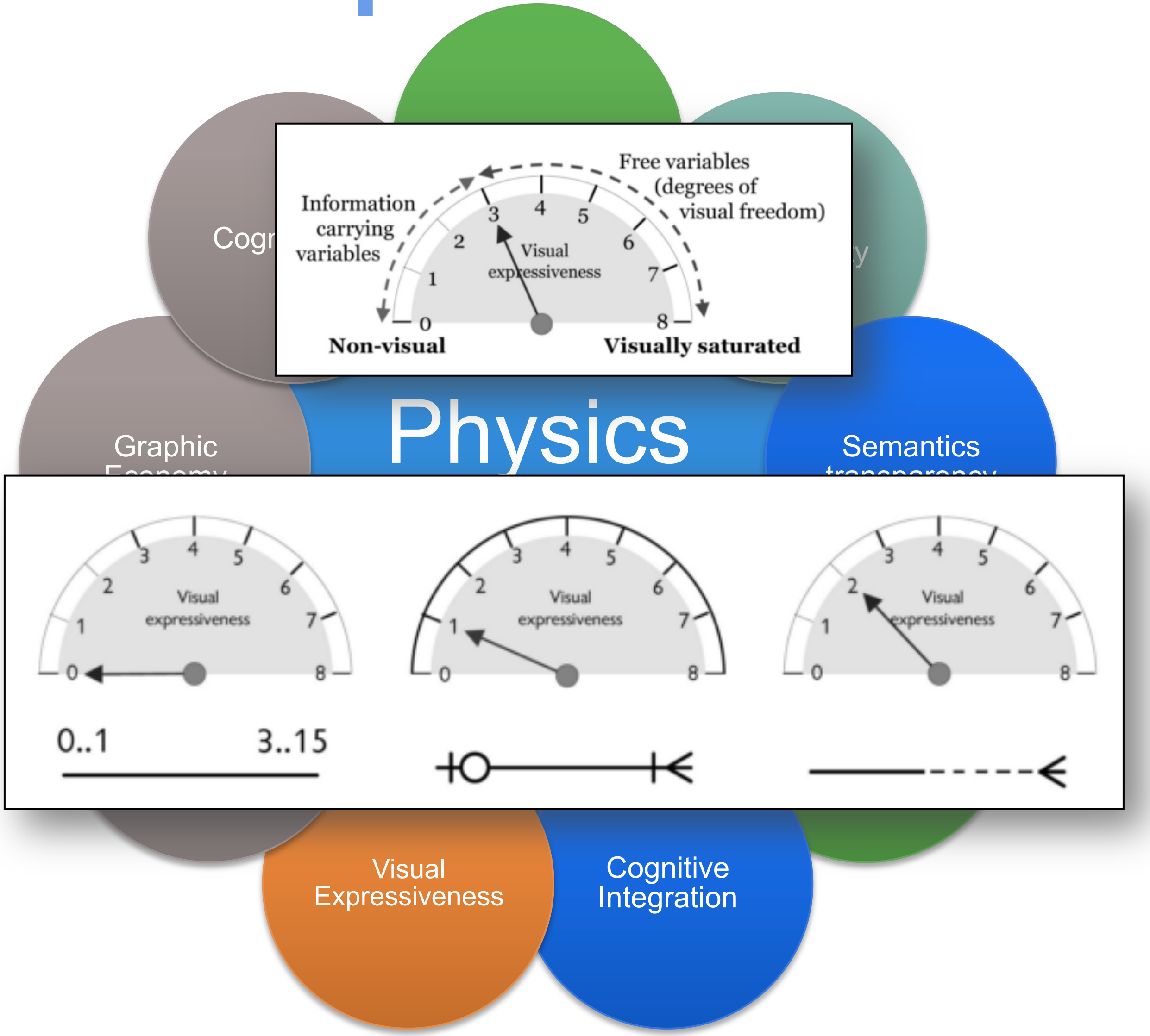
Dual Coding

Complexity Management

Visual Expressiveness

Cognitive Integration

Recommendations for Graphical DSLs



Cogr

y

Graphic Economy

Physics

Semantics transparency

Visual Expressiveness

Cognitive Integration

Recommendations for Graphical DSLs

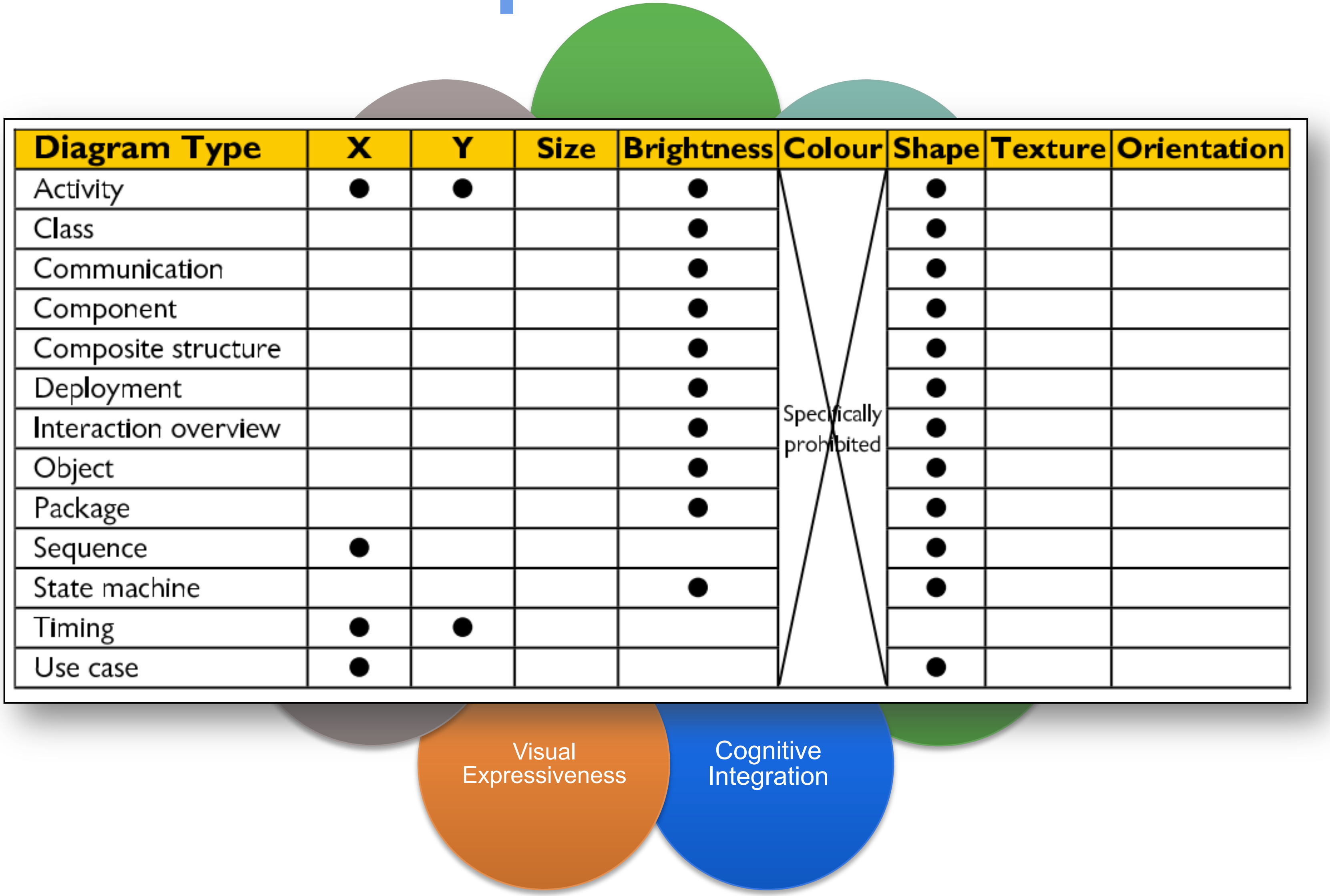
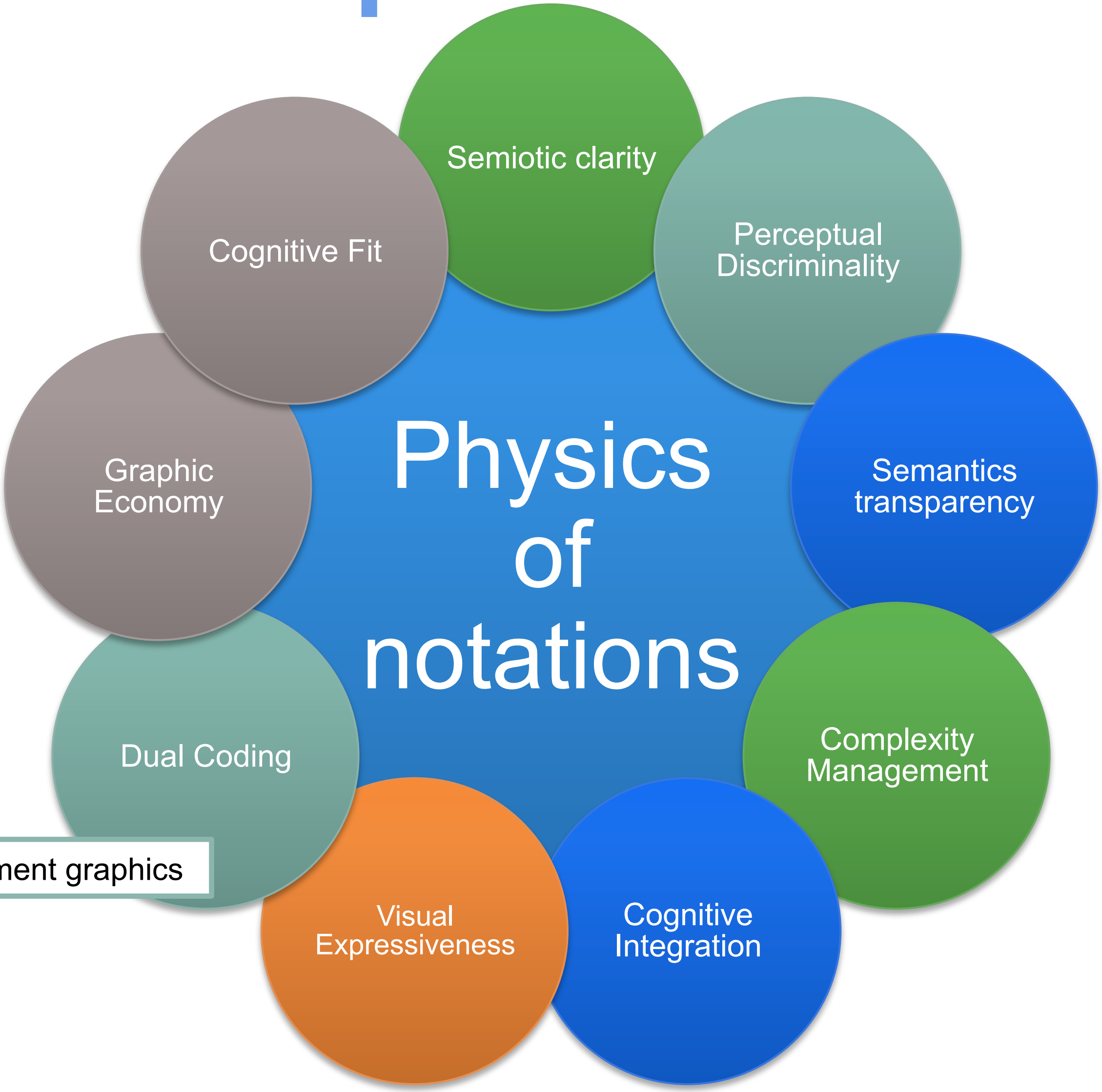


Diagram Type	X	Y	Size	Brightness	Colour	Shape	Texture	Orientation
Activity	●	●		●	Specifically prohibited	●		
Class				●		●		
Communication				●		●		
Component				●		●		
Composite structure				●		●		
Deployment				●		●		
Interaction overview				●		●		
Object				●		●		
Package				●		●		
Sequence	●					●		
State machine				●		●		
Timing	●	●						
Use case	●					●		

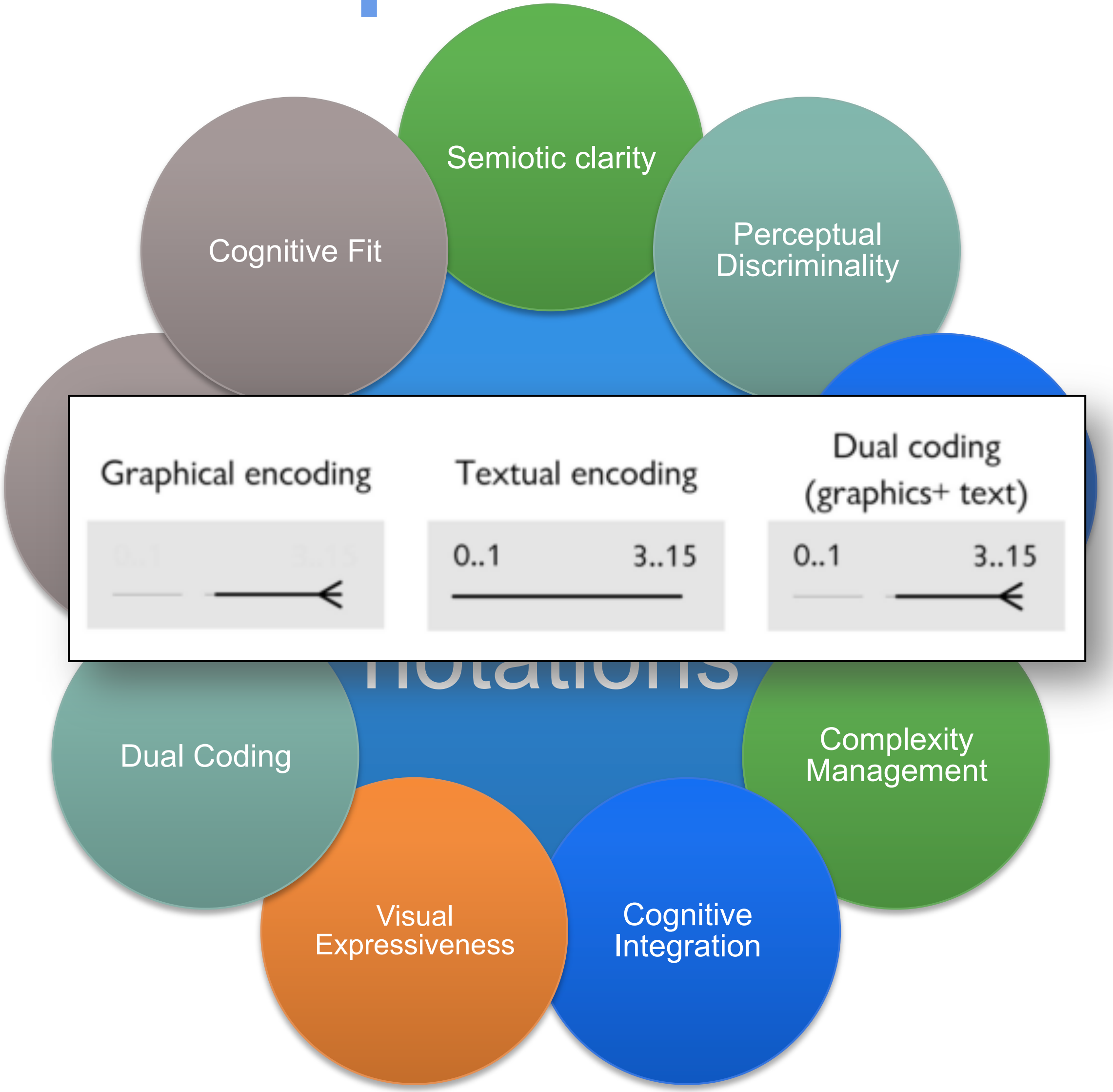
Visual Expressiveness
Cognitive Integration

Recommendations for Graphical DSLs

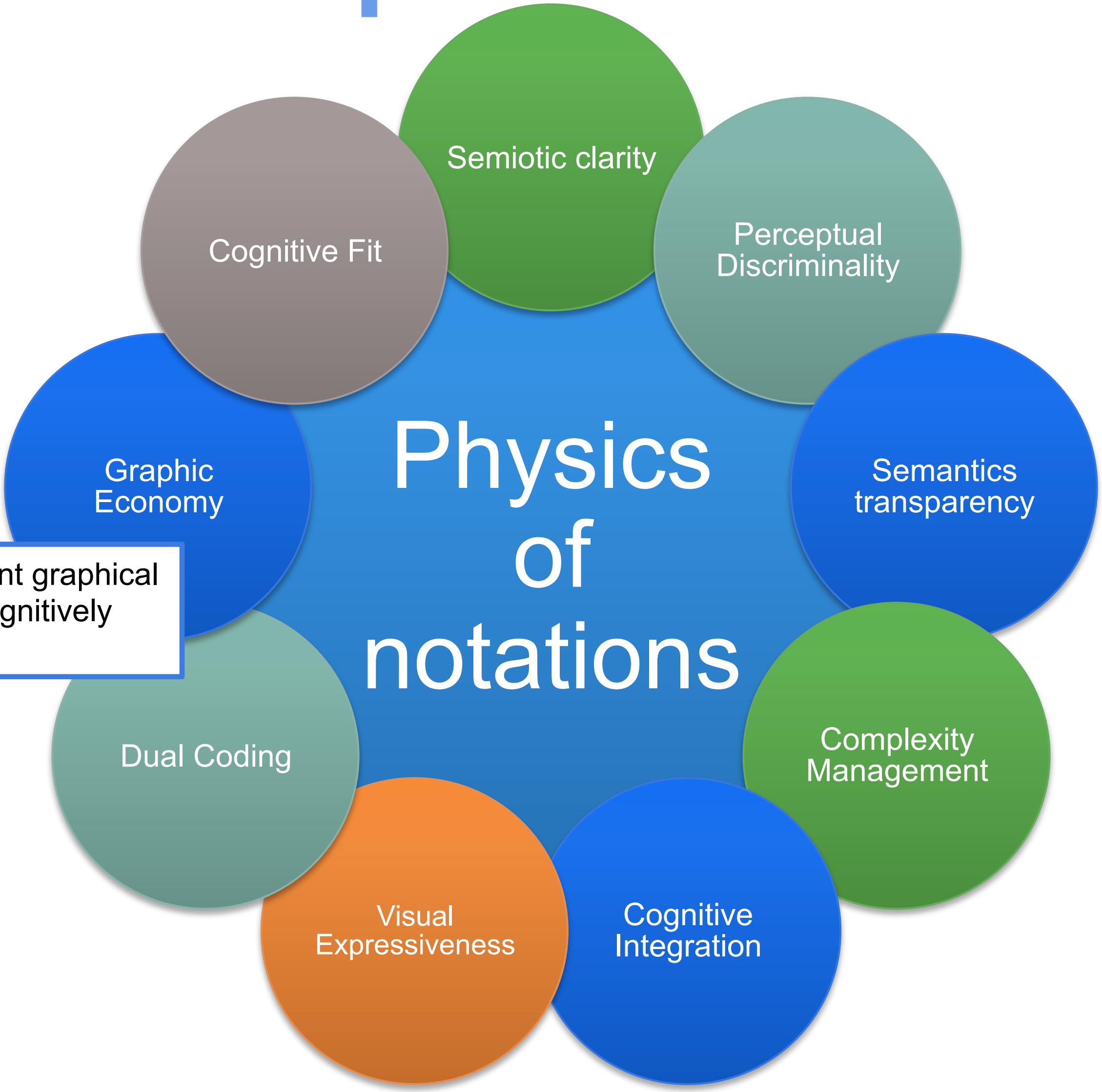


Use text to complement graphics

Recommendations for Graphical DSLs



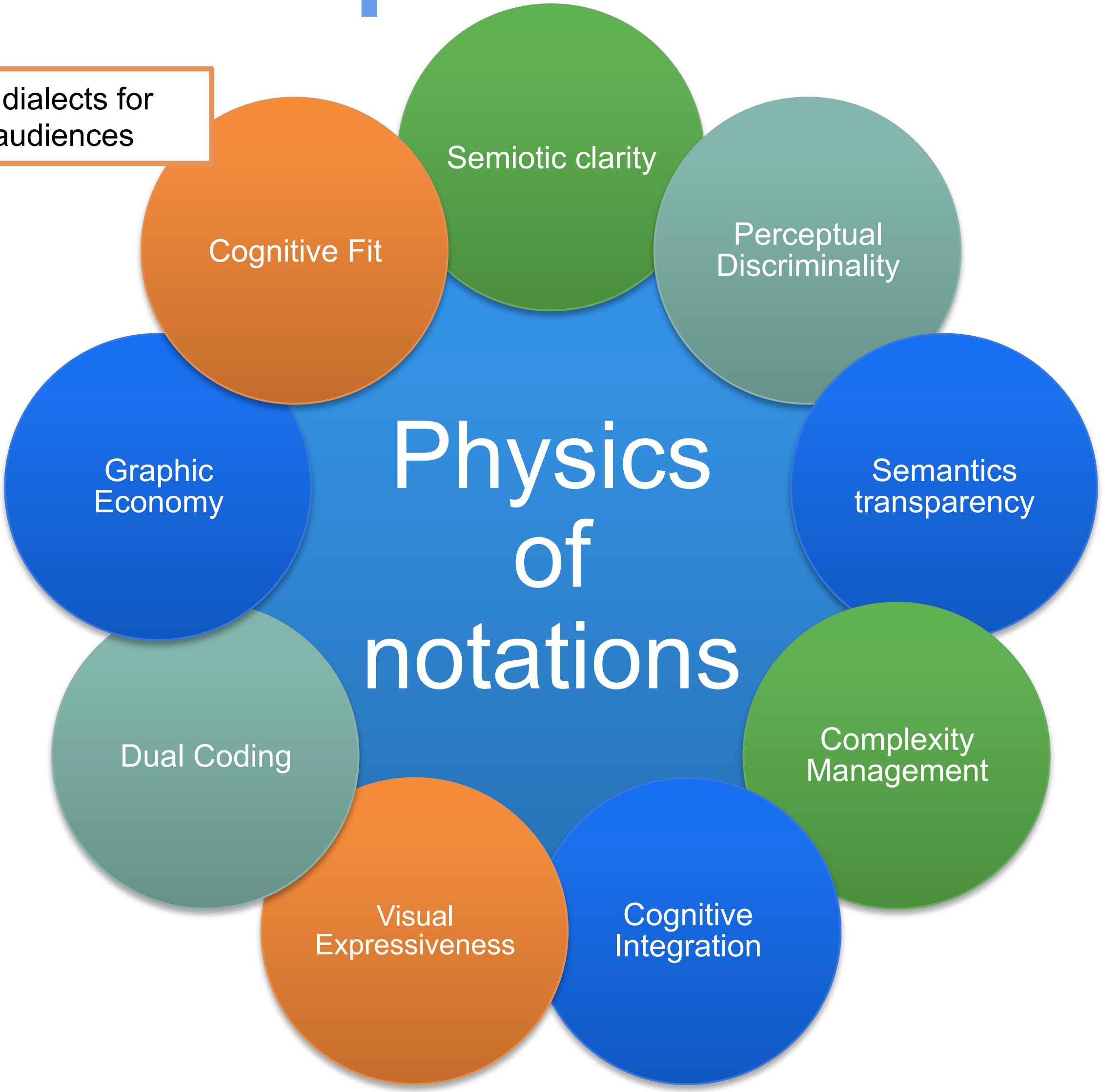
Recommendations for Graphical DSLs



The number of different graphical symbols should be cognitively manageable

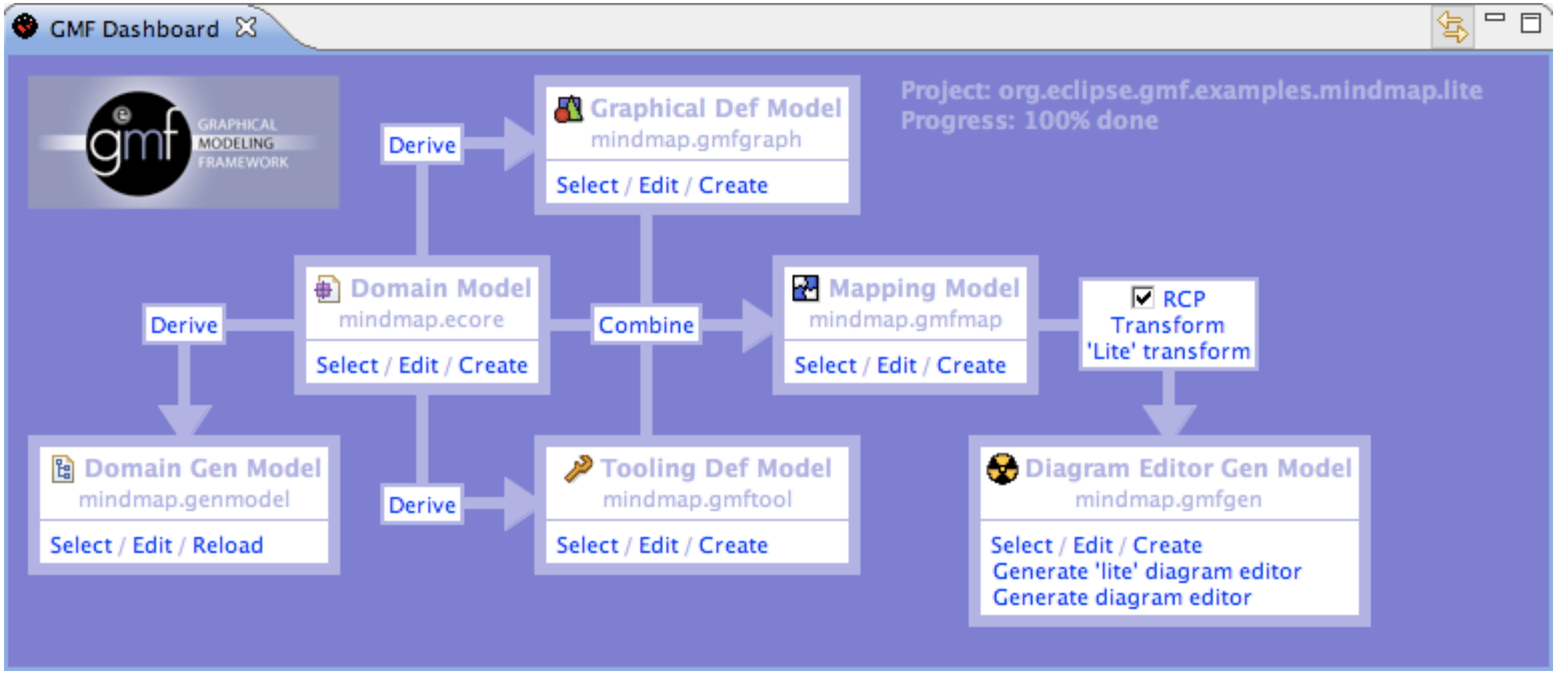
Recommendations for Graphical DSLs

Use different visual dialects for different tasks and audiences



Graphical Modeling Framework (GMF)

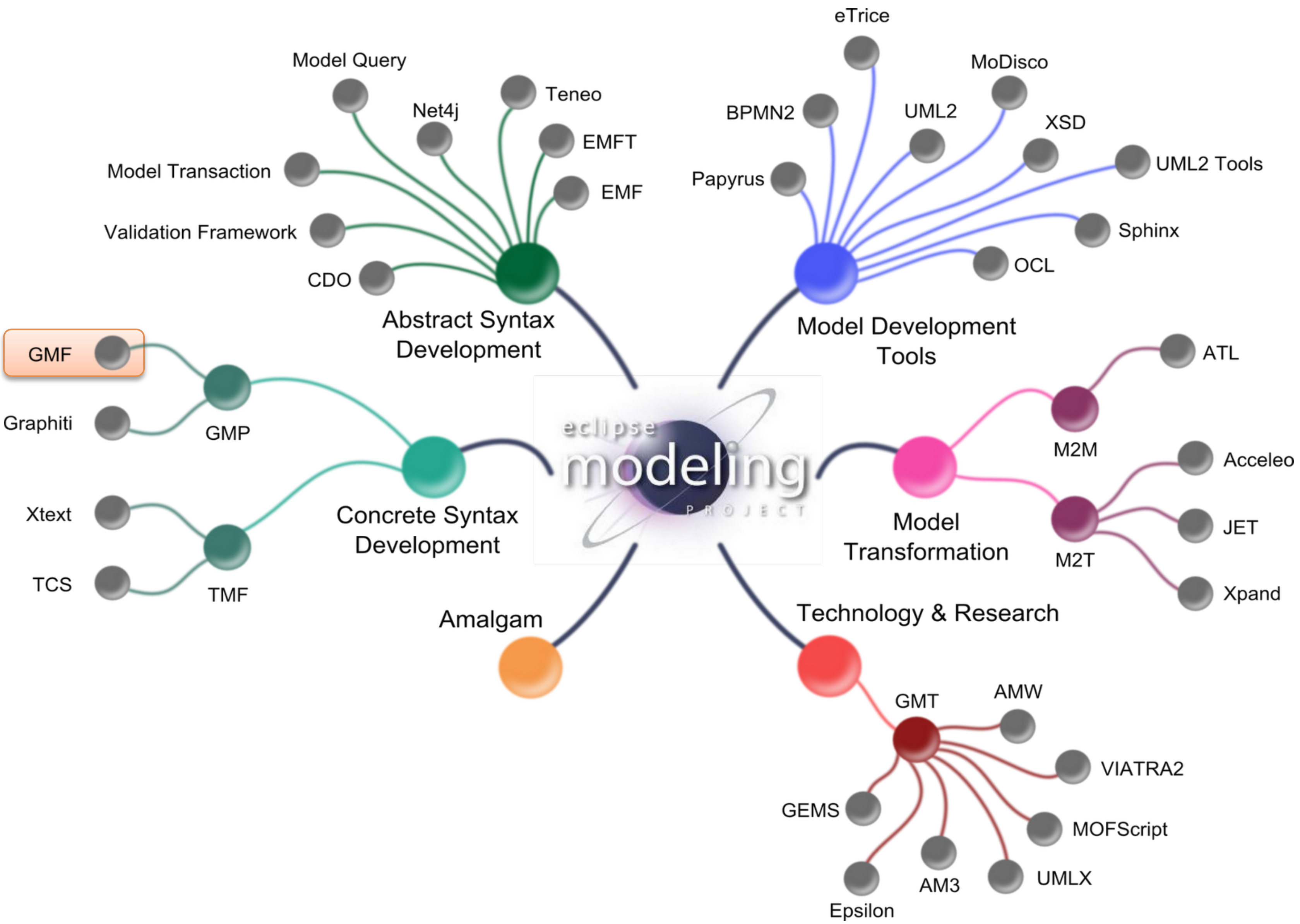
- Model-Driven Framework to develop graphical editors based on EMF and GEF
- GMF is part of Eclipse Modeling Project
- Provides a generative component to create the DSL tooling
- Provides a runtime infrastructure to facilitate the development of graphical DSLs



GMF

- Eclipse project
  - Eclipse Modelling components
  - Uses
    - EMF (Eclipse Modeling Framework)
    - GEF (Graphical Editing Framework)
- Model-driven framework for Graphical DSLs
  - Everything is a model
- DSL definition easy, tweaking hard

Eclipse Modeling Project



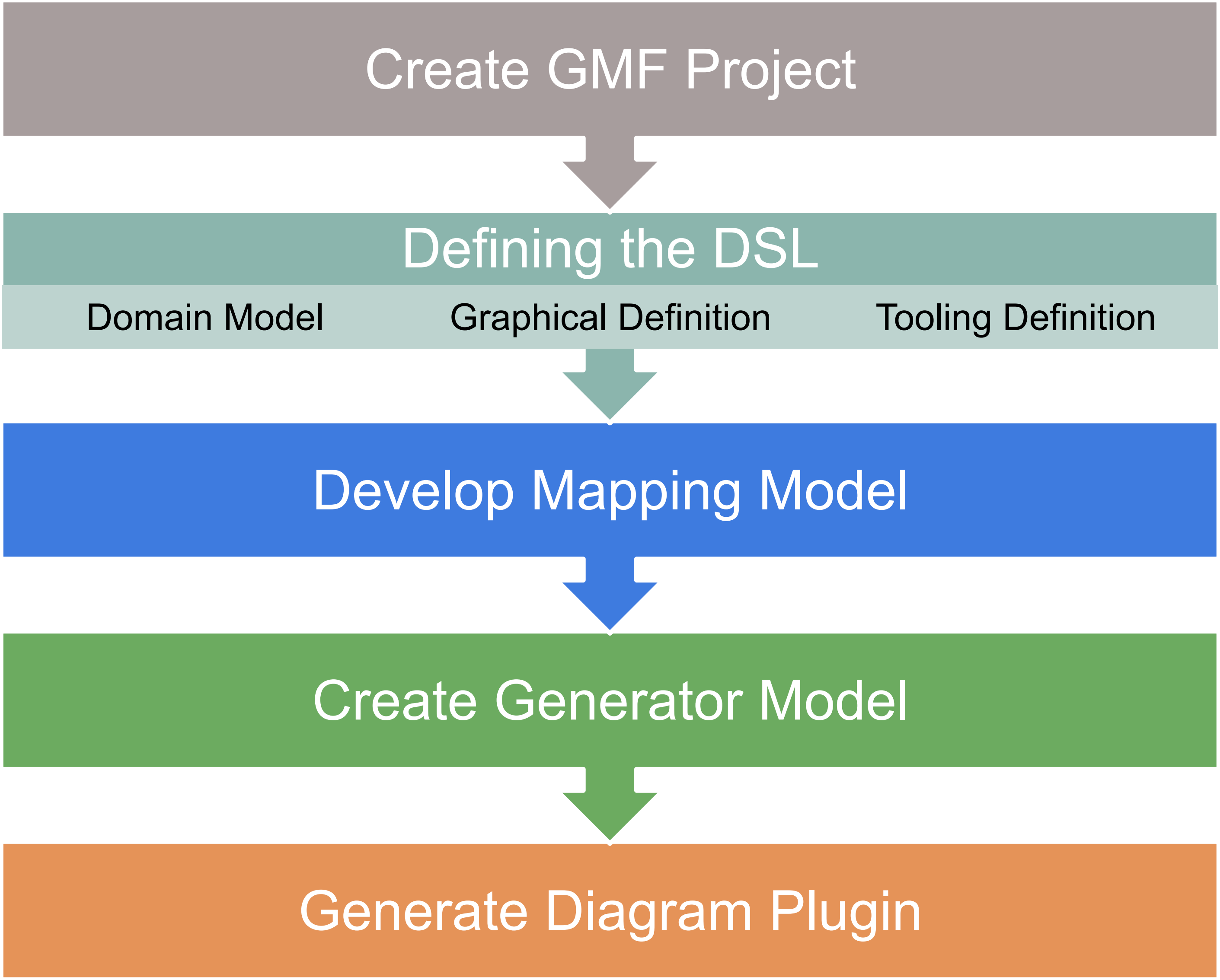
GMF features

- Tooling
  - Editors for notation, semantic and tooling
  - GMF Dashboard
  - Generator to produce the DSL implementation
- Runtime
  - Generated DSLs depend on the GMF Runtime to produce an extensible graphical editor

Main Advantages

- Consistent look and feel
- Diagram persistence
- Open editors can be extended by third-parties
- Already integrated with various Eclipse components
- Extensible notation metamodel to enable the isolation of notation from semantic concerns
- Future community enhancements will easily be integrated

Development Process



Create GMF Project

Defining the DSL

Domain Model

Graphical Definition

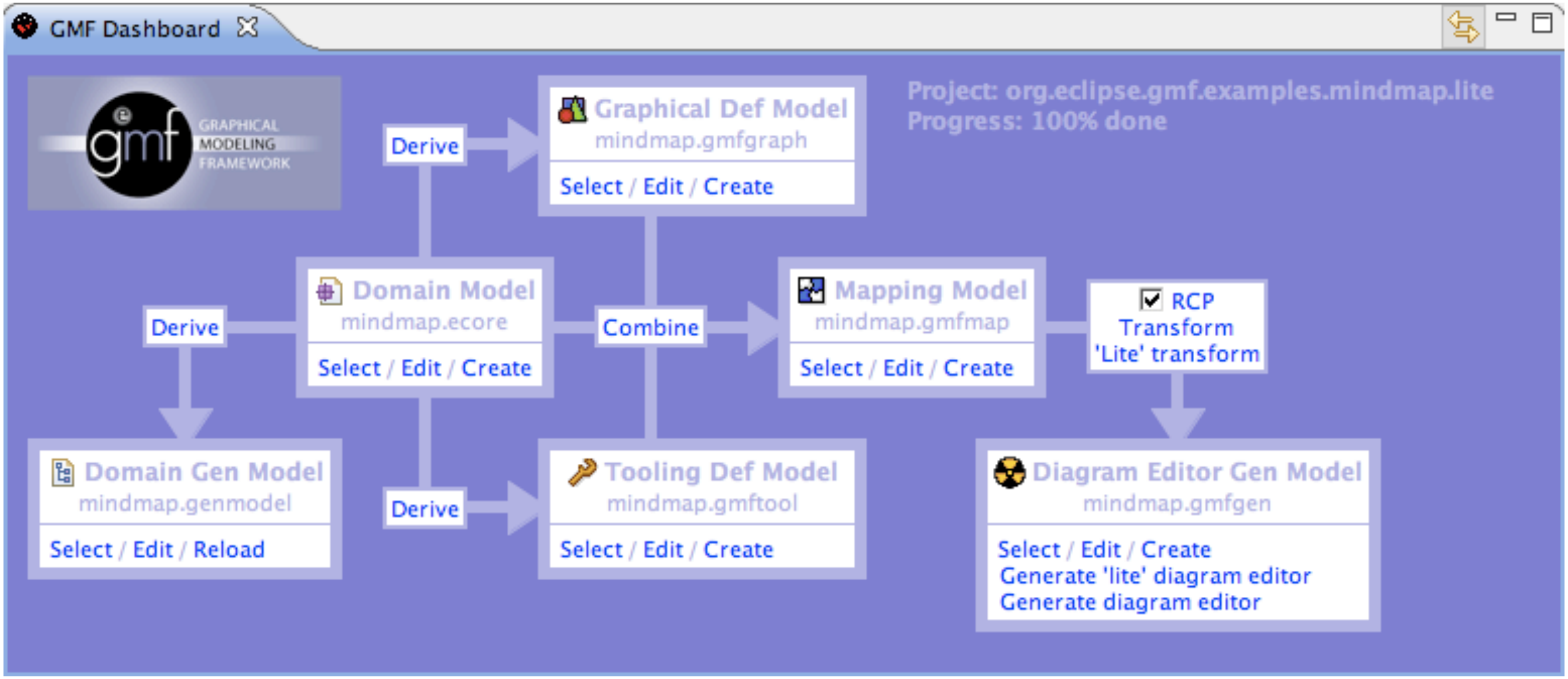
Tooling Definition

Develop Mapping Model

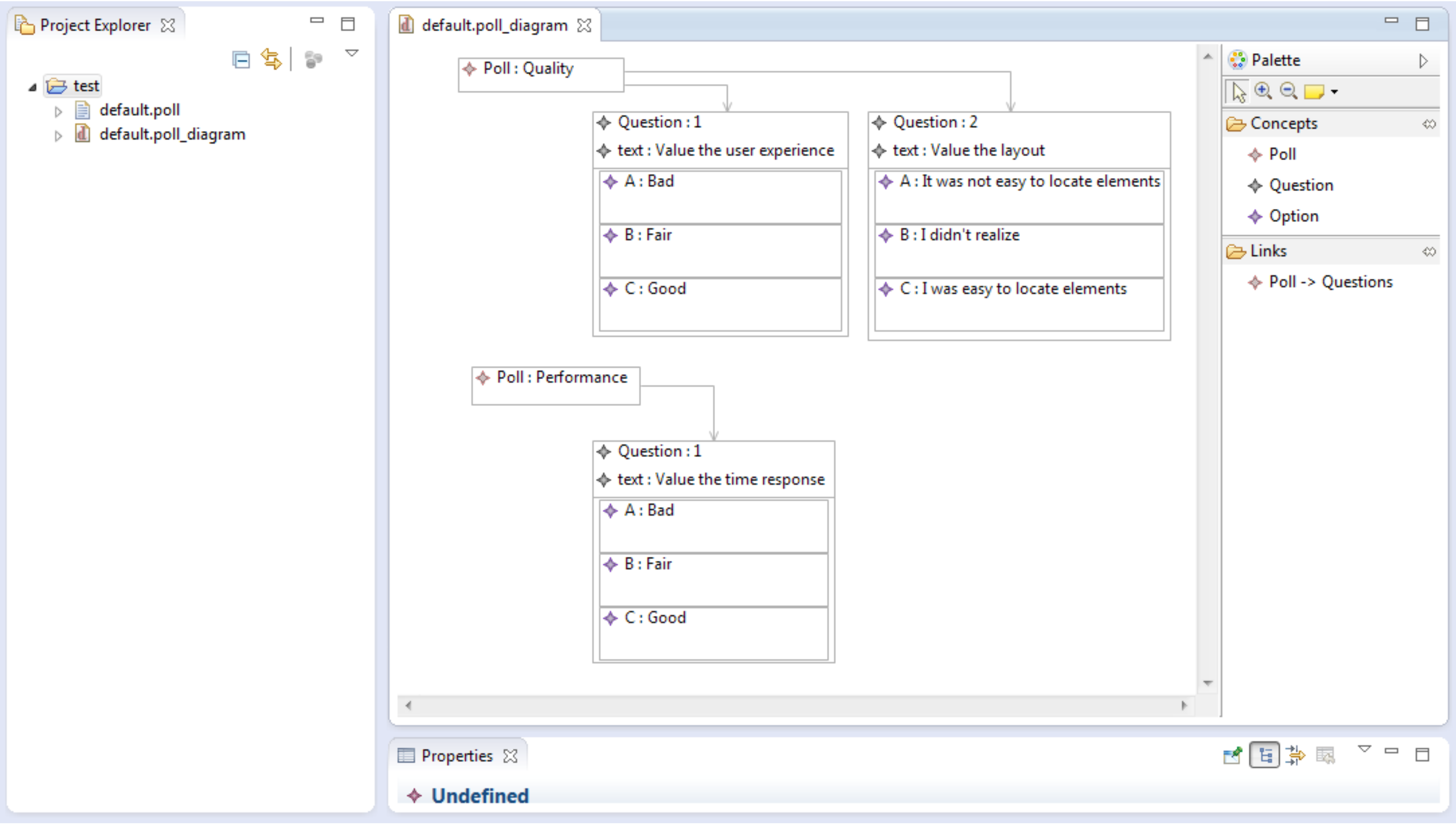
Create Generator Model

Generate Diagram Plugin

Development Process



Example (Graphical Notation)



Poll : Quality

Question : 1
text : Value the user experience
A : Bad
B : Fair
C : Good

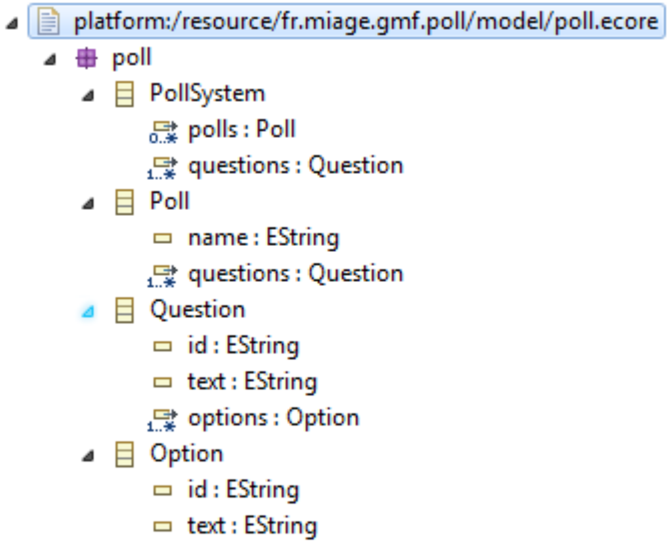
Question : 2
text : Value the layout
A : It was not easy to locate elements
B : I didn't realize
C : I was easy to locate elements

Poll : Performance

Question : 1
text : Value the time response
A : Bad
B : Fair
C : Good

Poll System Metamodel

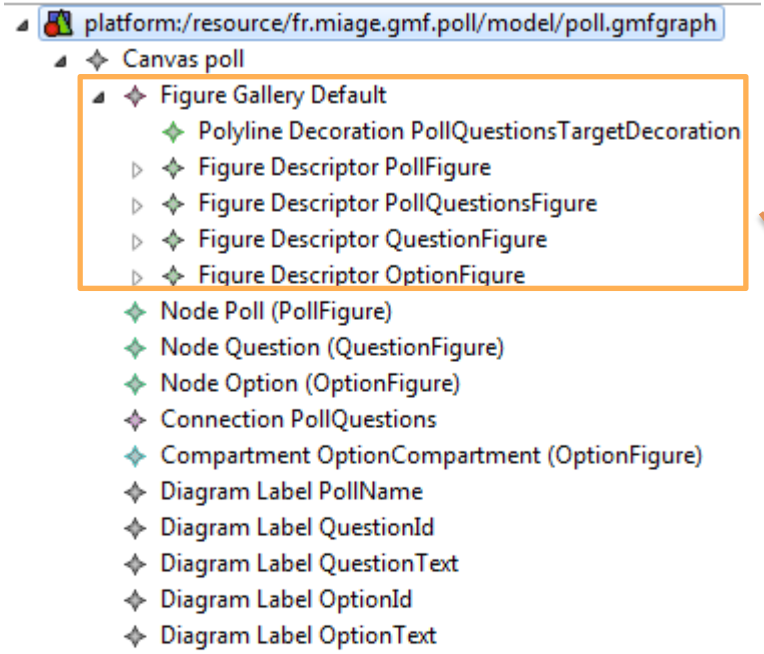
- Concepts
  - PollSystem
  - Poll
  - Question
  - Option
- Attributes
  - A Poll has a name
  - A Question has an identifier and a descriptive text
  - An Option has an identifier and a descriptive text
- Relationships
  - PollSystem is composed of polls and questions
  - Question has a set of options



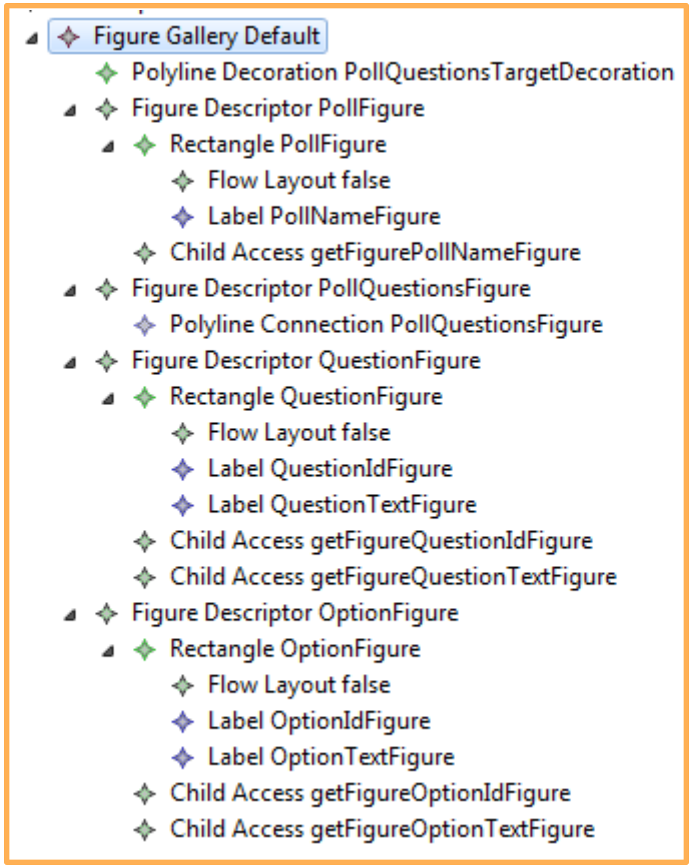
```
platform:/resource/fr.miage.gmf.poll/model/poll.ecore
  poll
    PollSystem
      polls : Poll
      questions : Question
    Poll
      name : EString
      questions : Question
    Question
      id : EString
      text : EString
      options : Option
    Option
      id : EString
      text : EString
```

Graphical Definition

- A model will represent a PollSystem
- A Poll will be a node
- A Question will be a rectangular node
- An Option will be a rectangular node included in the Question node



- platform:/resource/fr.miage.gmf.poll/model/poll.gmfgraph
  - Canvas poll
    - Figure Gallery Default
      - Polyline Decoration PollQuestionsTargetDecoration
      - Figure Descriptor PollFigure
      - Figure Descriptor PollQuestionsFigure
      - Figure Descriptor QuestionFigure
      - Figure Descriptor OptionFigure
    - Node Poll (PollFigure)
    - Node Question (QuestionFigure)
    - Node Option (OptionFigure)
    - Connection PollQuestions
    - Compartment OptionCompartment (OptionFigure)
    - Diagram Label PollName
    - Diagram Label QuestionId
    - Diagram Label QuestionText
    - Diagram Label OptionId
    - Diagram Label OptionText



- Figure Gallery Default
  - Polyline Decoration PollQuestionsTargetDecoration
  - Figure Descriptor PollFigure
    - Rectangle PollFigure
      - Flow Layout false
      - Label PollNameFigure
    - Child Access getFigurePollNameFigure
  - Figure Descriptor PollQuestionsFigure
    - Polyline Connection PollQuestionsFigure
  - Figure Descriptor QuestionFigure
    - Rectangle QuestionFigure
      - Flow Layout false
      - Label QuestionIdFigure
      - Label QuestionTextFigure
    - Child Access getFigureQuestionIdFigure
    - Child Access getFigureQuestionTextFigure
  - Figure Descriptor OptionFigure
    - Rectangle OptionFigure
      - Flow Layout false
      - Label OptionIdFigure
      - Label OptionTextFigure
    - Child Access getFigureOptionIdFigure
    - Child Access getFigureOptionTextFigure

Plan

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

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)

- Models and Languages
  - Perhaps a more concrete way to see models, metamodels and MDE (IDM in french)

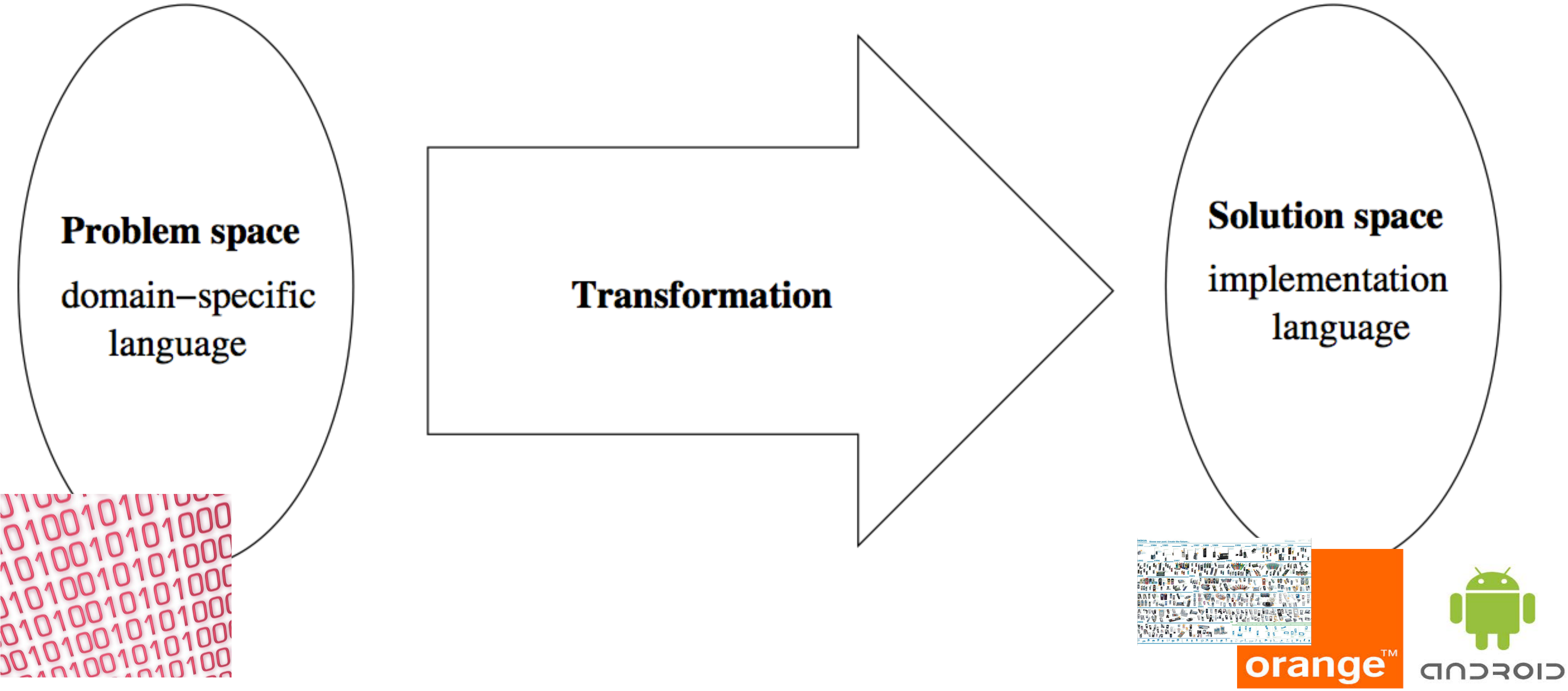
DSL,

Model,

Metamodel,

Summary

Abstraction Gap



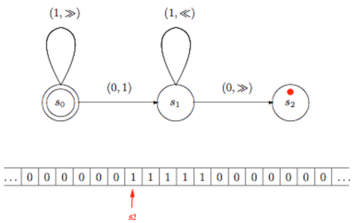
Problem space

domain–specific language

Transformation

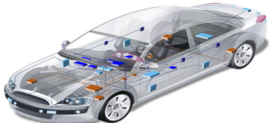
Solution space

implementation language



Google

twitter



Models/MDE

- In essence, a model is an **abstraction** of some aspect of a system under study.
- Some details are hidden or removed to **simplify** and focus attention.
- A model is an abstraction since **general** concepts can be formulated by abstracting common properties of instances or by extracting common features from specific examples
- **(Domain-specific) Languages** enable the specification or execution of models

Generative approach

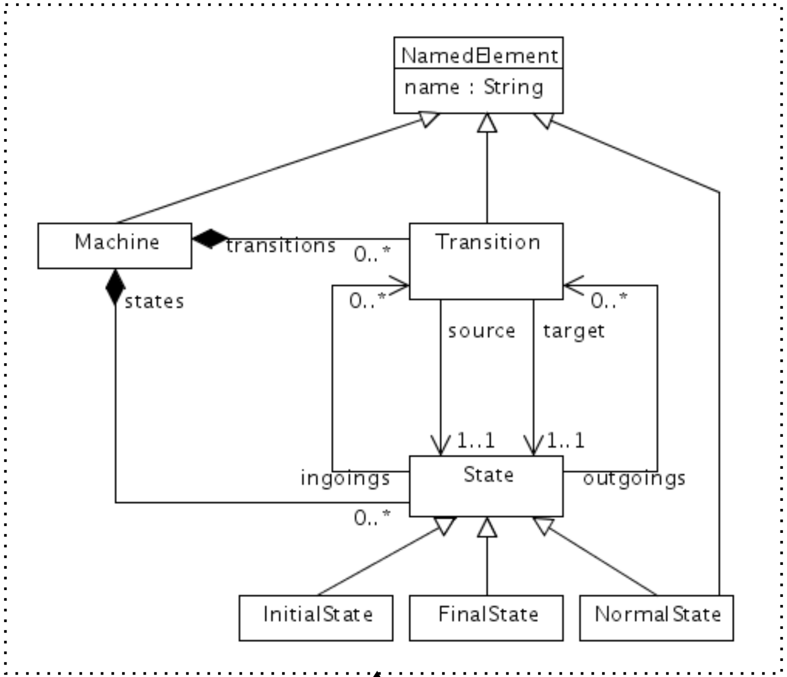
- Programming the generation of programs
  - Very old practice
  - Metaprogramming: generative language and target language are the same
    - Reflection capabilities

- Generalization of this idea:
  - from a specification written in one or more textual or graphical domain-specific languages
  - you generate customized variants

Grammar

```
machineDefinition:
  MACHINE OPEN_SEP stateList
  transitionList CLOSE_SEP;

stateList:
  state (COMMA state)*;

state:
  ID_STATE;

transitionList:
  transition (COMMA transition)*;

transition:
  ID_TRANSITION OPEN_SEP
  state state CLOSE_SEP;

MACHINE: ‘machine’;
OPEN_SEP: ‘{’;
CLOSE_SEP: ‘{’;
COMMA: ‘,’;
ID_STATE: ‘S’ ID;
ID_TRANSITION: ‘T’ (0..9)+;
ID: (a..zA..Z_) (a..zA..Z0..9)*;
```

MetaModel



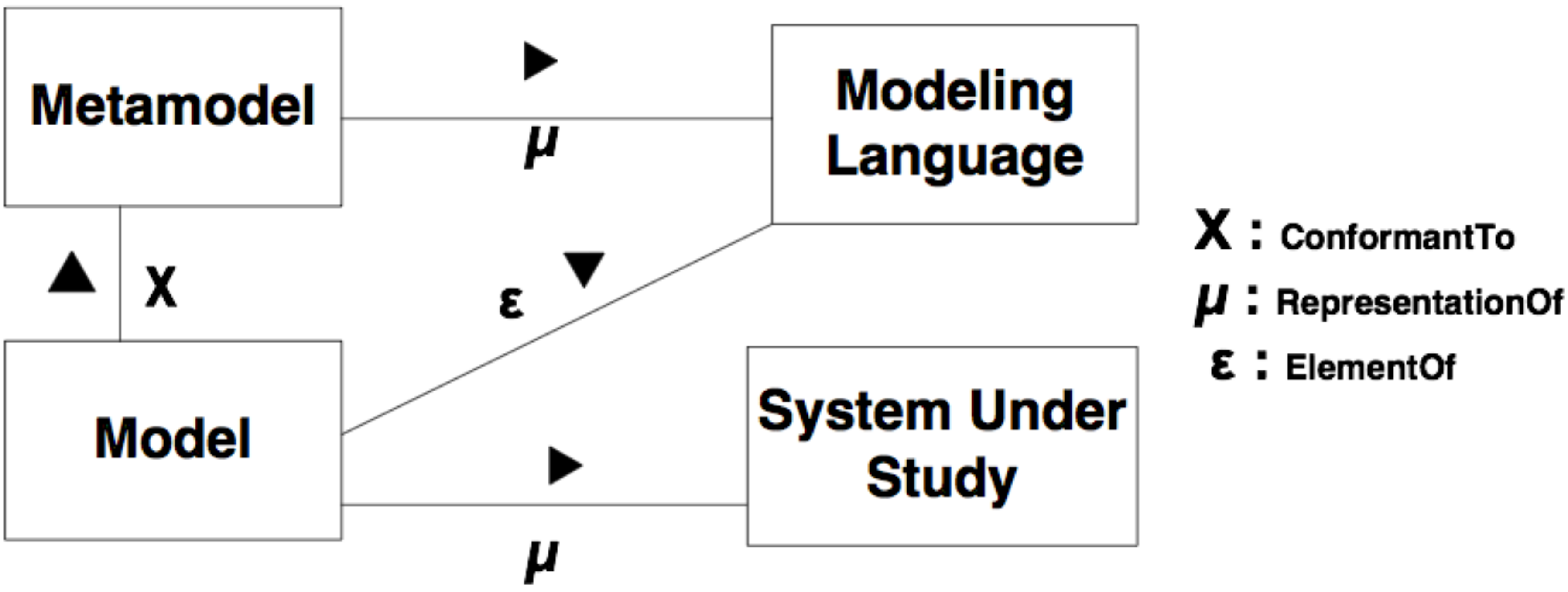
conforms To

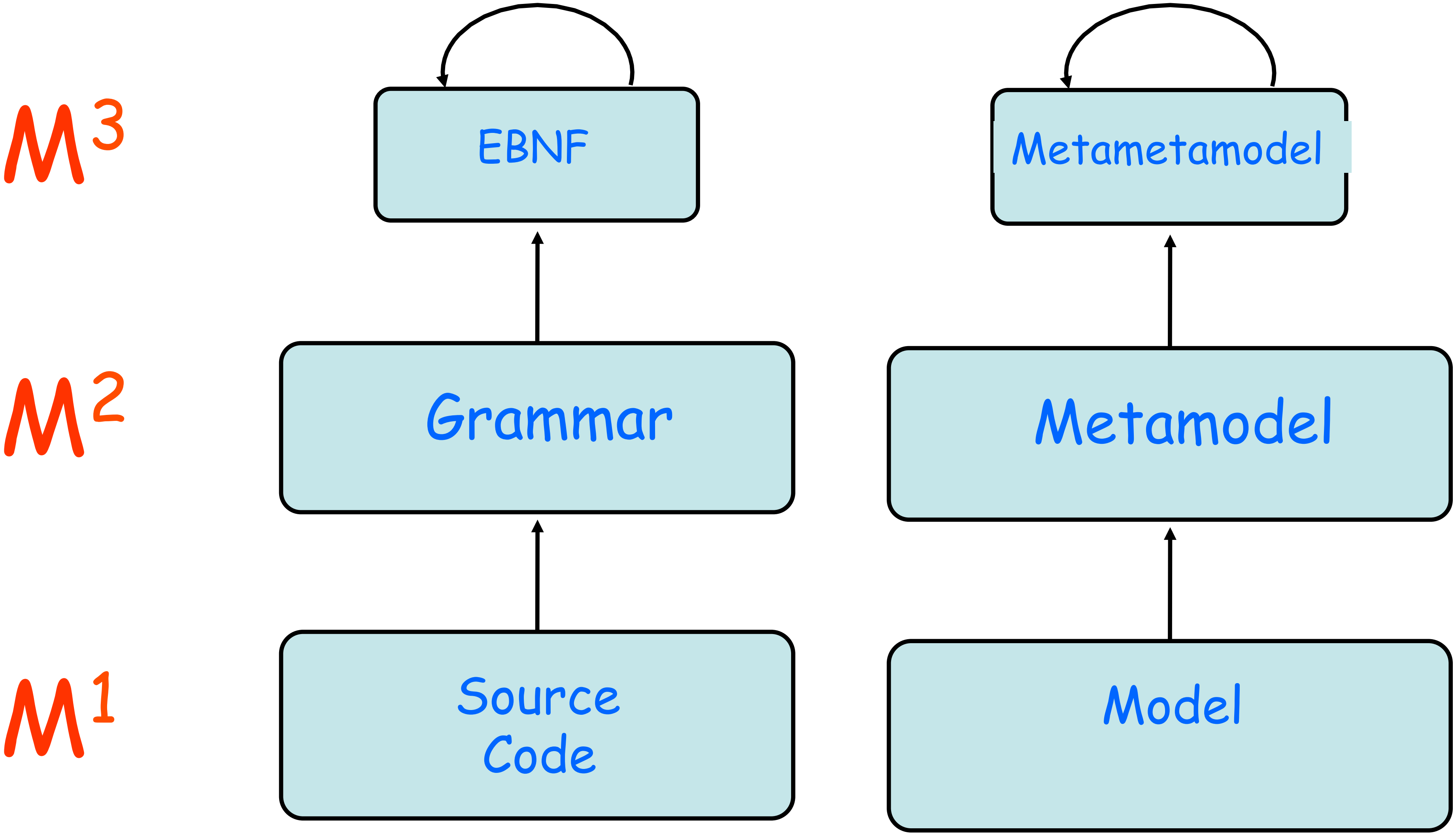
conforms To

```
machine {
  SOne STwo
  T1 { SOne STwo }
}
```

Source Code/Mode

Model, Metamodel, Metametamodel, DSML





M^3

EBNF

Metametamodel

M^2

Grammar

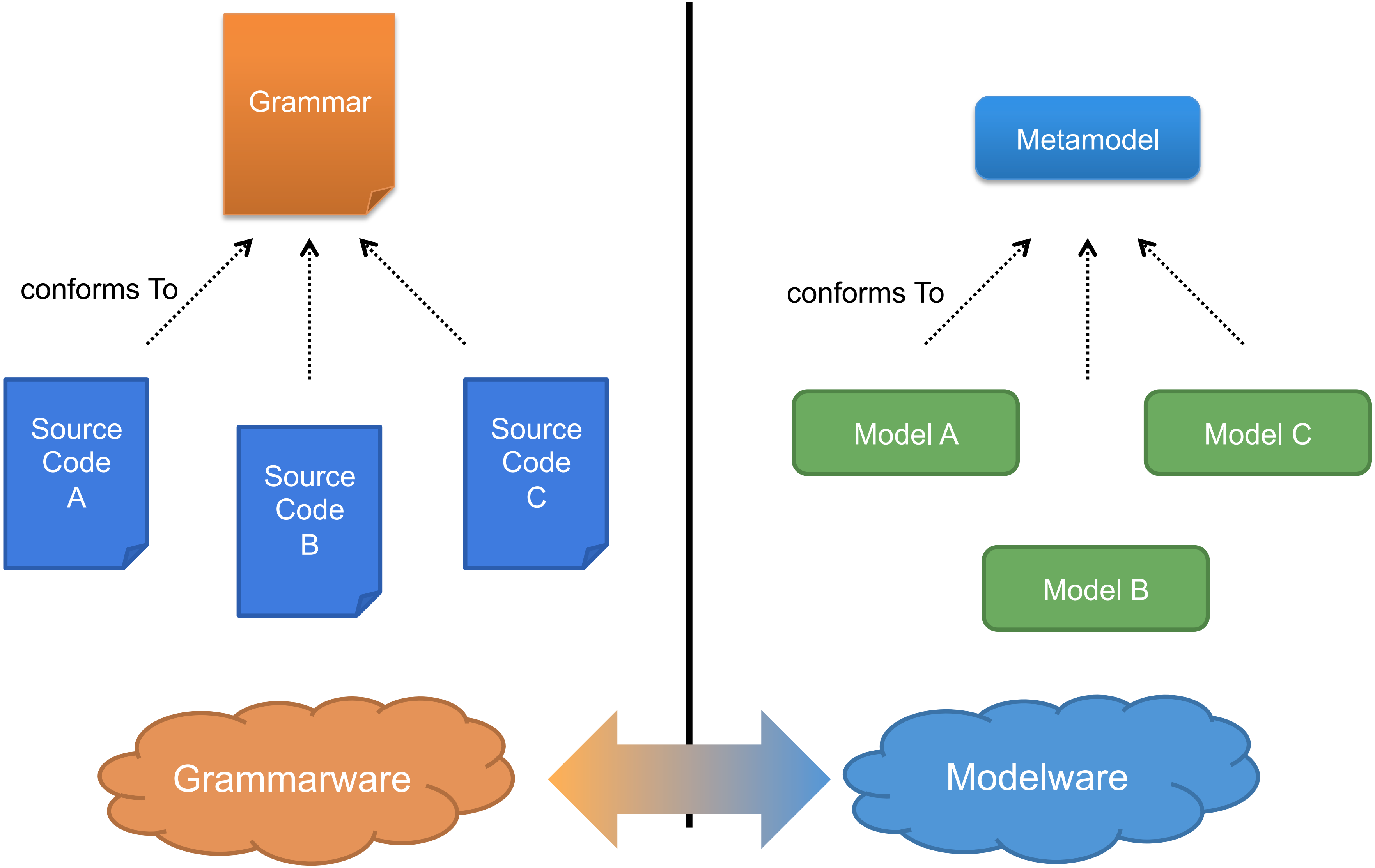
Metamodel

M^1

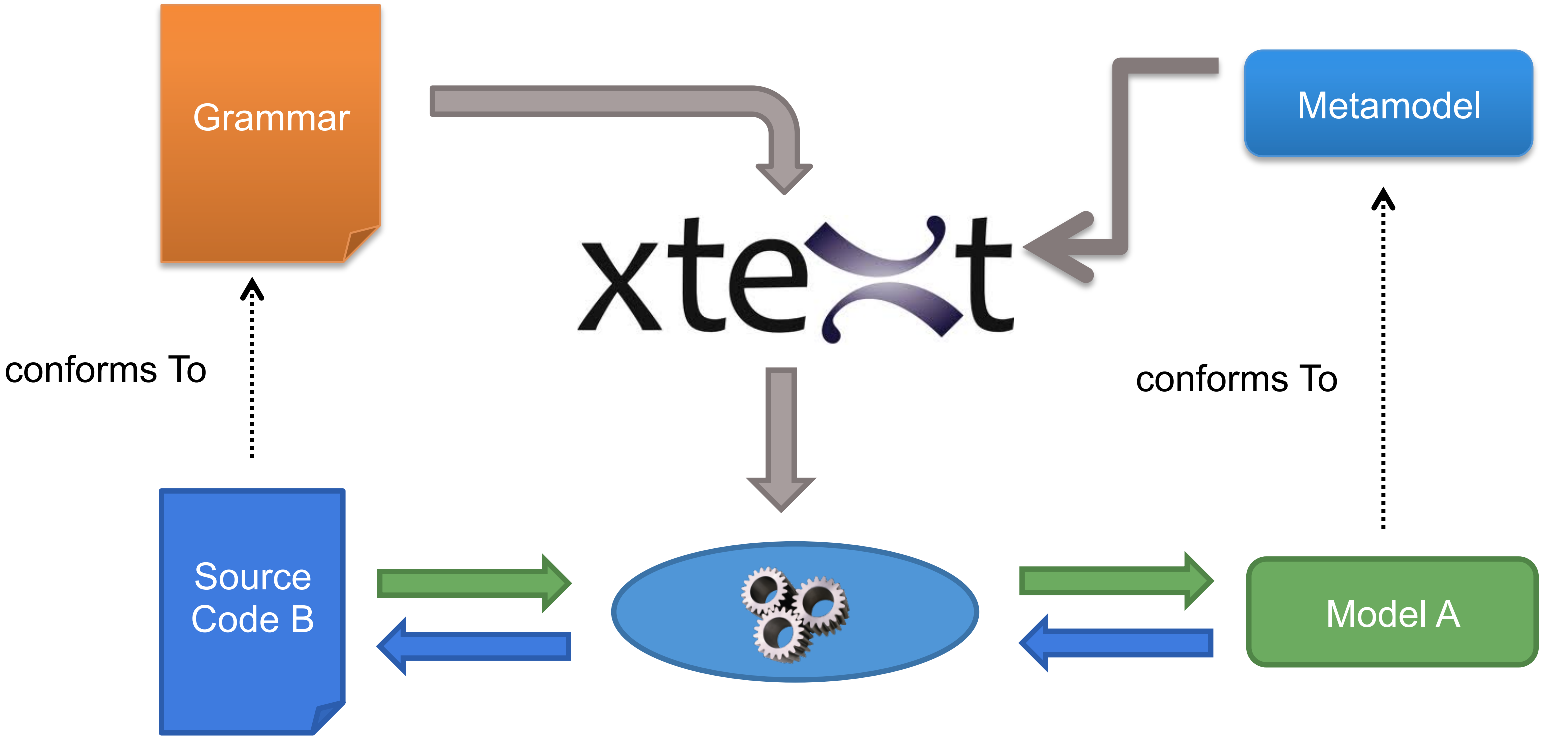
Source Code

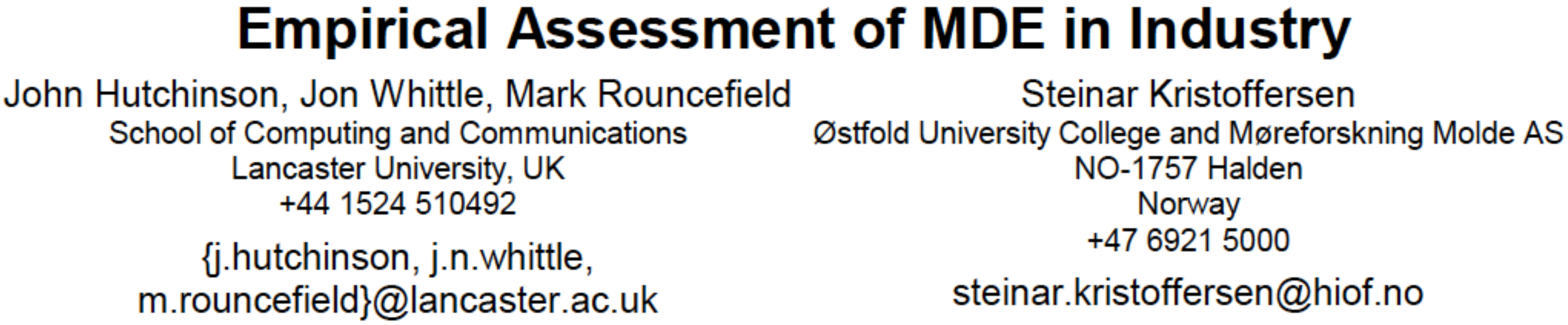
Model

Language and MDE



MDE, Grammar: there and back again

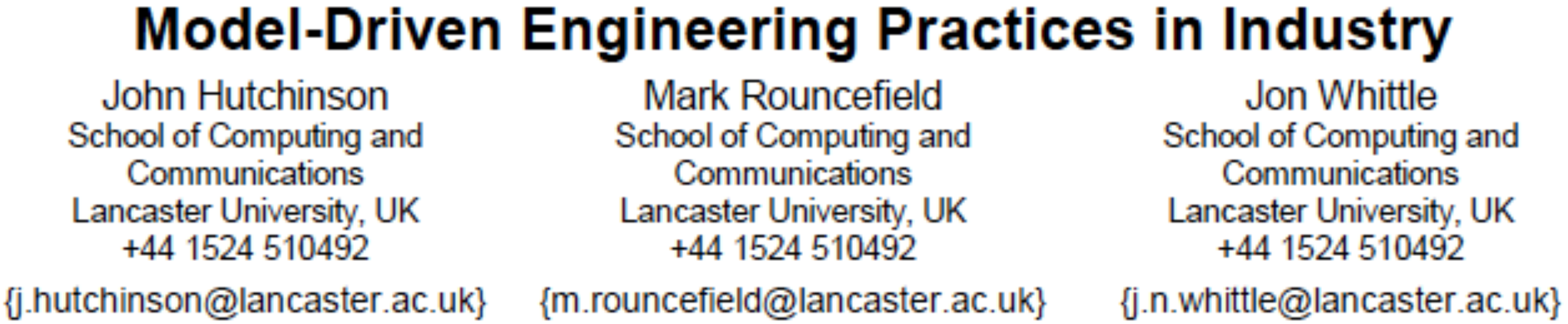




Empirical Assessment of MDE in Industry

John 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



Model-Driven Engineering Practices in Industry

John Hutchinson
School of Computing and
Communications
Lancaster University, UK
+44 1524 510492
{j.hutchinson@lancaster.ac.uk}

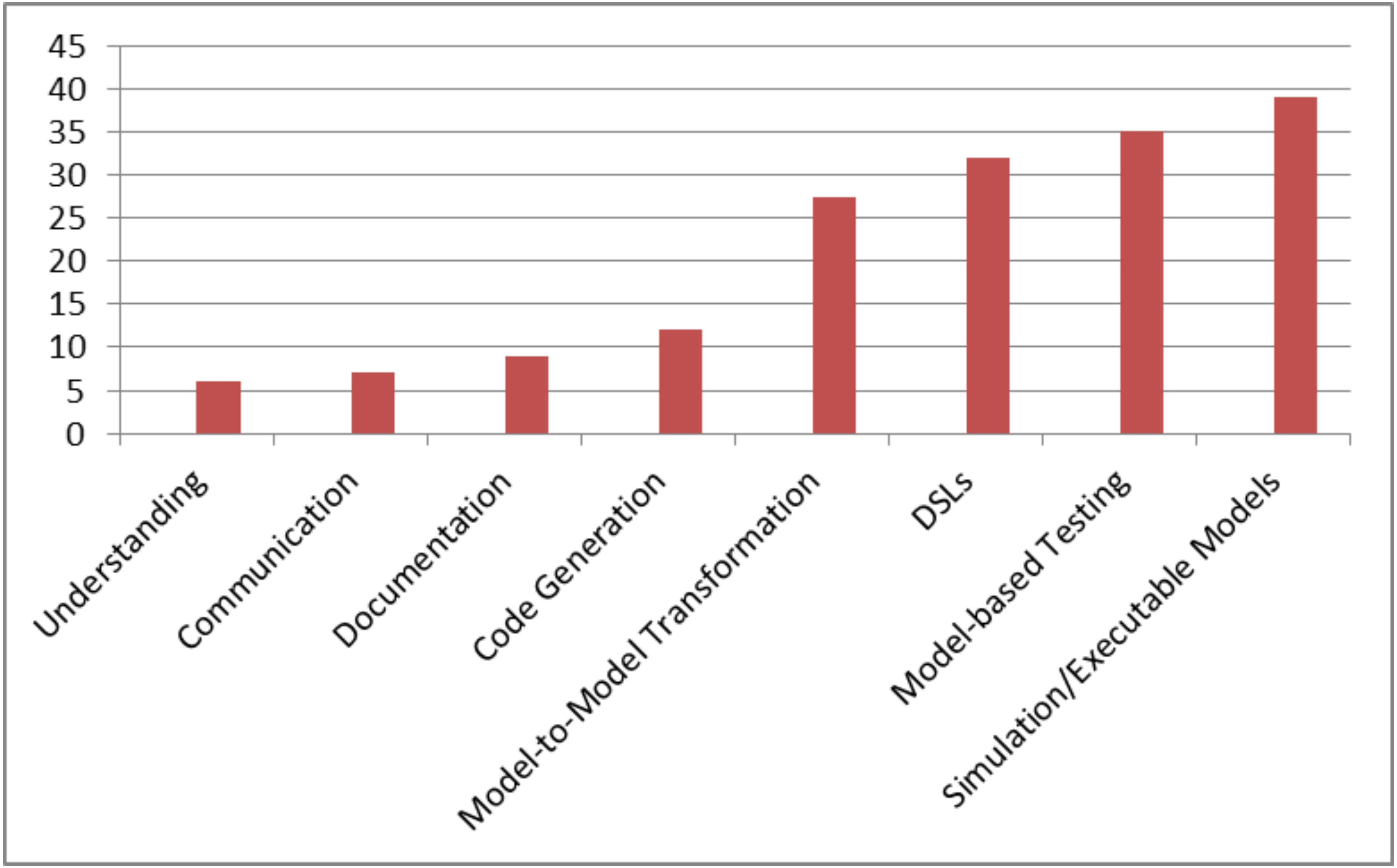
Mark Rouncefield
School of Computing and
Communications
Lancaster University, UK
+44 1524 510492
{m.rouncefield@lancaster.ac.uk}

Jon Whittle
School of Computing and
Communications
Lancaster University, UK
+44 1524 510492
{j.n.whittle@lancaster.ac.uk}

2011

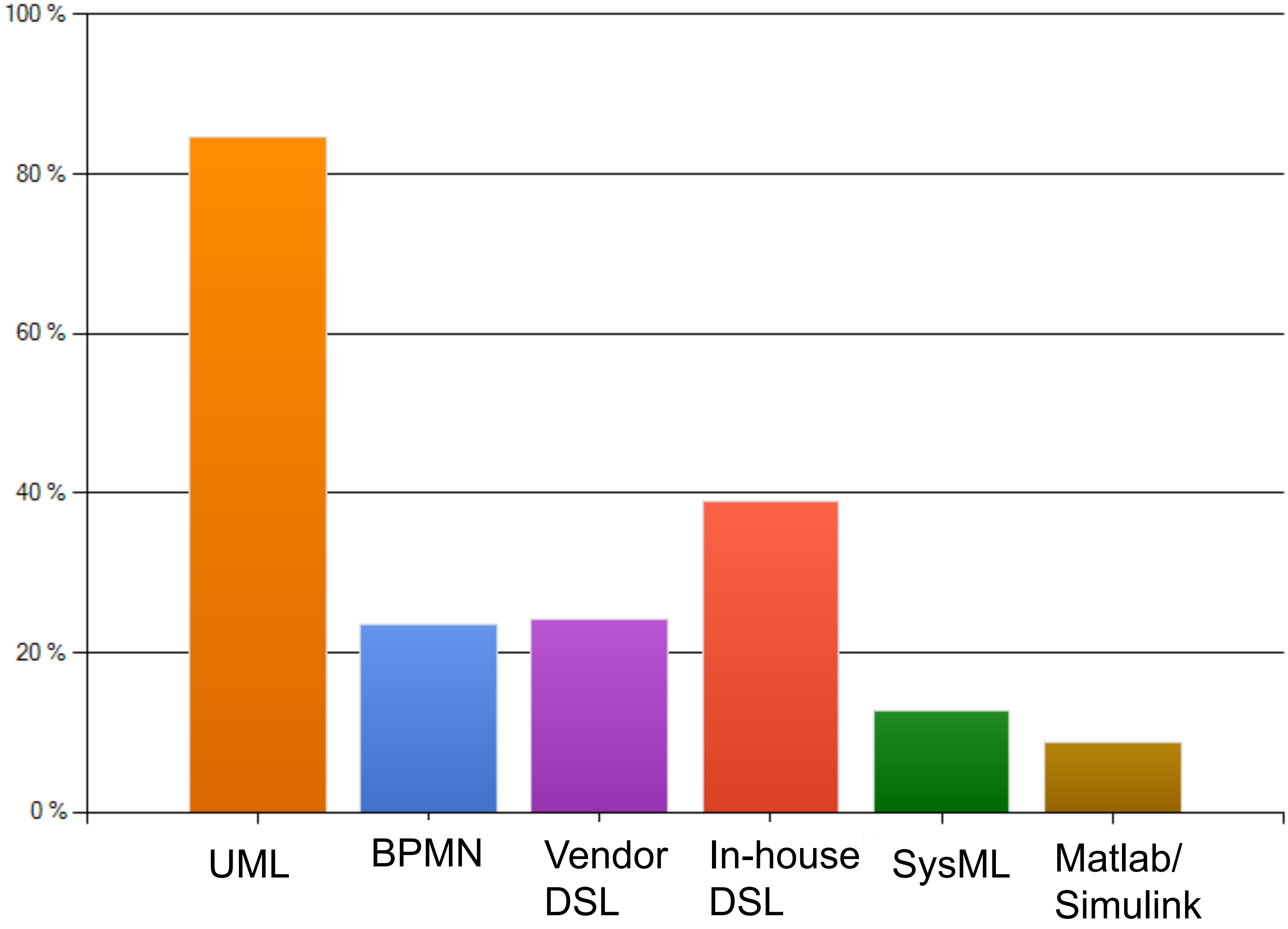
« Domain-specific languages are far more prevalent than anticipated »

What are models used for?

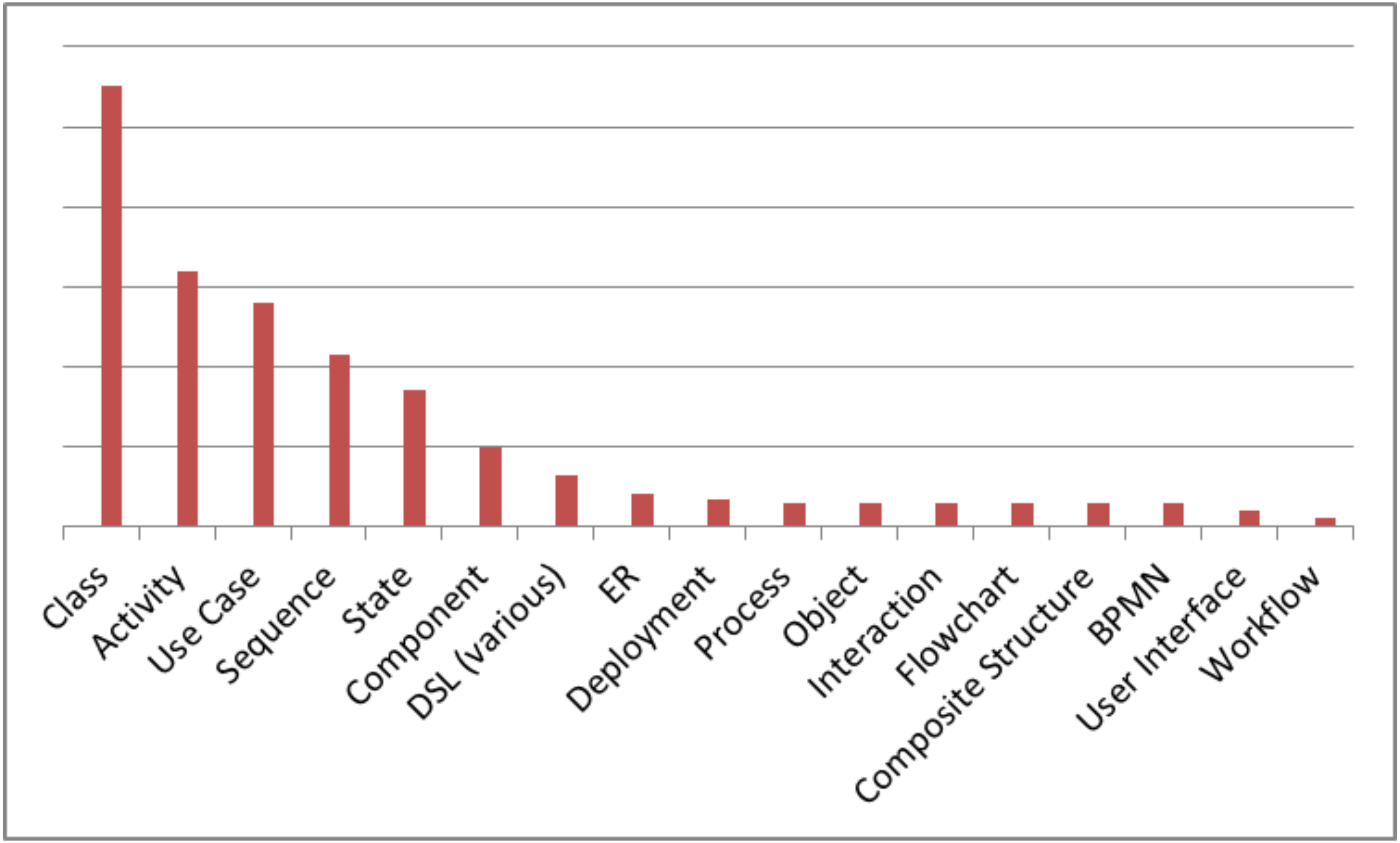


“Do not use” percentages for MDE activities

Which modeling languages do you use?



Which diagrams are used?

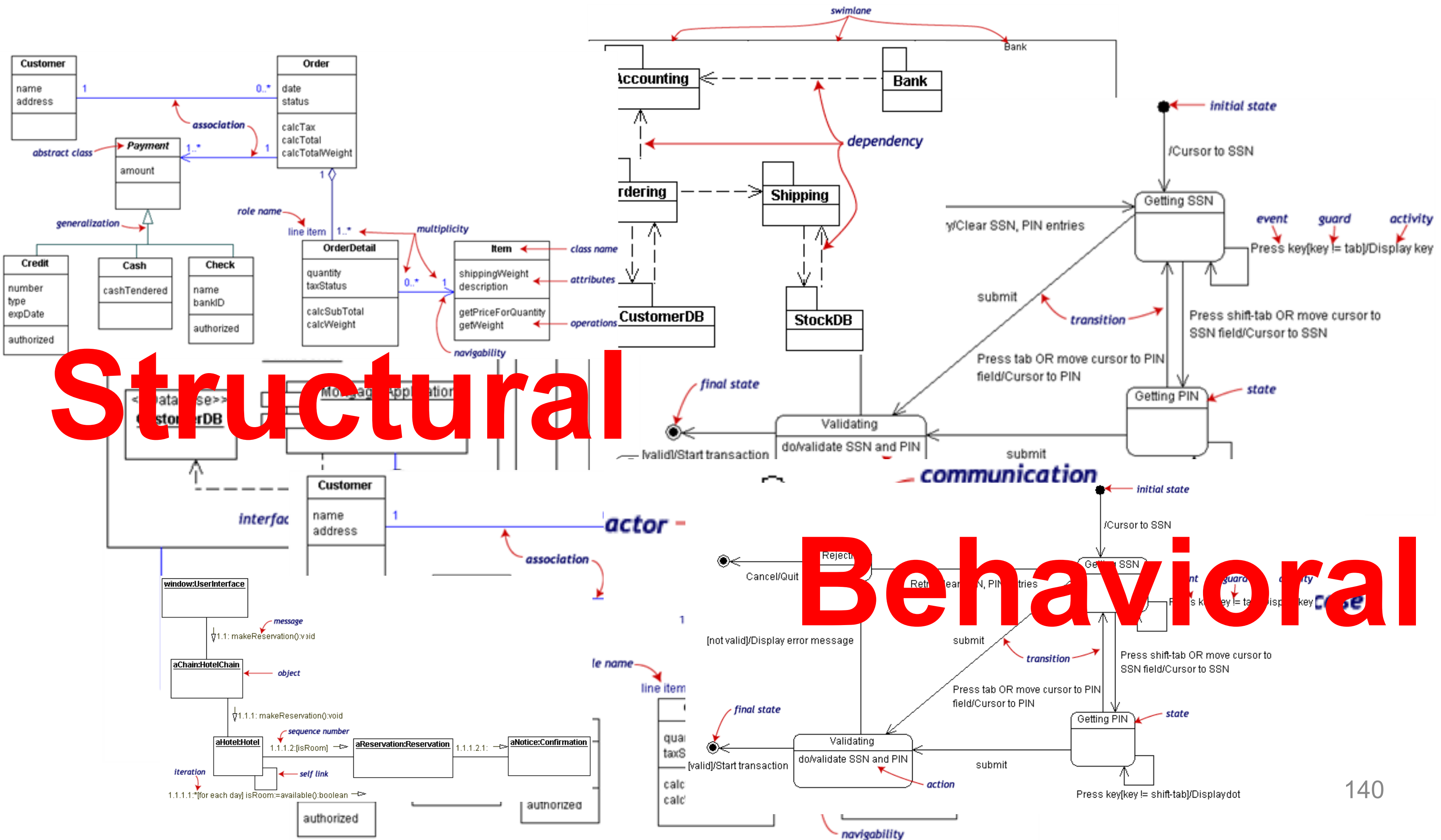


19 different diagram types are used regularly

Use of multiple languages (DSLs)

- 62% of those using custom DSLs also use UML
- Almost all users of SysML and BPMN also use UML
- UML is the most popular ‘single use’ language
  - 38% of all respondents
- UML used in combination with just about every combination of modeling languages
  - 14% of UML users combine with vendor DSL
  - 6% with both custom and vendor DSL

UML can be seen as a collection of domain-specific modeling languages



Structural

Behavioral

Xtext is built using MDE technologies

Xtext (and alternatives) democratize DSL development

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

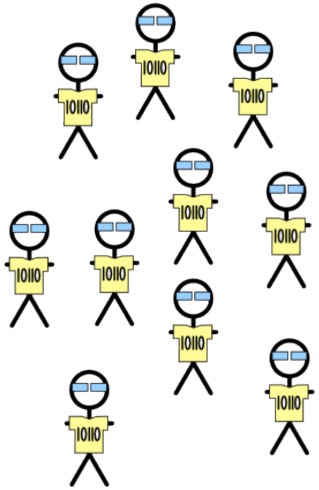
But my take away message is NOT

That DSLs should be used systematically, in every situations

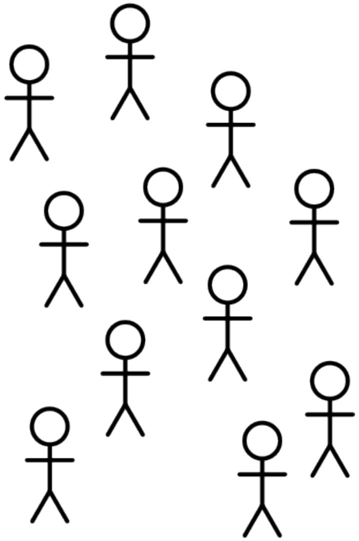
When Developing DSLs?

- Tradeoff cost/time of development versus producivity gained for solving problems
  - If you use your DSL for resolving one problem, just one time, hum…
  - DSL: reusable, systematic means to resolve a specific task in a given domain
- DSL development can pay off quickly
  - 5’ you can get a DSL
- But DSL development can be time-consuming and numerous worst practices exists

Actors

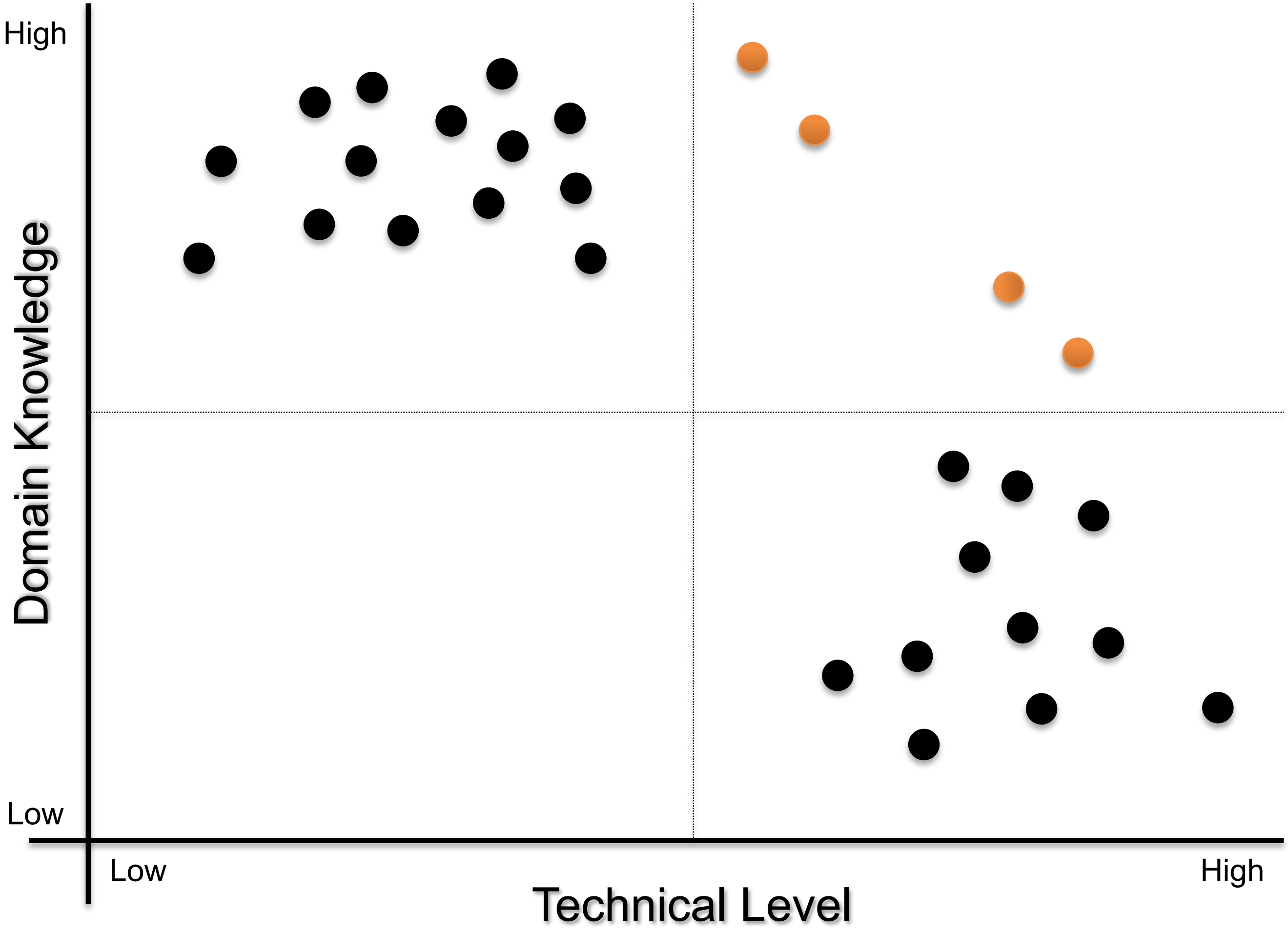


Developers



End-Users

Actors



Best Practices

- Limit Expressiveness
- Viewpoints
- Evolution
- Learn from GPLs
- Support
- Tooling

Worst Practices

- Initial conditions
  - Only Gurus allowed
    - Believe that only gurus can build languages ir that "I'm smart and don't need help"
  - Lack of Domain Understanding
    - Insufficiently understanding the problem domain or the solution domain
  - Analysis paralysis
    - Wanting the language to be theoretically complete, with its implementation assured

Worst Practices

- The source for Language Concepts
  - UML: New Wine in Old Wineskins
    - Extending a large, general-purpose modeling language
  - 3GL Visual Programming
    - Duplicanting the concepts and semantics of traditional programming languages
  - Code: The Library is the Language
    - Focusing the language on the current code's technical details
  - Tool: if you have a hammer
    - Letting the tool's technical limitations dictate language development

Worst Practices

- The resulting language
  - Too Generic / Too Specific
    - Creating a language with a few generic concepts or too many specific concepts, or a language that can create only a few models
  - Misplaced Emphasis
    - Too strongly emphasizing a particular domain feature
  - Sacred at Birth
    - Viewing the initial language version as unalterable

Worst Practices

- Language Notation
  - Predetermined Paradigm
    - Choosing the wrong representational paradigm or the basis of a blinkered view
  - Simplistic Symbols
    - Using symbols that are too simple or similar or downright ugly

Worst Practices

- Language Use
  - – Ignoring the use process
    - Failing to consider the language's real-life usage
  - – No training
    - Assuming everyone understands the language like its creator
  - – Pre-adoption Stagnation
    - Letting the language stagnate after successful adoption

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

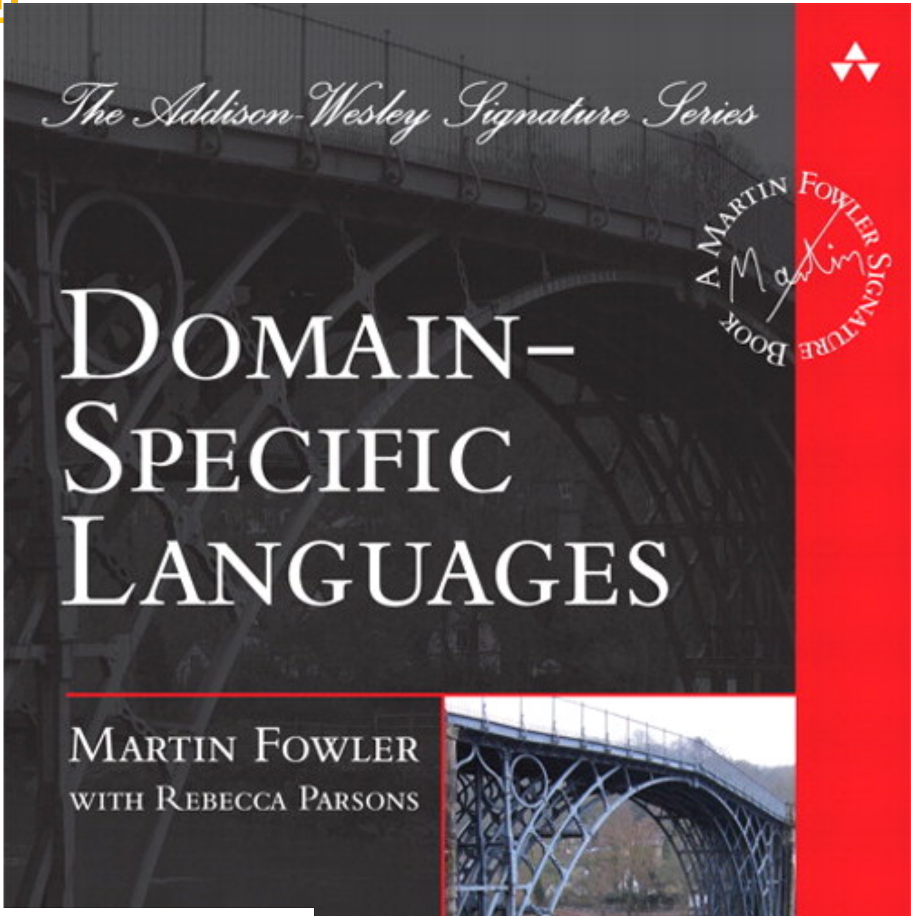
References

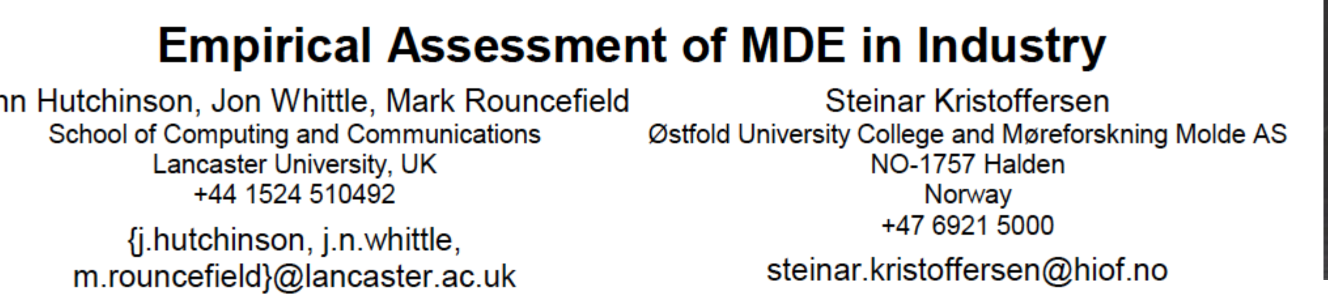
- Sebastian Erdweg, Tijs van der Storm, Markus Völter, Meinte Boersma, Remi Bosman, William R. Cook, Albert Gerritsen, Angelo Hulshout, Steven Kelly, Alex Loh, Gabriël D. P. Konat, Pedro J. Molina, Martin Palatnik, Risto Pohjonen, Eugen Schindler, Klemens Schindler, Riccardo Solmi, Vlad A. Vergu, Eelco 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

http://martinfowler.com/bliki/DomainSpecificLanguage.html

xtext

gmf GRAPHICAL MODELING FRAMEWORK


The Addison-Wesley Signature Series
A Martin Fowler Signature Book
Domain-Specific Languages
Martin Fowler
with Rebecca Parsons



Empirical Assessment of MDE in Industry

hn 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