

# Domain-Specific Languages

The Art Of Domain-Specific Languages  
*Let's Hack Our Own Languages!*

# Plan

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

# Plan

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

# 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 (mature and used in a variety of industries)

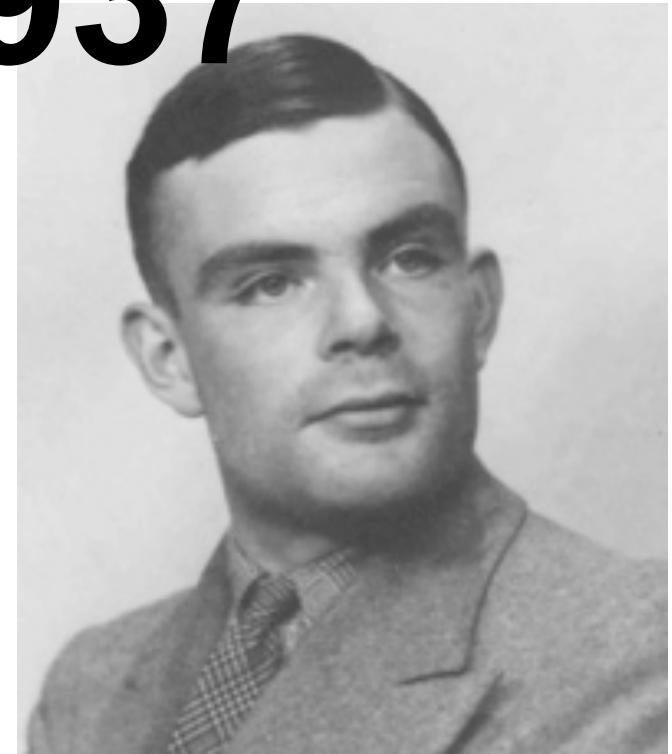
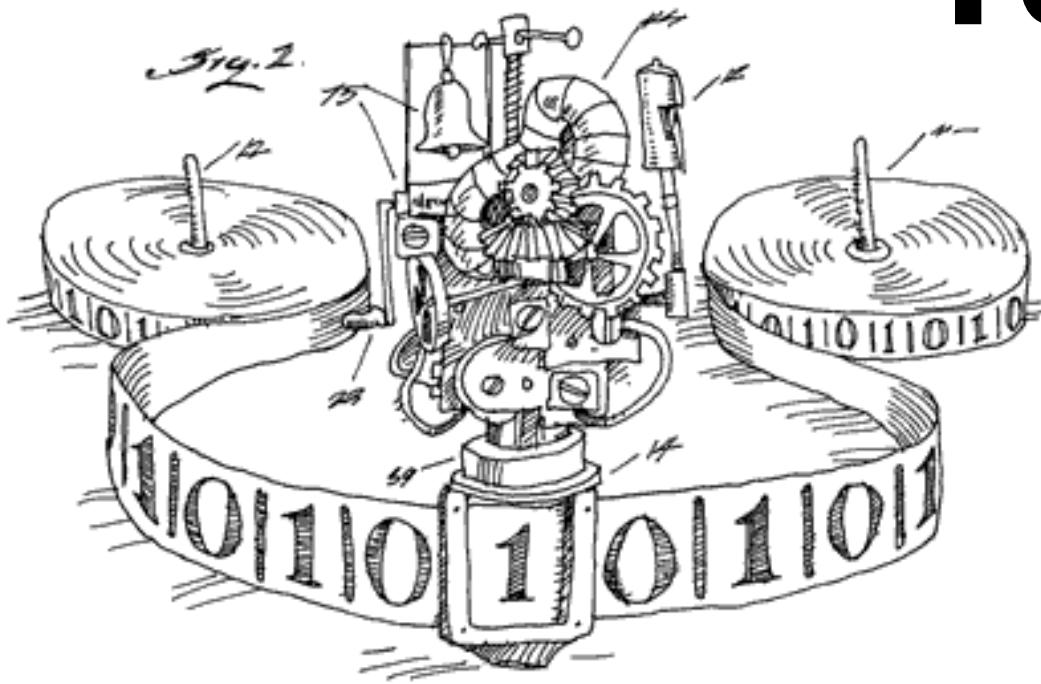
What are DSLs

Where are DSLs

Why DSLs (will) matter

# The (Hi)Story of Software Engineering / Computer Science

1937

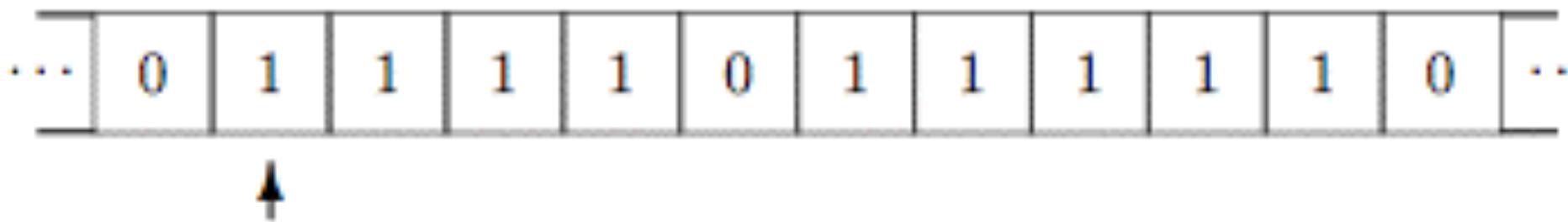


# Turing Machine

- Infinite tape divided into Cells (0 or 1)
- Read-Write Head
- Transition rules

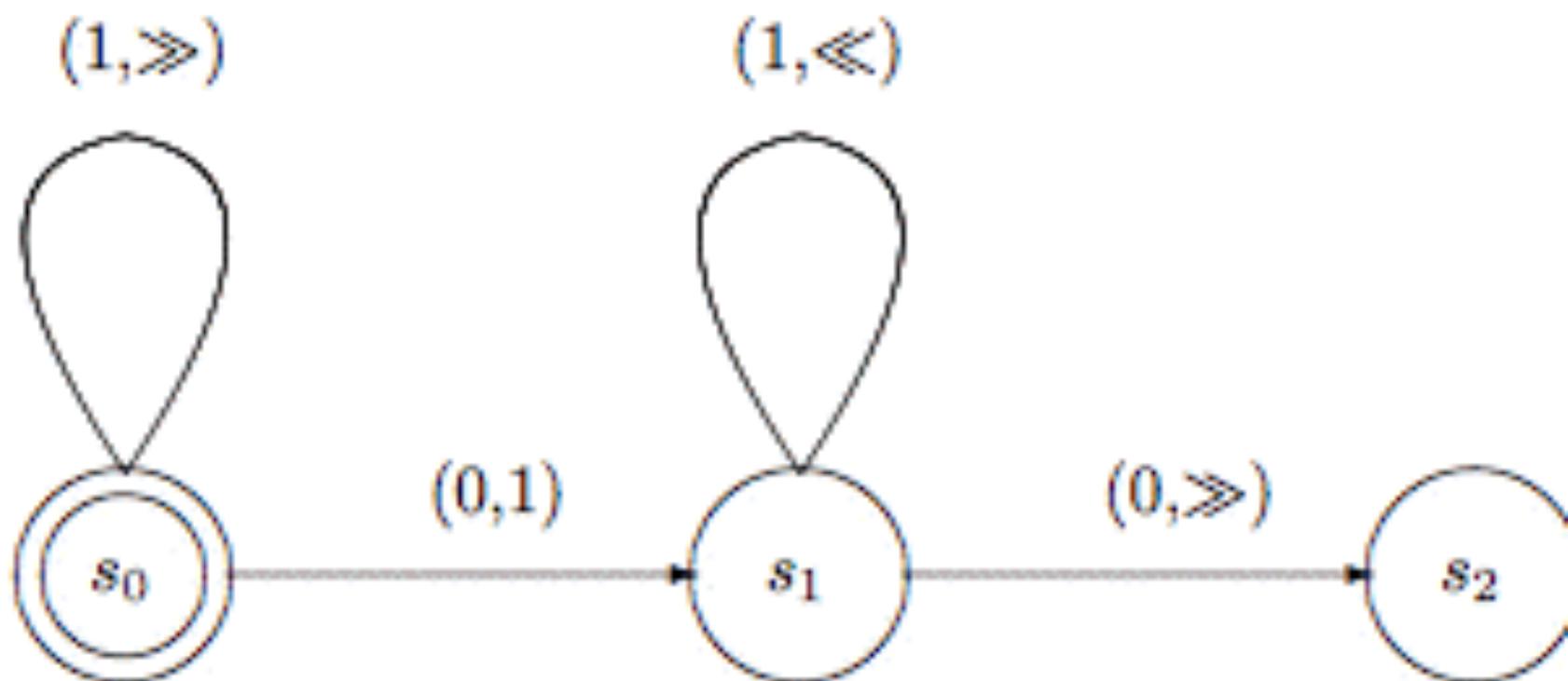
**Write a symbol  
or move to left (>>) or right  
(<<)**

*< State<sub>current</sub>, Symbol, State<sub>next</sub>, Action >*



# Turing Machine

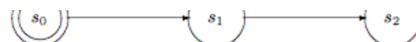
~ kind of state machine



# The (Hi)Story of Software Engineering Computer Science

0100  
010010101000  
1010010101000  
010010101000  
010010101000  
0010010101000  
0010010101000

0 1 1 1 1 0 1 1 1 1 1 0 ...



... 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 ...

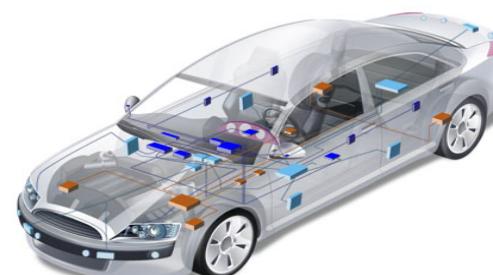


orange™

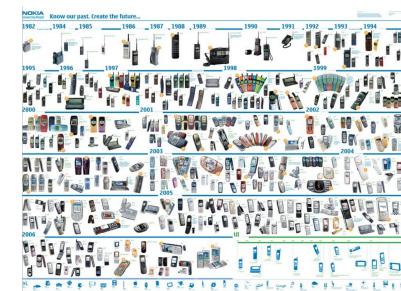
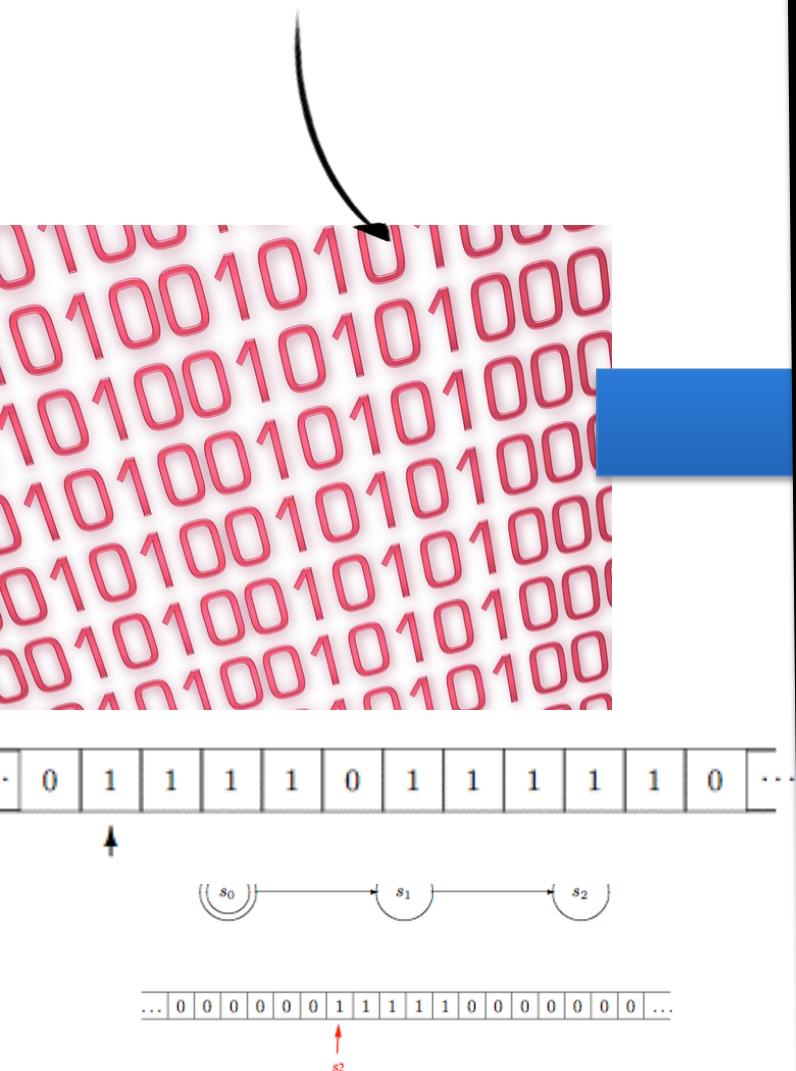


Google

twitter



# Software Languages

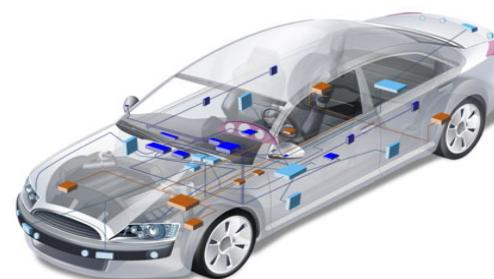


orange™



Google

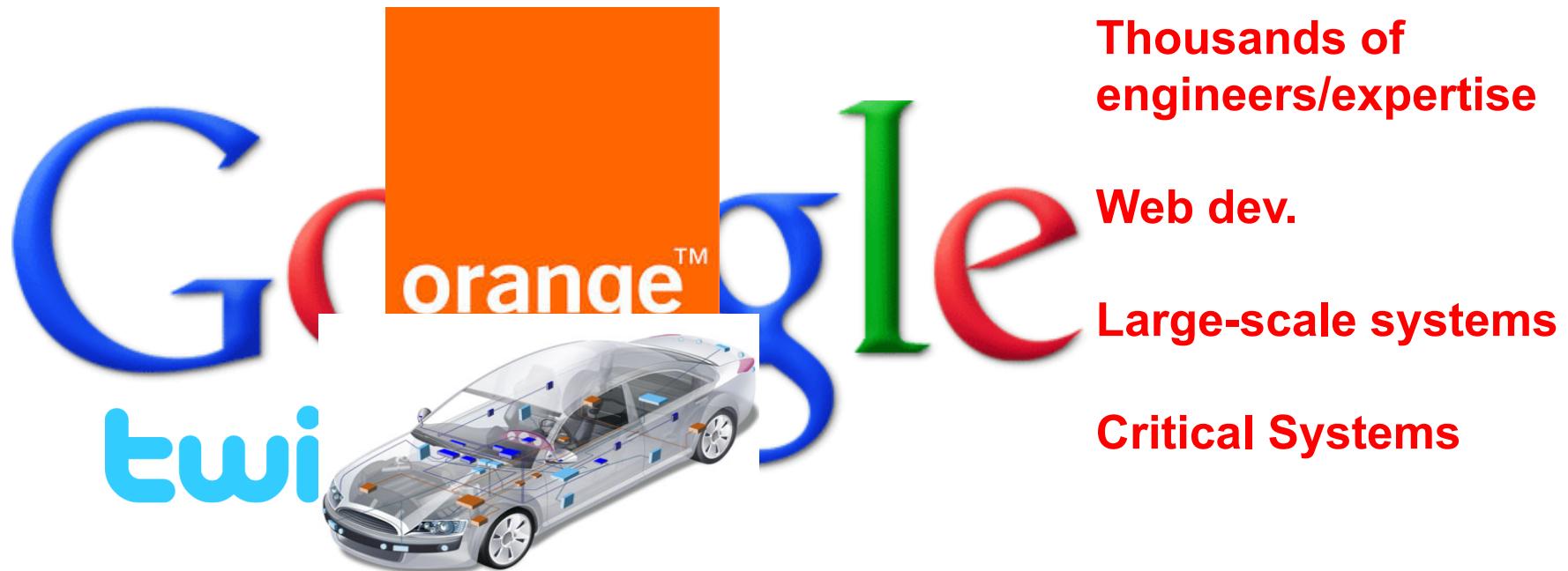
twitter



# Programming the Turing Machine

## Why aren't we using tapes, states and transitions after all ?

### Complex Systems



Distributed systems

Thousands of  
engineers/expertise

Web dev.

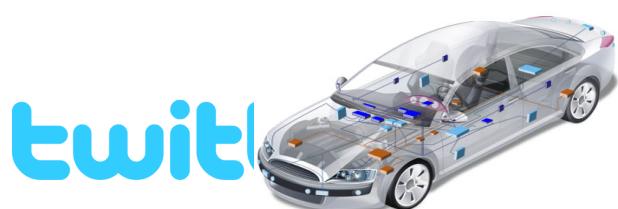
Large-scale systems

Critical Systems

Programming the Turing Machine

**Why aren't we using tapes, states and transitions after all ?**

**You cannot be serious**



[SUBMIT A LINK](#)[FEATURES](#) [REVIEWS](#) [PODCASTS](#) [VIDEO](#) [FORUMS](#) [MORE ▾](#)

3CD LIMITED EDITION BOX SET • 2CD • 2LP • DOWN  
AVAILABLE DECEMBER 10, 2013

# Implementing a Turing machine in Excel

Cory Doctorow at 2:20 pm Fri, Sep 20, 2013



142



24



The screenshot shows a Microsoft Excel spreadsheet titled "Turing Machine\_Successor.xlsx". The formula bar displays the formula: =IFERROR(VLOOKUP(VLOOKUP(\$B4&","&INDEX(\$A4:\$F4,M4,\$A4).StateTable,5,FALSE),DirectionTable,2, FALSE),0)+\$A4. The spreadsheet contains a grid of binary values (0s and 1s) representing a state transition table for a Turing machine. The columns represent states and symbols, and the rows represent transitions.

## Formulas are Turing complete

**2013 GIFT GUIDE**

**AN ASTRONAUT'S GUIDE TO LIFE ON EARTH**

# Formulas are Turing complete

**Youtube video <https://t.co/RTfJAxXV>**

<http://fr.slideshare.net/Felienne/spreadsheets-are-code-online>

# Esoteric programming languages

- Designed to test the boundaries of computer programming language design, as a proof of concept, as software art, or as a joke.
  - extreme paradigms and design decisions
  - Eg <https://esolangs.org/wiki/Brainfuck>
- Usually, an esolang's creators do not intend the language to be used for mainstream programming.

(brainfuck)

What does it compute?

```
+++++++[>+++++>++++++>+++<<<-  
]>++.>+.++++++  
.++,>++.<<+++++++.>,.++,.----,.----.>+.
```

# Quizz Time

- Why assembly language is not the mainstream language?
- Why spreadsheets are not used for building Google?
- Why esoteric languages are not used for mainstream programming?

# Programming the Turing Machine

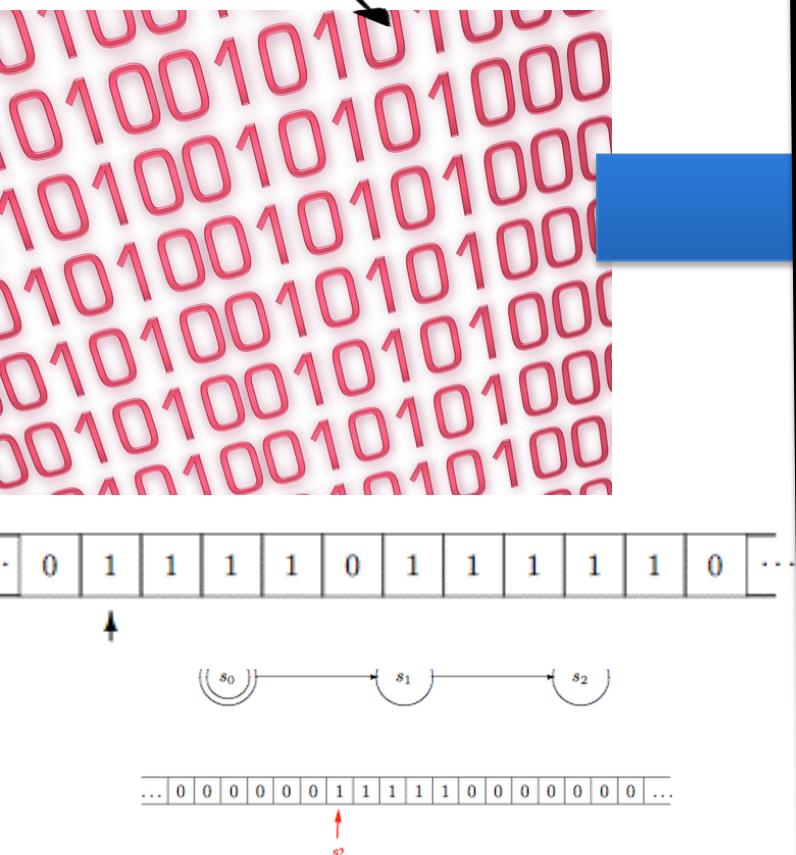
## Why aren't we using tapes, states and transitions after all ?

## Software Languages

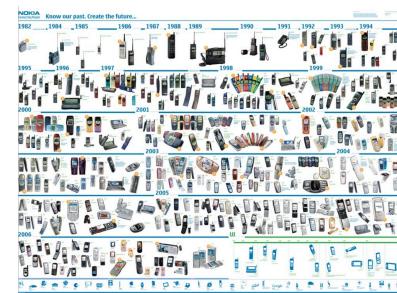


Not fun. Over complicated.  
Hard to write and  
understand. No abstractions.  
Poor language constructs.  
Tooling Support?

# Languages



# Complex Systems



orange™

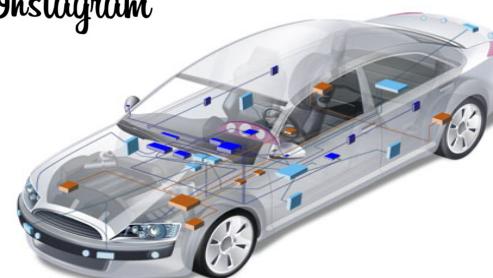


ANDRO

# Google



## Instagram



# How Language Shapes Thought

The languages we speak affect our perceptions of the world

*By Lera Boroditsky*

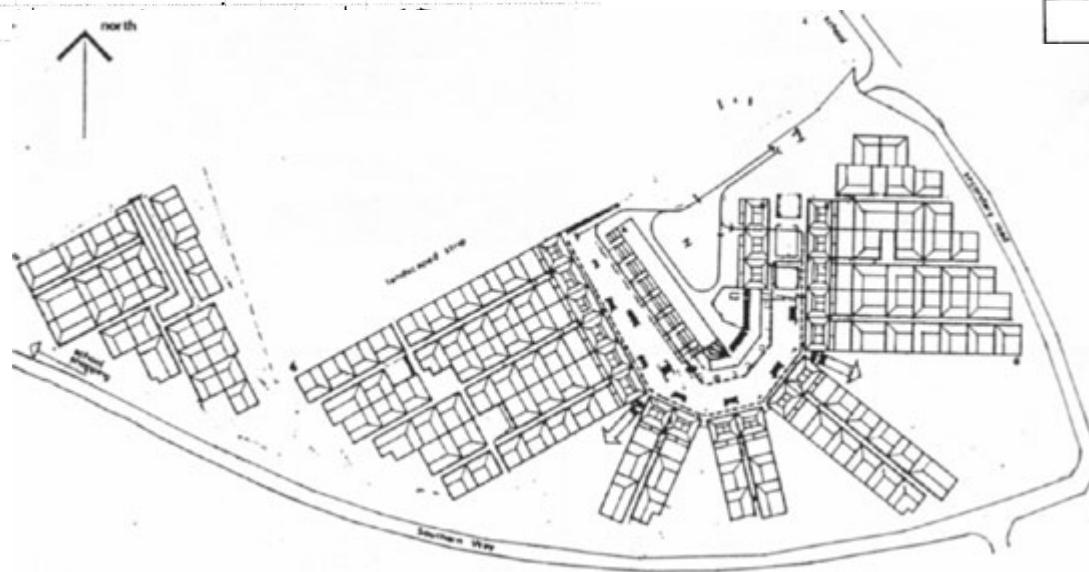
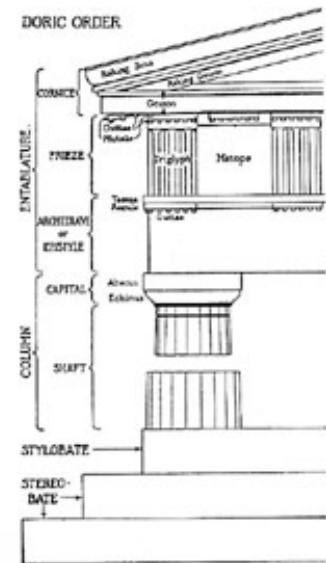
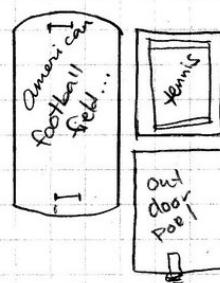
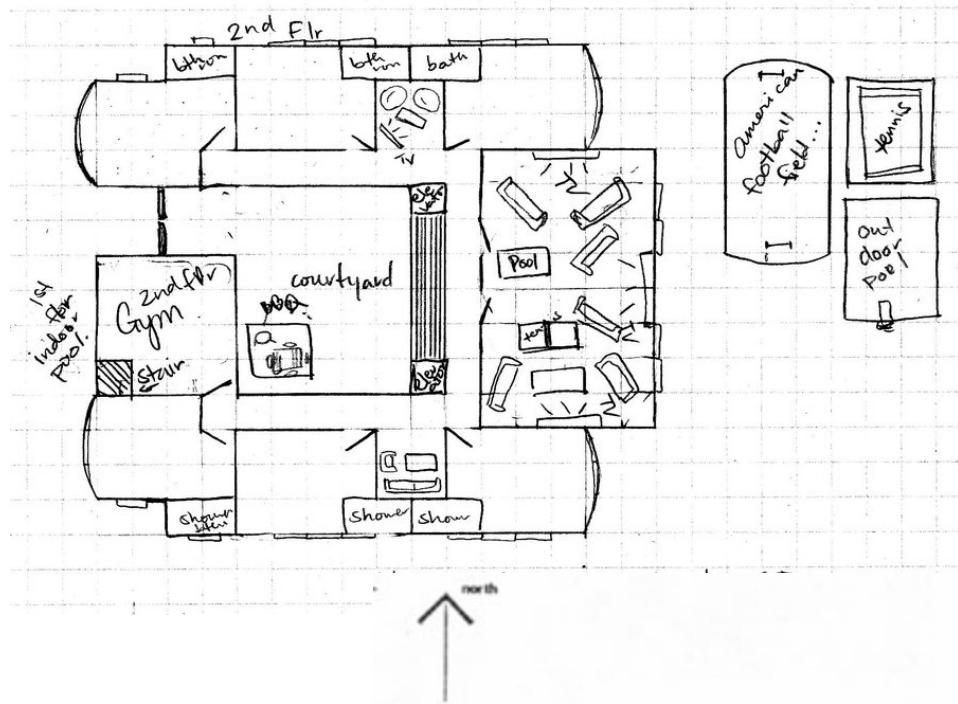
“Even variations in grammar can profoundly affect how we see the world.”

She's talking about real languages; **what about synthetic, programming languages?**

# What is a language?

- « A system of signs, symbols, gestures, or rules used in **communicating** »
- « The **special** vocabulary and usages of a scientific, professional, or other group »
- « A system of symbols and rules used for communication with or between computers. »

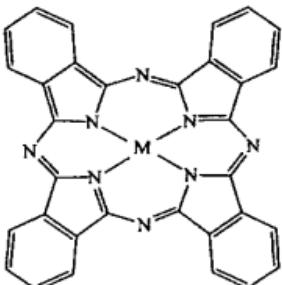
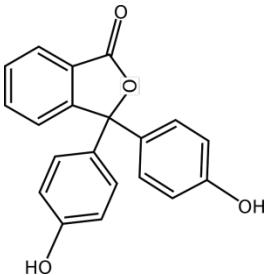
# Architecture



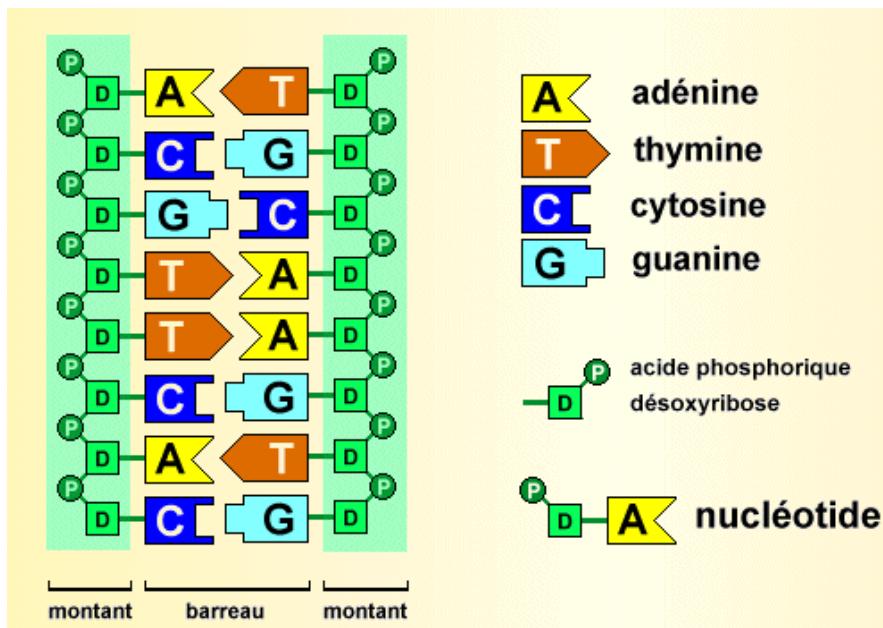
# Cartography



# Biology



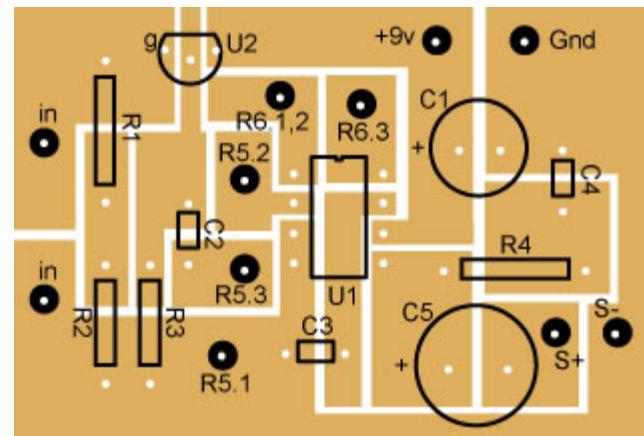
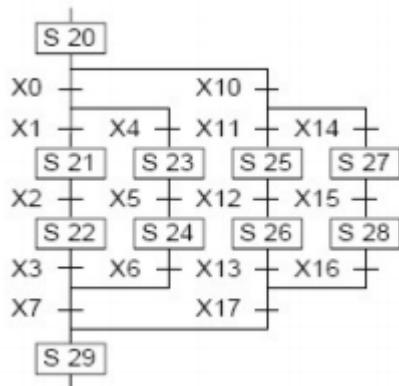
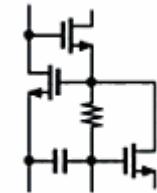
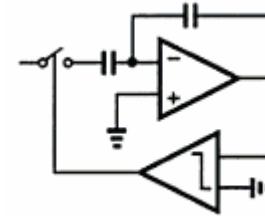
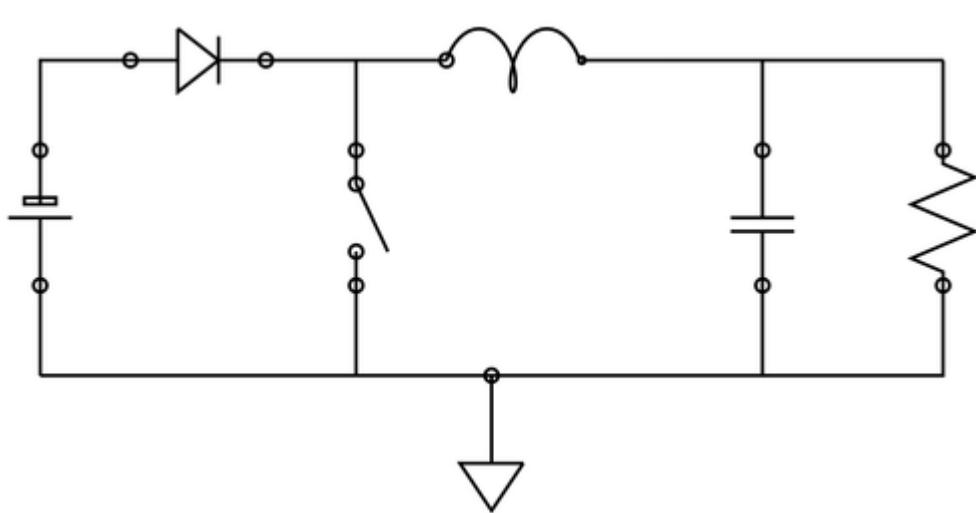
phthalocyanine



60	70	80	90	100
AGACCCCCAG	CAACCCCCGG	GGCCGTGCGG	CCTCGGTCGT	GTCGTGTGAT
160	170	180	190	200
AGACCCCGCG	TACGAATGCC	GGTCCACCAA	CAACCCGTGG	GCTTCGCAGC
260	270	280	290	300
CTGCCGGGCA	TGTACAGTC	TTGTCGGCAG	TTCTTCCACA	AGGAAGACAT
360	370	380	390	400
GGCTTGCTGG	GGCCCCCGCC	ACCAGCACTA	CAGACCTCCA	GTACGTCGTG
460	470	480	490	500
GGCCTATCCC	ACGCTCGCCG	CCAGCCACAG	AGTTATGCTT	GCCGAGTACA
560	570	580	590	600
GAAAGGGTGG	CGCCGATGAA	GAGACTATT	AAGCTGGAA	ACAAGGTGGT
660	670	680	690	700
ATAGTGGTTA	ACTTCACCTC	CAGACTCTTC	GCTGATGAAC	TGGCCGCCCT
760	770	780	790	800
AAAATATACA	GGCATTTGGC	CTGGGGTGCG	TATGCTCACG	TGAGACATCT
860	870	880	890	900
CCTGGAGGAG	GTTCGCCCCG	ACAGCCTGCG	CCTAACGCGG	ATGGATCCCT
960	970	980	990	1000
AGCAACACCC	AGCTAGCAGT	GCTACCCCCA	TTTTTTAGCC	GAAAGGATTC
1060	1070	Pvu II site	1090	1100
TGCCCGCAGCA	ACTGGGGCAC	GCTATTCTGC	AGCAGCTGTT	GGTGTACCA
1160	1170	1180	1190	1200
ACTTGATCTA	TATACCCACCA	ATGTGTCATT	TATGGGGCCG	ACATATCGTC
1260	1270	1280	1290	1300
CTGTCATGT	ACCTTTGTAT	CCTATCAGCC	TTGGTTCCCA	GGGGGTGTCT
1360	1370	1380	1390	1400
TGTTTGAGGG	GGTGGTGCCA	GATGAGGTGA	CCAGGATAGA	TCTCGACCAG
1460	1470	1480	1490	1500
TCAGAGTCCT	AGTTCTATAT	TTAACCTTGG	CCCCAGACTG	CACGTGTATG
1560	1570	1580	1590	1600
CGATTTGAAG	CGGGGGGGGT	ATGGCGTCAT	CTGATATTCT	GTGGGTTGCA
1660	1670	1680	1690	1700
AAAAACTTACC	GTCTACCTGC	CGGACACTGA	ACCCCTGGGTG	GTAGAGACCG
1760	1770	1780	1790	1800
AAGCTTCATC	GTGGTGCCCT	GCCCTCAAAT	TCTCACAAAG	GCTTGAGGAT

CTG.

# Electronics

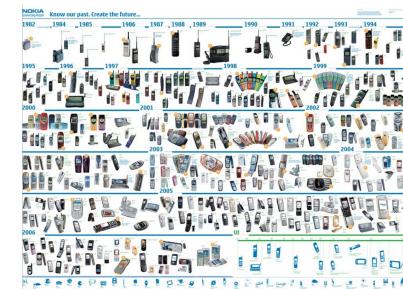
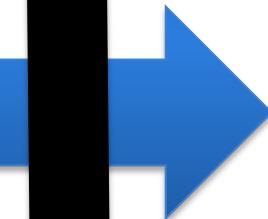
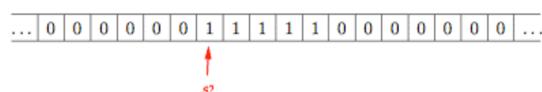
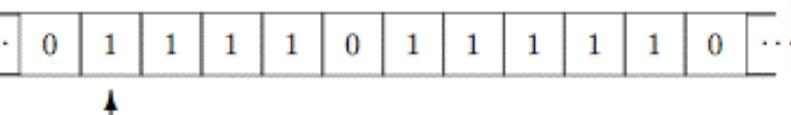
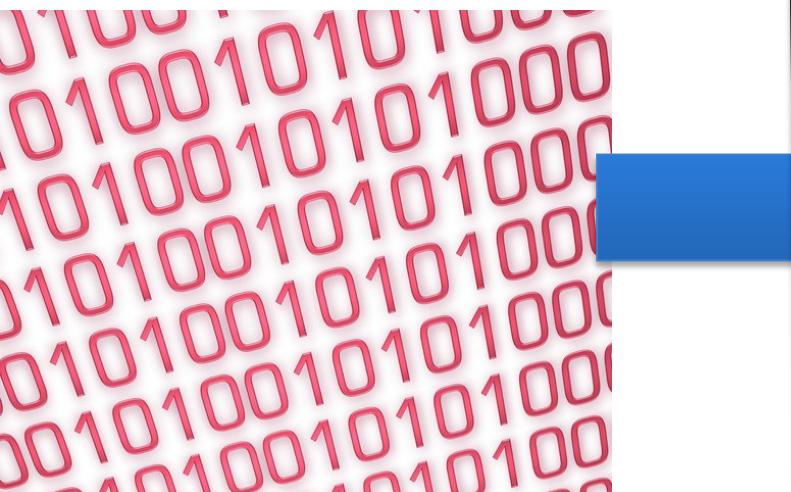


# In Software Engineering

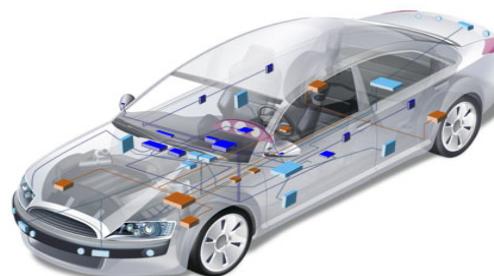
« Languages are the primary way in which system developers communicate, design and implement software systems »

# General Purpose Languages

Assembly ?  
COBOL ? LISP ? C ? C++ ?  
Java? PHP ? C# ? Ruby ?



Google  
twitter



# Limits of General Purpose Languages (1)

- **Abstractions and notations** used are not natural/suitable for the stakeholders



```
if (newGame) resources.free();
s = FILENAME + 3;
setLocation(); load(s);
loadDialog.process();

try { setGamerColor(RED); }
catch(Exception e) { reset(); }
while (notReady) { objects.make();
if (resourceNotFound) break;

byte result; // сменить на int!
music();
System.out.print("");
```



# Limits of General Purpose Languages (2)

- Not targeted to a **particular** kind of problem, but to any kinds of software problem.



# Domain Specific Languages

- Targeted to a **particular** kind of problem, with dedicated notations (textual or graphical), support (editor, checkers, etc.)
- Promises: more « efficient » languages for resolving a set of specific problems in a domain



# Domain Specific Languages (DSLs)

- Long history: used for almost as long as computing has been done.
- You're using DSLs in a daily basis
- You've learnt many DSLs in your curriculum
- Examples to come!

# HTML

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "DTD/xhtml1-transitional.dtd">
<html xml:lang="en" lang="en" xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Hello World</title>
  </head>
  <body>
    <p>My first Web page.</p>
  </body>
</html>
```

Domain: web (markup)

# CSS

```
.CodeMirror {  
    line-height: 1;  
    position: relative;  
    overflow: hidden;  
}  
  
.CodeMirror-scroll {  
    /* 30px is the magic margin used to hide the element's real scrollbars */  
    /* See overflow: hidden in .CodeMirror, and the paddings in .CodeMirror-sizer */  
    margin-bottom: -30px; margin-right: -30px;  
    padding-bottom: 30px; padding-right: 30px;  
    height: 100%;  
    outline: none; /* Prevent dragging from highlighting the element */  
    position: relative;  
}  
.CodeMirror-sizer {  
    position: relative;  
}
```

Domain: web (styling)

# SQL

```
SELECT Book.title AS Title,  
       COUNT(*) AS Authors  
  FROM Book  
 JOIN Book_author  
    ON Book.isbn = Book_author.isbn  
GROUP BY Book.title;
```

```
INSERT INTO example  
(field1, field2, field3)  
VALUES  
( 'test' , 'N' , NULL);
```

Domain: database (query)

# Makefile

```
PACKAGE      = package
VERSION      = ` date "+%Y.%m%d%" `
RELEASE_DIR  = ..
RELEASE_FILE = $(PACKAGE)-$(VERSION)

# Notice that the variable LOGNAME comes from the environment in
# POSIX shells.
#
# target: all - Default target. Does nothing.
all:
    echo "Hello $(LOGNAME), nothing to do by default"
    # sometimes: echo "Hello ${LOGNAME}, nothing to do by default"
    echo "Try 'make help'"

# target: help - Display callable targets.
help:
    egrep "^# target:" [Mm]akefile

# target: list - List source files
list:
    # Won't work. Each command is in separate shell
    cd src
    ls

    # Correct, continuation of the same shell
    cd src; \
    ls
```

Domain: software building

# Lighttpd configuration file

```
server.document-root = "/var/www/servers/www.example.org/pages/"

server.port = 80

server.username = "www"
server.groupname = "www"

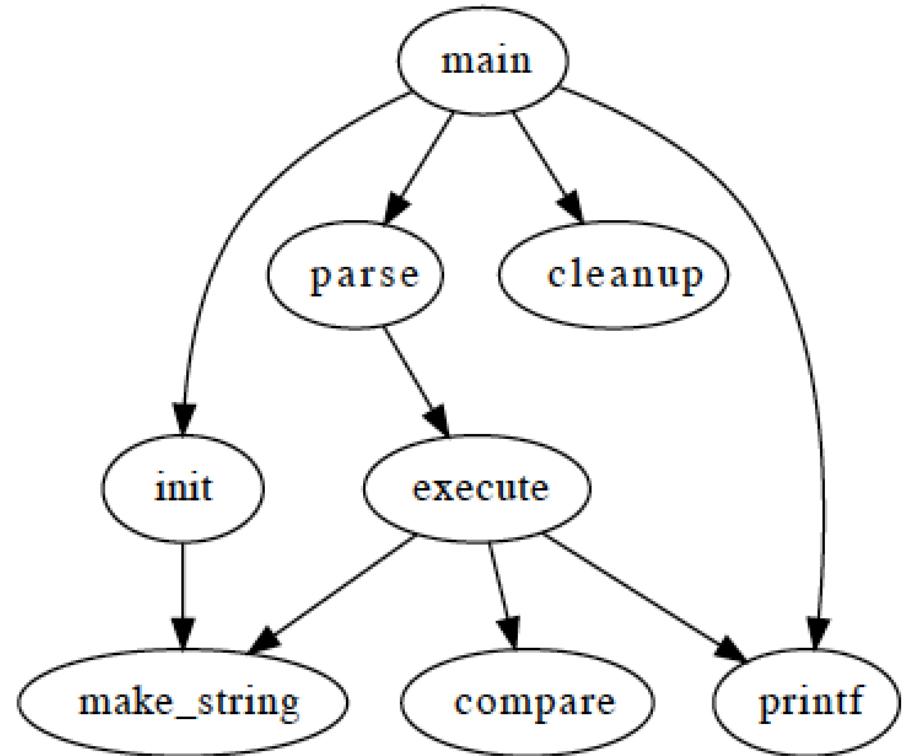
mimetype.assign = (
    ".html" => "text/html",
    ".txt" => "text/plain",
    ".jpg" => "image/jpeg",
    ".png" => "image/png"
)

static-file.exclude-extensions = ( ".fcgi", ".php", ".rb", "~", ".inc" )
index-file.names = ( "index.html" )
```

Domain: web server (configuration)

# Graphviz

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



Domain: graph (drawing)

# Regular expression

```
<TAG\b[^>]*>(.*)?</TAG>
```

Domain: strings (pattern matching)

# OCL

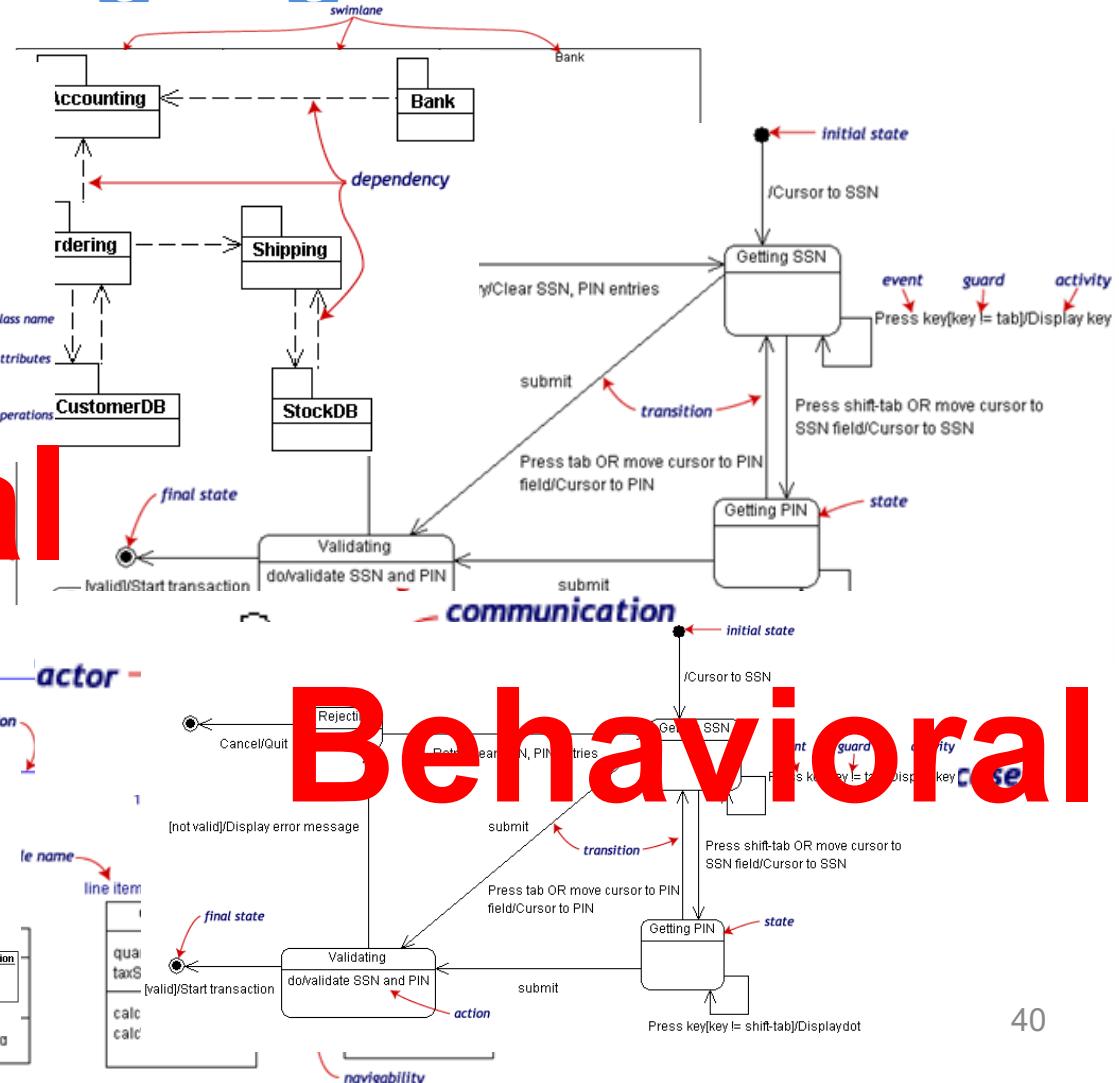
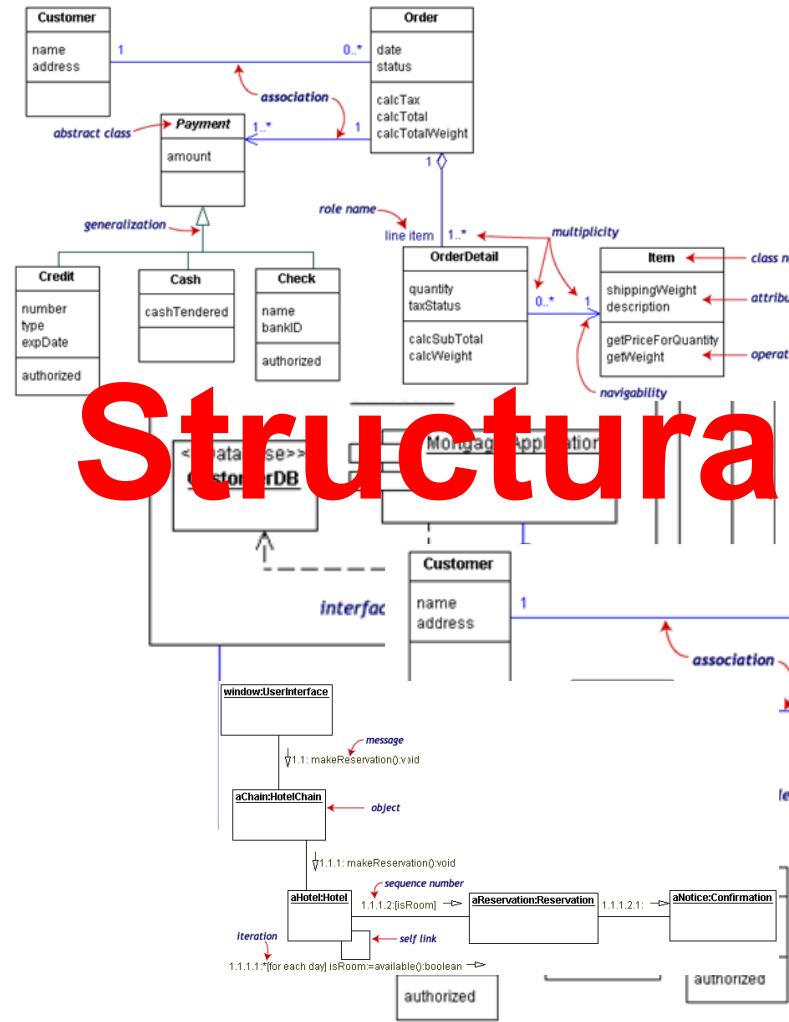
```
self.questions->size  
self.employer->size  
self.employee->select (v | v.wages>10000 )->size  
Student.allInstances  
->forAll( p1, p2 |  
           p1 <> p2 implies p1.name <> p2.name )
```

Domain: model management

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

Structural

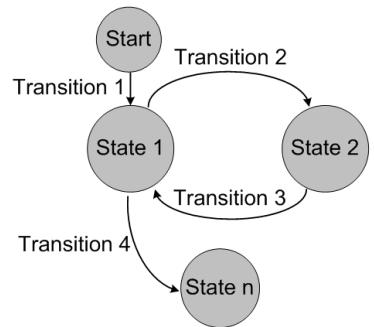
Behavioral



# BIBTEX



## Graphviz



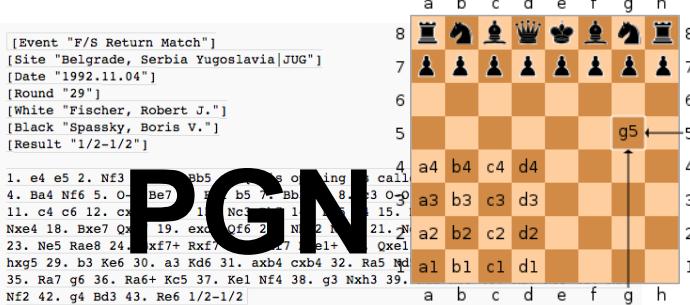
Finite State  
Machine



Domain-Specific Languages (DSLs)

```
[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"]
```

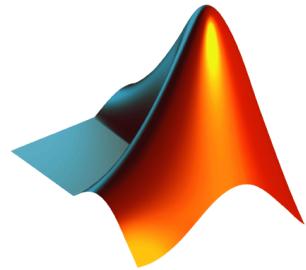
```
1. e4 e5 2. Nf3 Nc6 3. Bb5 a6 4. c3 d5 5. O-O Bb7 6. b3 Bb5 7. Bb7 Nf6 8. e3 0-0 9. Bb5 Nc6 10. d4 Nf6 11. c4 c6 12. c5 Nc6 13. Nc3 Nf6 14. Nf3 Nc6 15. Nf3 Nf6 16. Nf3 Nc6 17. Nf3 Nf6 18. Nf3 Nc6 19. exd5 Nf6 20. Nf3 Nf6 21. Nf3 Nf6 22. Nf3 Nf6 23. Ne5 Rae8 24. Nxf7+ Rxf7 25. Nf3 Nf6 26. Nf3 Nf6 27. Nf3 Nf6 28. Nf3 Nf6 29. b3 Ke6 30. a3 Kd6 31. axb4 cxb4 32. Ra5 Nd5 33. Ra7 g6 34. Ra6+ Kc5 35. Ke1 Nf4 36. g3 Nxh3 37. Ra5 Nd5 38. Kd2 Nf4 39. Ra5 Nd5 40. Ra5 Nd5 41. Ra5 Nd5 42. Ra5 Nd5 43. Re6 1/2-1/2
```



## PGN



## Make



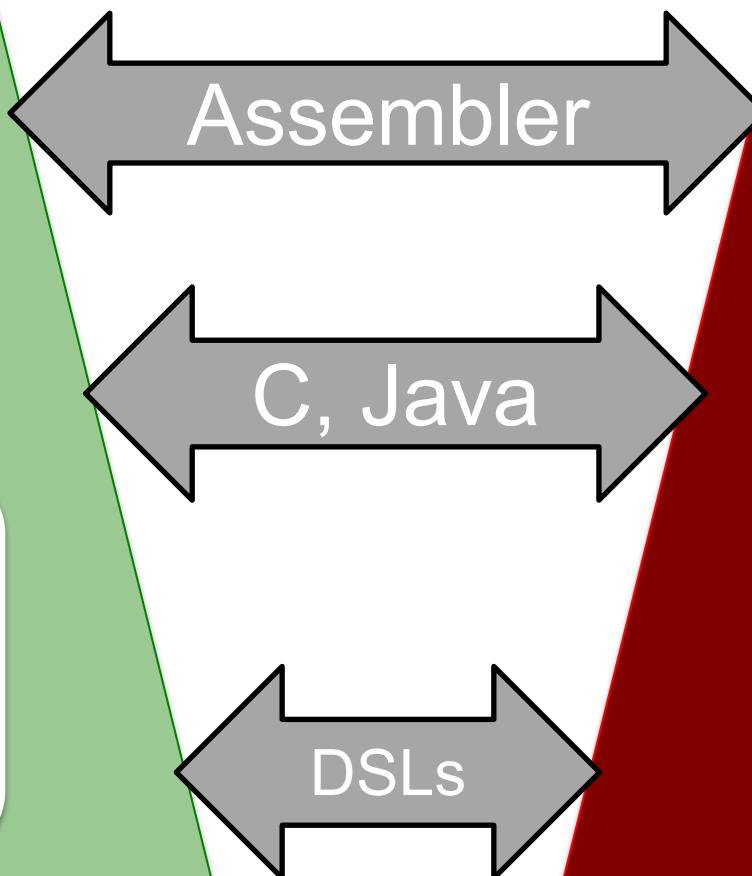
## Matlab



# Abstraction Gap

Problem  
Space

Solution  
Space



Google

twitter



« Another lesson we should have learned from the recent past is that the development of 'richer' or 'more powerful' programming languages was a mistake in the sense that these baroque monstrosities, these conglomerations of idiosyncrasies, are really unmanageable, both mechanically and mentally.

aka General-Purpose Languages

I see a great future for very systematic and very modest programming languages »

1972

aka Domain-Specific Languages

ACM Turing Lecture, « The Humble Programmer »  
Edsger W. Dijkstra

# **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**

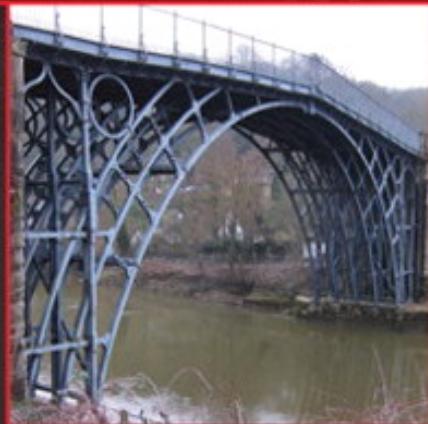
**« Domain-specific  
languages are far more  
prevalent than anticipated »**

*The Addison-Wesley Signature Series*

# DOMAIN-SPECIFIC LANGUAGES

---

MARTIN FOWLER  
WITH REBECCA PARSONS



A MARTIN FOWLER SIGNATURE  
Book Martin Fowler



# 2011



# What is a domain-specific language ?

- « Language **specially** designed to perform a task in a **certain domain** »
- « A formal processable language targeting at a **specific viewpoint or aspect** of a software system. Its **semantics and notation** is designed in order to support working with that viewpoint as good as possible »
- « A computer language that's targeted to a particular kind of problem, **rather than a general purpose language** that's aimed at any kind of software problem. »

# GPL (General Purpose Language)

A GPL provides notations that are used to describe a computation in a human-readable form that can be translated into a machine-readable representation.

A GPL is a formal notation that can be used to describe problem solutions in a precise manner.

A GPL is a notation that can be used to write programs.

A GPL is a notation for expressing computation.

A GPL is a standardized communication technique for expressing instructions to a computer. It is a set of syntactic and semantic rules used to define computer programs.

# Promises of domain-specific languages

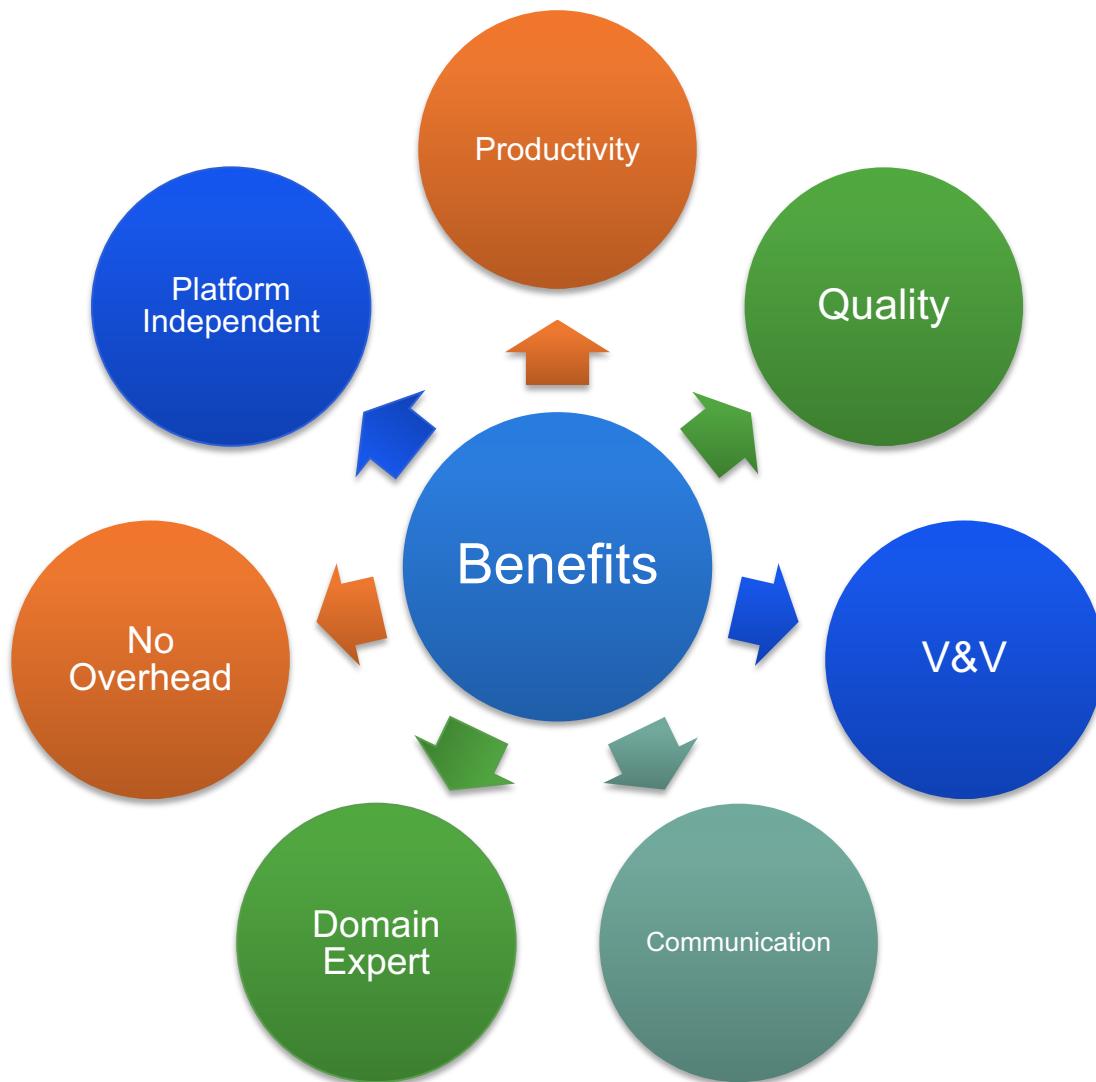
Higher abstractions

Avoid redundancy

Separation of concerns

Use domain concepts

# Promises of domain-specific languages



# GeneralPL vs DomainSL

The boundary isn't as clear as it could be. Domain-specificity is not black-and-white, but instead gradual: a language is more or less domain specific



	GPLs	DSLs
Domain	large and complex	smaller and well-defined
Language size	large	small
Turing completeness	always	often not
User-defined abstractions	sophisticated	limited
Execution	via intermediate GPL	native
Lifespan	years to decades	months to years (driven by context)
Designed by	guru or committee	a few engineers and domain experts
User community	large, anonymous and widespread	small, accessible and local
Evolution	slow, often standardized	fast-paced
Deprecation/incompatible changes	almost impossible	feasible

# External DSLs vs Internal DSLs

- An **external** DSL is a completely separate language and has its own custom syntax/tooling support (e.g., editor)
- An internal DSL is more or less a set of APIs written on top of a host language (e.g., Java).
  - Fluent interfaces

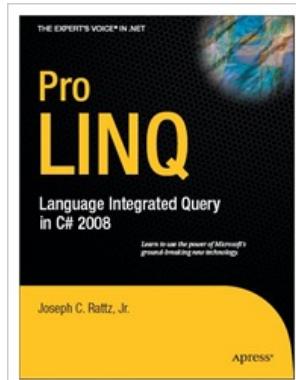
# External vs Internal DSL (SQL example)

```
-- Select all books by authors born after 1920,  
-- named "Paulo" from a catalogue:  
SELECT *  
FROM t_author a  
JOIN t_book b ON a.id = b.author_id  
WHERE a.year_of_birth > 1920  
    AND a.first_name = 'Paulo'  
ORDER BY b.title
```

```
Result<Record> result =  
create.select()  
    .from(T_AUTHOR.as("a"))  
    .join(T_BOOK.as("b")).on(a.ID.equal(b.AUTHOR_ID))  
    .where(a.YEAR_OF_BIRTH.greaterThan(1920))  
    .and(a.FIRST_NAME.equal("Paulo")))  
    .orderBy(b.TITLE)  
    .fetch();
```

# Internal DSL (LINQ/C# example)

```
// DataContext takes a connection string
DataContext db = new DataContext("c:\\northwind\\northwnd.mdf");
// Get a typed table to run queries
Table<Customer> Customers = db.GetTable<Customer>();
// Query for customers from London
var q =
    from c in Customers
    where c.City == "London"
    select c;
foreach (var cust in q)
    Console.WriteLine("id = {0}, City = {1}", cust.CustomerID, cust.City);
```



# Internal DSL

- « Using a host language (e.g., Java) to give the host language the feel of a particular language. »
- Fluent Interfaces
  - « The more the use of the API has that language like flow, the more fluent it is »

```
Result<Record> result =
    create.select()
        .from(T_AUTHOR.as("a"))
        .join(T_BOOK.as("b")).on(a.ID.equal(b.AUTHOR_ID))
        .where(a.YEAR_OF_BIRTH.greaterThan(1920))
        .and(a.FIRST_NAME.equal("Paulo")))
        .orderBy(b.TITLE)
        .fetch();
```

```
-- Select all books by authors born after 1920,
-- named "Paulo" from a catalogue:
SELECT *
FROM t_author a
JOIN t_book b ON a.id = b.author_id
WHERE a.year_of_birth > 1920
AND a.first_name = 'Paulo'
ORDER BY b.title
```

# SQL in... Java

## DSL in GPL

```
Connection con = null;

// create sql insert query
String query = "insert into user values(" + student.getId() + ","
    + student.getFirstName() + "','" + student.getLastName()
    + "','" + student.getEmail() + "','" + student.getPhone()
    + "')";

try {
    // get connection to db
    con = new CreateConnection().getConnection("checkjdbc", "root",
        "root");

    // get a statement to execute query
    stmt = con.createStatement();

    // executed insert query
    stmt.execute(query);
    System.out.println("Data inserted in table !");
}
```

# Regular expression in...

## Java

### DSL in GPL

```
public class RegexTestStrings {  
    public static final String EXAMPLE_TEST = "This is my small example "  
        + "string which I'm going to " + "use for pattern matching.";  
  
    public static void main(String[] args) {  
        System.out.println(EXAMPLE_TEST.matches("\\w.*"));  
        String[] splitString = (EXAMPLE_TEST.split("\\s+"));  
        System.out.println(splitString.length); // Should be 14  
        for (String string : splitString) {  
            System.out.println(string);  
        }  
        // Replace all whitespace with tabs  
        System.out.println(EXAMPLE_TEST.replaceAll("\\s+", "\t"));  
    }  
}
```

# Terminology

- Traditional dichotomy between internal DSL and external DSL (Fowler et al., 2010)
  - Fluent APIs
  - Internal DSLs
  - (deeply) embedded DSLs
  - External DSLs
- Boundary between DSL and GPL is not that clear (Voelter et al., 2013)
  - What is and what is not a DSL is still a debate

# Internal DSLs vs External DSL

- Both internal and external DSLs have strengths and weaknesses
  - learning curve,
  - cost of building,
  - programmer familiarity,
  - communication with domain experts,
  - mixing in the host language,
  - strong expressiveness boundary
- Focus of the course
  - **external DSL:** a completely separate language with its own custom syntax and tooling support (e.g., editor)

# Plan

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

# Contract

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

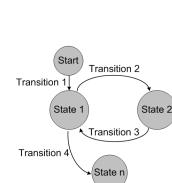
# DSL = Syntax + Services

## Specialized notation:

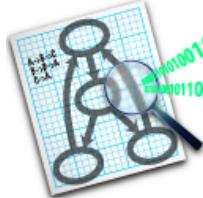
Textual or Graphical  
Specific Vocabulary  
Idiomatic constructs

## Specialized tools/IDE:

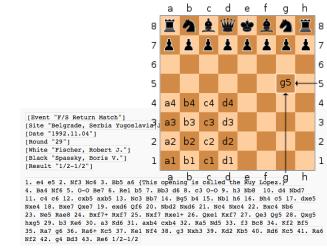
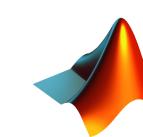
Editor with auto-completion, syntax highlighting, etc.  
Compiler  
Interpreter  
Debugger  
Profiler  
Syntax/Type Checker  
...



BIBT<sub>E</sub>X



SQL



# 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)

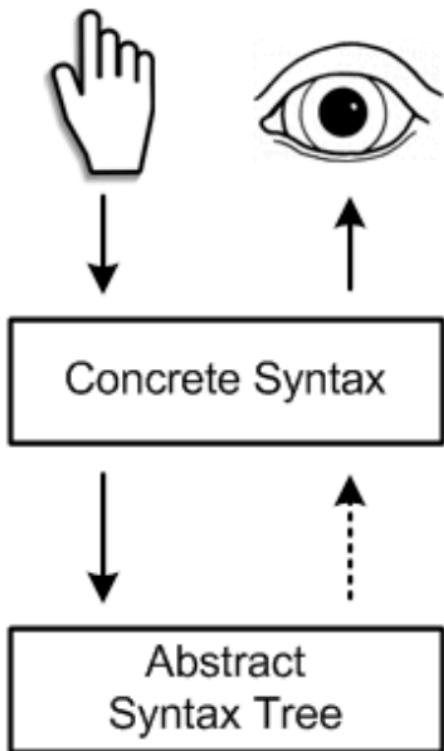
The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** Java – strategoxt-sugar-papers/test/BookHandler.sugj – Eclipse – /Users/seba/tmp/ecli...
- Toolbar:** Includes icons for file operations, search, and transform.
- Left Panel (Problems View):** Shows 1 error and 1 warning.
  - Error:** expected element edition of namespace lib (line 18)
  - Warning:** skipping validation of quoted attribute value (line 14)
- Central Editor:** Displays the XML schema code for BookHandler.sugj. The code includes imports for xml.Sugar, xml.Editor, and xml.schema.BookSchema, and defines a public class BookHandler with a method appendBook that takes a ContentHandler parameter and throws SAXException. The schema includes elements for book, author, editions, edition, and edit, with specific attributes like title, name, year, and publisher.
- Outline View:** Shows the structure of the schema, including the BookHandler class and its nested elements and attributes.
- Bottom Status Bar:** Shows "Writable" and "Smart Insert".

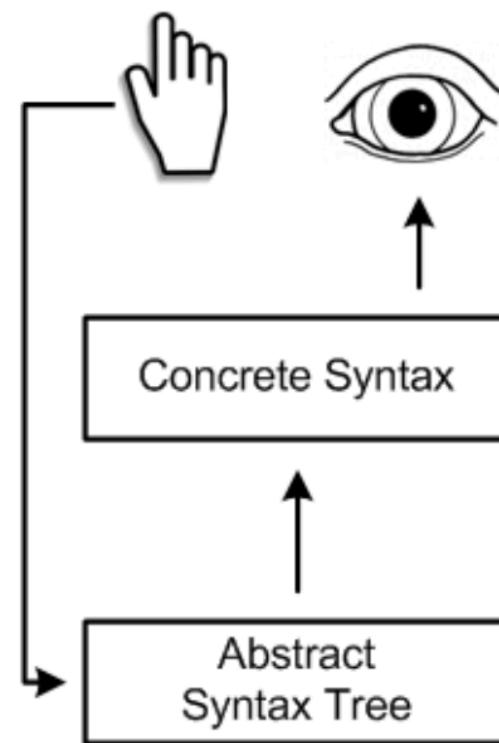
Sebastian Erdweg, Tillmann Rendel, Christian Kästner, and Klaus Ostermann. Sugarj: Library-based syntactic language extensibility. OOPSLA'11

# 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 {
    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
landing    [tp->speed == 0 mps] -> landed
            [tp->speed > 0 mps] -> landing
landed     [ ] -> beforeFlight
crashed    [ ] -> beforeFlight
}
```



```

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 M
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 expected"
16     and e2 : BoolType()
17       else error "bool expected"
18
19 Eq(e1, e2) + Gt(e1, e2) :
20   where e1 : IntType()
21     else error "int expected"

generate.str
6
7 sm-to-java :
8   machine@Machine(m, exten
9   public class [m] [<ext
10  String current = [<
11    [vardefs]
12
13  String next(String e
14    [cond-stat*]
15    while(true) {
16      [uncond-stat*]
17    }
18  ]
19  ]
20
21 where ...
22
23
24

VendingMachine.ATOML
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]

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.

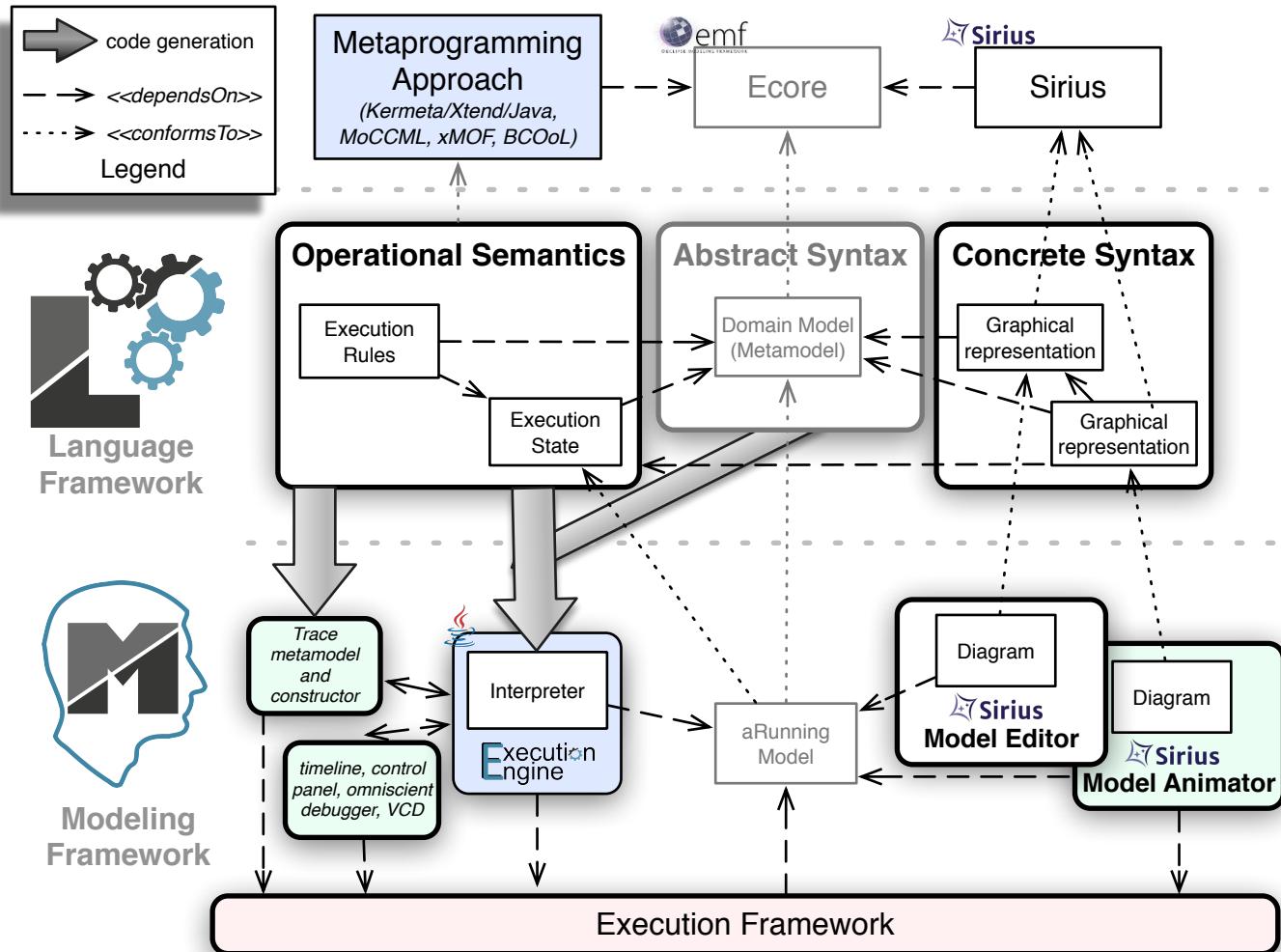
With the Spoofax language workbench, you can write the grammar of your language using the high-level SDF grammar formalism. Based on this grammar, basic editor services such as syntax highlighting and code folding are automatically provided. Using high-level descriptor languages, these services can be 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

# GEMOC Studio



EMF, a popular, open source,  
easy-to-use modeling  
framework for developping  
DSLs

Your domain model in 5'

# Eclipse Modeling: Overview

- Eclipse Modeling is the umbrella project for **all things about modeling** that happen on the Eclipse platform:

*The Eclipse Modeling Project (EMP) focuses on the evolution and promotion of model-based development technologies within the Eclipse community by providing a unified set of modeling frameworks, tooling, and standards implementations.*

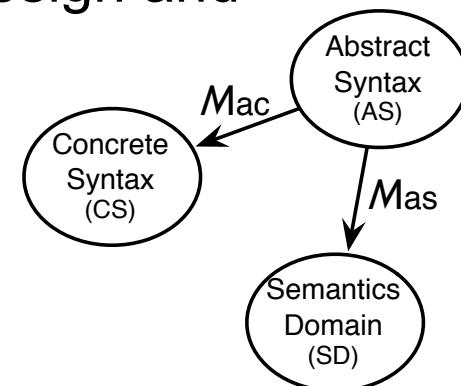
- Eclipse Modeling is **not formally related to OMG**, but implements several of their standards.
- It is fair to say that **many leading edge modeling** tools are hosted/developed at Eclipse Modeling.
- Everything **Open Source** under the Eclipse Public License

# Eclipse Modeling: Overview

The answer to "What is Eclipse Modeling?" depends on who you ask!

A set of Eclipse projects dedicated to...

- ... **Modeling**: modeling tools
  - Model Development Tools (UML2, OCL, SysML, MARTE, BPMN2, etc.)
- ... **Metamodeling**: workbench for language design and implementation
  - Abstract Syntax Development (EMF)
  - Concrete Syntax Development (GMP, TMF)
  - Model Transformation (M2M, M2T)
- See <http://www.eclipse.org/modeling>



# Eclipse Modeling



```

culture corn {
    activity LABOUR from 1 jan to 28 feb
    using 1 Tractor and 1 People

    activity SEMIS from 15 mar to 15 apr [
        after LABOUR && no rain since 3 days && temperature > 10 °C
    ] using 1 Tractor and 2 People

    activity IRRIGATION weekly from 15 jun to 15 aug [
        after SEMIS
    ] using 1 Tractor and 1 People

    activity FERTILISATION from 15 mar to 15 jun [
        after SEMIS is done since 30 days &&
        no rain since 1 days
    ] using 1 Tractor and 1 People

    activity RECOLTE from 1 sept to 30 sept [
        grain is "mature"
    ] using 1 Tractor and 2 People
}

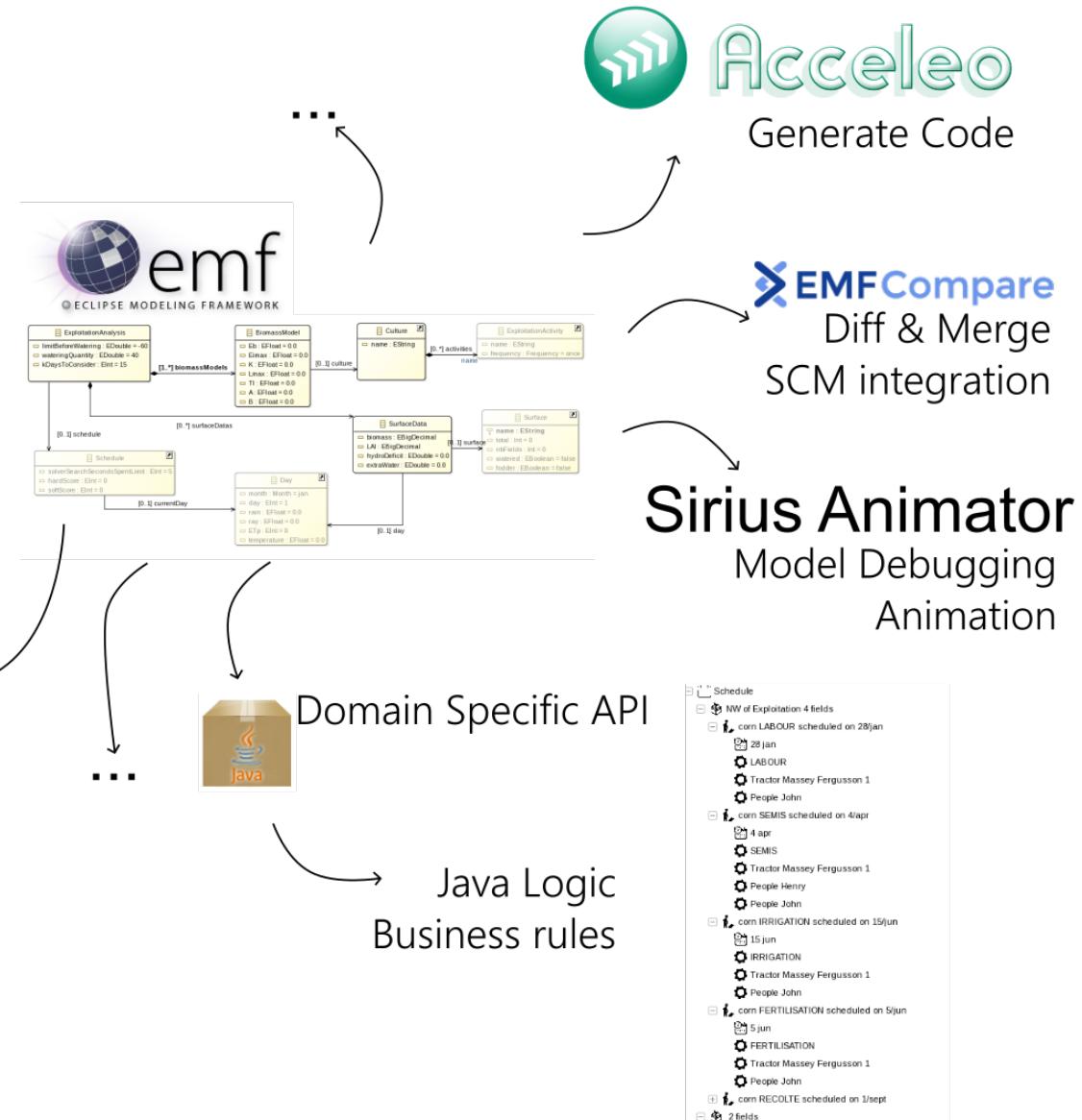
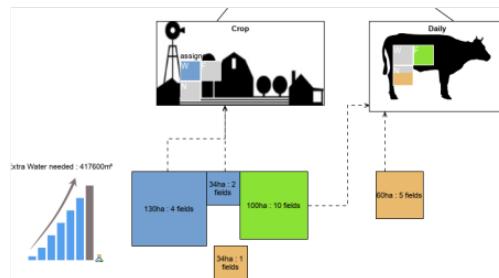
culture wheat {
    activity LABOUR from 1 sept to 30 sept [
        no rain since 3 days
    ] using 1 Tractor and 1 People

    activity SEMIS from 1 oct to 31 oct [
        after LABOUR &&
        no rain since 3 days &&
        temperature > 5°C
    ] using 1 Tractor and 1 People

    activity FERTILISATION from 1 feb to 28 feb [
        after SEMIS is done since 30 days &&
        no rain since 1 days
    ] using 1 Tractor and 1 People

    activity RECOLTE from 1 jun to 30 jun [
        grain is "mature"
    ] using 1 Tractor and 1 People
}

```

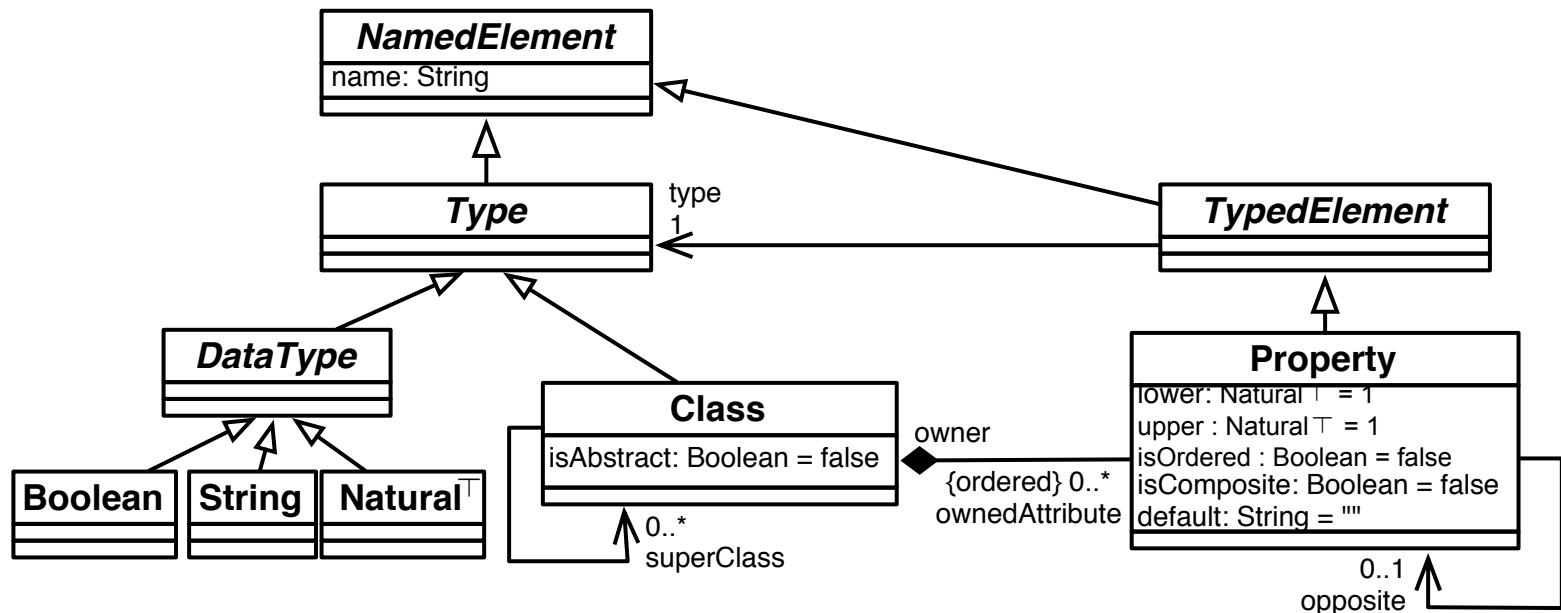


# EMF: Overview

- What is it?
  - **MetaModeling** (think of UML/OCL)
  - Interoperability (think of XMI)
  - Editing tool support (think Eclipse)
  - Code generation (think of MDA)
- EMF serves as the foundation: It provides the Ecore metamodel, and frameworks and tools around it for tasks such as
  - Editing
  - Transactions
  - Validation
  - Query
  - Distribution/Persistence (CDO, Net4j, Teneo)
- See <http://www.eclipse.org/modeling/emf>

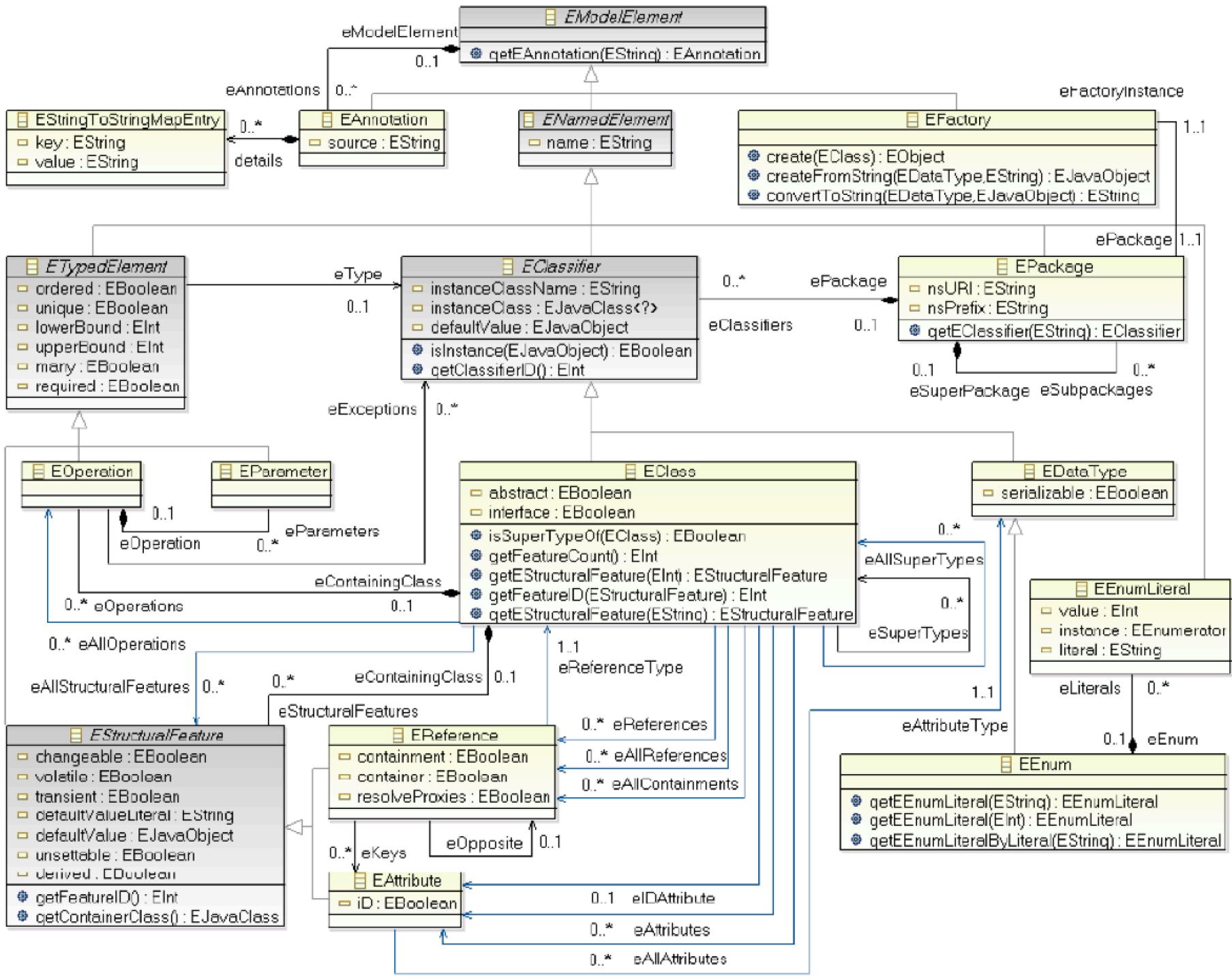
# OMG (Essential) MOF

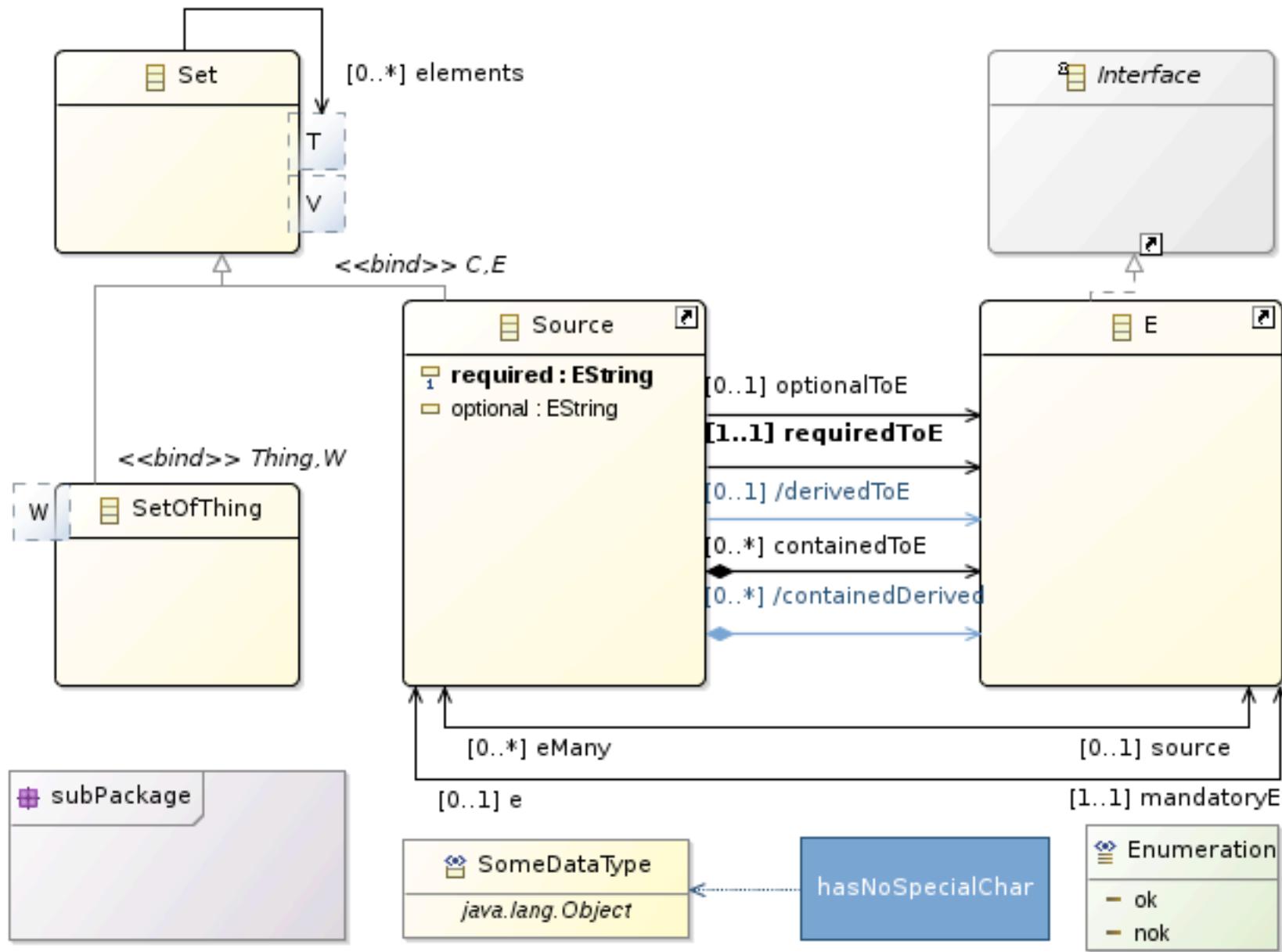
- Provides language constructs for specifying a DSL metamodel
  - mainly based on Object-Oriented constructs: *package*, *classes*, *properties* (*attribute* and *reference*), and (multiple) *inheritance*.
  - specificities: composition, opposite...
- Defined as a model, called *metametamodel*:



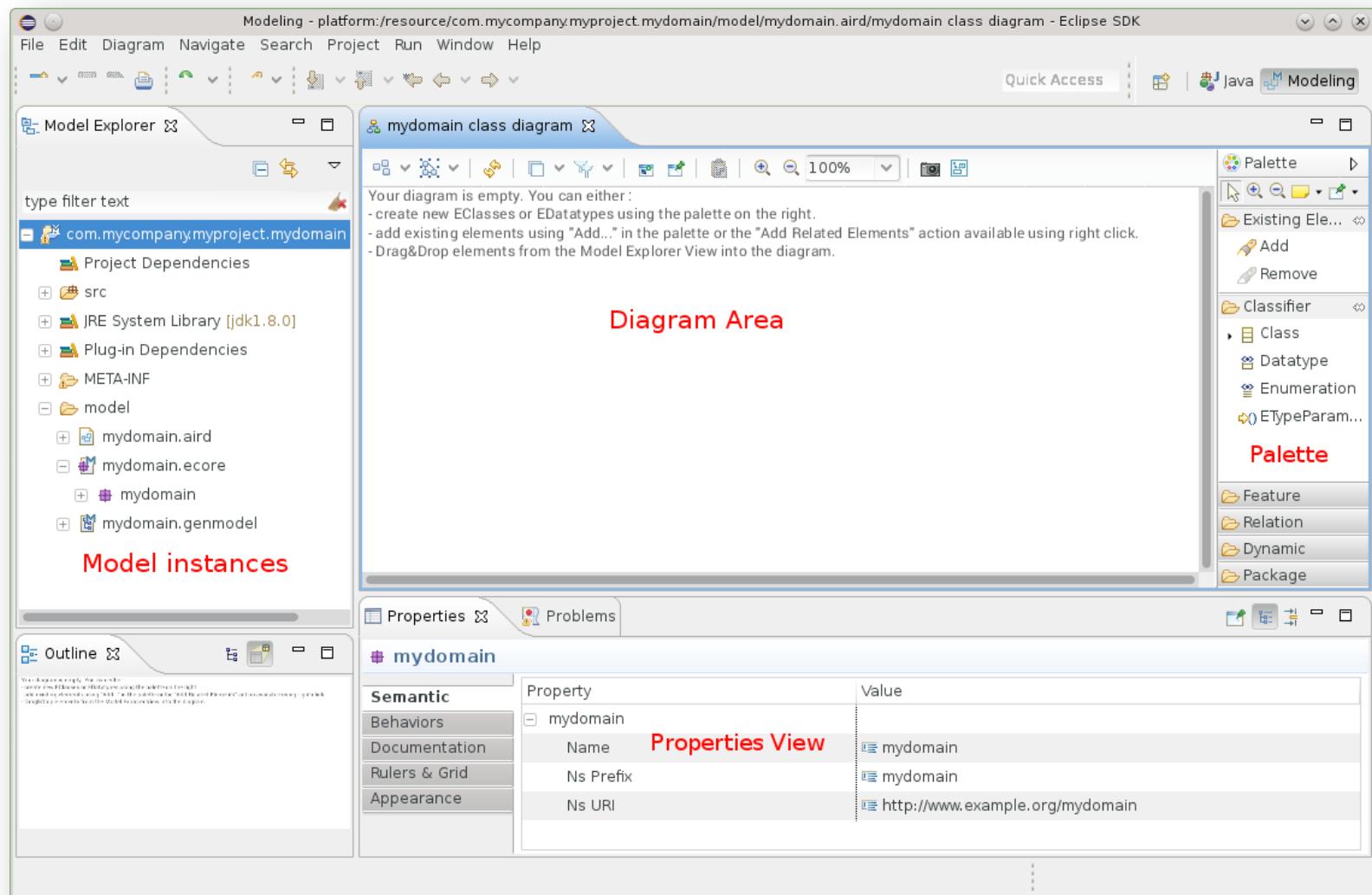
# Ecore: a metamodel for metamodels

- Ecore is an implementation proposed by EMF, and aligned to EMOF
- Provides a language to build languages
- A metamodel is a model; and its metamodel is Ecore.
  - So a metamodel is an Ecore model!
- Ecore has concepts like:
  - Class – inheritance, have properties
  - Property – name, multiplicity, type
- Essentially this is a simplified version of class modeling in UML

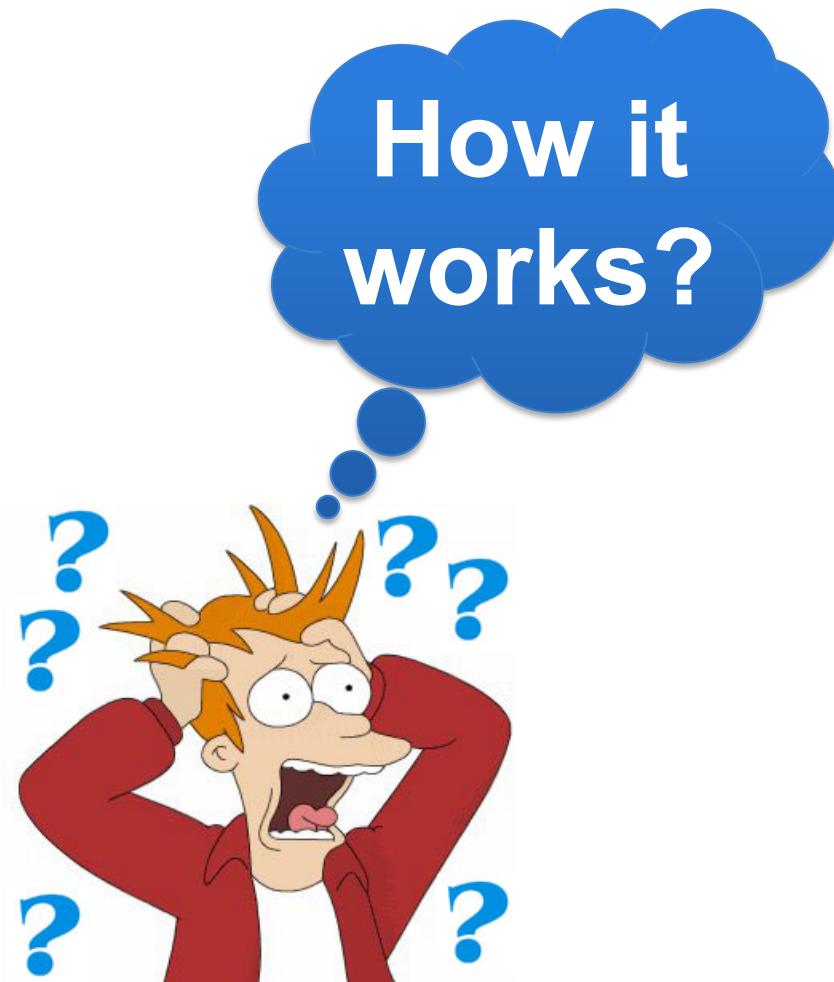
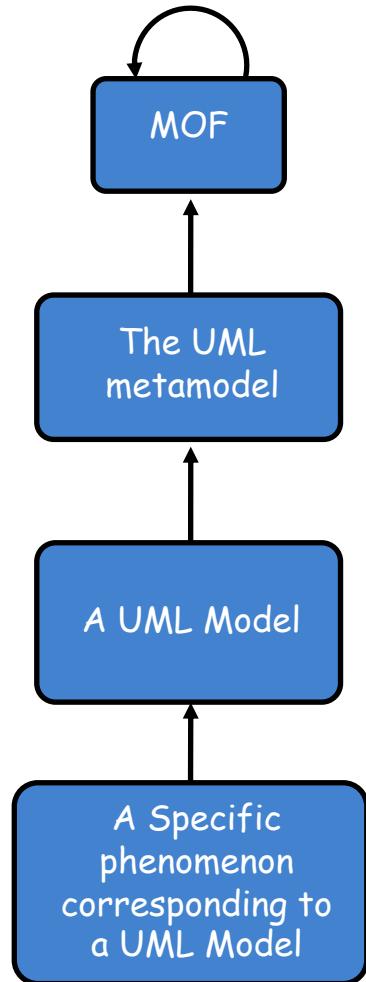




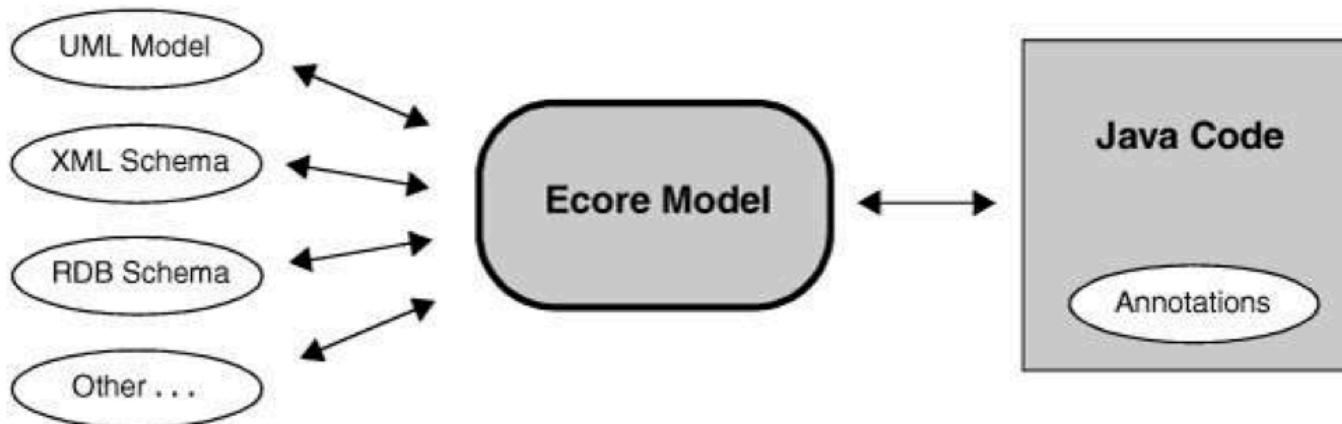
# Ecore Tools



# Implementation with Java



# EMF

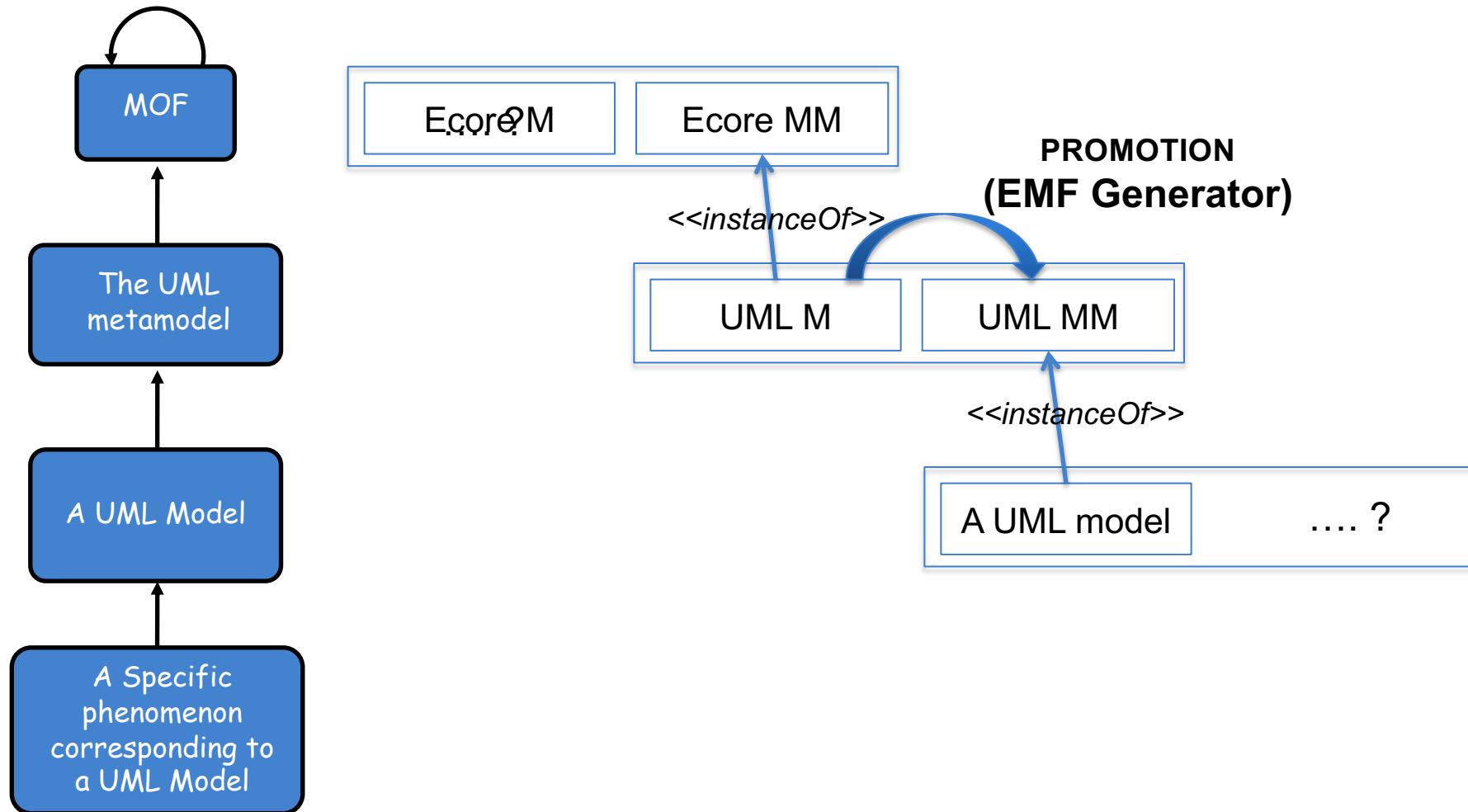


An Ecore model and its sources  
(from *EMF: Eclipse Modeling Framework 2<sup>nd</sup>*)

# Implementation with Java

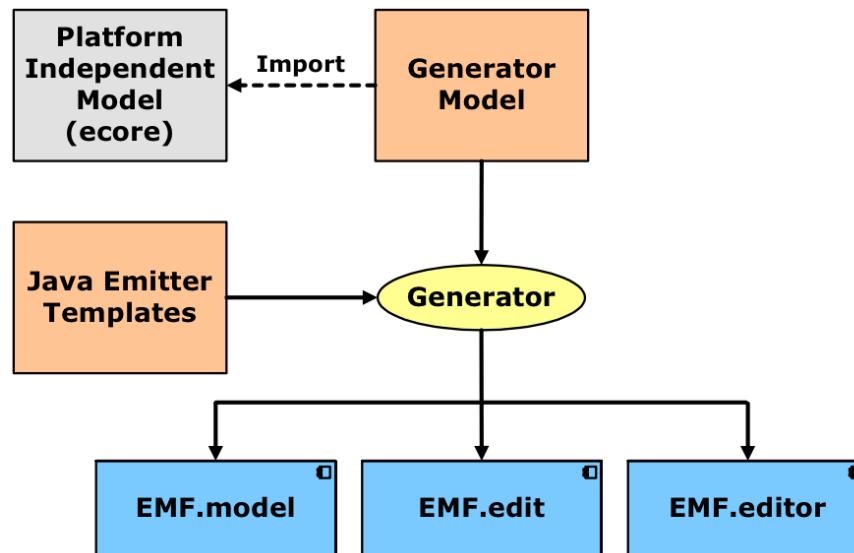
- EMF is a software (E)framework
- Model driven..., but implemented using a programming language!
- Reification MDE → Java:
  - Metamodels are represented with EClasses
  - Models are represented with EObjects

# Implementation with Java

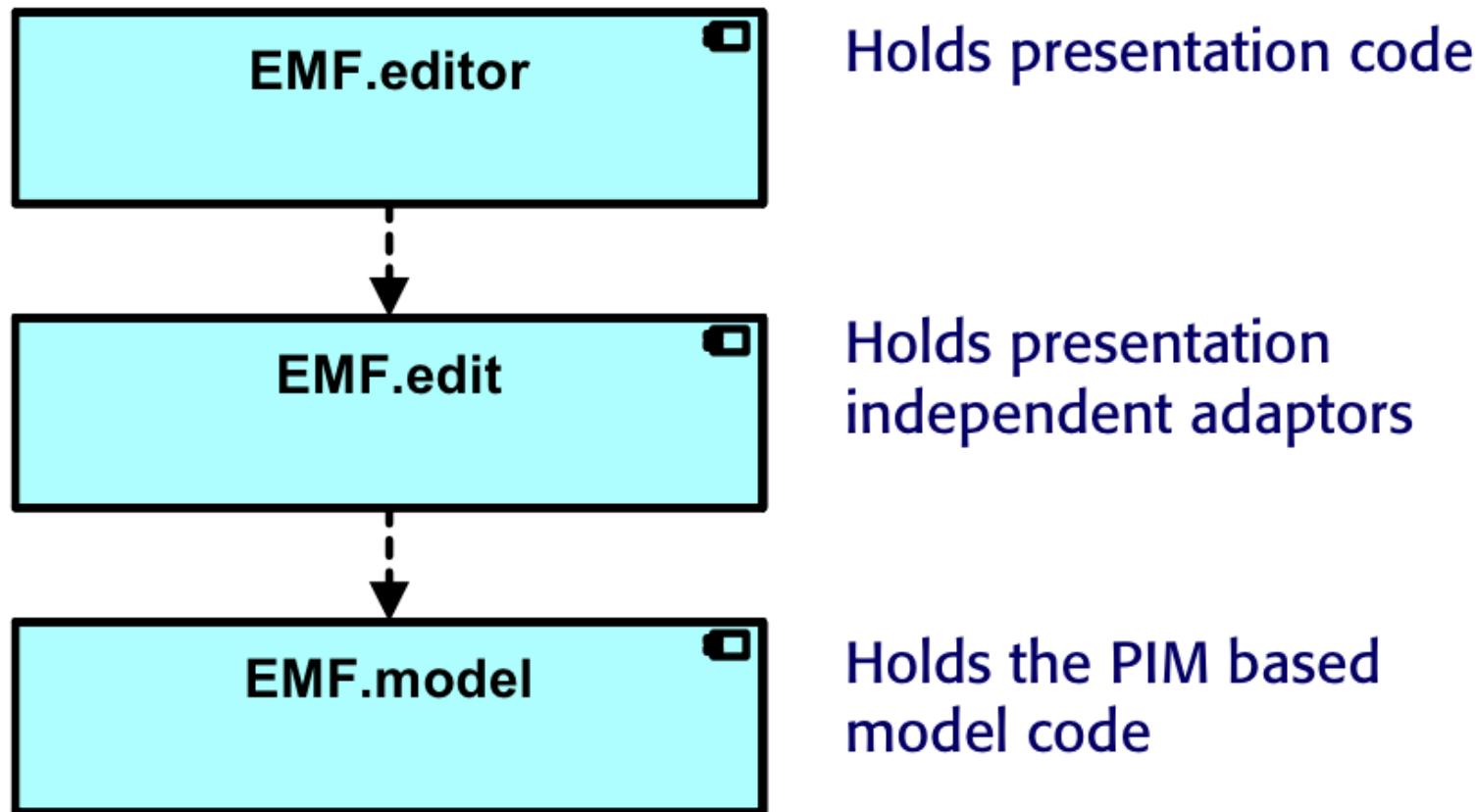


# EMF Toolset from 30.000 Feet

- The EMF Generator do not work on the .ecore
- EMF defines a .genmodel in parallel:
  - New/ Other/ Eclipse Modeling Framework/ EMF Model
  - We can customize the code generator!
  - The IDE takes care of maintaining the consistency (or not!)



# EMF Toolset from 30.000 Feet

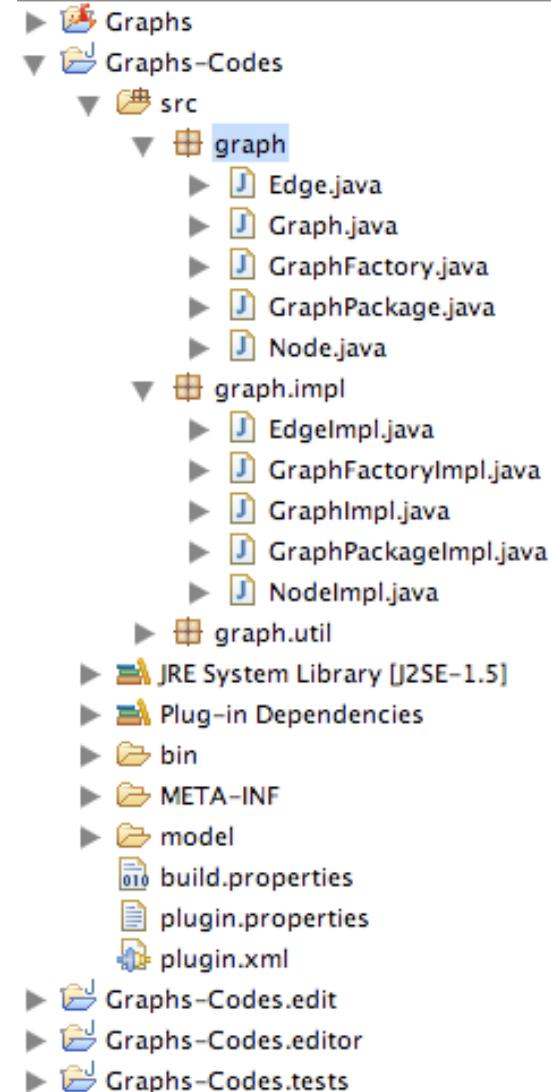


# EMF Toolset from 30.000 Feet

Actions available on the metamodel:

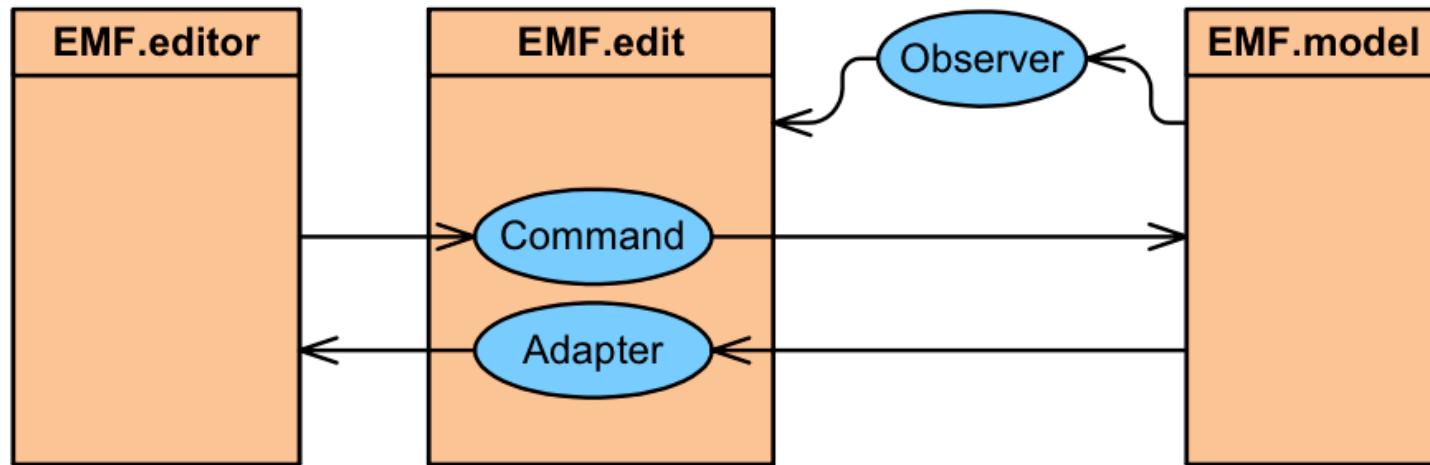
1. *Generate Model Code*: Java Classes corresponding to the metamodel
2. *Generate Edit Code*: Plugin supporting the edition
3. *Generate Editor Code*: Plugin for a tree based model editor
4. *Generate Test Code*: Plugin for unit testing

Actions available from the .genmodel, and into an EMF Project.



# EMF: open the box

- The EMF.edit separates the GUI from the business model
- To understand the EMF.edit plug-in, it is essential to understand three basic design patterns
  - Observer pattern
  - Command pattern
  - Adapter pattern



# EOperation Implementation

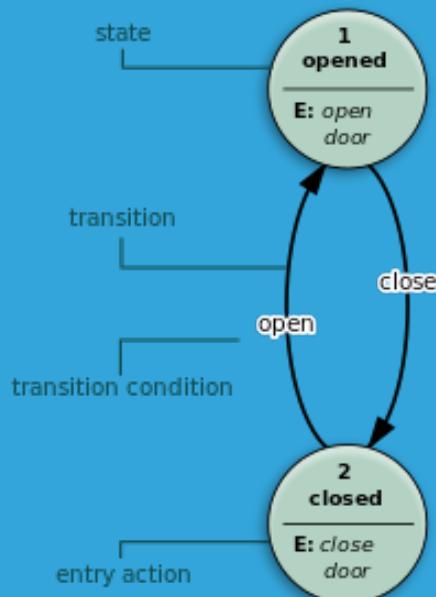
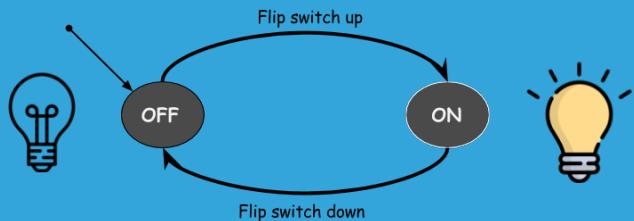
Localization of the methods in the generated code

1. In the subpackage graph.impl
2. In the class GraphImpl
3. Scattered in the code automatically generated by EMF...

```
/**  
 * @generated NOT  
 */  
public int order () {  
    return this.getEdges ().size ();  
}
```

Do not forget to mark (@generated NOT) to prevent crushing!

Part 1: define a metamodel  
to help you developing  
state machines...



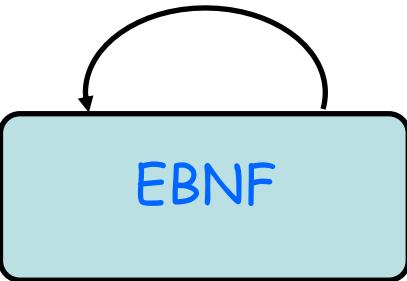
KEEP  
CALM  
AND  
DO IT  
YOURSELF

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

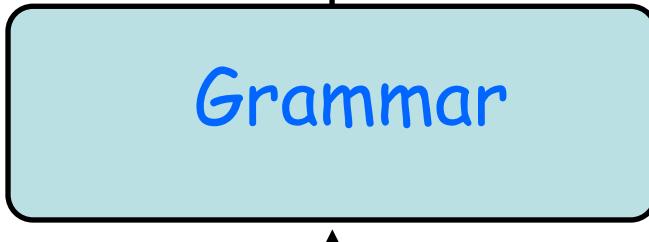
Your textual DSL in 5'  
(incl. editors, serializers)

# Foundations (or some course refresh)

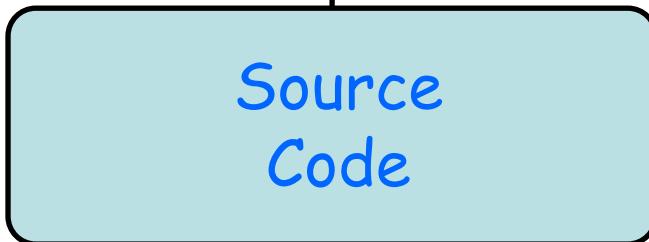
M<sup>3</sup>



M<sup>2</sup>



M<sup>1</sup>



Java Grammar

```
CHARLITERAL
: '\'' 
| '\"' EscapeSequence
| '\"'
| '\\'
;

STRINGLITERAL
: '\"' 
| '\'' EscapeSequence
| '\"'*
| '\''
;

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

variableModifiers
: (
    FINAL
    | annotation
)*
;

classDeclaration
: normalclassDeclaration
| enumDeclaration
```

Java Program

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

```
classOrInterfaceDeclaration
: classDeclaration
| interfaceDeclaration
;

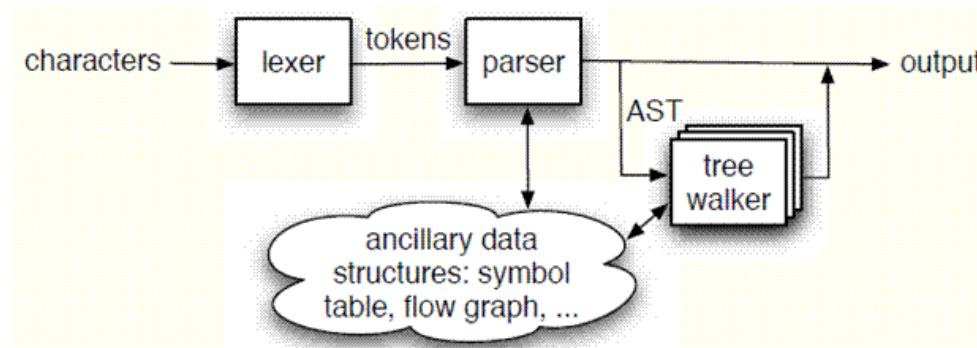
modifiers
: (
    annotation
    PUBLIC
    PROTECTED
    PRIVATE
    STATIC
    ABSTRACT
    FINAL
    NATIVE
    SYNCHRONIZED
    TRANSIENT
    VOLATILE
    STRICTFP
)*
;

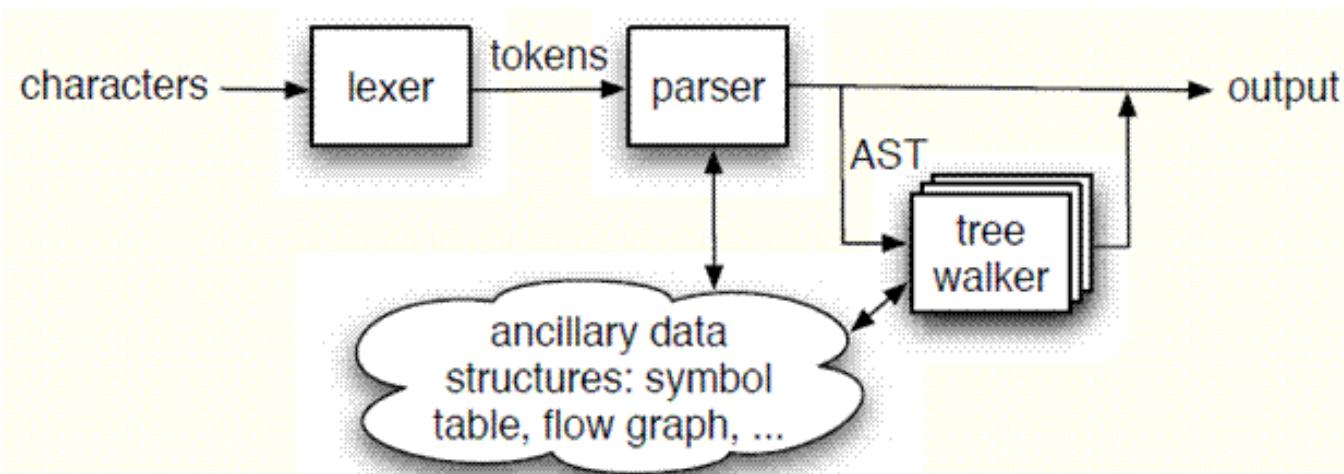
variableModifiers
: (
    FINAL
    | annotation
)*
;

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'
|   '\"'
)
;
```

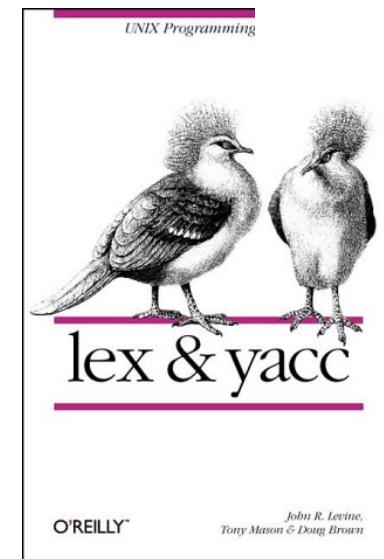
```

classOrInterfaceDeclaration
:   classDeclaration
|   interfaceDeclaration
;

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

variableModifiers
:   (
    FINAL
|   annotation
)*
;

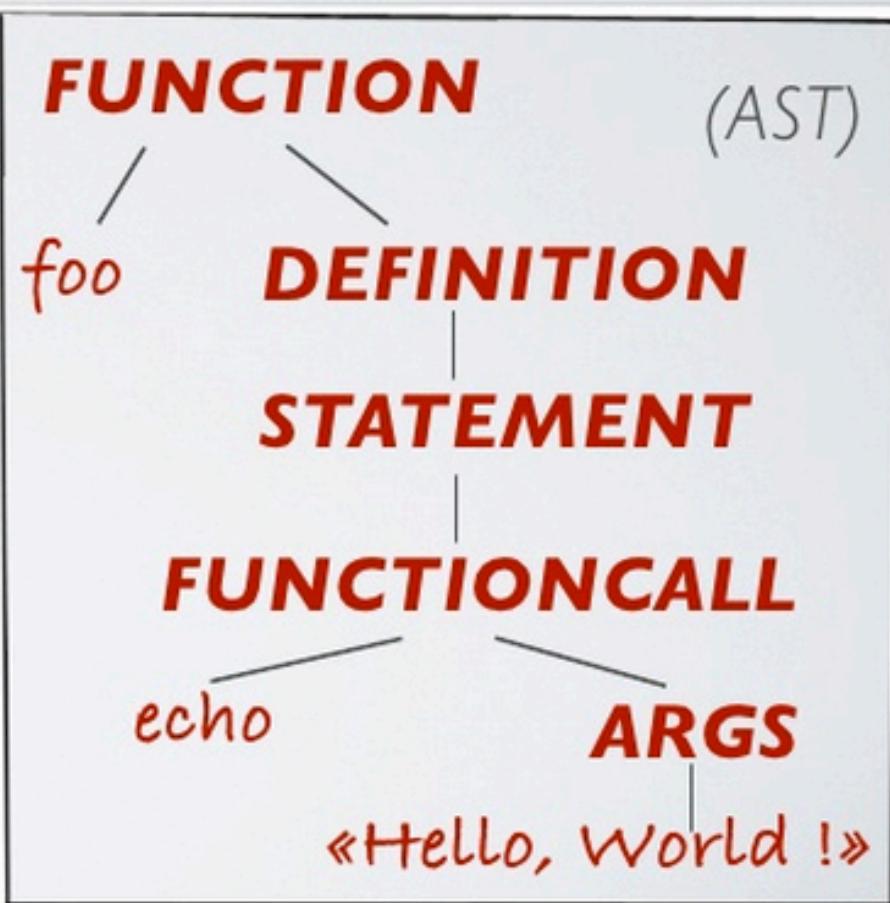
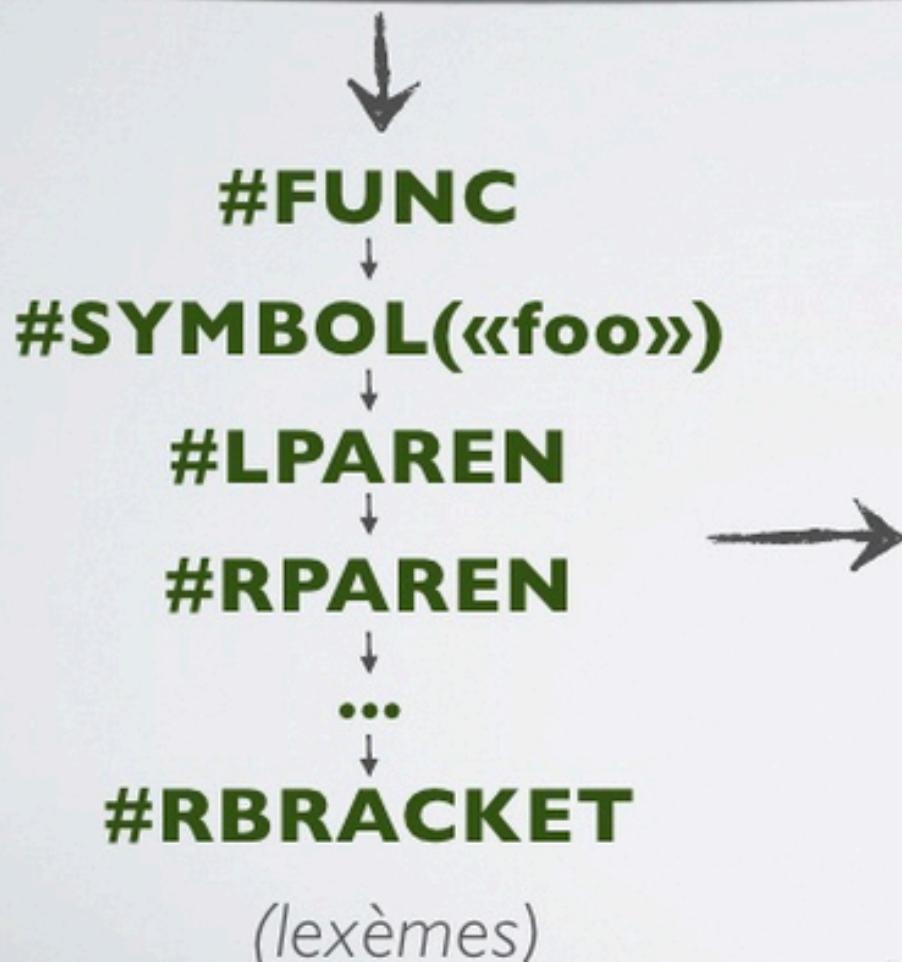
classDeclaration
:   normalClassDeclaration
|   enumDeclaration
;
```



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}")
}

```

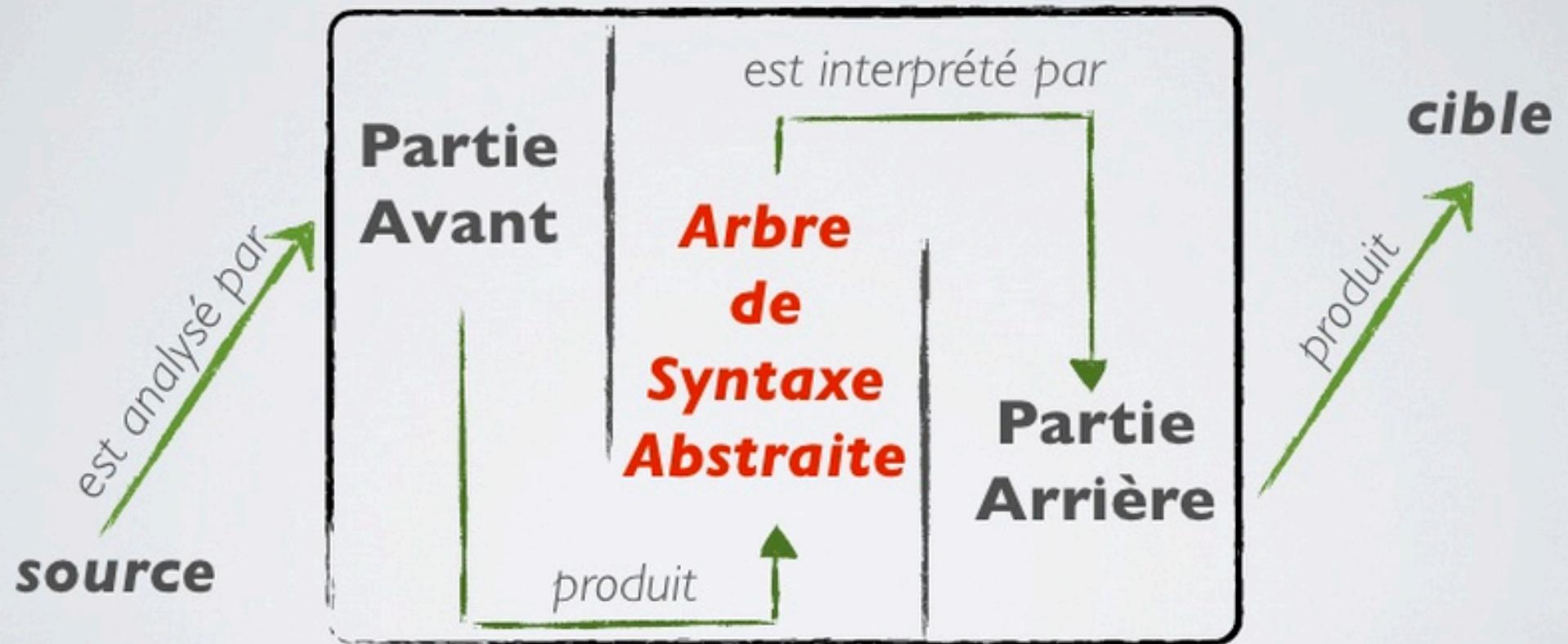
## Scala AST (example)

```

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"))),
    ))), Literal(Constant(())))
)

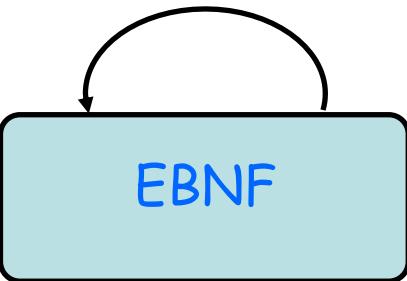
```

# Compilation (en français)

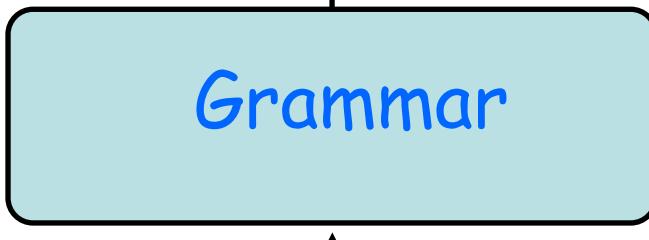


# DSL? The same!

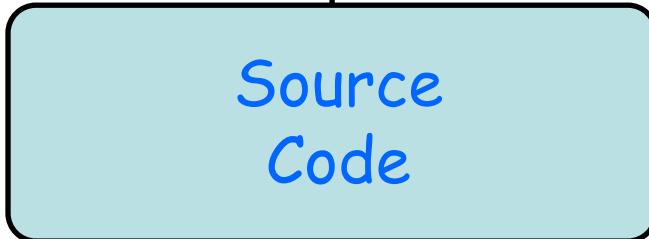
M<sup>3</sup>



M<sup>2</sup>

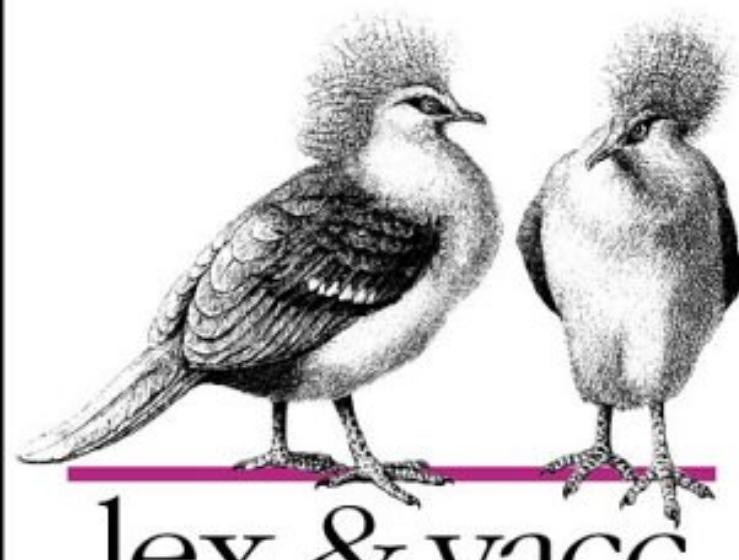


M<sup>1</sup>



DSL  
specification/program

*UNIX Programming Tools*



# lex & yacc

O'REILLY™

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

The  
Pragmatic  
Programmers

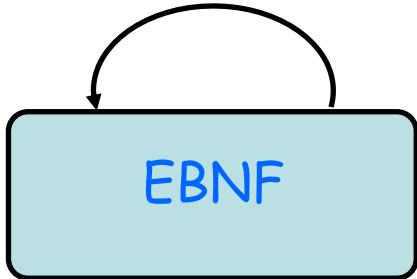
## The Definitive **ANTLR** Reference

Building Domain-  
Specific Languages

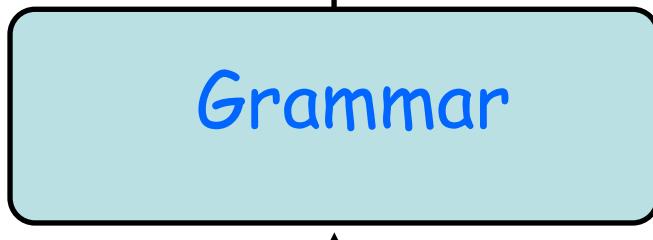


Terence Parr

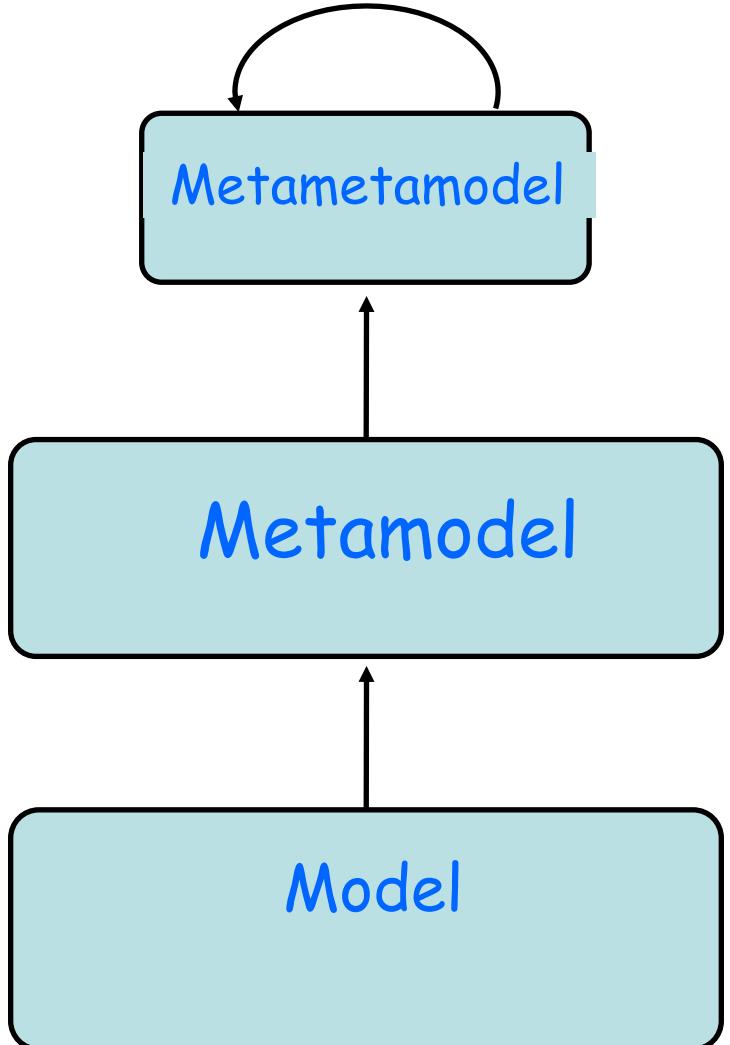
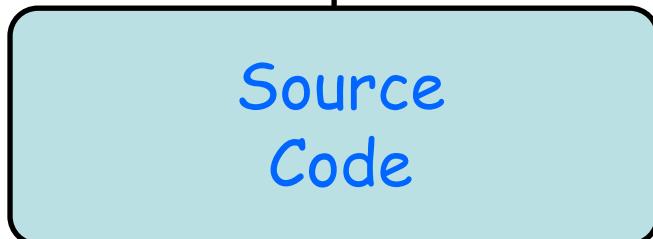
M<sup>3</sup>



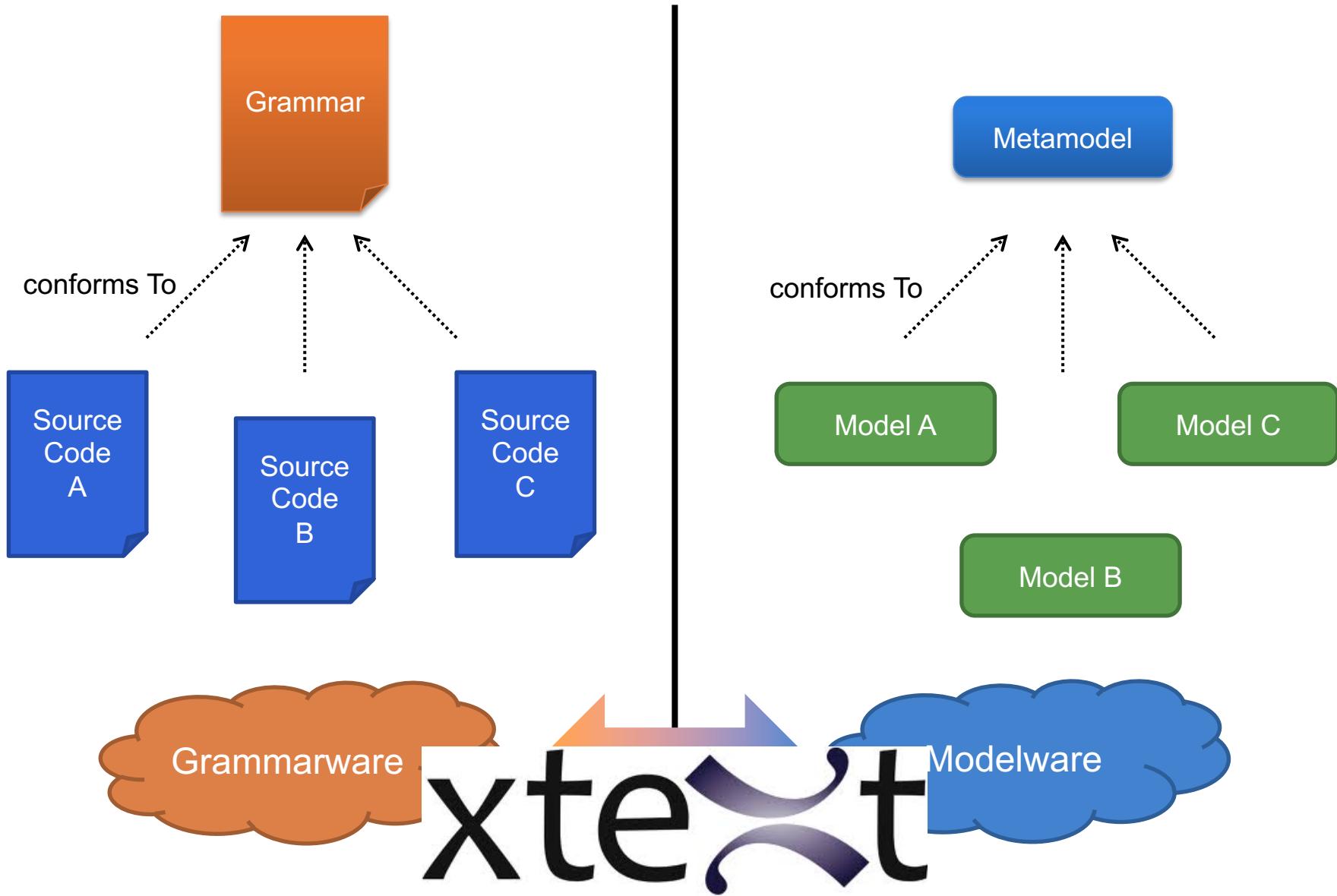
M<sup>2</sup>



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)

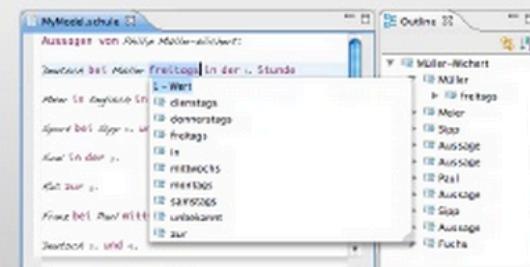
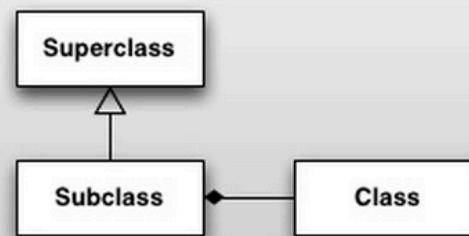
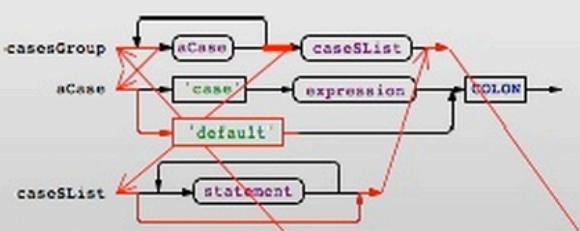
Model

Grammar

Xtext  
Generator



## Xtext Runtime

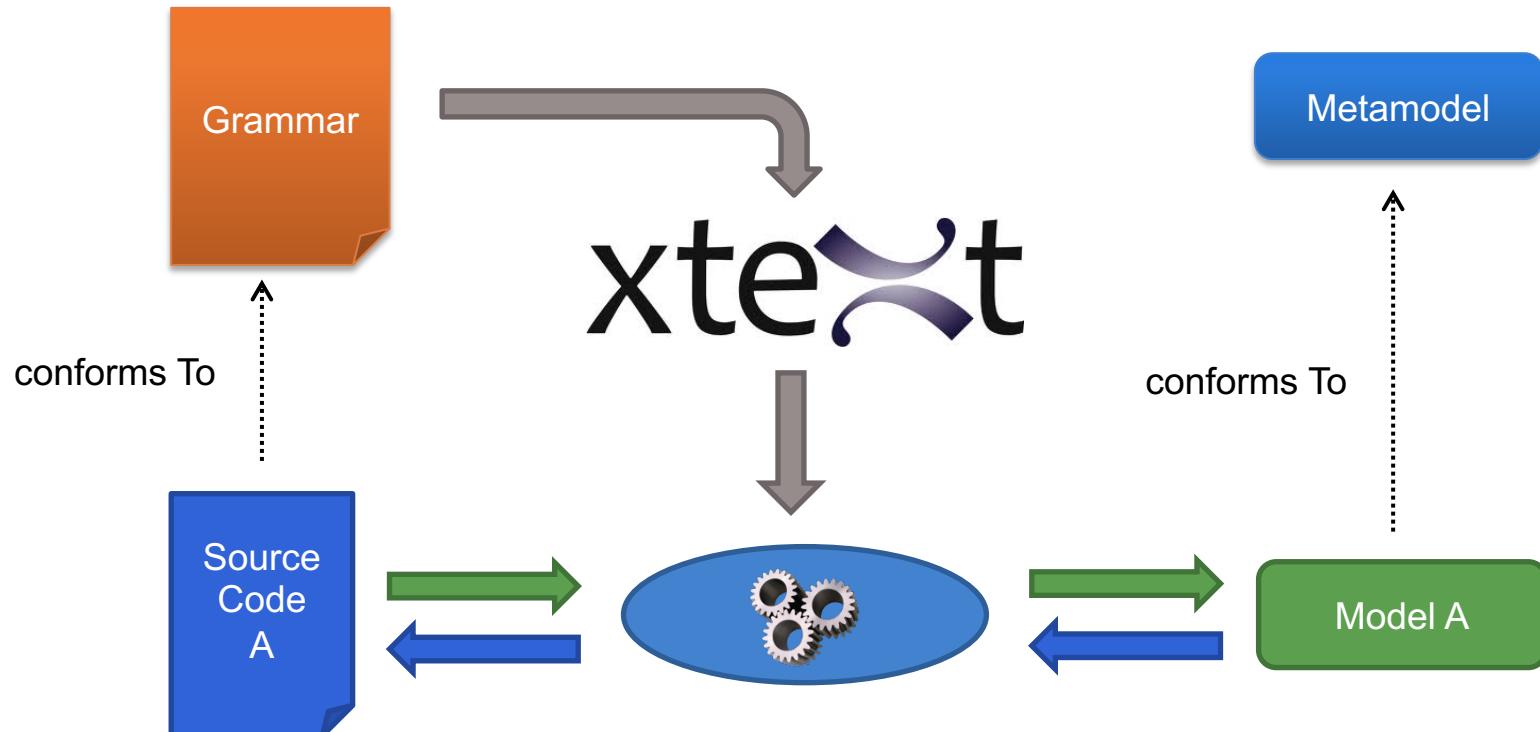


LL(\*) Parser

ecore meta model

editor

# Xtext, Grammar, Metamodel

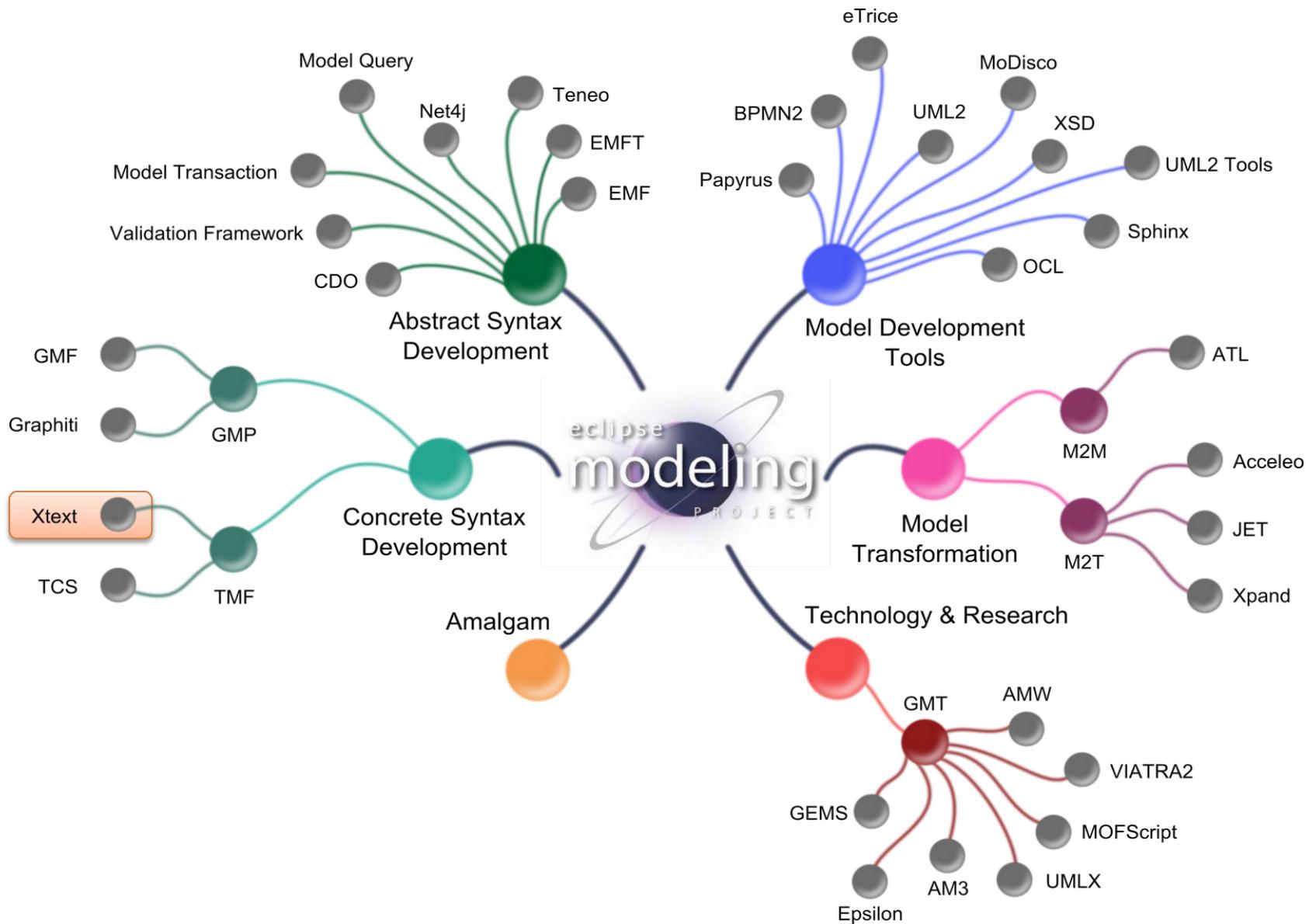


# Xtext Project

- 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



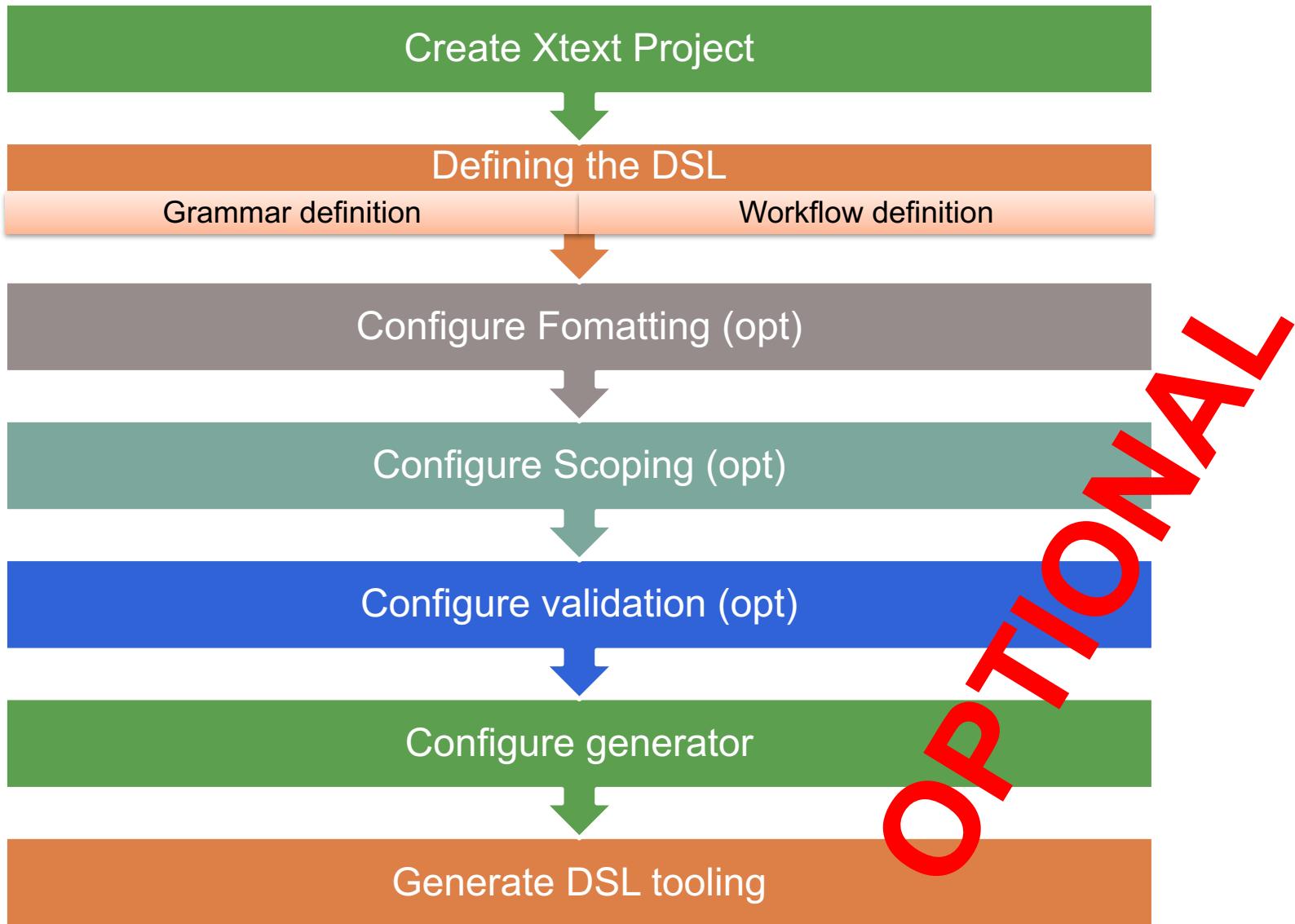
# The Grammar Language of Xtext

- 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

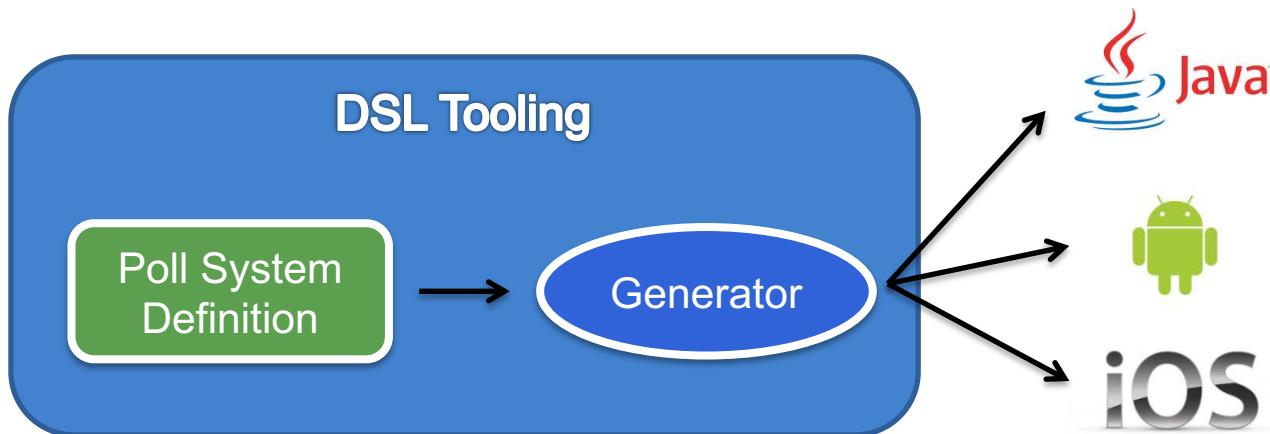
- Consistent look and feel
- Textual DSLs are a resource in Eclipse
- Open editors can be extended
- Complete framework to develop DSLs
- Easy to connect to any Java-based language

# Development Process



# Motivating Scenario

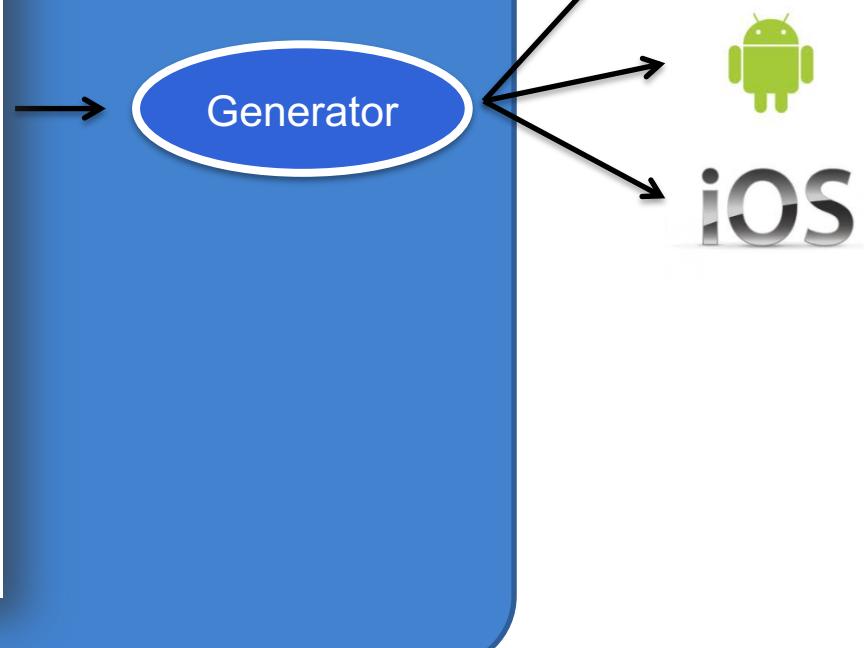
- Poll System application
  - Define a Poll with the corresponding questions
  - Each question has a text and a set of options
  - Each option has a text
- Generate the application in different platforms



# Motivating Scenario (2)

## 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"  
            }  
        }  
    }  
}
```

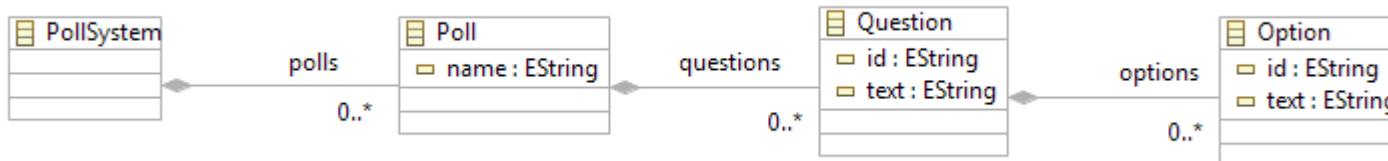


# Grammar Definition

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;
```



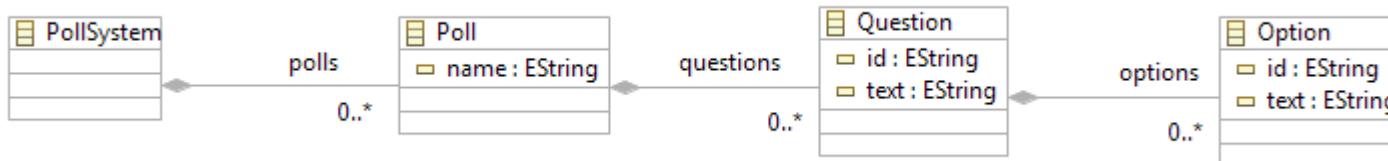
# Grammar Definition

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;
```

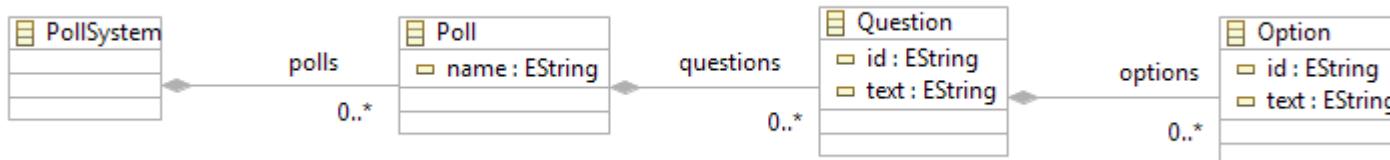


# Grammar Definition

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;
```

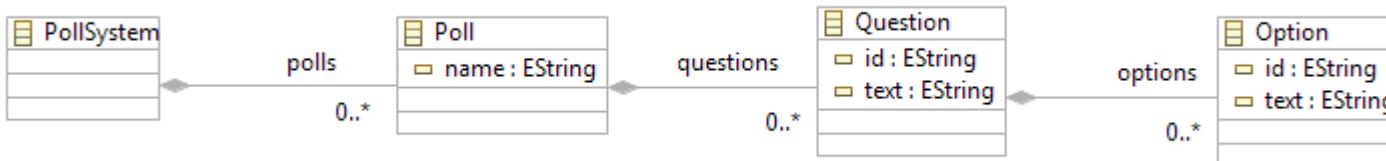


# Grammar Definition

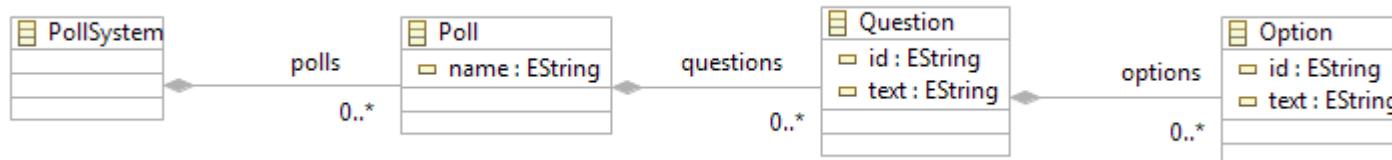
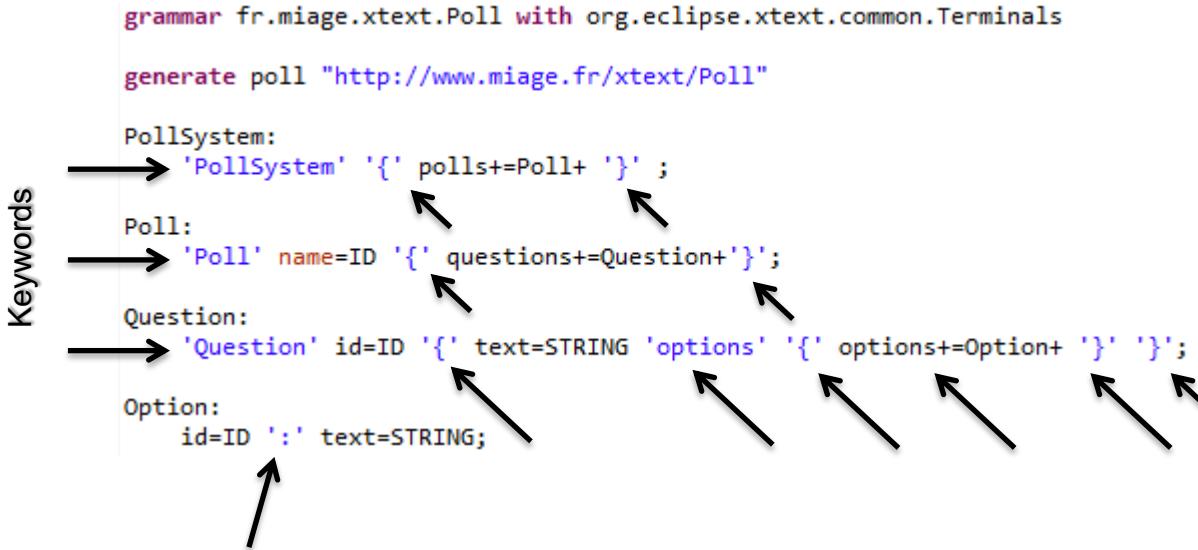
grammar fr.miage.xtext.Poll with org.eclipse.xtext.common.Terminals

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;



# Grammar Definition



# Grammar Definition

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

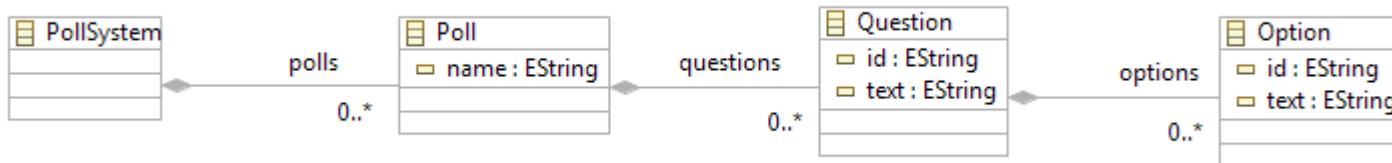
generate poll "http://www.miage.fr/xtext/Poll"

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

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

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

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



# Grammar Definition

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

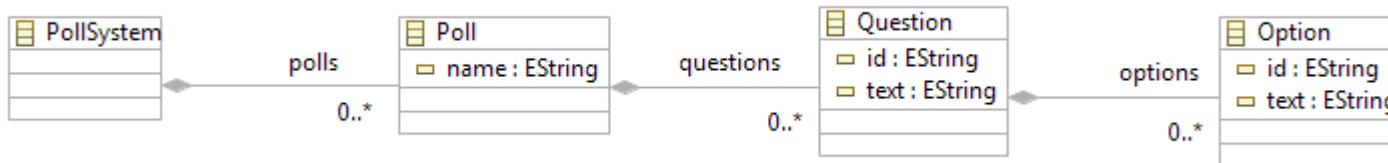
generate poll "http://www.miage.fr/xtext/Poll"

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

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

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

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



# Grammar Definition

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

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

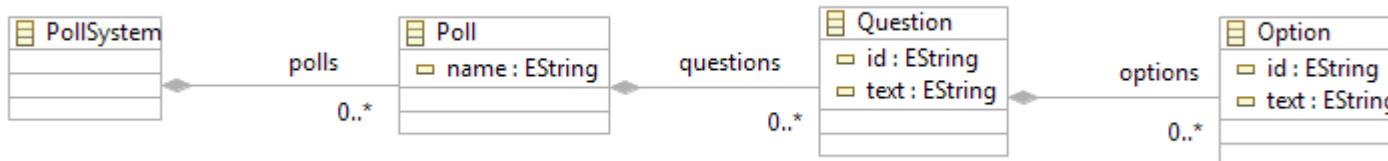
PollSystem:
    'PollSystem' '{' polls+=Poll+ '}';
    ^-----^

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

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

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

Containment



# Grammar Definition

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

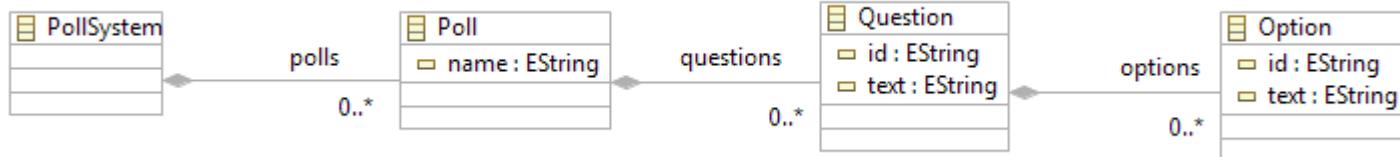
generate poll "http://www.miage.fr/xtext/Poll"

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

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

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

```
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 Definition

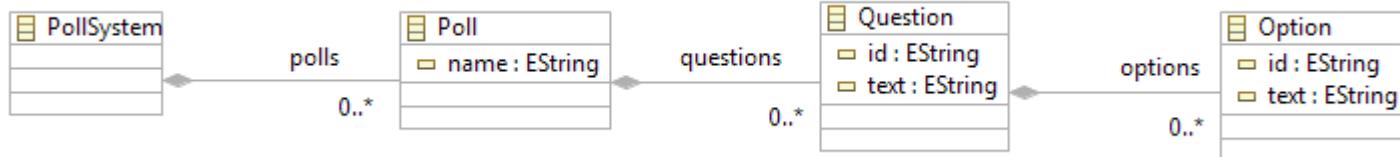
```
grammar fr.miage.xtext.Poll with org.eclipse.xtext.common.Terminals

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;
```

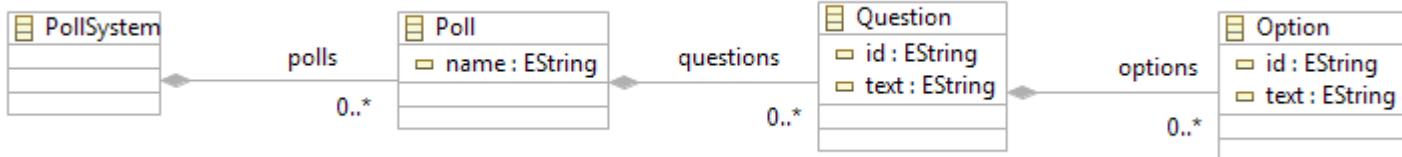
```
PollSystem {
    Poll Quality {
        Question q1 {
            "Value the user experience"
            options {
                A : "Bad"
                B : "Fair"
                C : "Good"
            }
        }
        Question q2 {
            "Value the layout"
            options {
 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

```
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"
            }
        }
    }
}
```



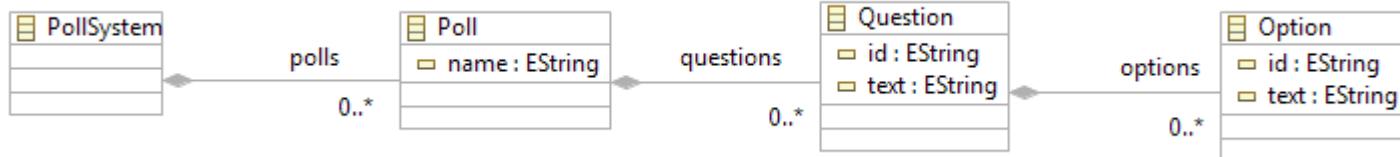
# Grammar Definition

```
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;
```

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

Live Demonstration

The screenshot shows the Eclipse IDE interface with the Package Explorer and a code editor.

**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

**Code Editor (Questionnaire.xtext):**

```
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  
      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  
  org.xtext.example.videogenerator  
  org.xtext.example.videogenerator.sdk  
  org.xtext.example.videogenerator.tests  
  org.xtext.example.videogenerator.ui

1 grammar org.xtext.example.mydsl.Questionnaire  
2  
3 generate questionnaire "http://www.xtext.org"  
4  
5   system '{' polls+=Poll+ '}';  
6   name=ID '{' questions+=Questi  
7   Question' id=ID '{' text=\$T  
8   -ID ':' text=STRING ;  
9  
10   Import...  
11   Export...  
12   Refresh F5  
13   Assign Working Sets...  
14  
15   Validate  
16   Run As ►  
17   Debug As ►  
18   Replace With ►  
19   Team ►  
20   Compare With ►  
21  
22   Properties ⌘I

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 MWE2 Workflow  
Run Configurations...

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

system '{' polls+=Poll+ '}';
name=ID '{' questions+=Questi
Question' id=ID '{' text=$T
-ID ':' text=STRING ;

Import...
Export...

Refresh F5
Assign Working Sets...

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

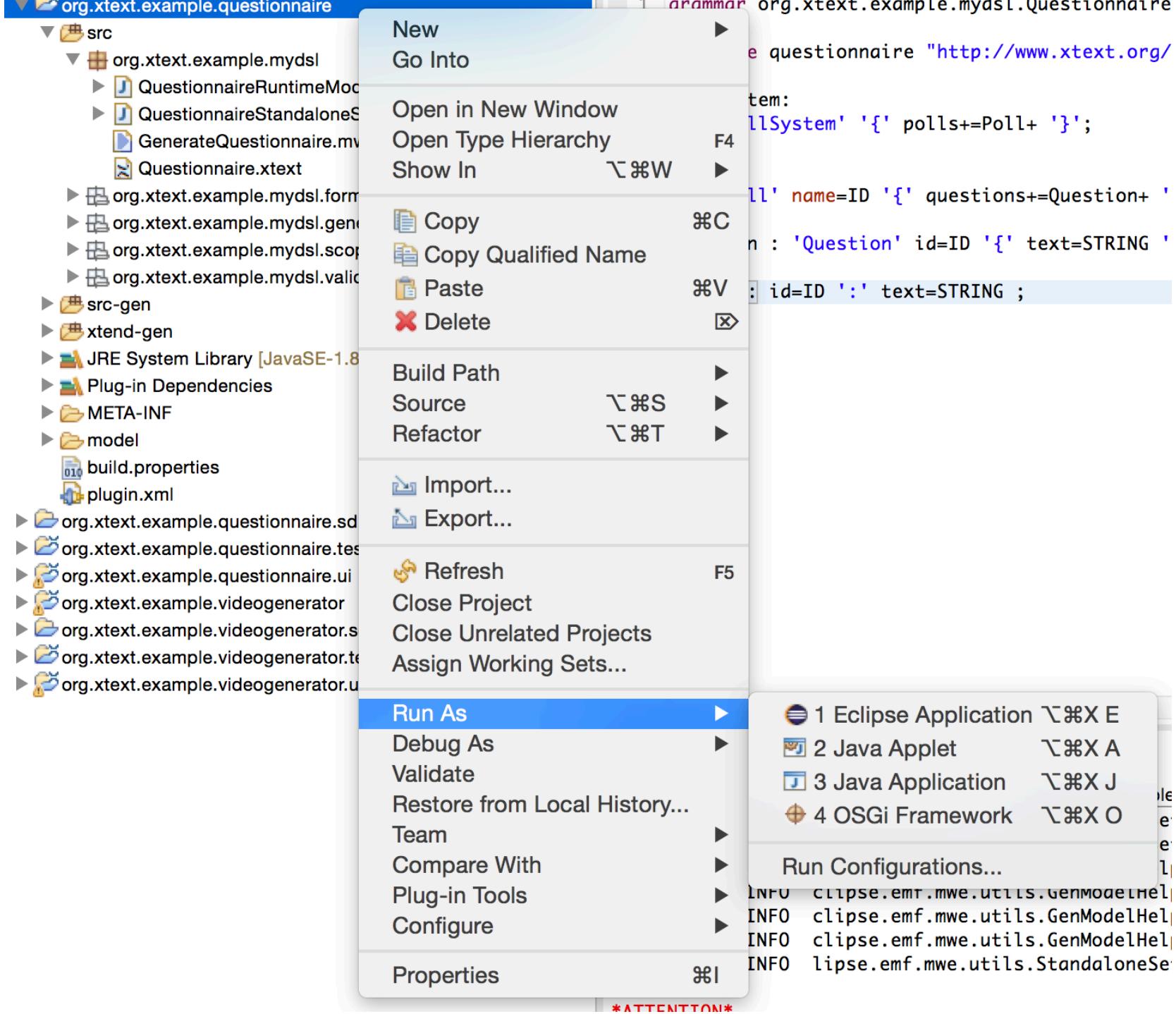
Properties ⌘I
```

Problems Javadoc Declaration Console

<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

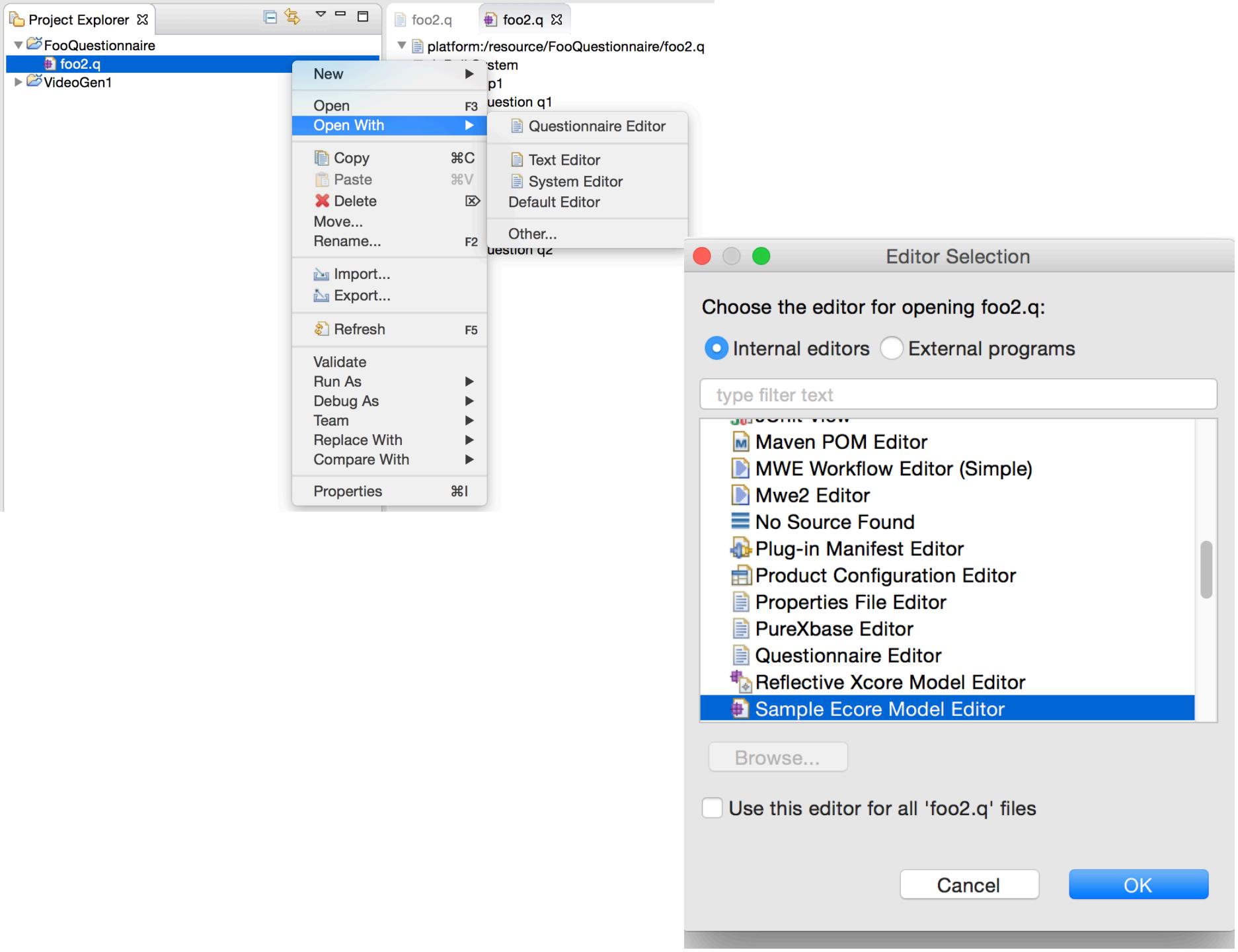
```
PollSystem {

    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"
            ]
        }
    }
}
```



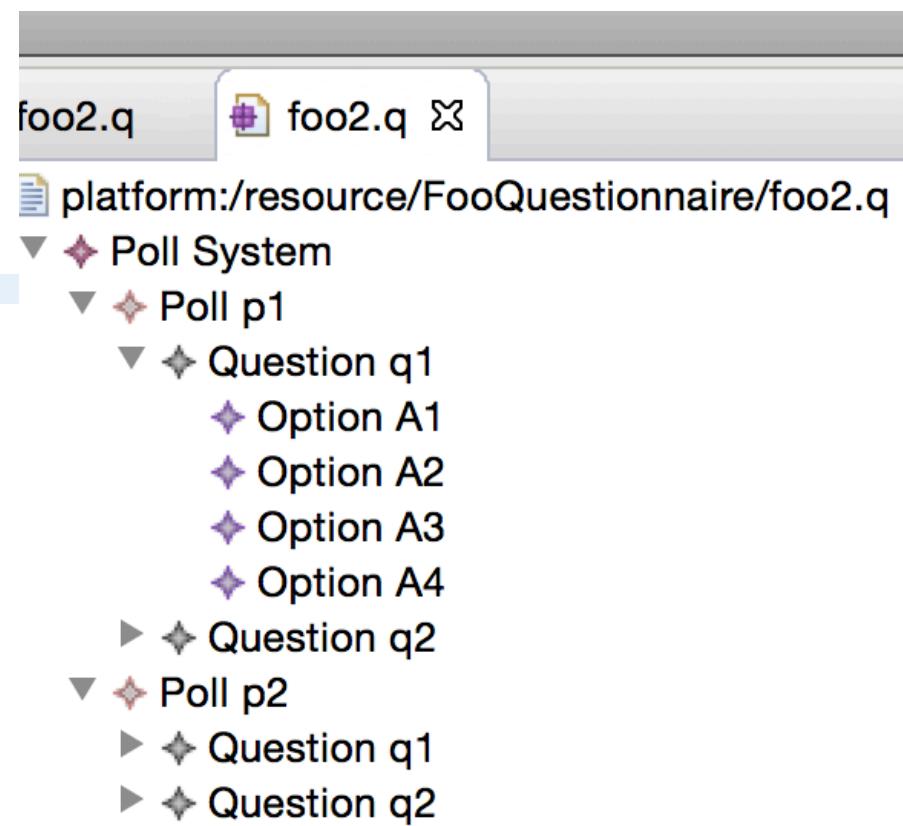
```
2.q ✎
ollSystem {

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"
        }
    }
}
}
```



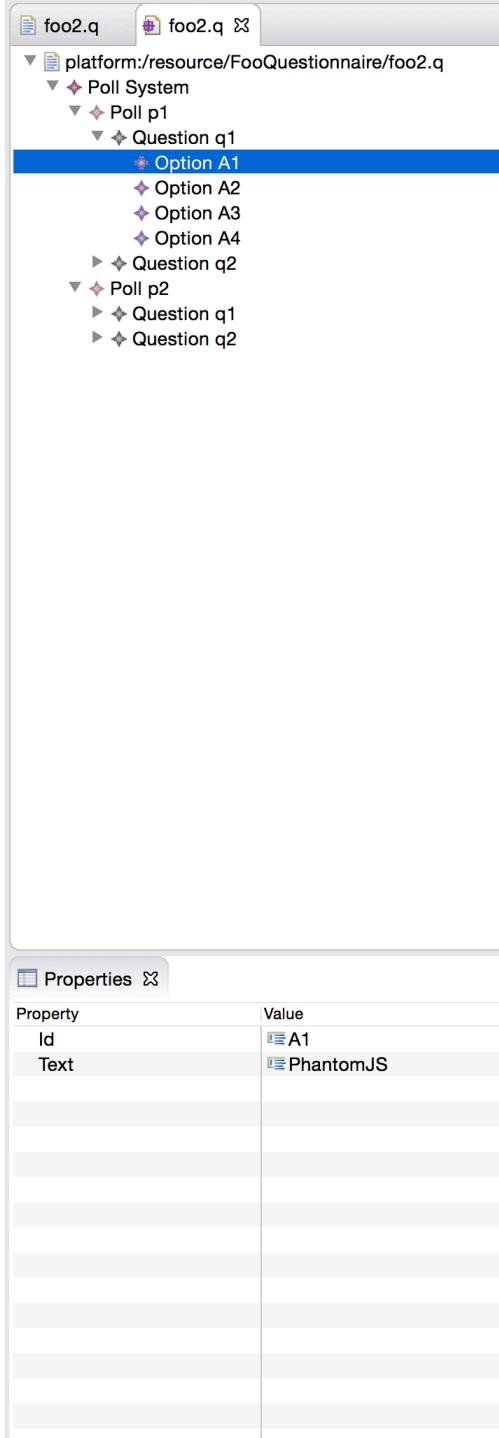
```
2.q ✎
ollSystem {

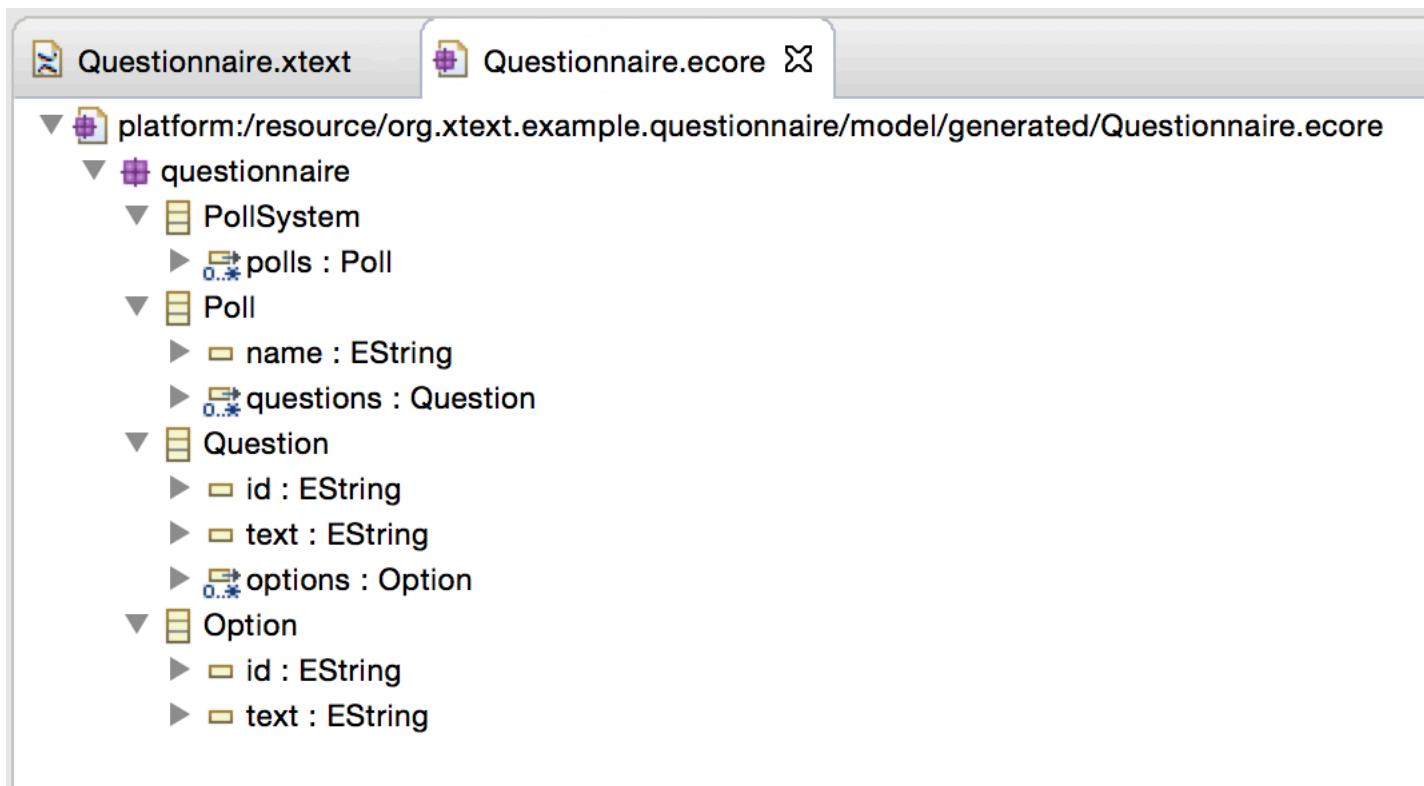
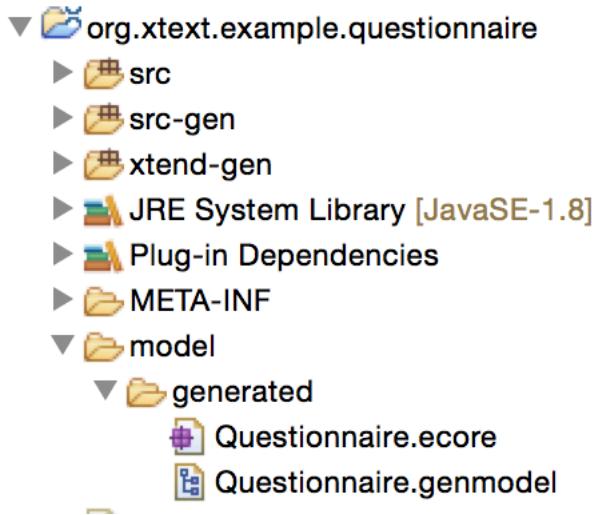
    Poll p1 {
        Question q1 {
            "What is the best JavaScript framework for testing?"
            options [
                A1: "PhantomJS"
                A2: "Jasmine"
                A3: "Mocha"
                A4: "I prefer to develop my own framework"
            ]
        }

        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"
            ]
        }
    }
}
```



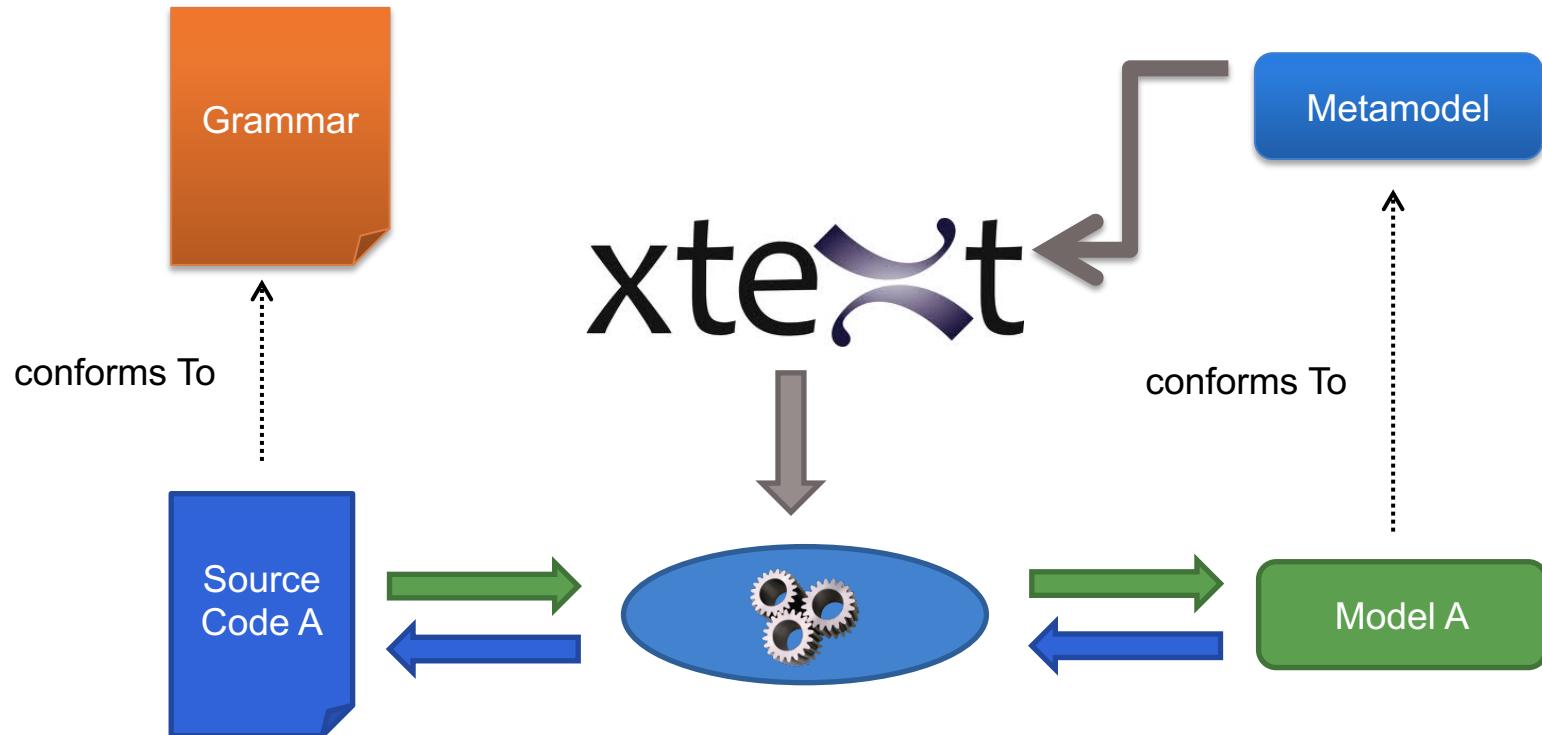


From Metamodel

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)* '}' )?
    '}';

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)?
    '}';
```

## Part 2: define a textual syntax for your statemachine metamodel...

```
fsm my_fsm
state opened entry "open door"
state[init] closed entry "close door"
transition open closed -> opened [on]
transition close opened -> closed [off]
```



KEEP  
CALM  
AND  
DO IT  
YOURSELF

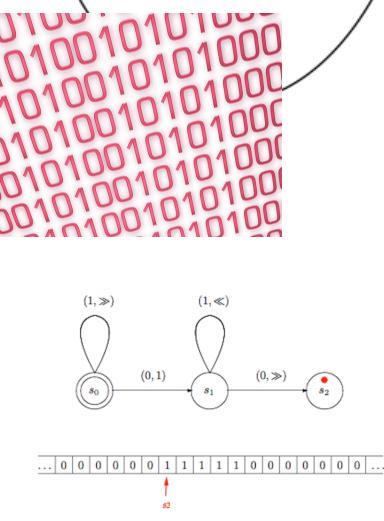
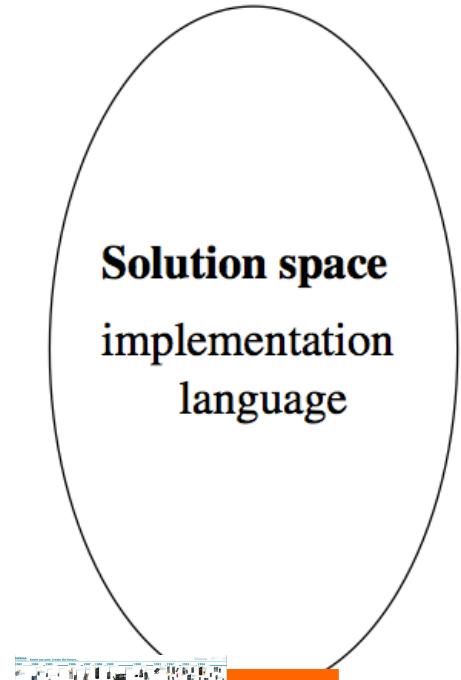
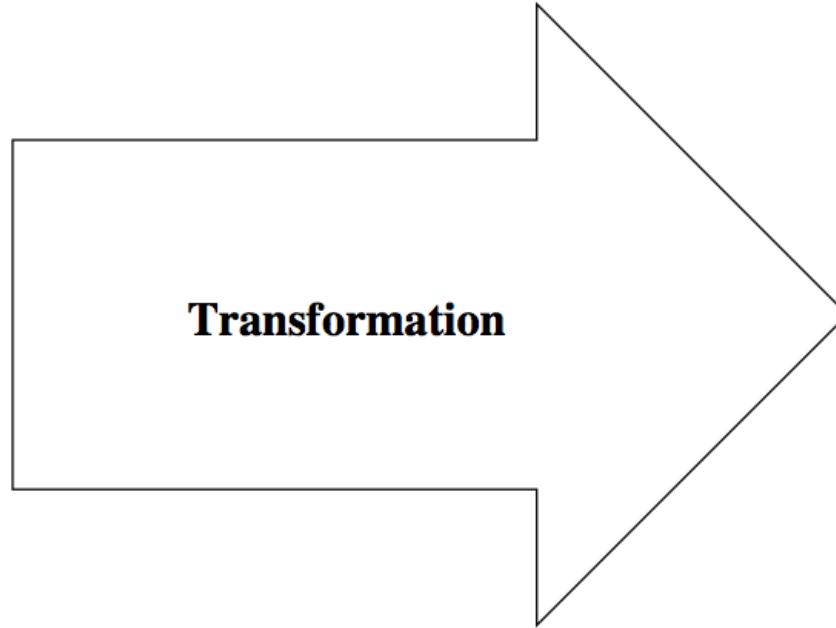
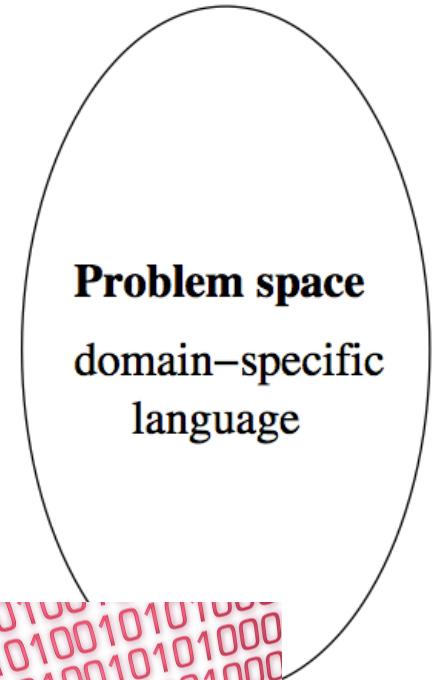
DSL,

Model,

Metamodel,

Summary

# Abstraction Gap



# 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

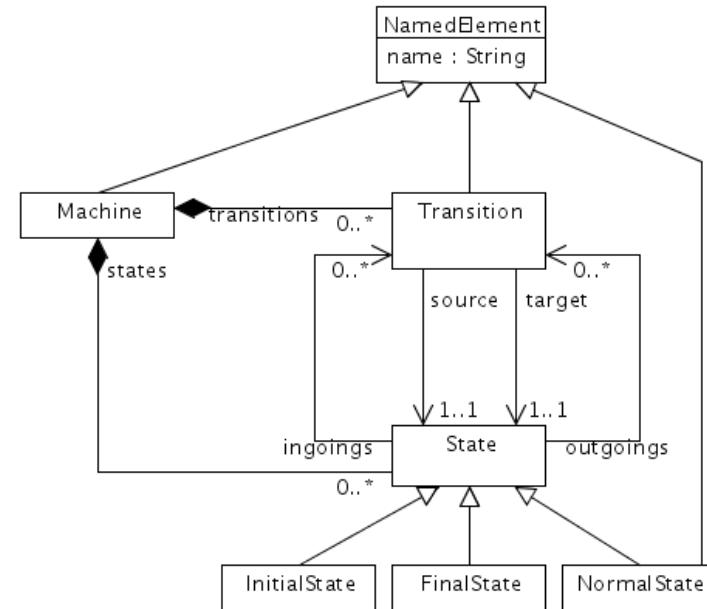
# Generative approach

- 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



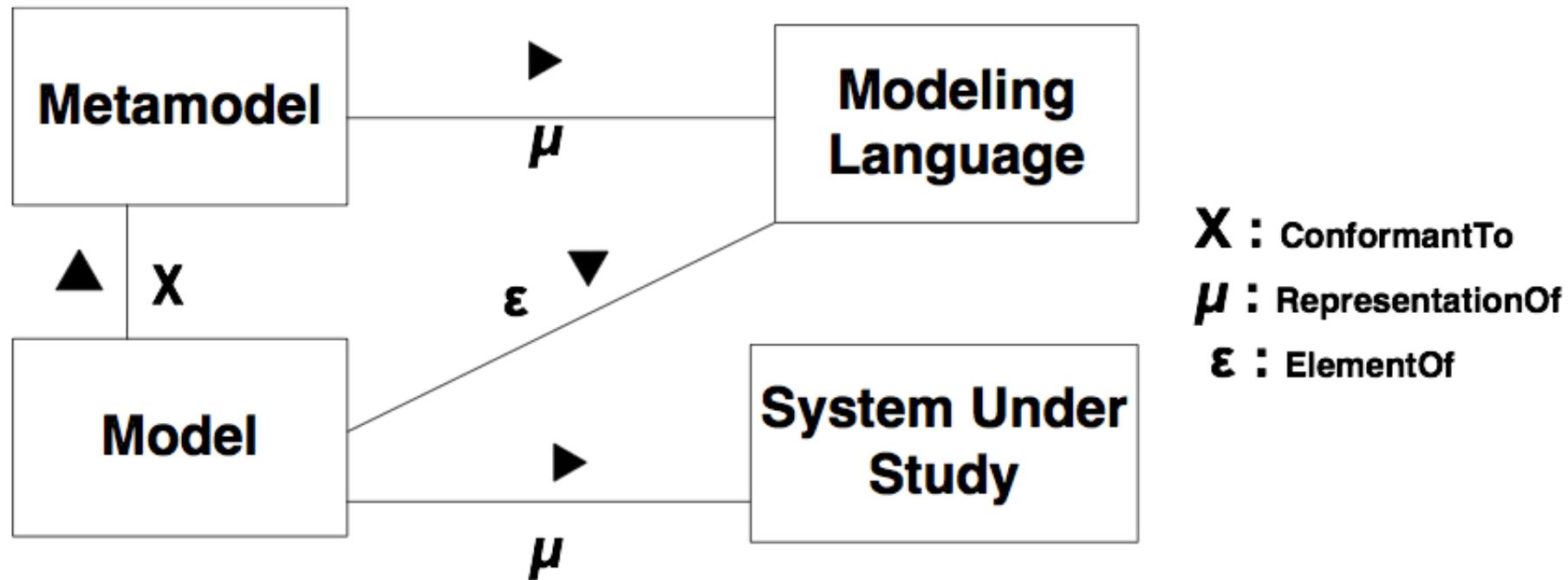
conforms To

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

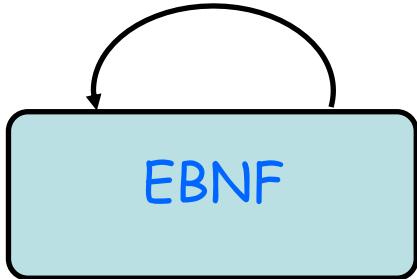
conforms To

Source Code/Model

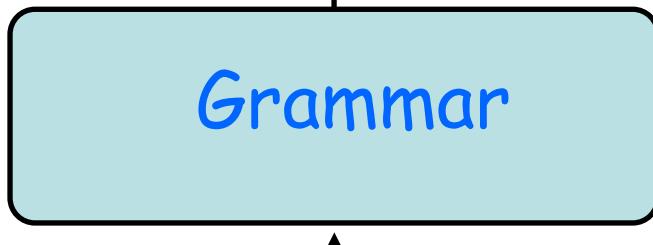
# Model, Metamodel, Metametamodel, DSML



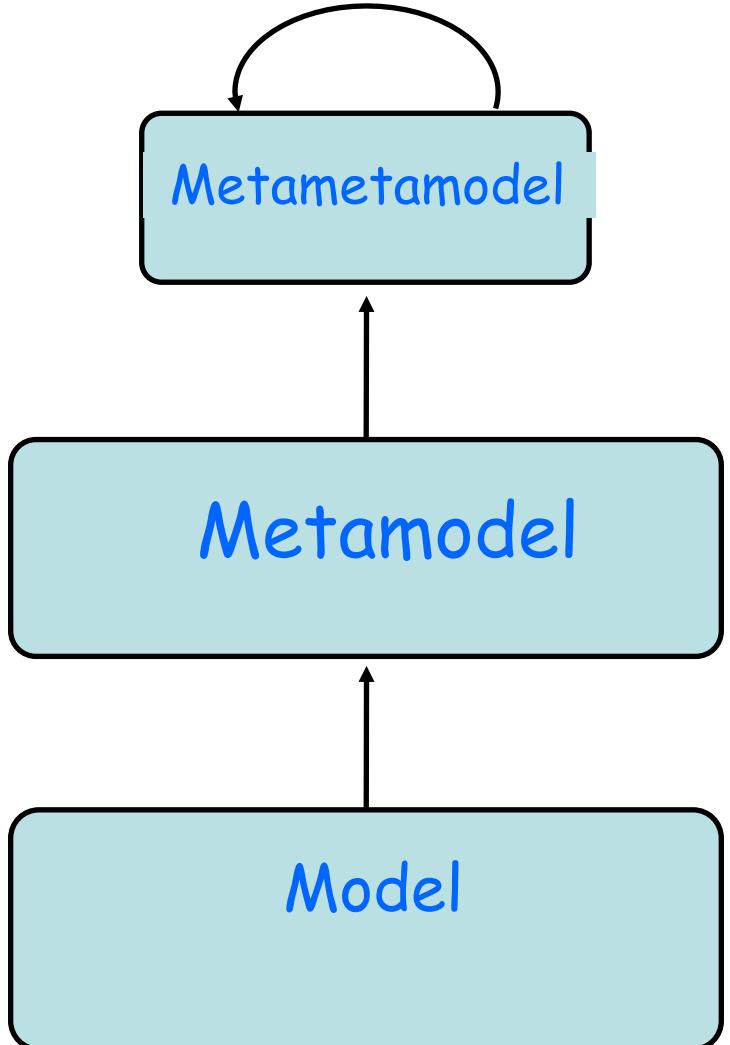
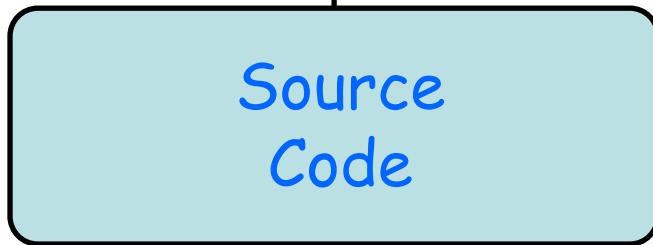
M<sup>3</sup>



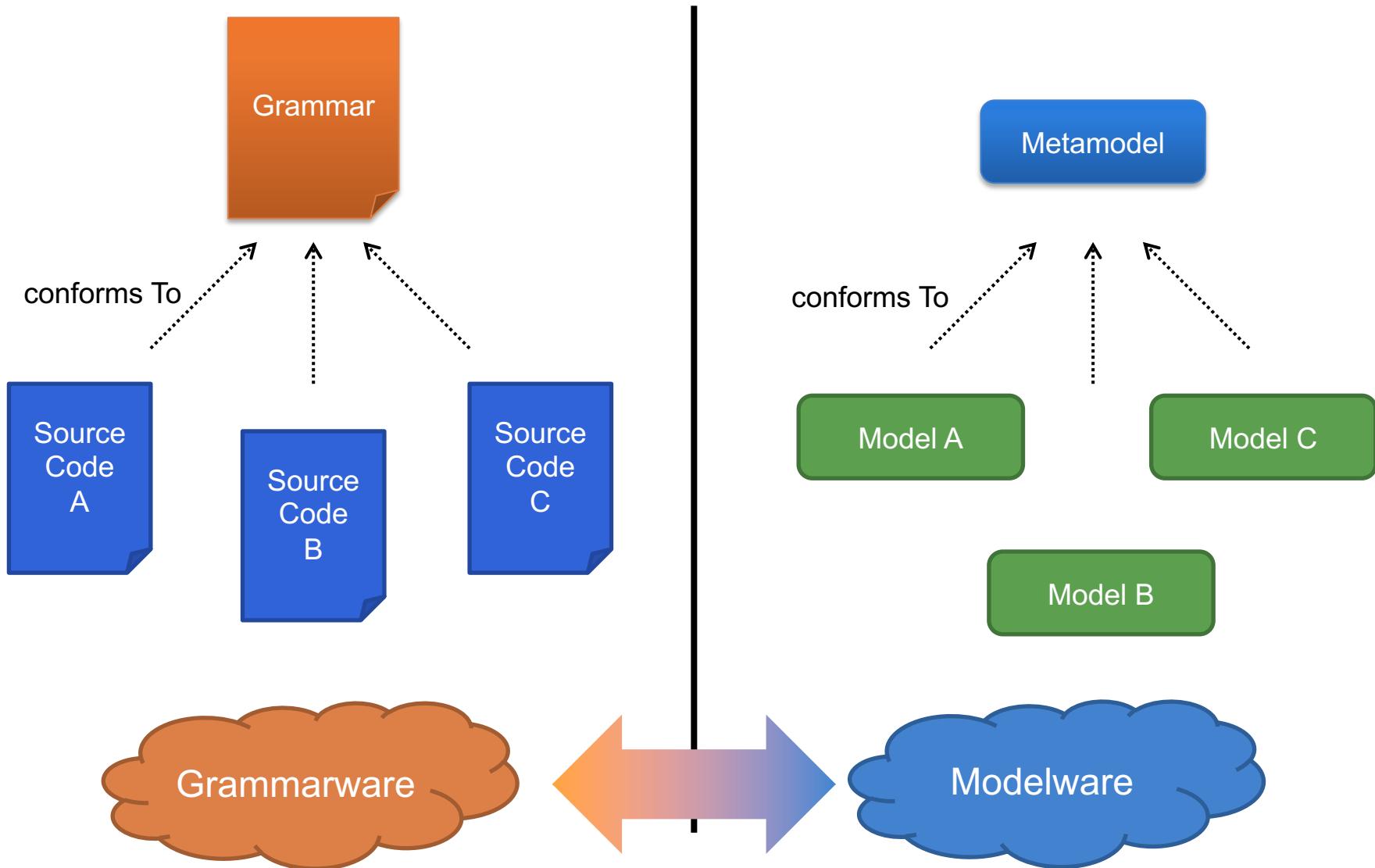
M<sup>2</sup>



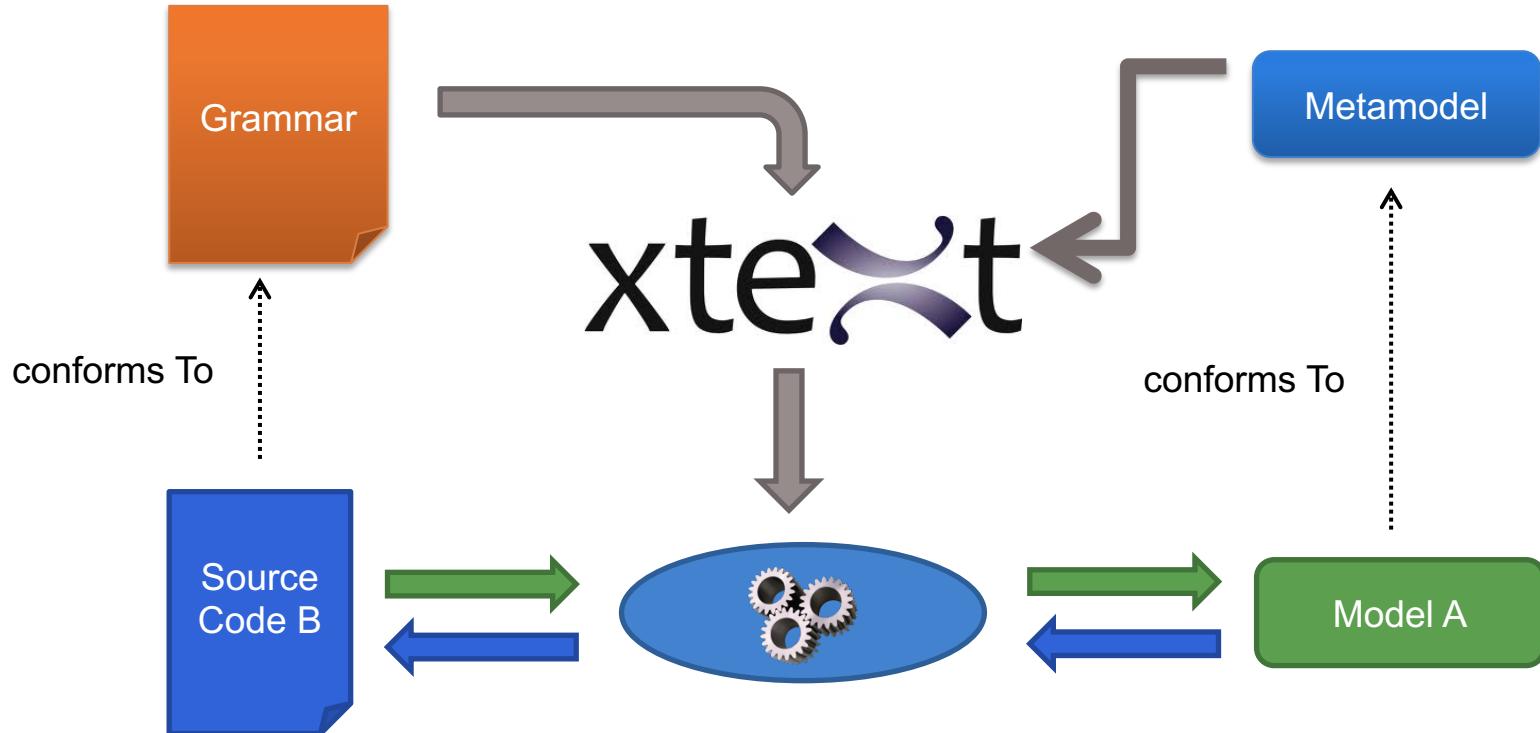
M<sup>1</sup>



# 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}

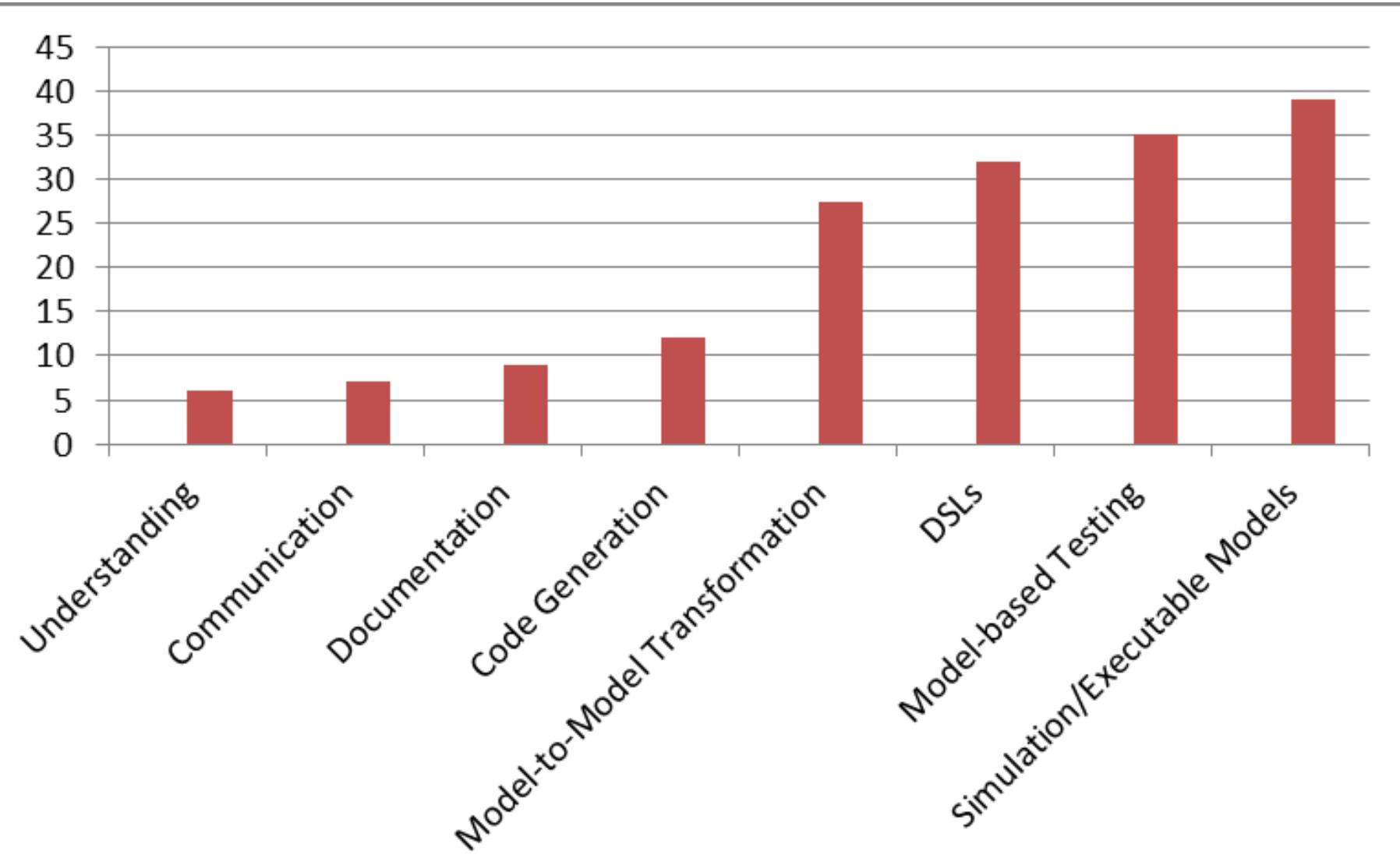
Jon Whittle  
School of Computing and  
Communications  
Lancaster University, UK  
+44 1524 510492

{j.n.whittle@lancaster.ac.uk}

**2011**

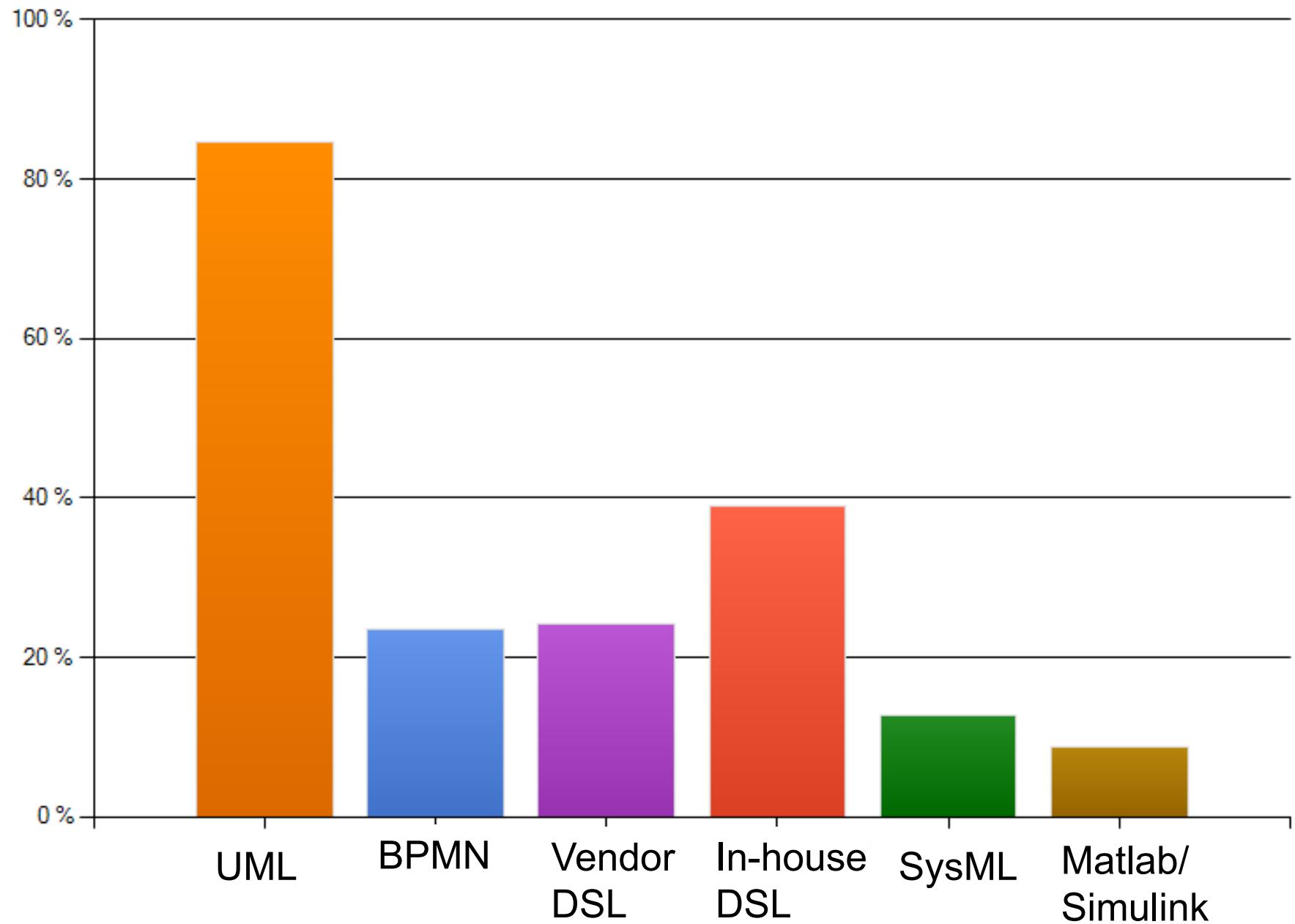
**« Domain-specific  
languages are far more  
prevalent than  
anticipated »**

# What are models used for?

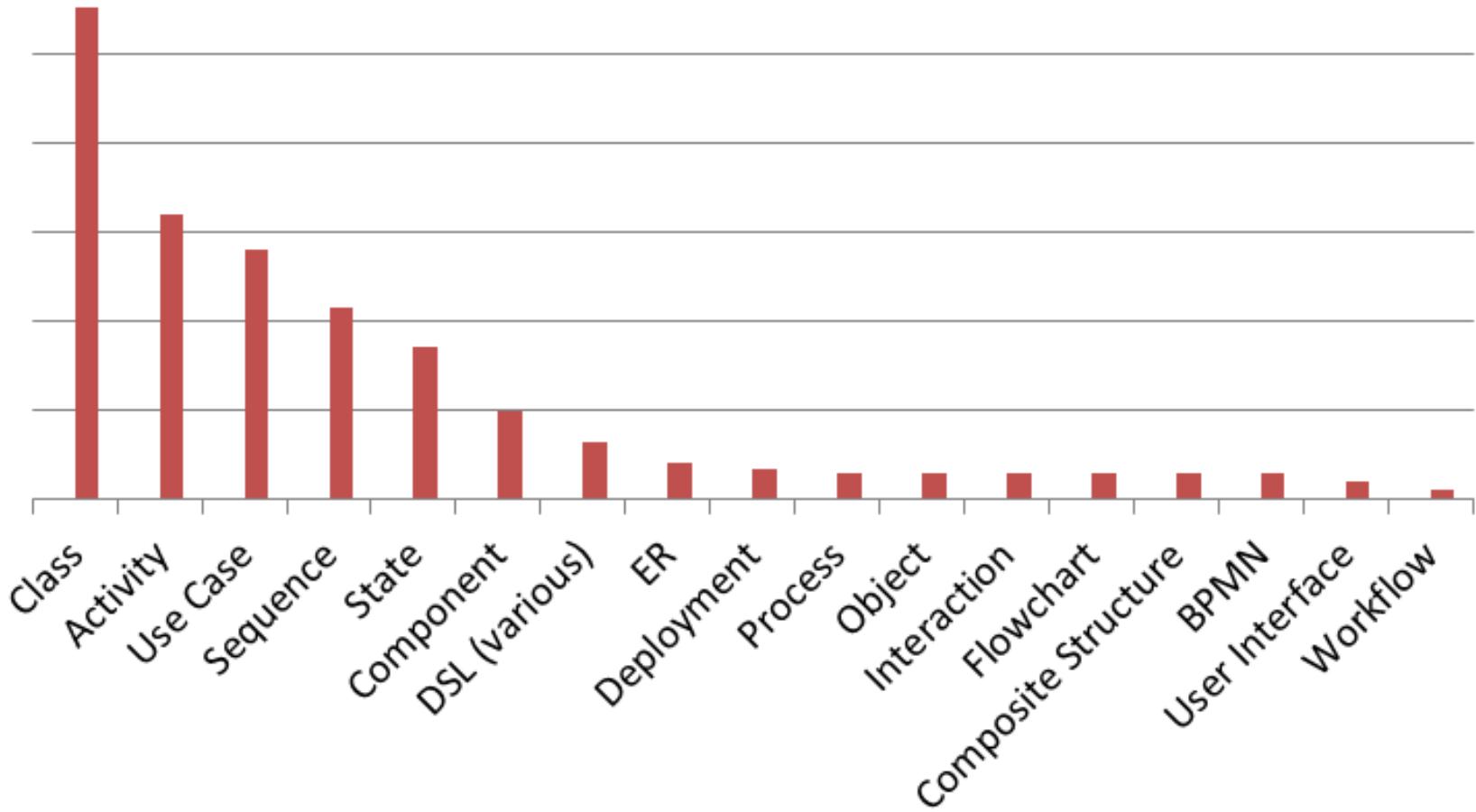


"Do not use" percentages for MDE activities

# Which modeling languages do you use?



# Which diagrams are used?



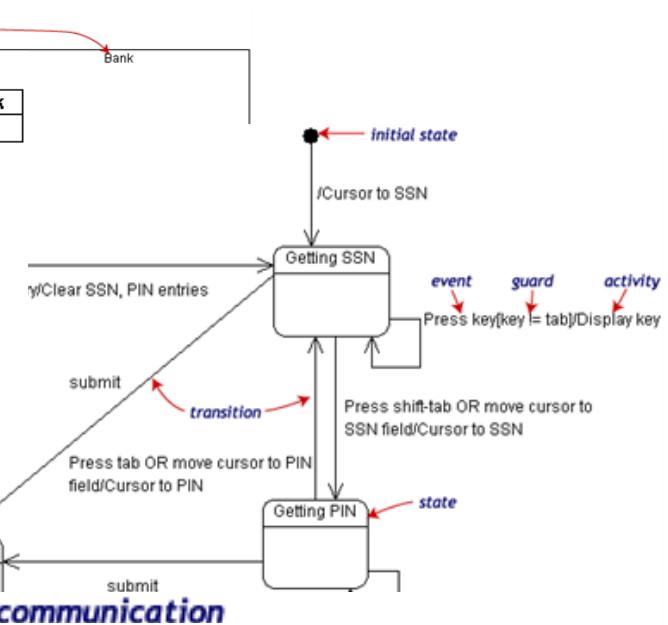
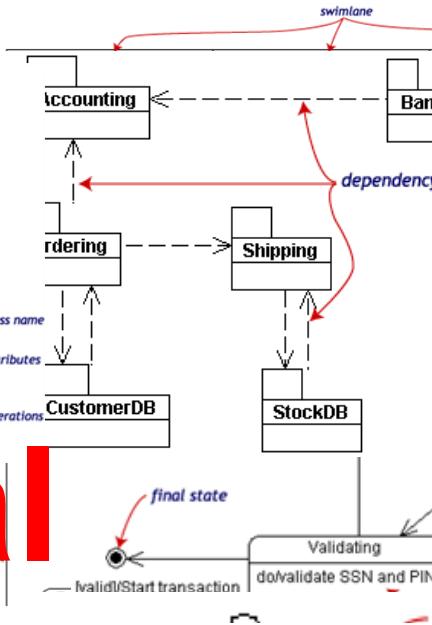
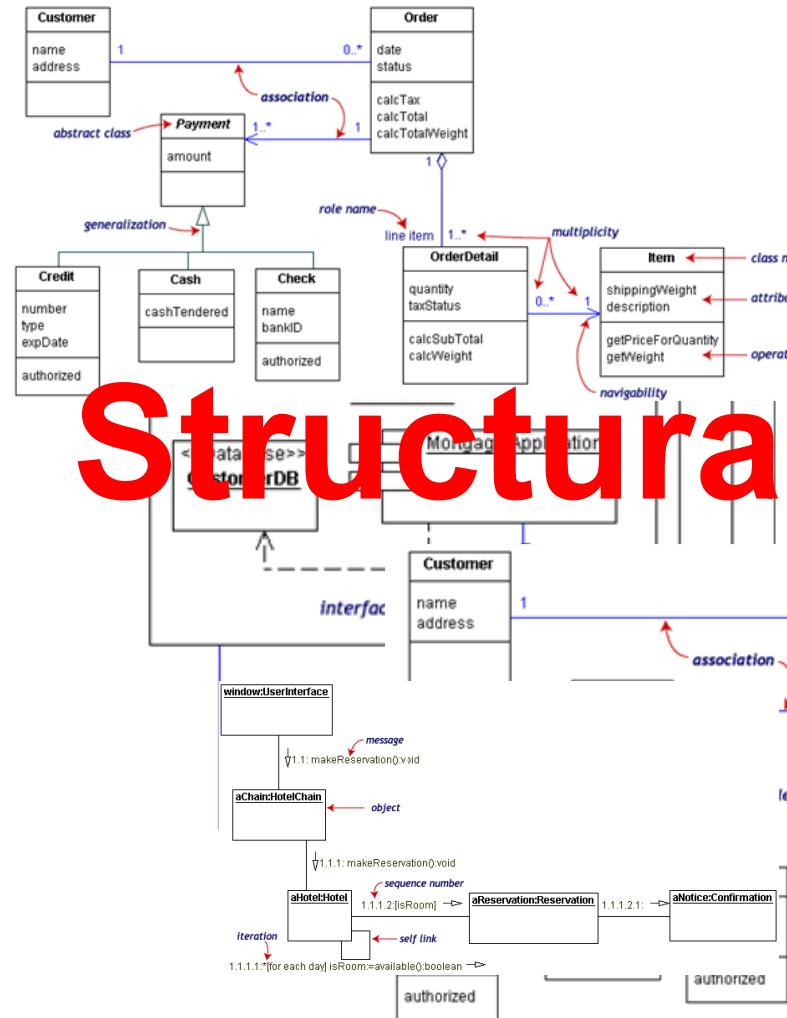
19 different diagram types are used regularly

## Use of multiple languages (DSLs)

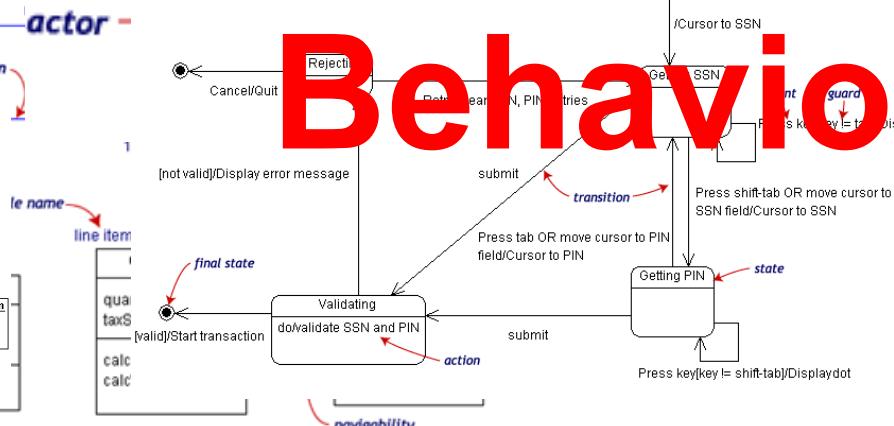
- 62% of those using custom DSLs also use UML
- Almost all users of SysML and BPMN also use UML
- UML is the most popular ‘single use’ language
  - 38% of all respondents
- UML used in combination with just about every combination of modeling languages
  - 14% of UML users combine with vendor DSL
  -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

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

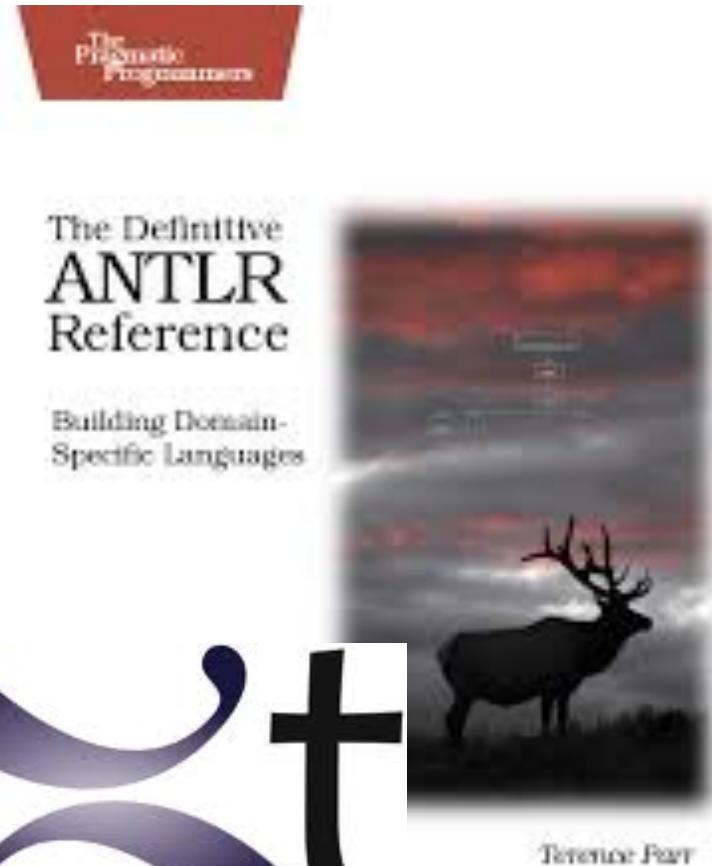
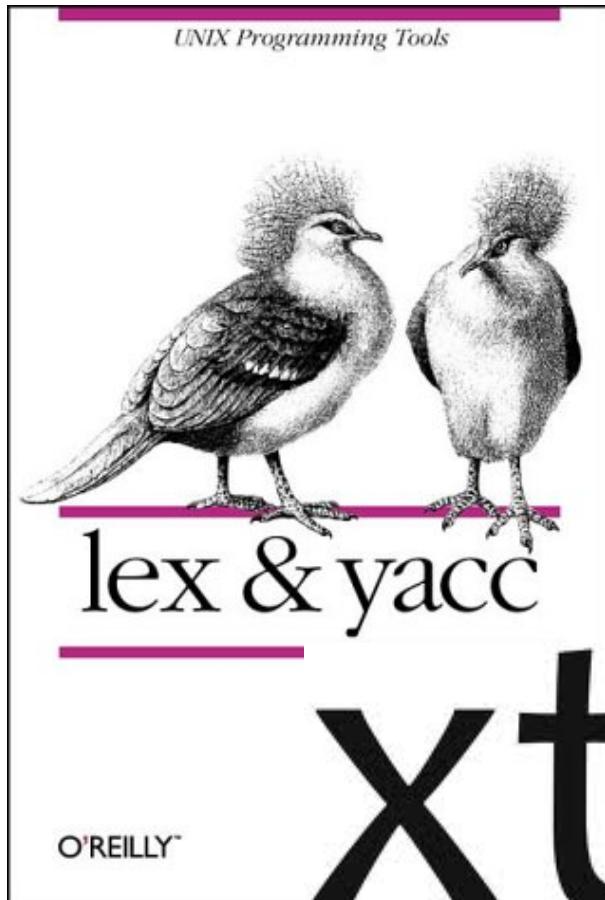
**Structural**



**Behavioral**



# Xtext is built using MDE technologies



xtext

**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 productivity 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

# 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 or 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
    - Duplicating 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

# Questions ?

# Engineering Modeling Languages

Turning Domain Knowledge into Tools



CRC Press  
A CHAPMAN & HALL BOOK



## {S} spoofax

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](mailto:j.hutchinson,j.n.whittle,m.rouncefield@lancaster.ac.uk)



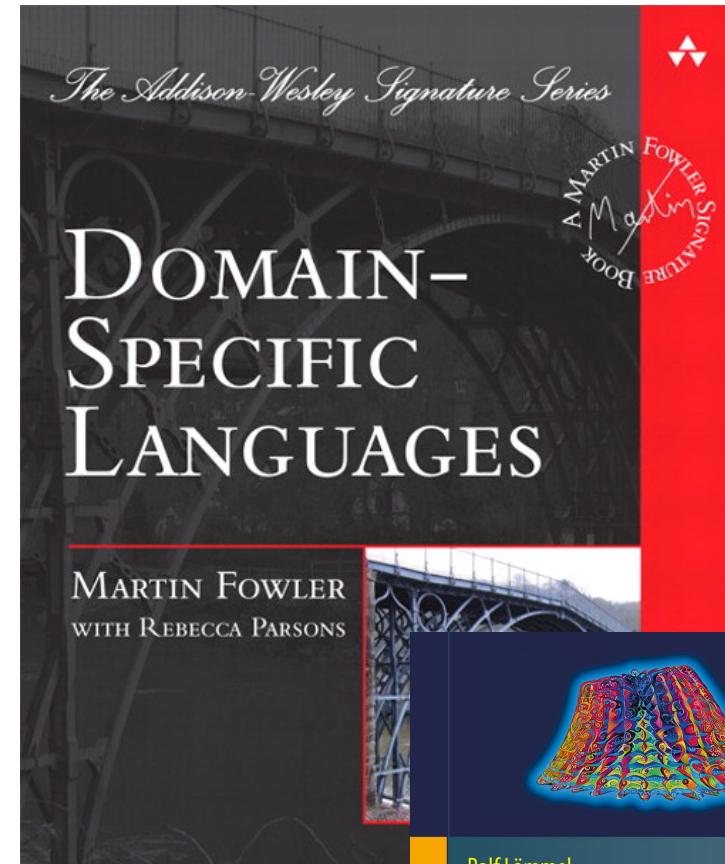
## DSL Engineering

*Designing, Implementing and Using  
Domain-Specific Languages*

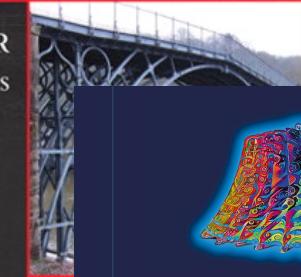
Markus Voelter

with Sebastian Benz, Christian Dietrich, Birgit Engelmann  
Mats Helander, Lennart Kats, Eelco Visser, Guido Wachsmuth

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



MARTIN FOWLER  
WITH REBECCA PARSONS



Ralf Lämmel

## Software Languages

Syntax, Semantics,  
and Metaprogramming