

Domain-Specific Languages

Mathieu Acher

Maître de Conférences

mathieu.acher@irisa.fr

Material

<https://github.com/acherm/teaching-MDE1617>

IDM (MDE) in practice

bref.
CANAL à 30 ans.

ETAPE 1 : DONNE TON PRENOM

MATHIEU

→ OK

Online Generator

← → C bref30ans.canalplus.fr/#c

ETAPE 2 : CHOISIS 3 BONS SOUVENIRS



Variant





40 ans et pas une ride

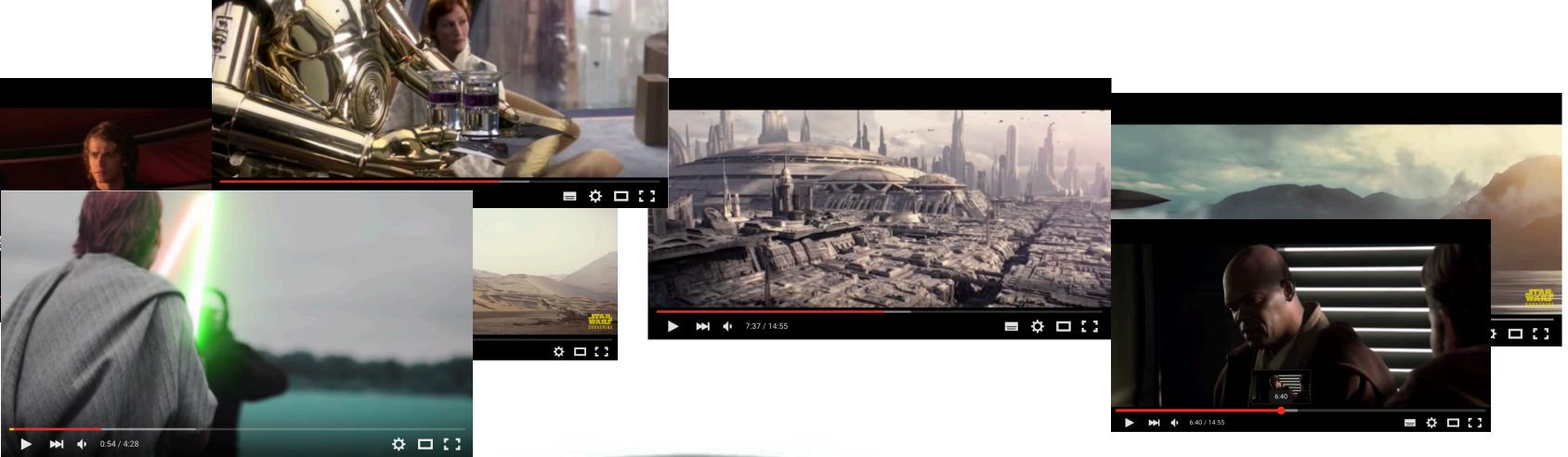
Découvrir un nouvel épisode...

Déjà 1768 épisodes générés !



Jean-Marc JEZEQUEL

Professeur des universités en informatique,
Directeur de l'IRISA depuis 2012

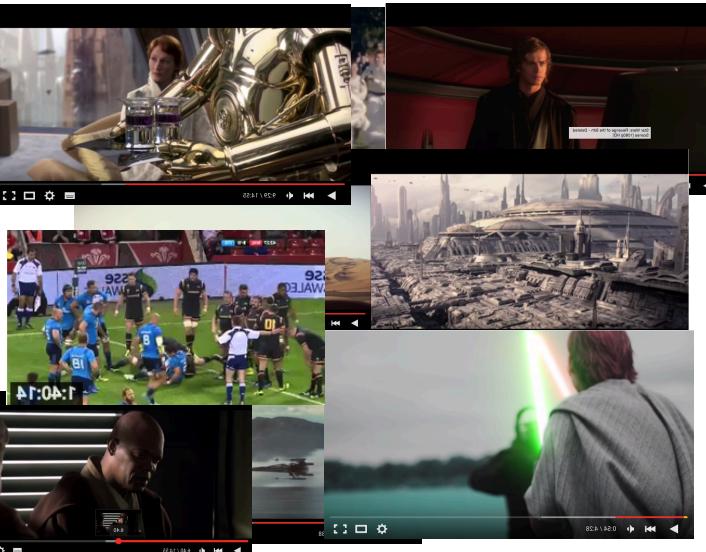




Generator
~ composition of
video sequences

**video
variants**





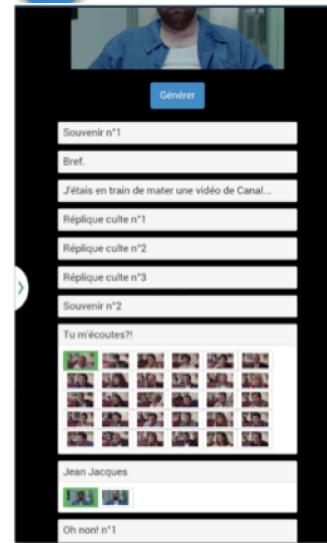
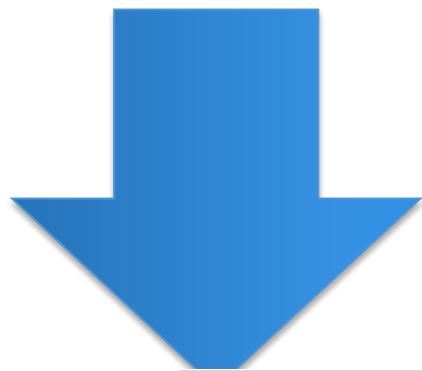
```

foo1.videogen ✎

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

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

```



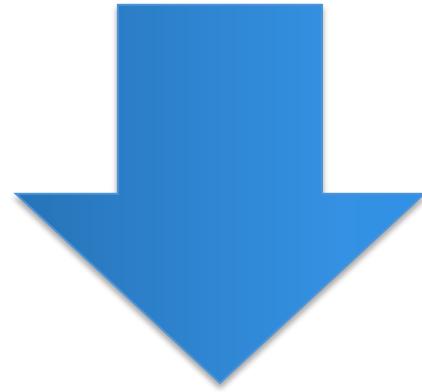
- ## Website/online
- Random generation
 - Configurator
 - Game
 - ...



```
foo1.videogen ✘

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

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



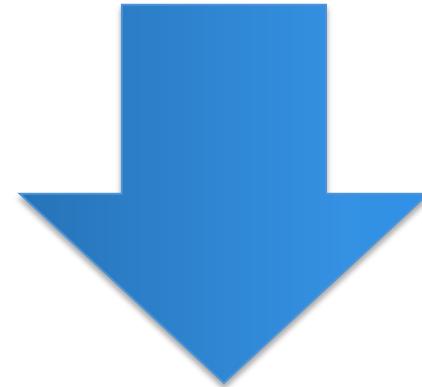
 FFmpeg

foo1.videoogen

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

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

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



#2 How to transform
models/programs?

#3 How to manage
variability/variants?

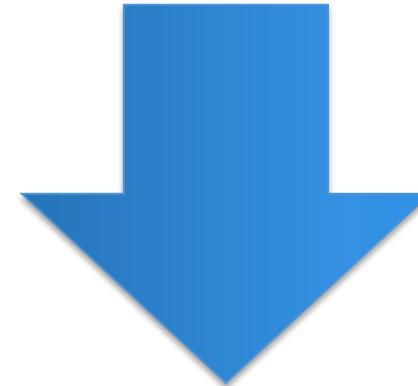
#4 How do
frameworks
internally work?

foo1.videoogen

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

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

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



#2 How to transform
models/programs?

#3 How to manage
variability/variants?

#4 How do
frameworks
internally work?

Plan

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

Contract

- Better understanding/source of inspiration of software languages and DSLs
 - Revisit of history and existing languages
- Foundations and practice of Xtext
 - State-of-the-art language workbench (Most Innovative Eclipse Project in 2010, mature and used in a variety of industries)
- Models and Languages
 - Perhaps a more concrete way to see models, metamodels and MDE (IDM in french)

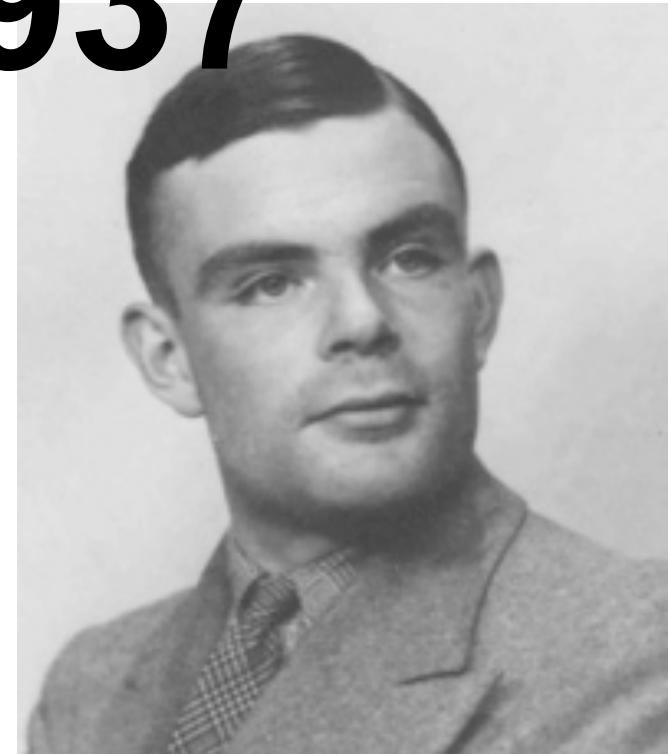
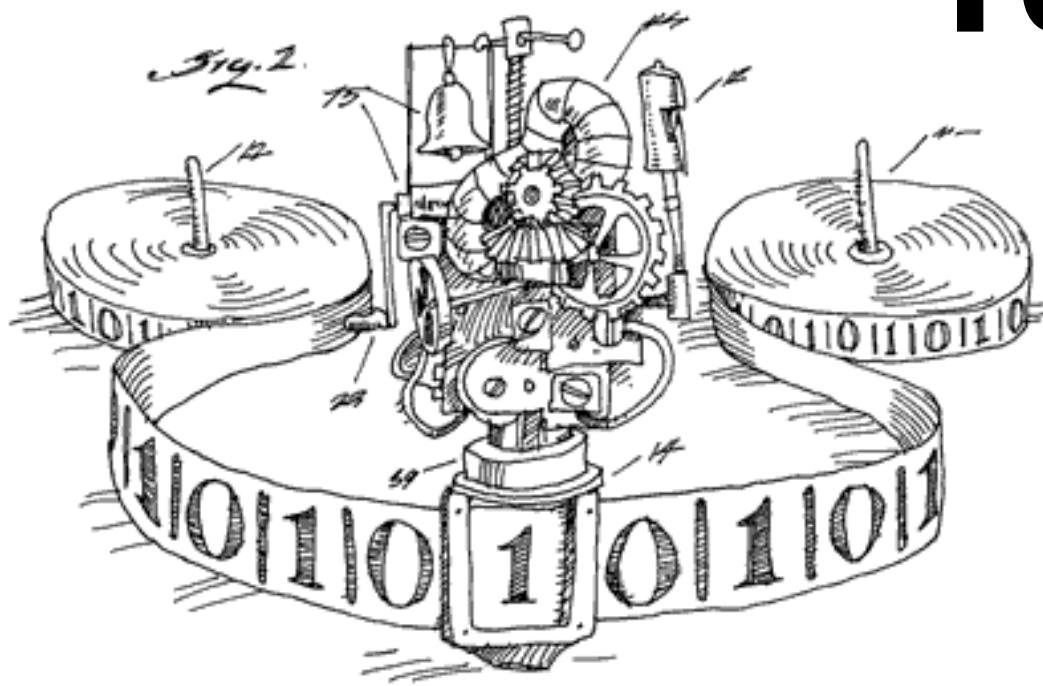
What are DSLs

Where are DSLs

Why DSLs (will) matter

The (Hi)Story of Software Engineering / Computer Science

1937

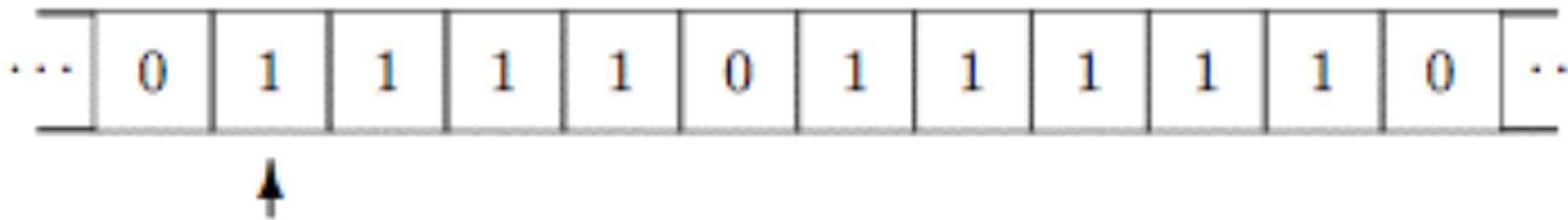


Turing Machine

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

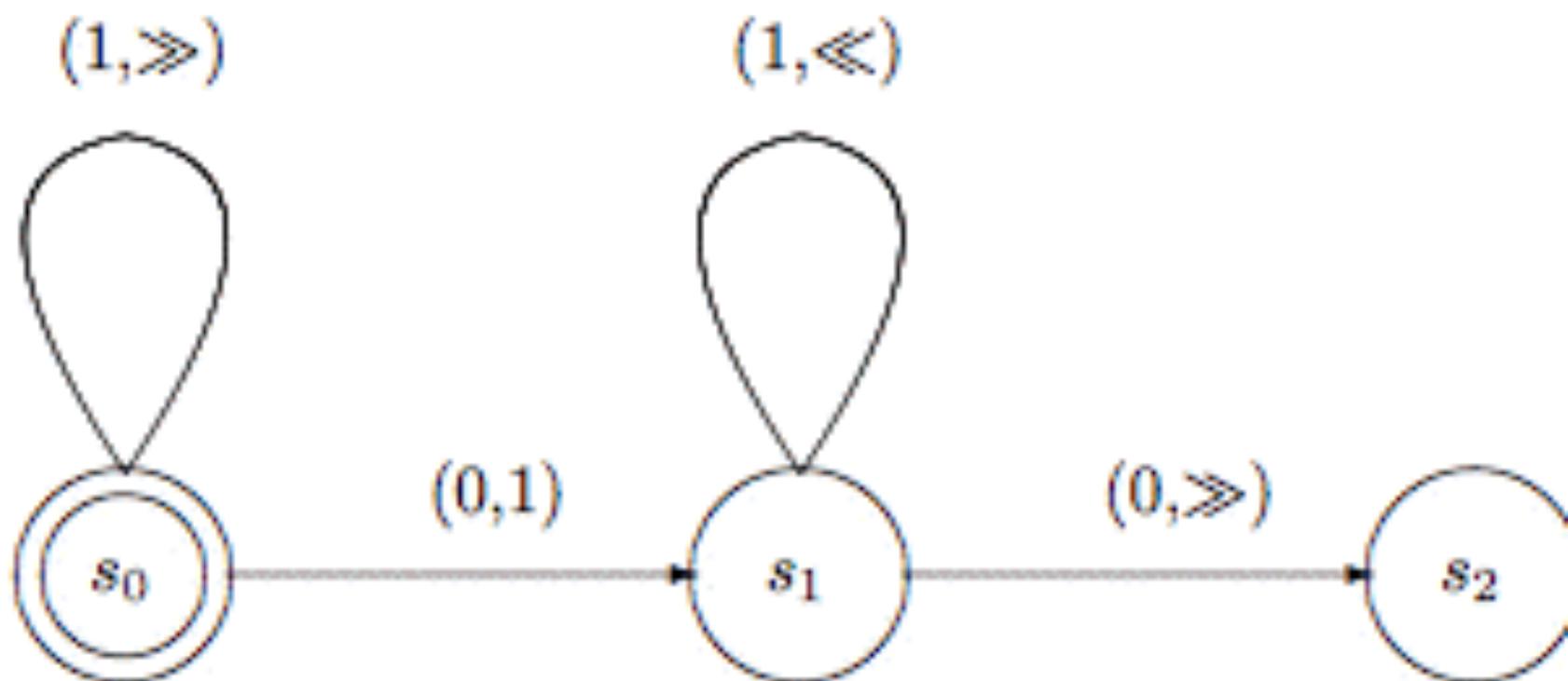
**Write a symbol
or move to left (>>) or right
(<<)**

< State_{current}, Symbol, State_{next}, Action >



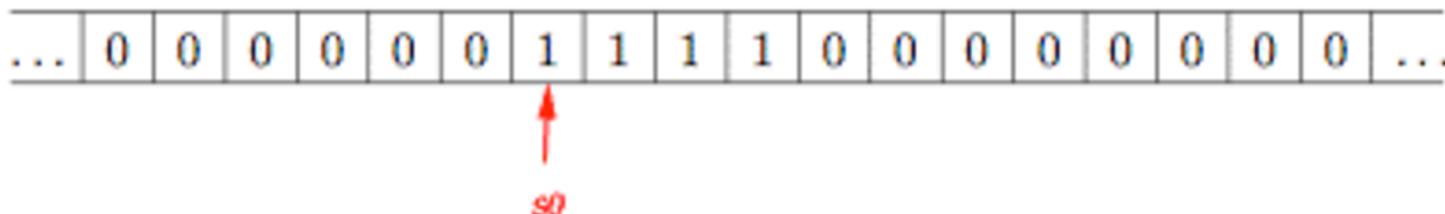
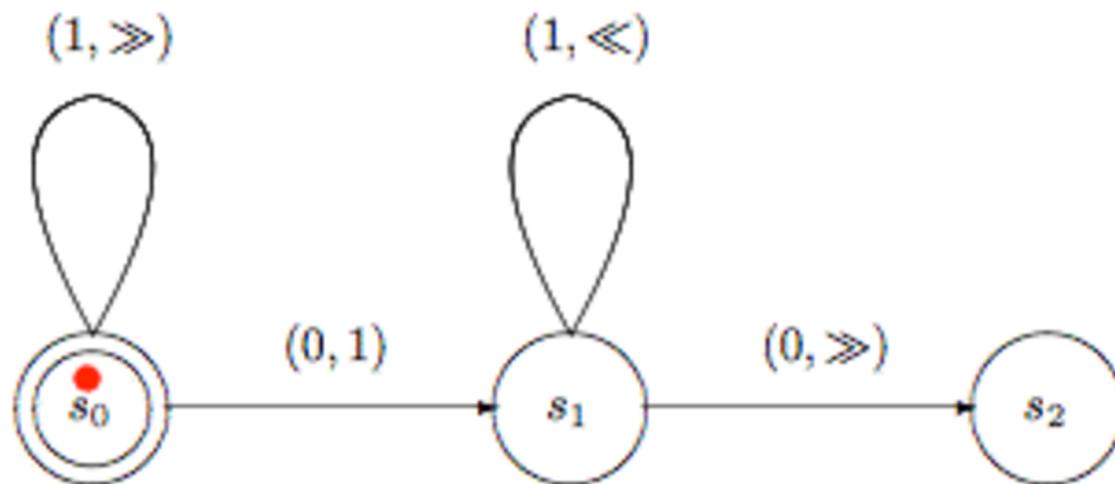
Turing Machine

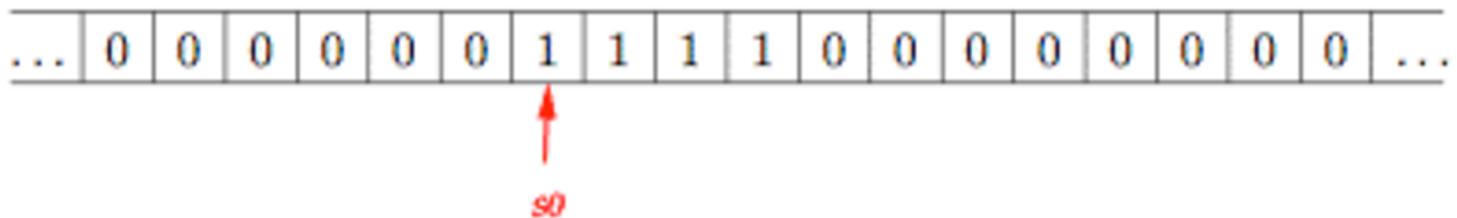
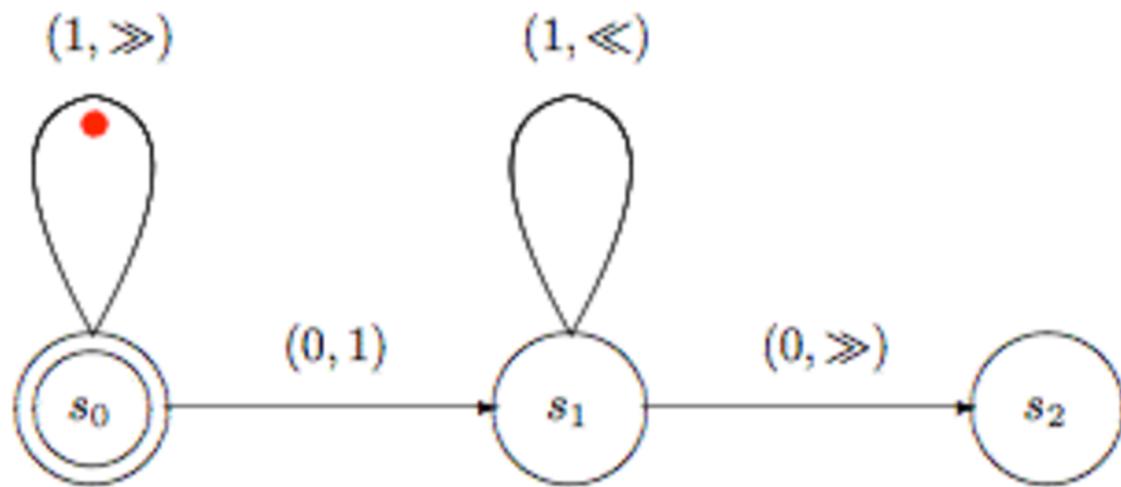
~ kind of state machine

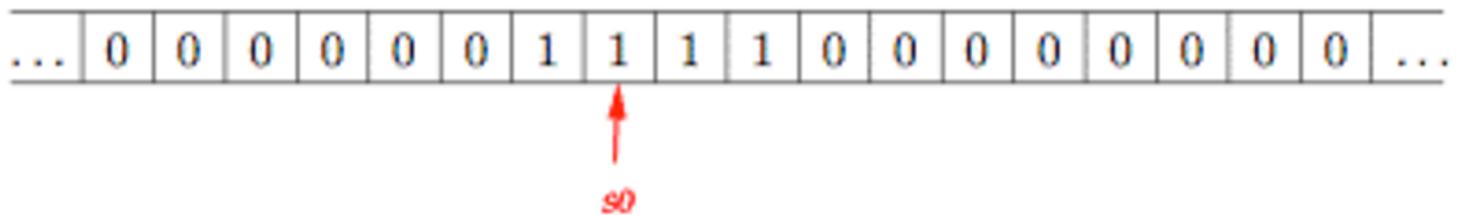
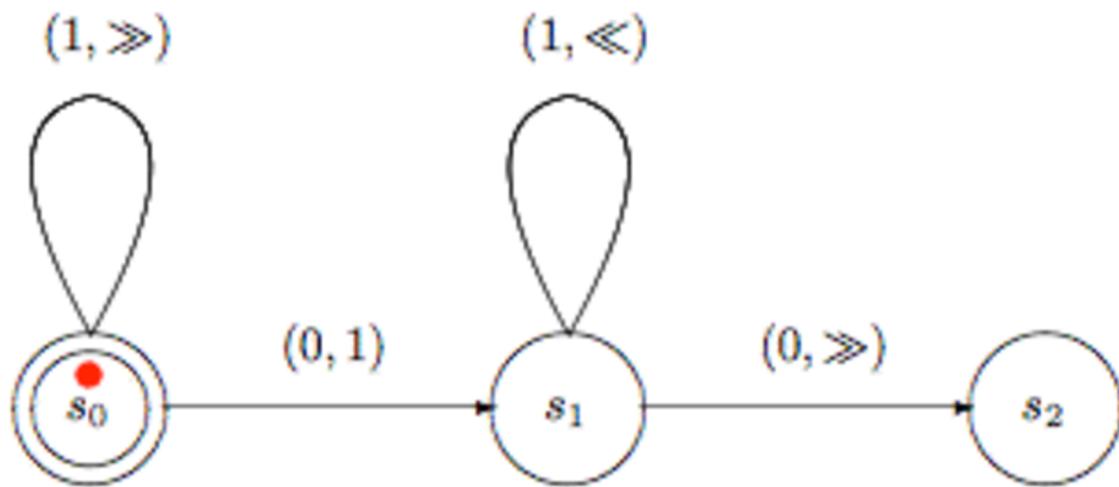


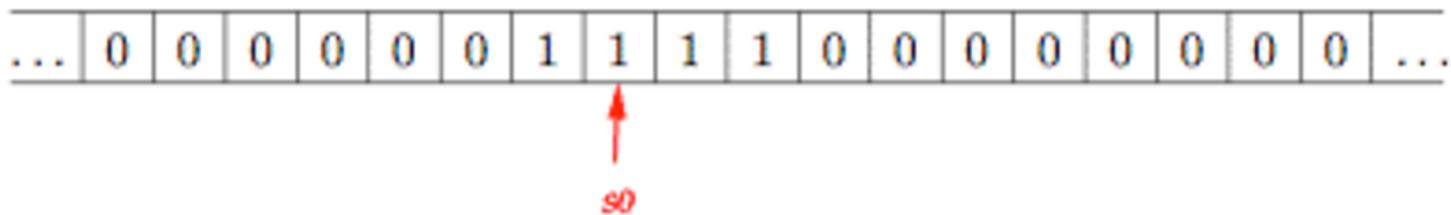
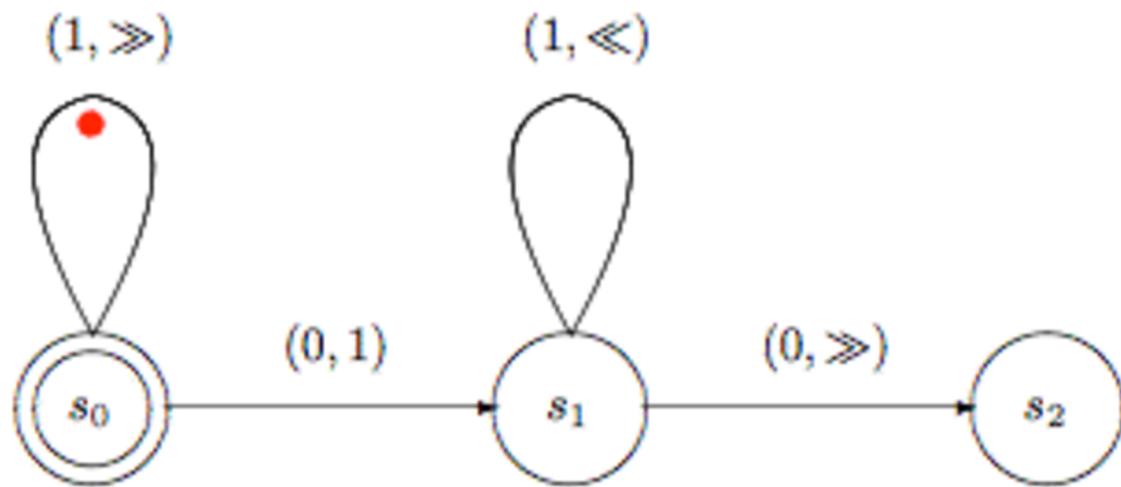
Successor (add-one) function

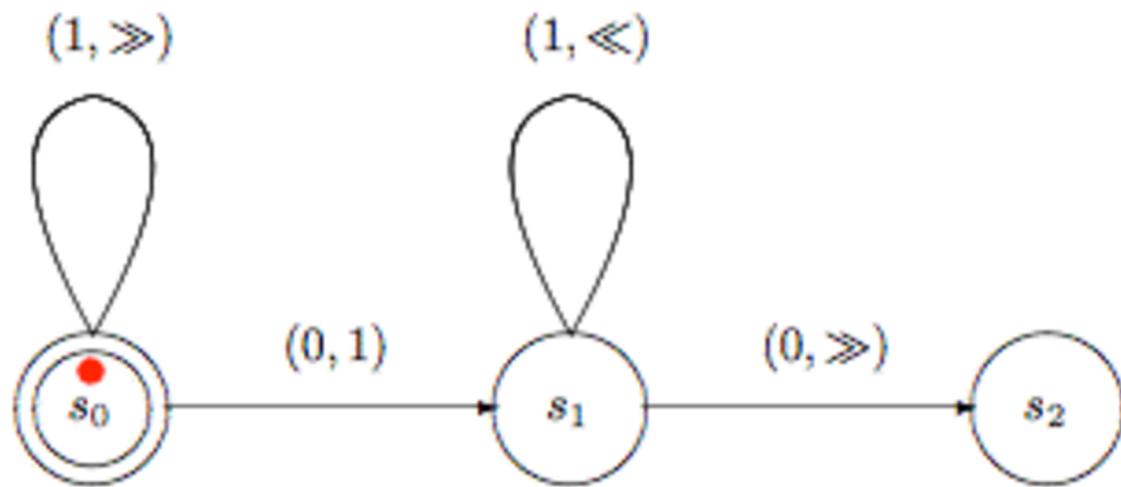
assuming that number n as a block of $n+1$ copies of the symbol '1' on the tape (here, $n=3$)

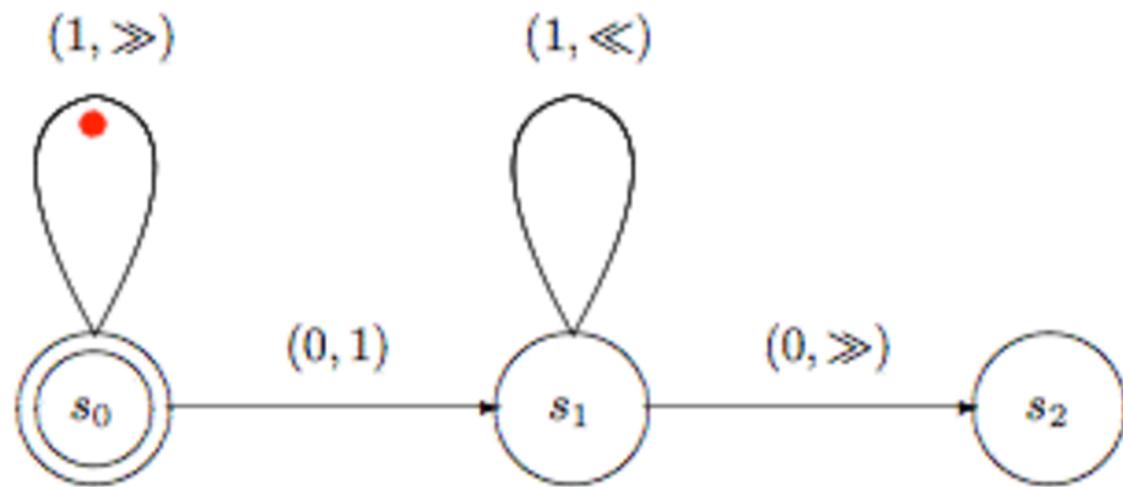


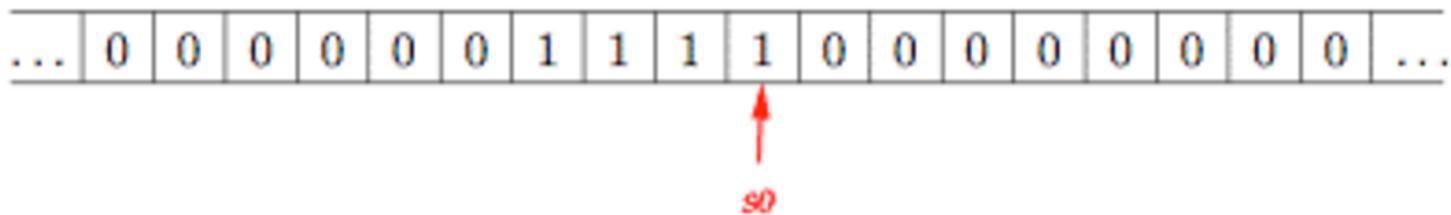
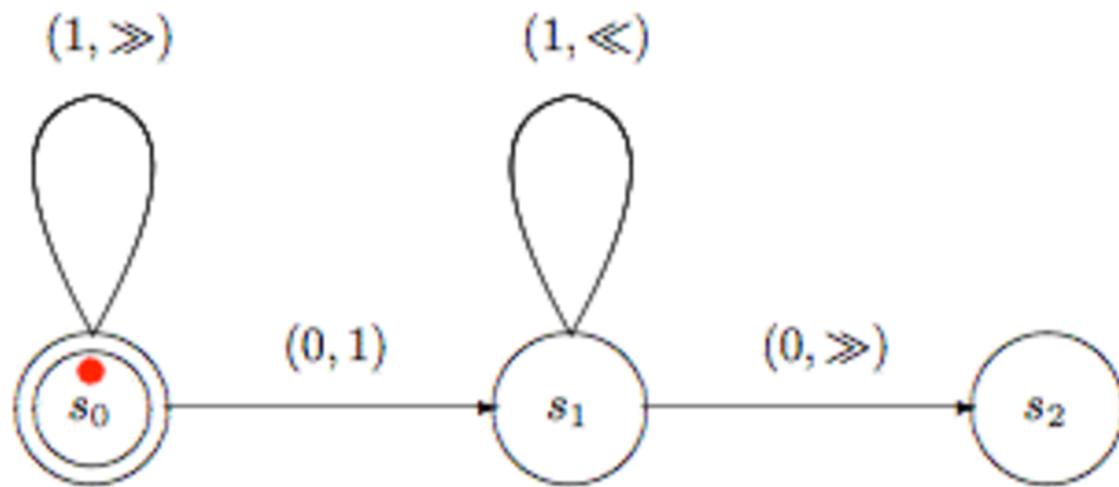


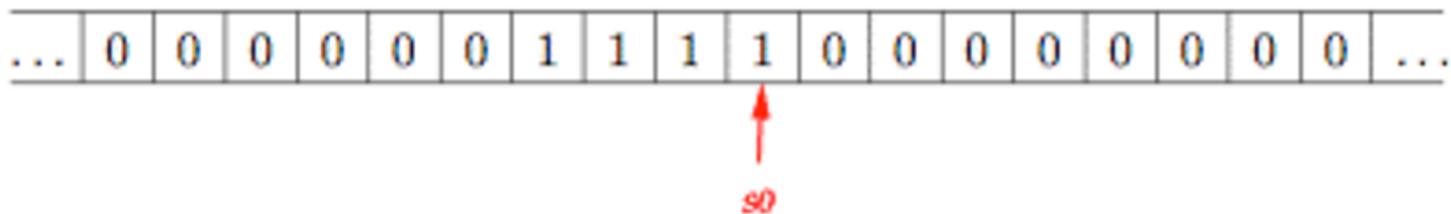
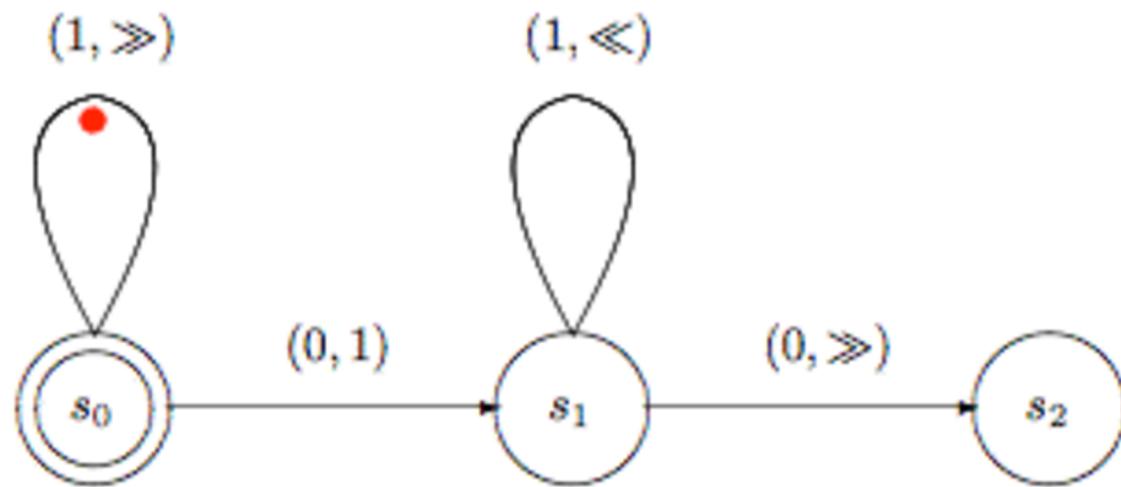


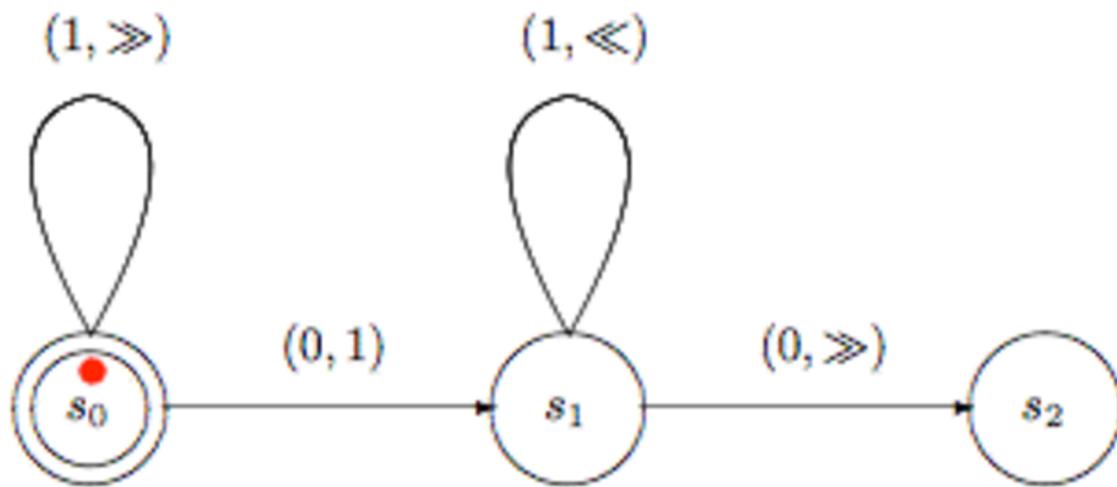


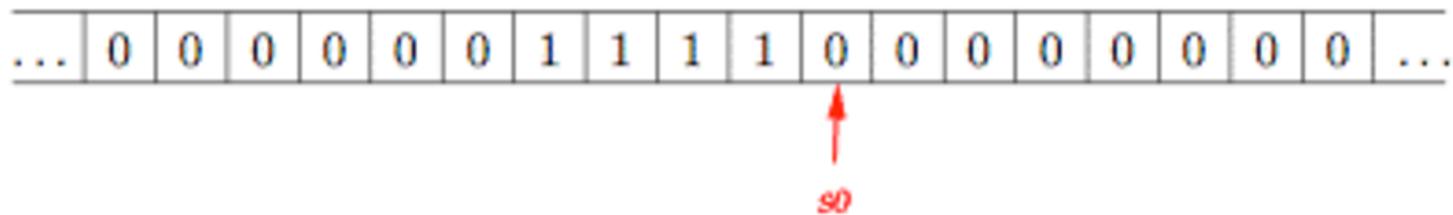
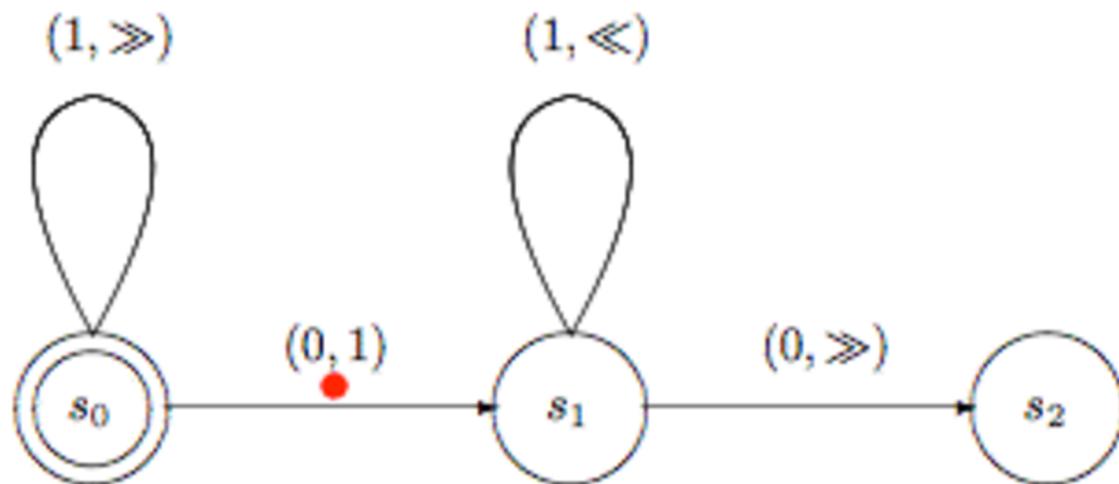


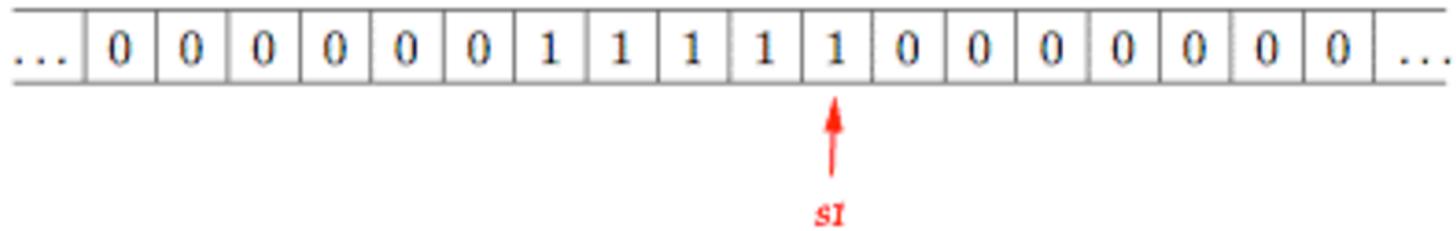
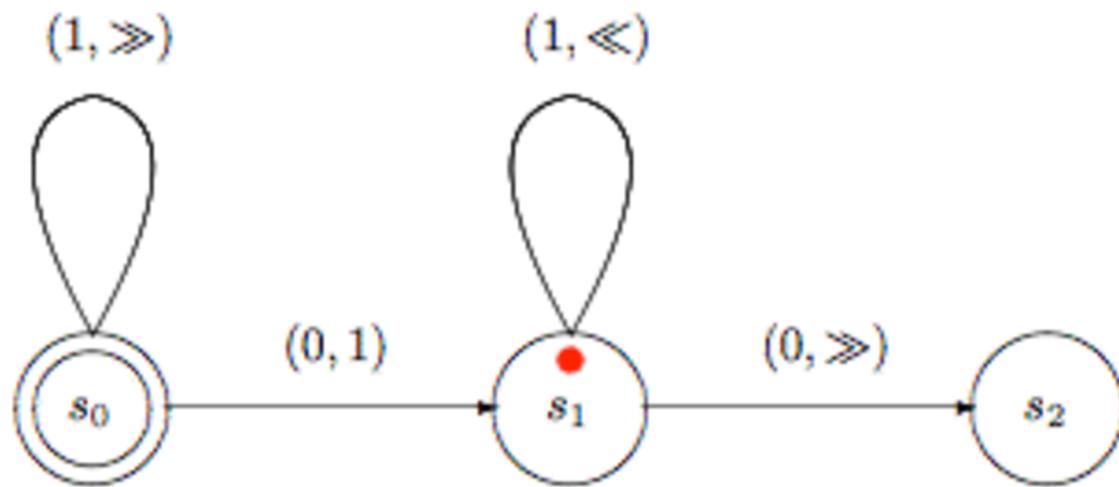


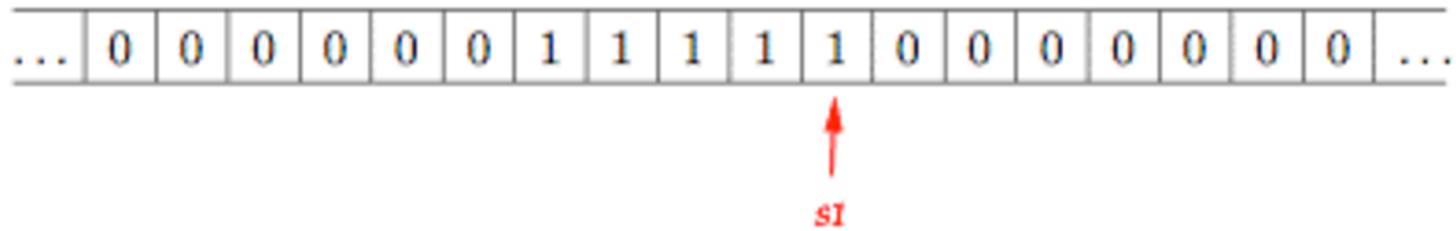
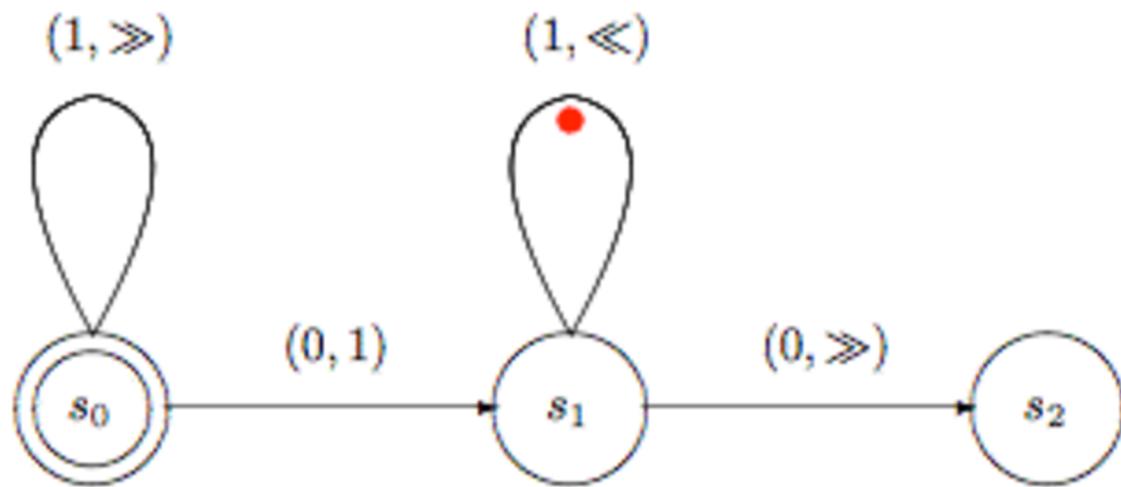


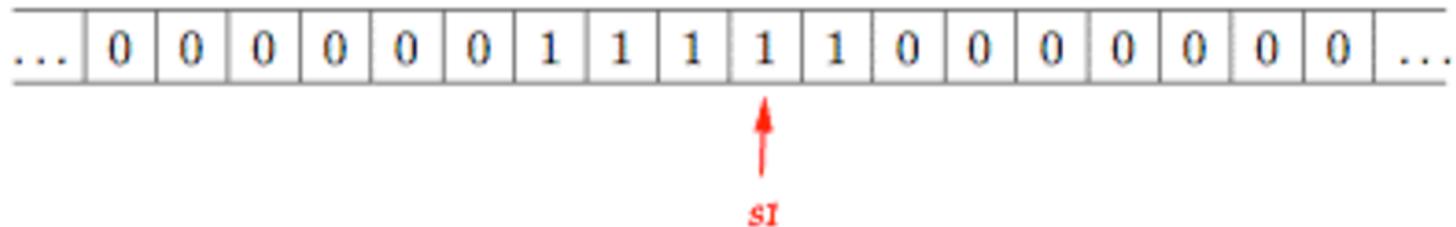
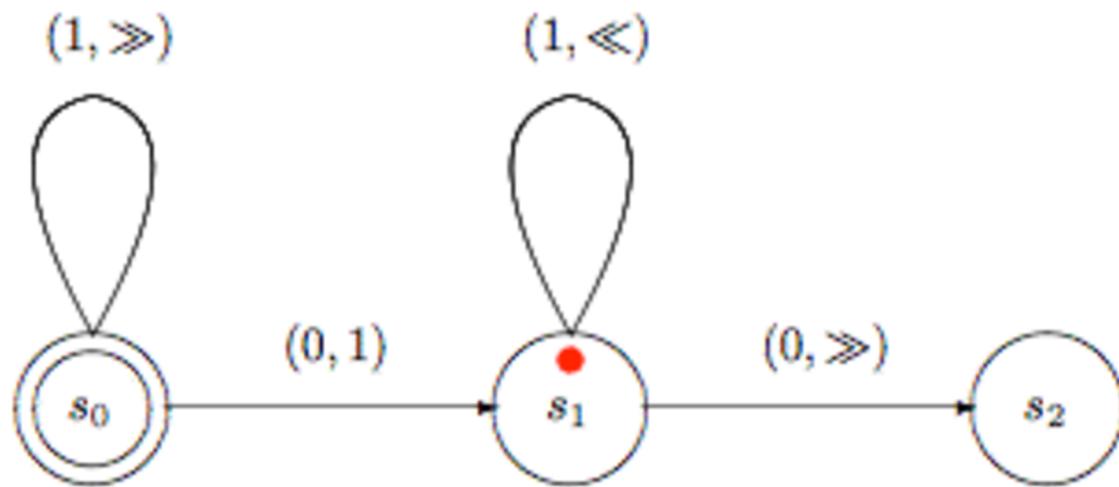


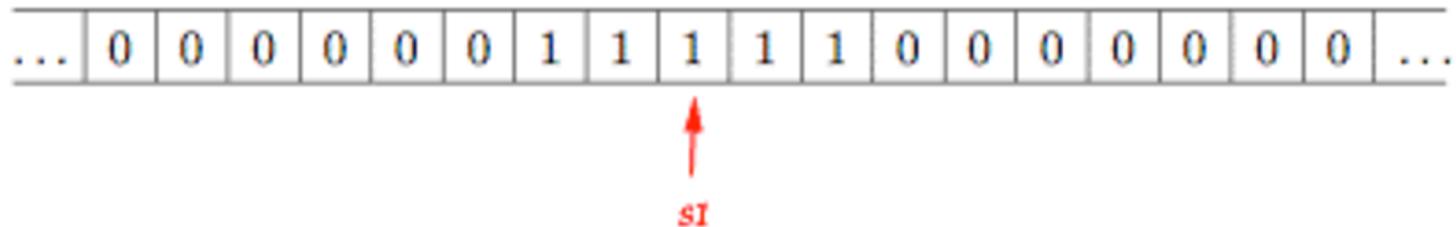
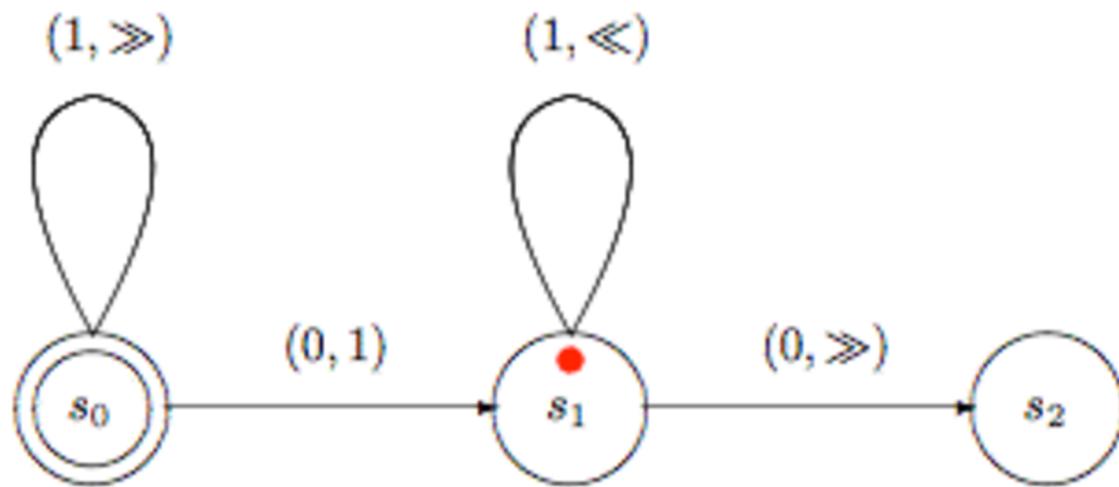


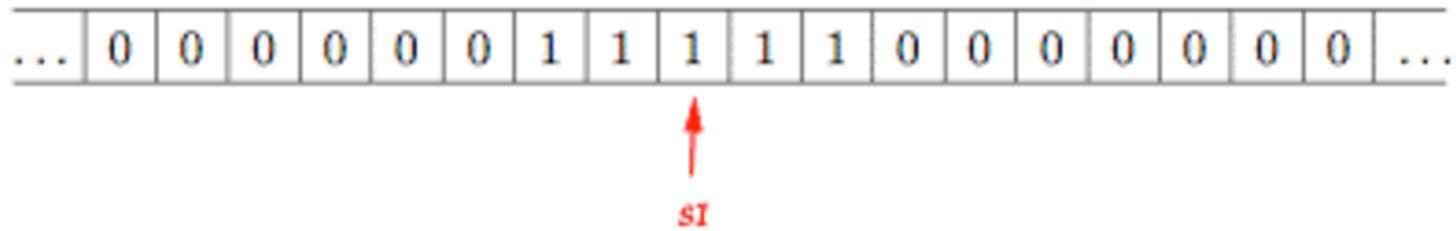
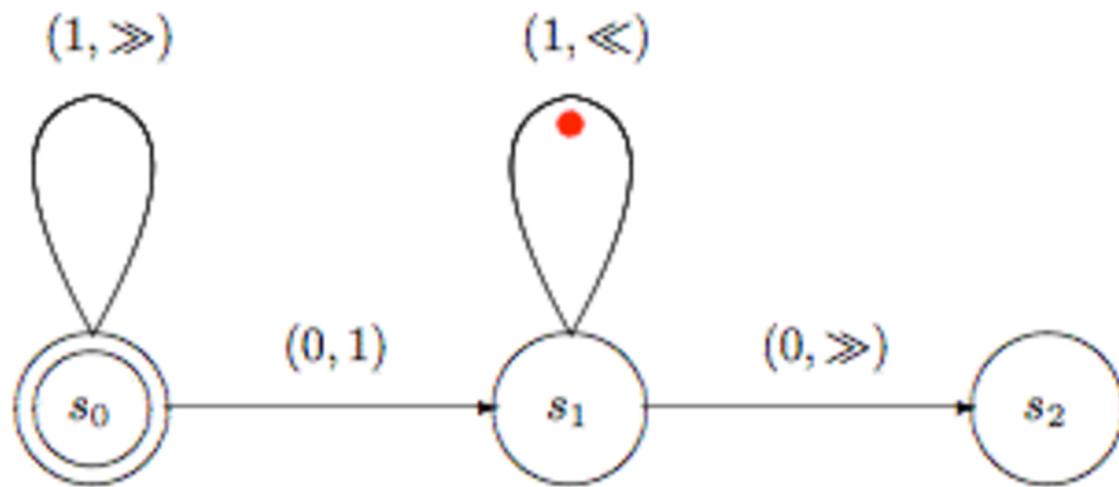


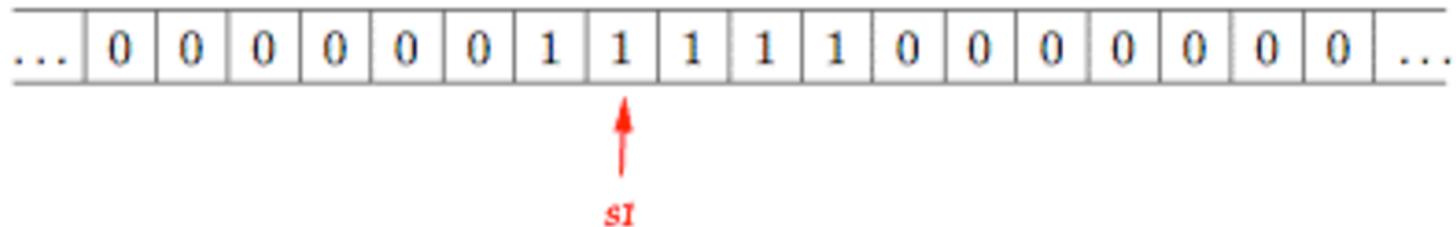
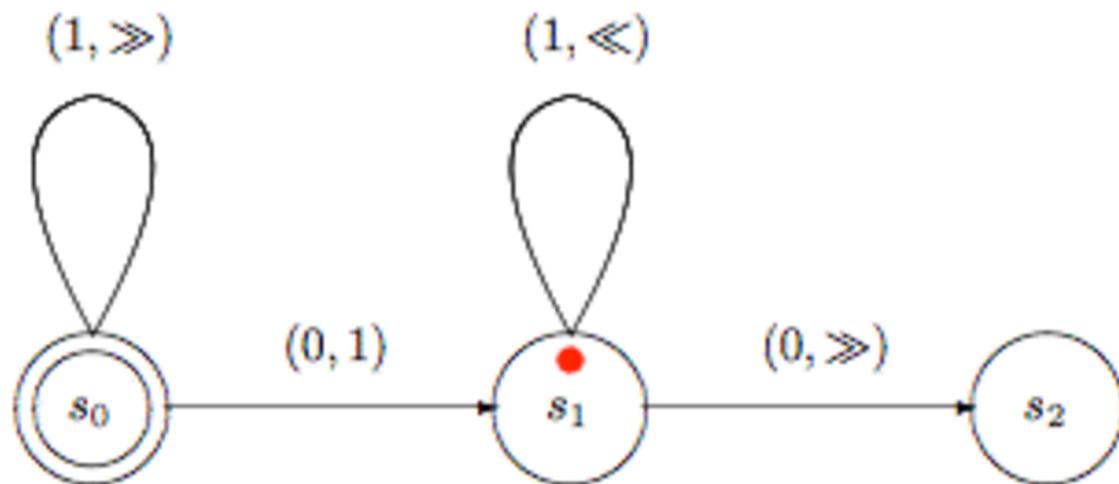


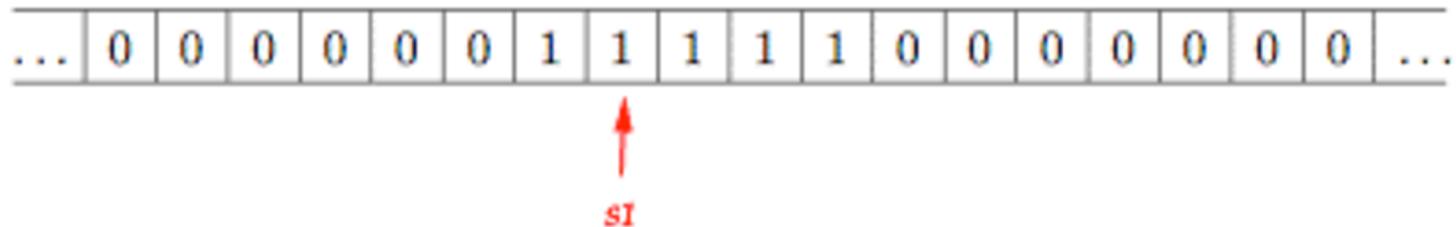
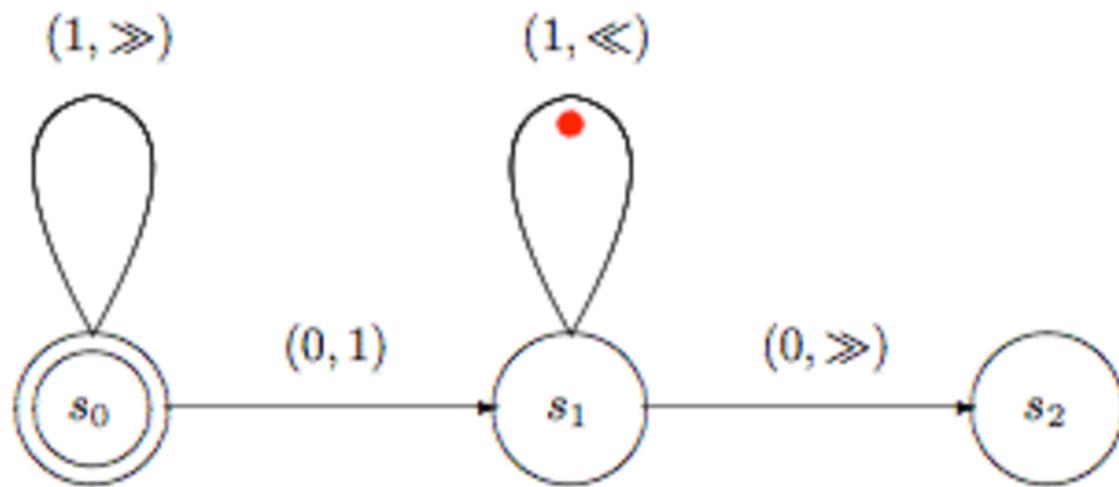


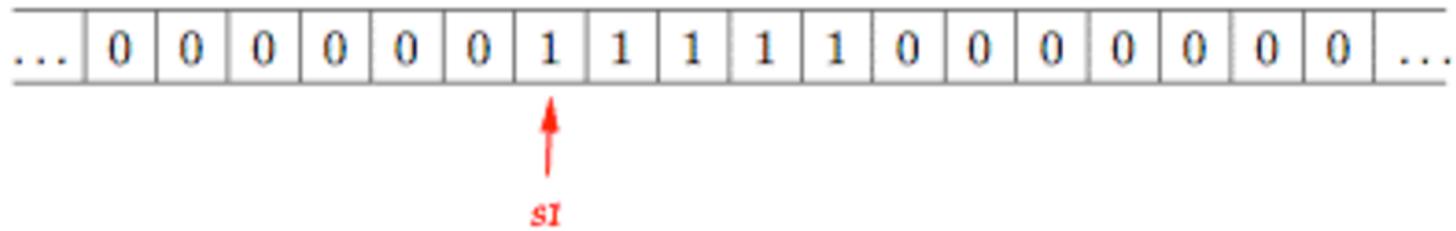
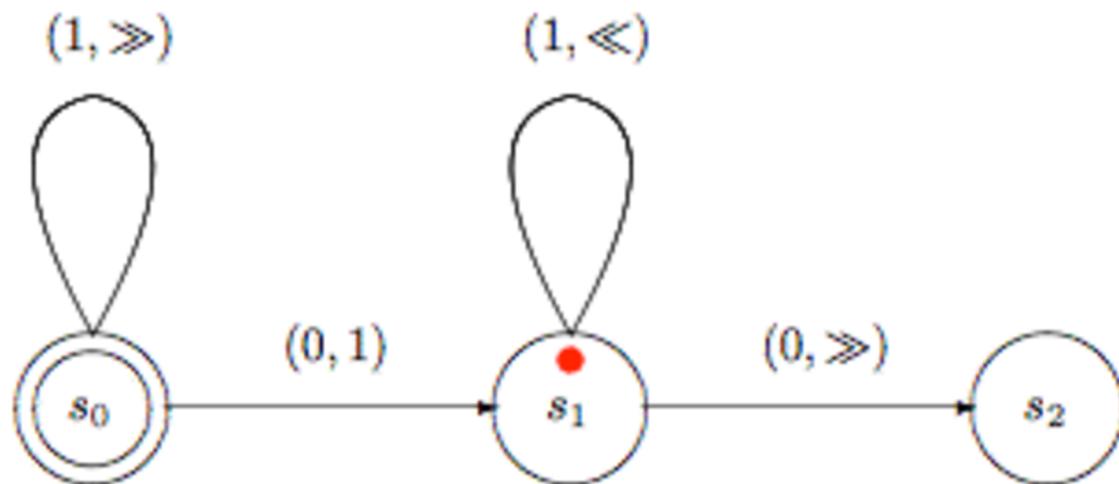


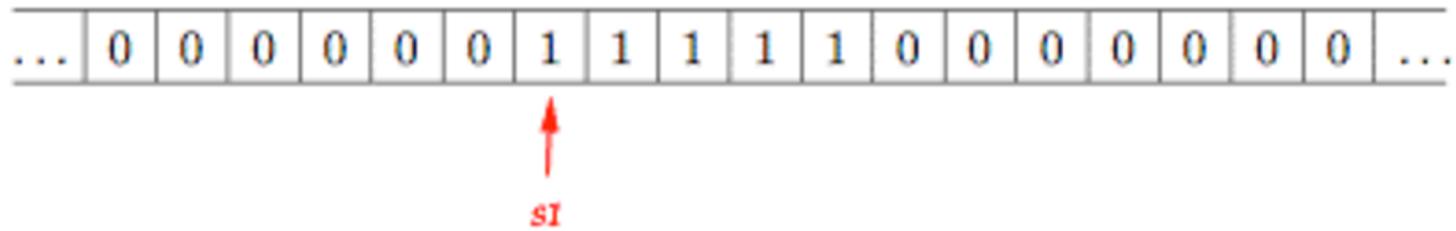
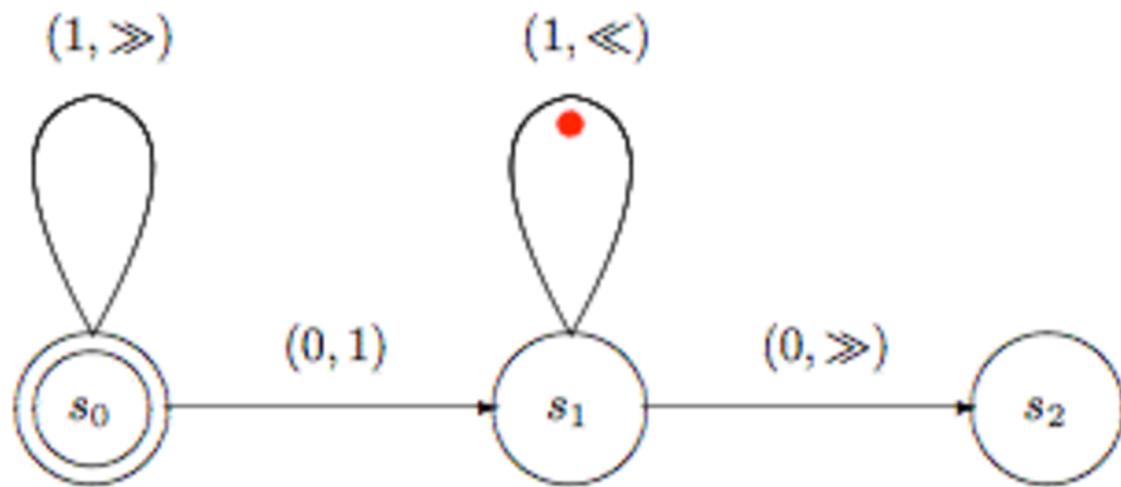


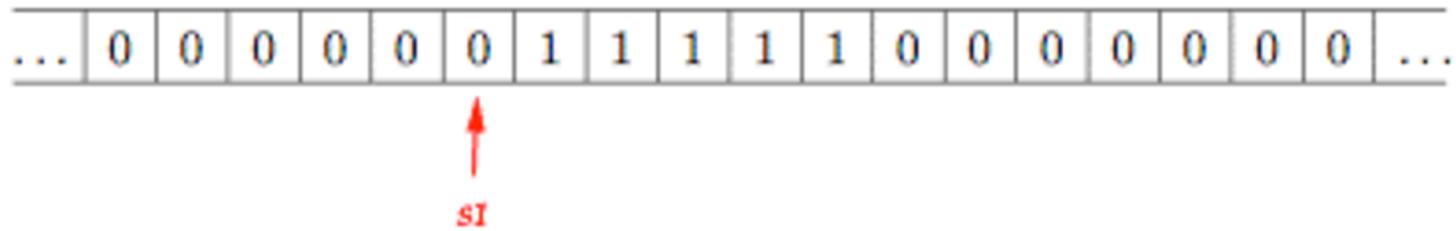
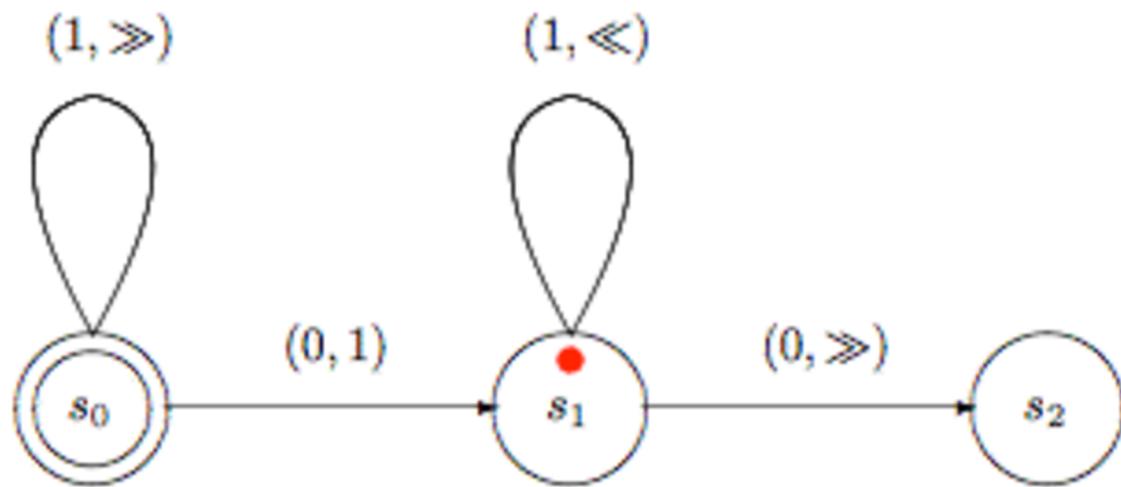


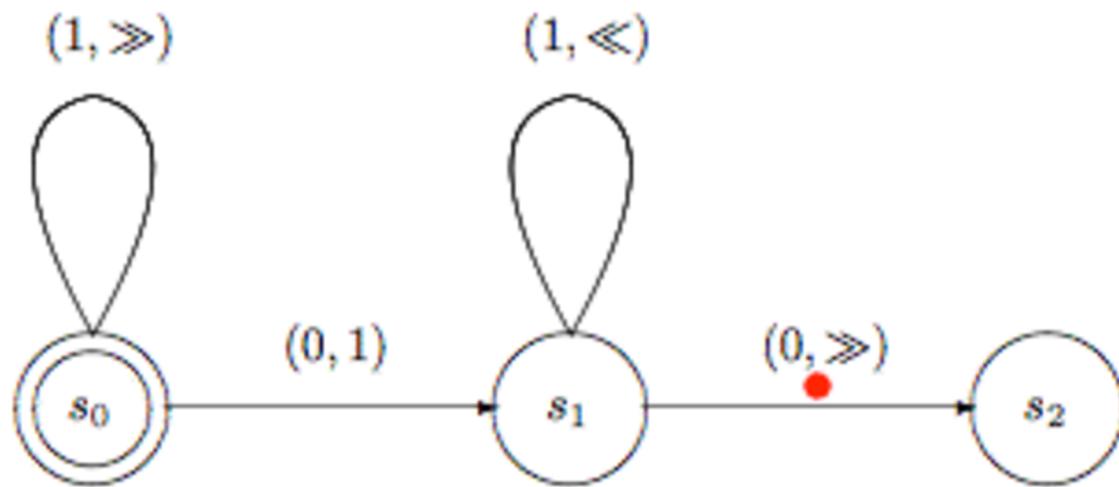






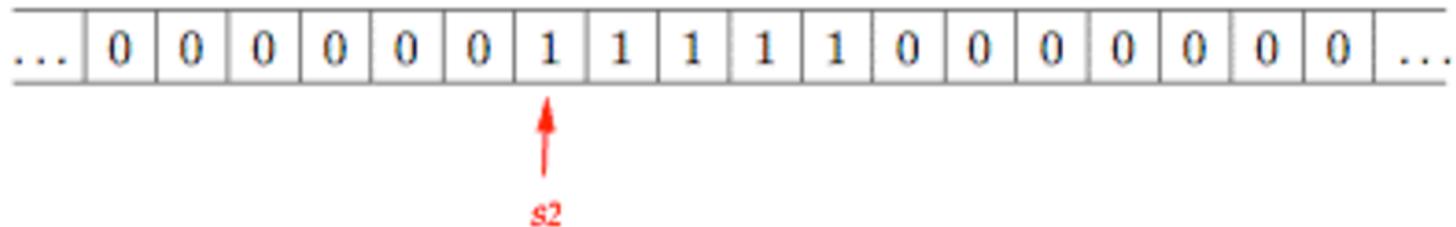
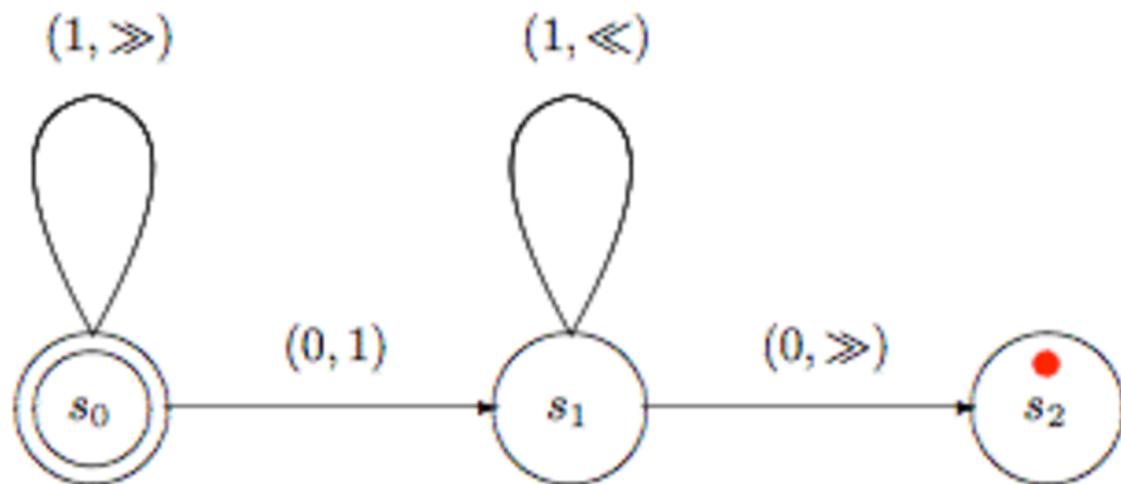




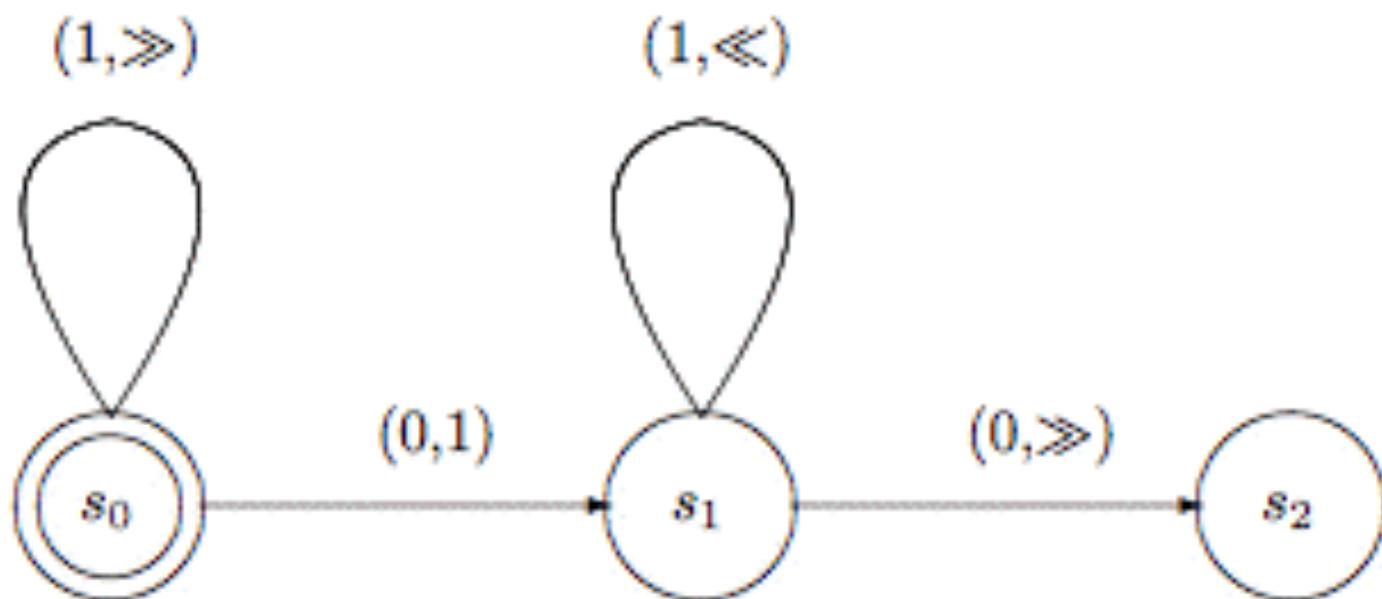


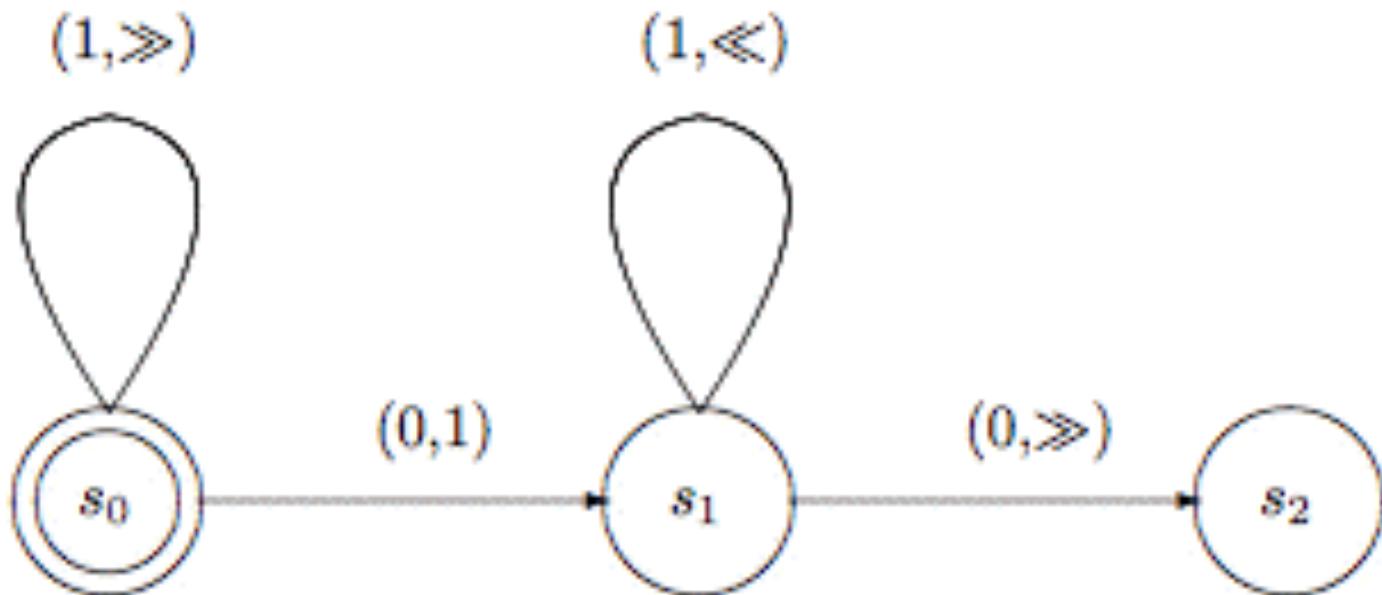
... 0 0 0 0 0 0 1 1 1 1 1 1 0 0 0 0 0 0 0 ...

s1



Question: what does it compute?



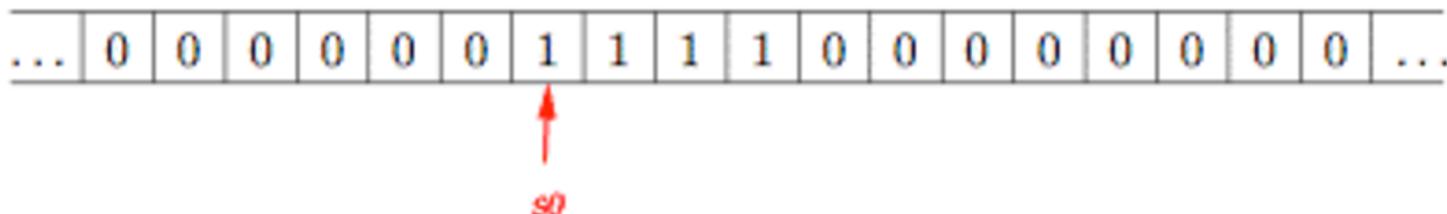
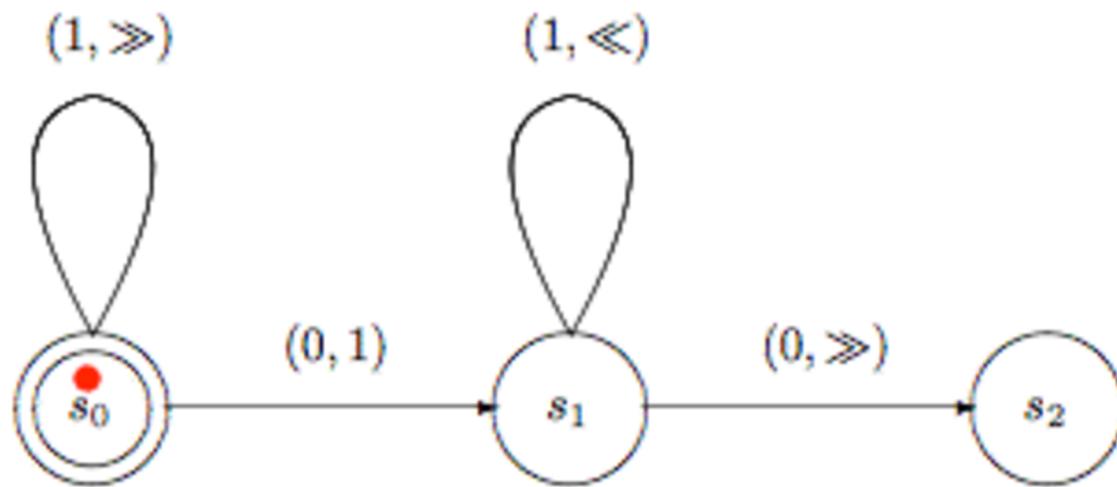


```
function succ (n) {  
    return n + 1;  
}
```

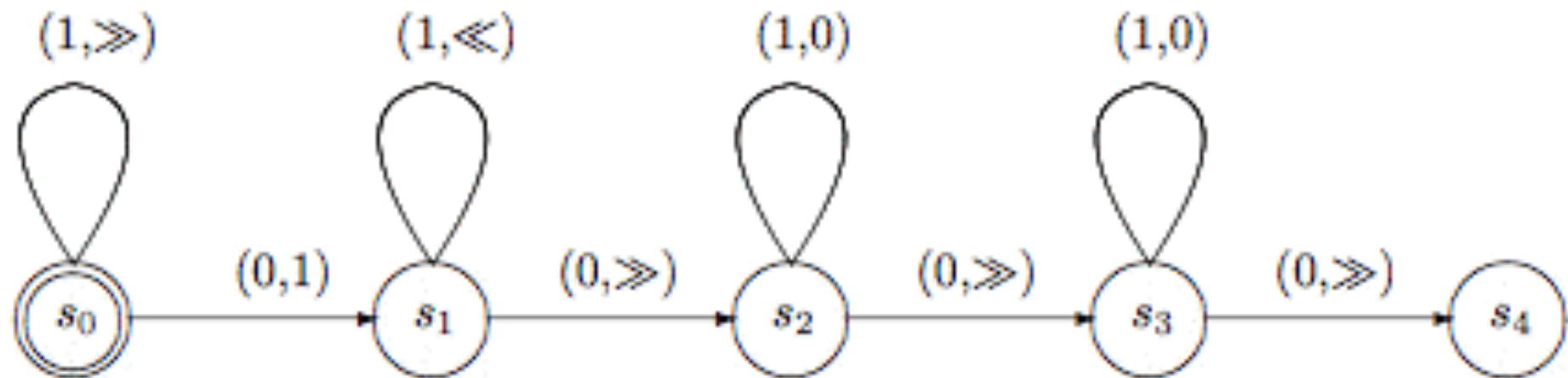
(lambda (x) (+ x 1))

Successor (add-one) function

assuming that number n as a block of $n+1$ copies of the symbol '1' on the tape (here, $n=3$)



Addition of n+m



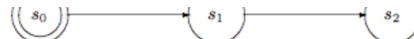
<http://graphics.stanford.edu/~seander/bithacks.html>

Maybe you prefer to use bit operations?

The (Hi)Story of Software Engineering Computer Science

0100
010010101000
1010010101000
010010101000
010010101000
0010010101000
0010010101000
0010010101000

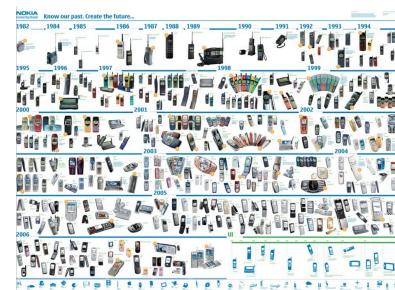
· 0 1 1 1 1 0 1 1 1 1 1 0 ...



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



s_2



orange™

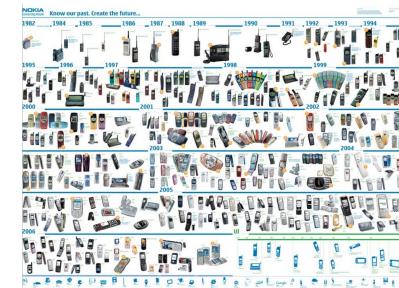
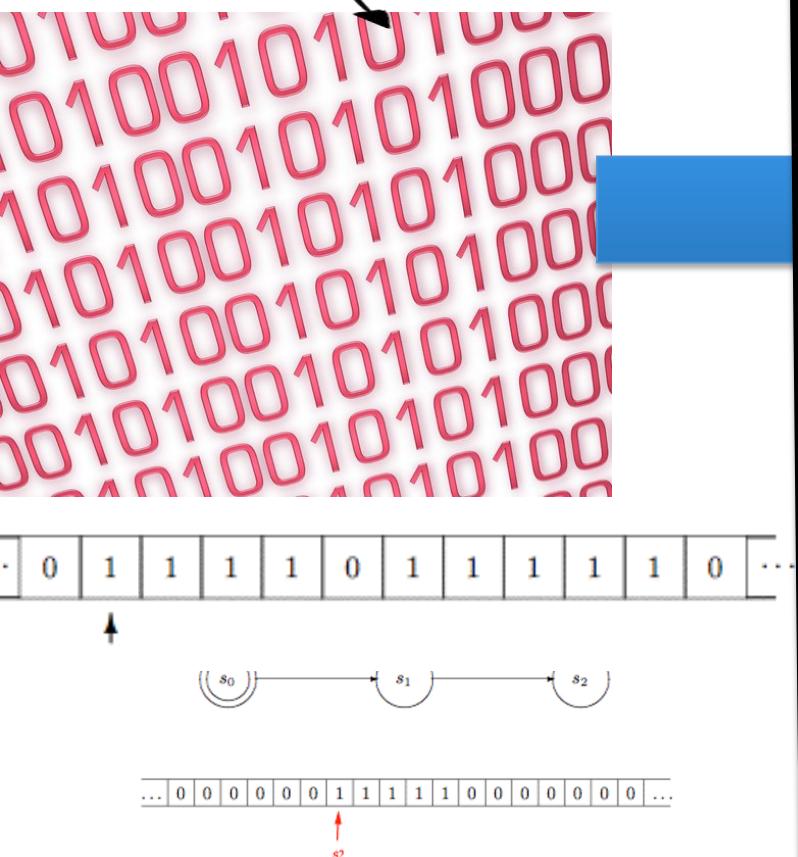


Google

twitter



Software Languages



orange™



ANDRO

Google

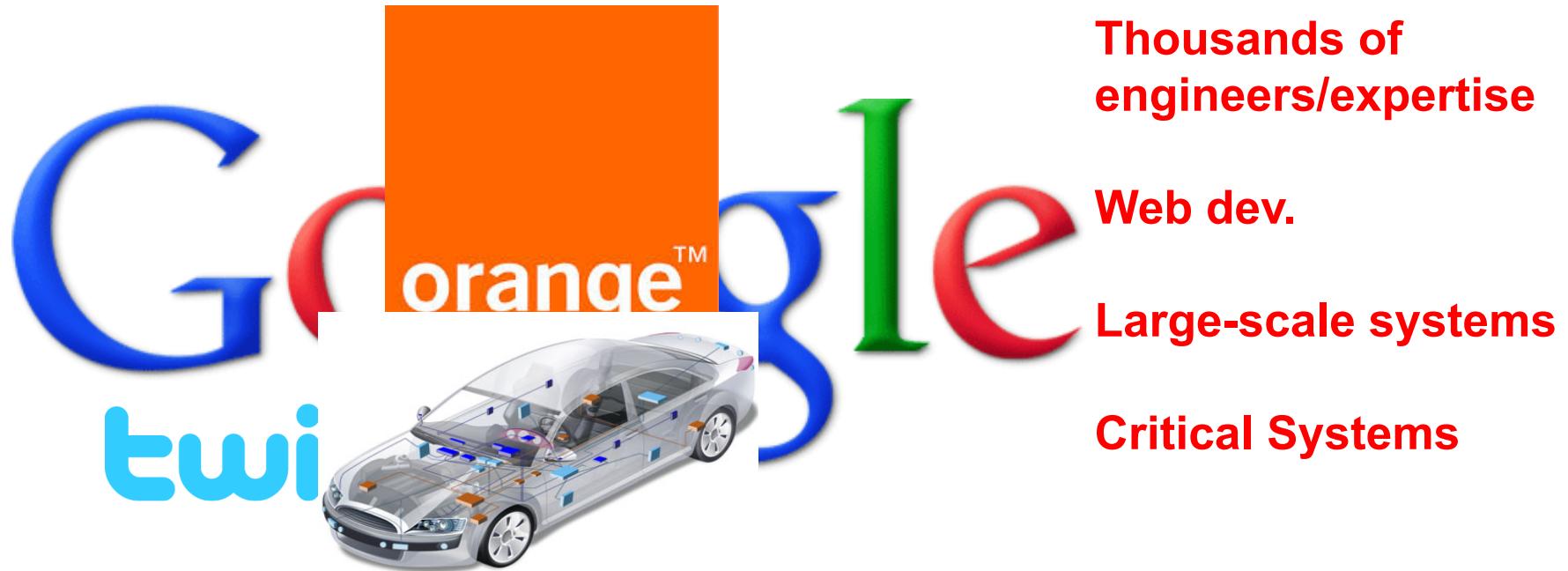
twitter



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.

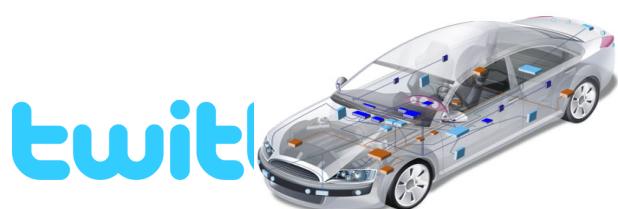
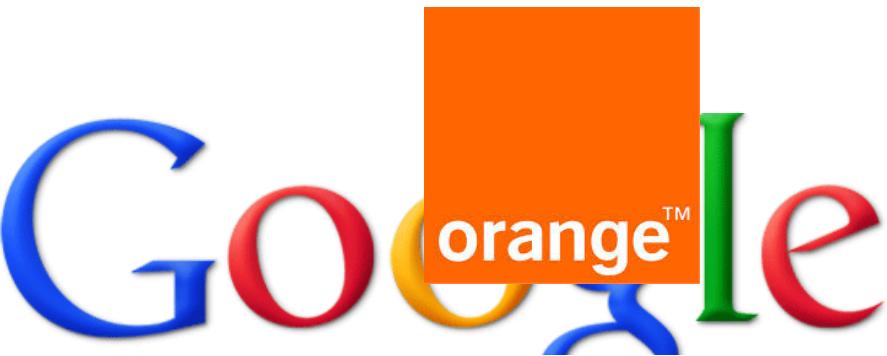
Large-scale systems

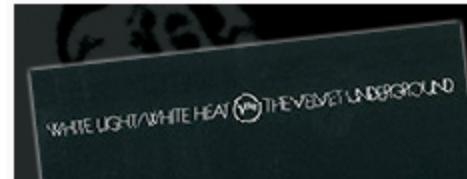
Critical Systems

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 spreadsheet contains a grid of binary values (0s and 1s) representing a state transition table for a Turing machine. The columns and rows are labeled with binary numbers from 0 to 11. The formula bar at the top shows a complex formula involving VLOOKUP, IFERROR, and INDEX functions, which is used to calculate the next state based on the current state and input.

Formulas are Turing complete

2013 GIFT GUIDE

AN ASTRONAUT'S GUIDE TO LIFE ON EARTH

Formulas are Turing complete

Turing Machine Successor																	
File		Home		Insert		Review		Add-Ins		Data		Page Layout		Formulas		Cells	
Paste	Cut	Format Painter	Find & Select	Font	Font Color	Font Style	Font Size	Font	Font	Font	Font	Font	Font	Font	Font	Font	Font
Clipboard	Format Cells	Format Selection	Format Painter	Font	Font	Font	Font	Font	Font	Font	Font	Font	Font	Font	Font	Font	Font
A1	B1	C1	D1	E1	F1	G1	H1	I1	J1	K1	L1	M1	N1	O1	P1	Q1	
1																	
2																	
3																	
4	4 S1	-	-	-	-	1	1	1	-	-	-	-	-	-	-	-	-
5	5 S1	-	-	-	-	1	1	1	-	-	-	-	-	-	-	-	-
6	6 S1	-	-	-	-	1	1	1	-	-	-	-	-	-	-	-	-
7	7 S1	-	-	-	-	1	1	1	-	-	-	-	-	-	-	-	-
8	8 S2	-	-	-	-	1	1	1	-	-	-	-	-	-	-	-	-
9	9 S2	-	-	-	-	1	1	1	-	-	-	-	-	-	-	-	-
10	10 S2	-	-	-	-	1	1	1	-	-	-	-	-	-	-	-	-
11	9 S3	-	-	-	-	1	1	1	1	-	-	-	-	-	-	-	-
12	8 S3	-	-	-	-	1	1	1	1	-	-	-	-	-	-	-	-
13	7 S3	-	-	-	-	1	1	1	1	-	-	-	-	-	-	-	-
14	6 S3	-	-	-	-	1	1	1	1	-	-	-	-	-	-	-	-
15	7 S4	-	-	-	-	1	1	1	1	-	-	-	-	-	-	-	-
16	7 S4	-	-	-	-	1	1	1	1	-	-	-	-	-	-	-	-
17	7 S4	-	-	-	-	1	1	1	1	-	-	-	-	-	-	-	-
18	7 S4	-	-	-	-	1	1	1	1	-	-	-	-	-	-	-	-
19	7 S4	-	-	-	-	1	1	1	1	-	-	-	-	-	-	-	-
20	7 S4	-	-	-	-	1	1	1	1	-	-	-	-	-	-	-	-
21	7 S4	-	-	-	-	1	1	1	1	-	-	-	-	-	-	-	-
22	7 S4	-	-	-	-	1	1	1	1	-	-	-	-	-	-	-	-
23	7 S4	-	-	-	-	1	1	1	1	-	-	-	-	-	-	-	-
24	7 S4	-	-	-	-	1	1	1	1	-	-	-	-	-	-	-	-
25	7 S4	-	-	-	-	1	1	1	1	-	-	-	-	-	-	-	-
26	7 S4	-	-	-	-	1	1	1	1	-	-	-	-	-	-	-	-
27	7 S4	-	-	-	-	1	1	1	1	-	-	-	-	-	-	-	-
28	7 S4	-	-	-	-	1	1	1	1	-	-	-	-	-	-	-	-
29	7 S4	-	-	-	-	1	1	1	1	-	-	-	-	-	-	-	-
30	7 S4	-	-	-	-	1	1	1	1	-	-	-	-	-	-	-	-

Youtube video <https://t.co/RTfJAxXYaX>

<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?

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

Questions to the audience

- 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?

The answer to such « thought-provoking » questions seems obvious at first glance

- Help to define the good properties of software languages we expect
- Help to understand why there is still innovation in language design

Programming the Turing Machine

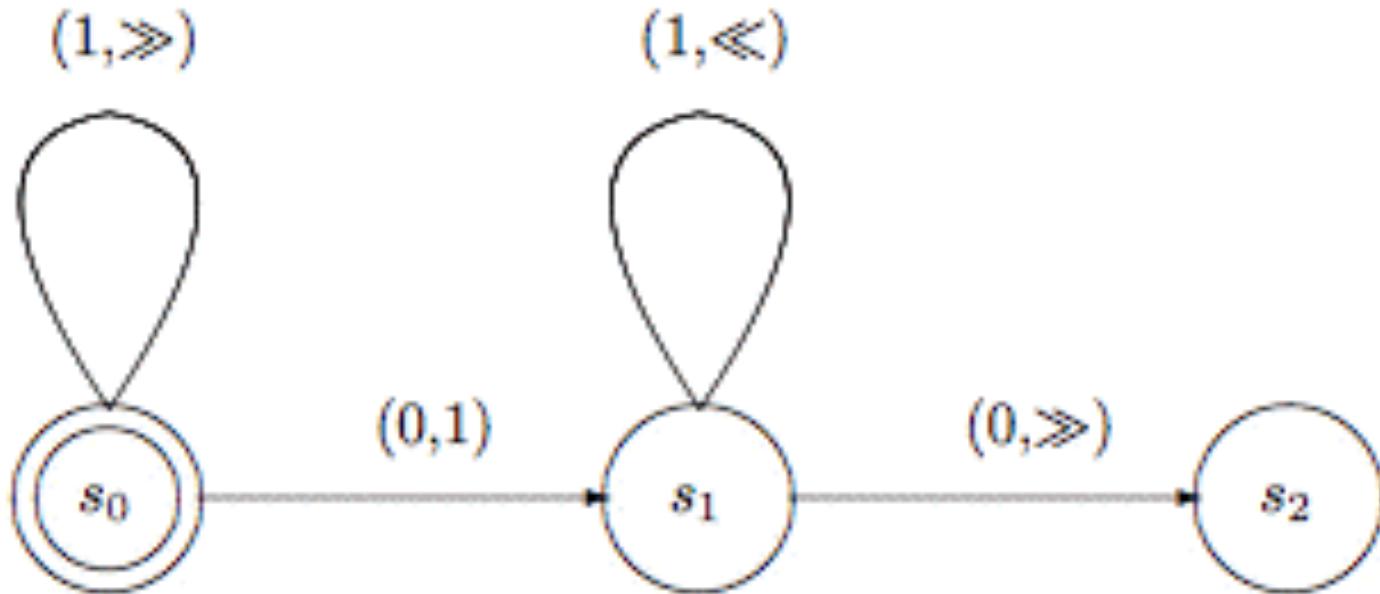
Why aren't we using tapes, states and transitions after all ?

Software Languages



Hard to write and understand.
No abstractions.
Hard to debug and test.
Poor language constructs. Poor tooling support.
Performance.
Usability, productivity,
reusability, safety,
expressiveness, learnability.

Question: what does it compute?

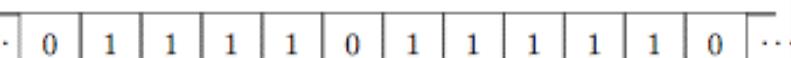
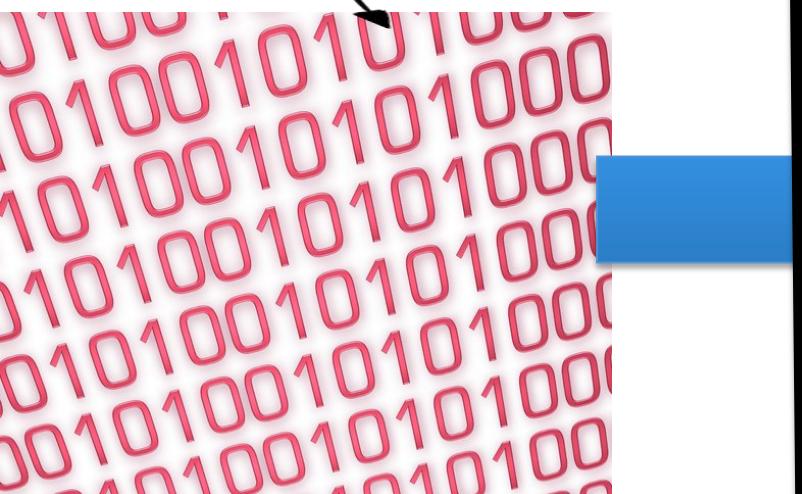


**Performance, usability,
productivity, reusability, safety,
expressiveness, learnability.**

Qualities and challenges

- Cognitive dimensions (see references after)
- Abstractions
 - Eg Kramer “Abstraction and Modelling - A Complementary Partnership” MODELS’08
- Separation of concerns/modularity
 - Eg Tarr et al., ICSE’99
- Scalability
 - Growing a language (like Scala)
- Performance
- ...

Languages

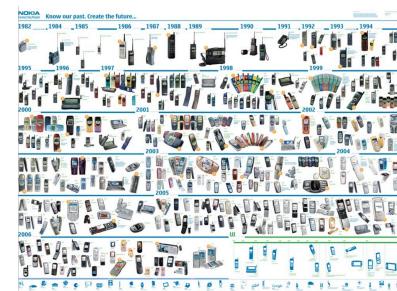


• 0 1 1 1 1 0 1 1 1 1 1 1 0 ...



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

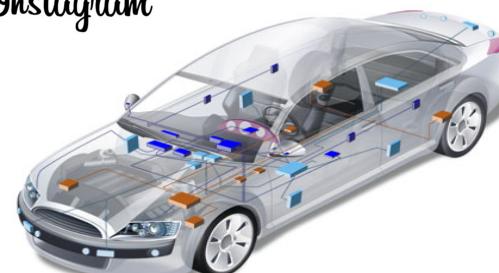
Complex Systems



orange™



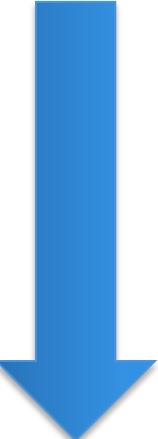
Orange



We need languages

1. At a high level of abstraction
 1. Still general-purpose
 2. Generation of other artefacts written in other languages
 3. Transformation, refinement
2. Multiplicity of languages
 1. Divide and conquer
 2. Specific to a problem or “domain”
 3. Induce a way to “compose” languages

(Combemale et al. “On the Globalization of Domain-Specific Languages”)



How Language Shapes Thought

The languages we speak affect our perceptions of the world

By Lera Boroditsky

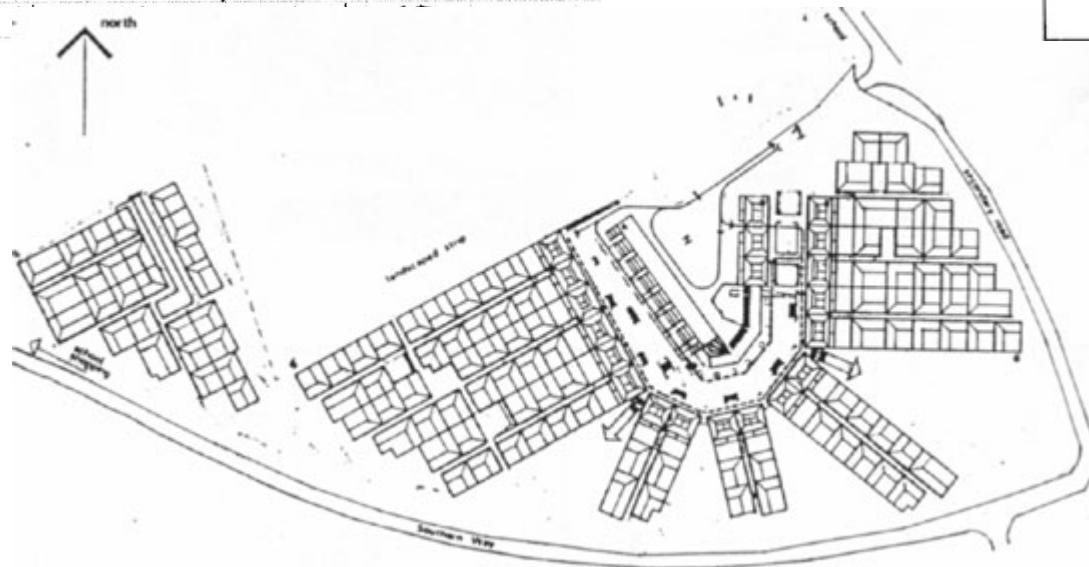
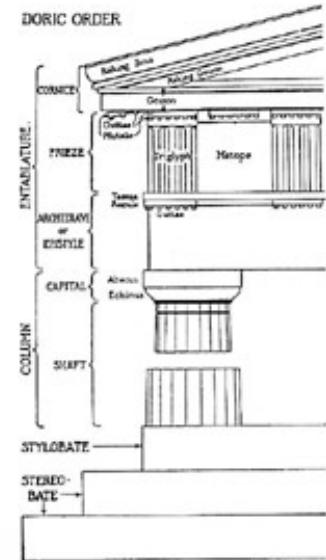
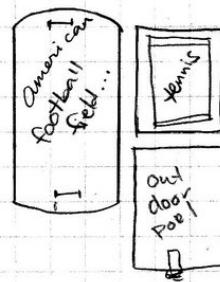
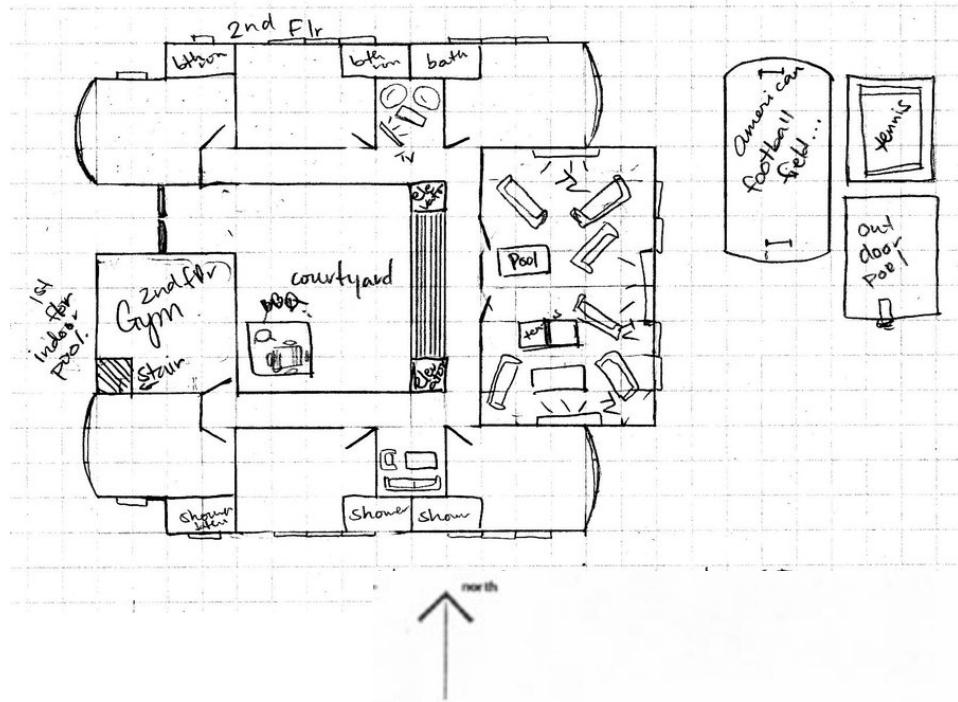
“Even variations in grammar can profoundly affect how we see the world.”

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. »

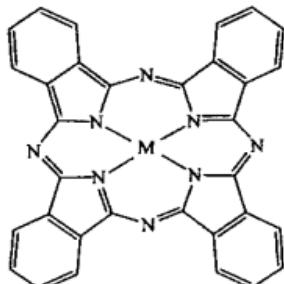
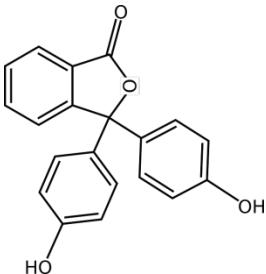
Architecture



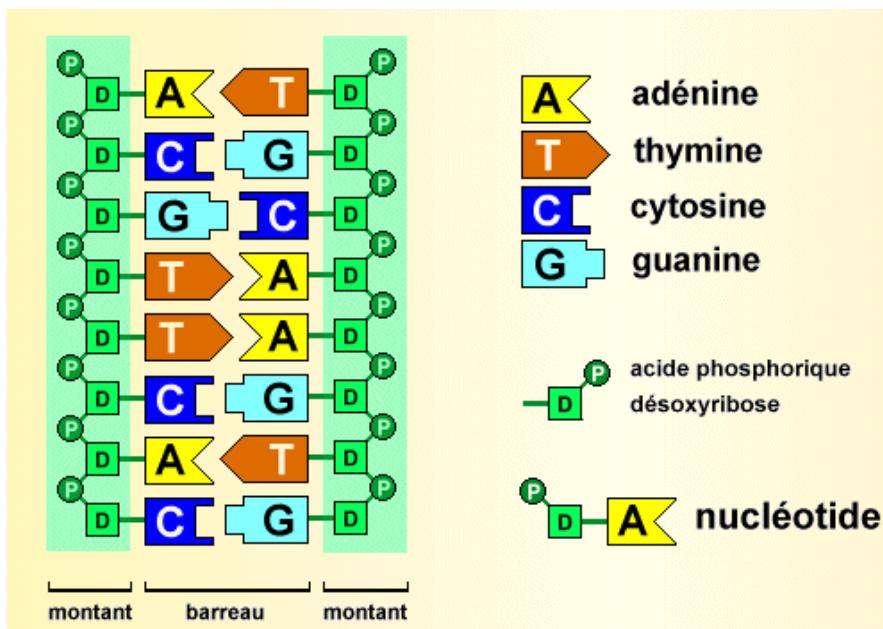
Cartography



Biology



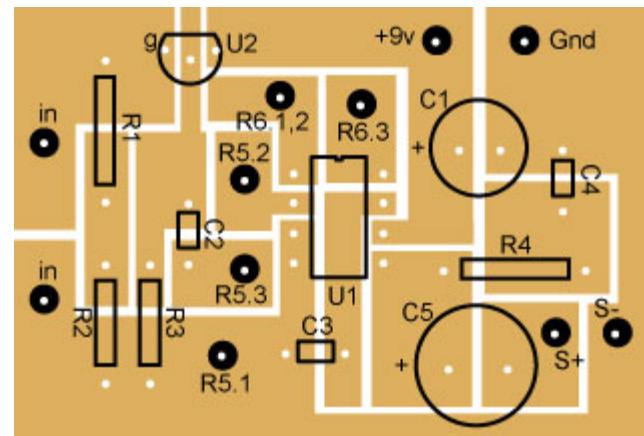
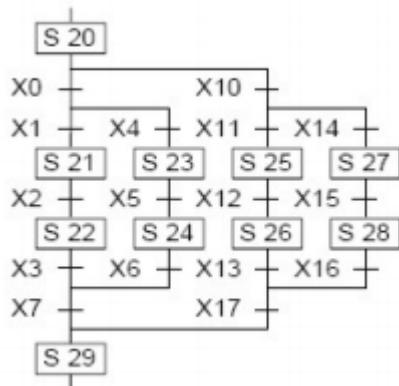
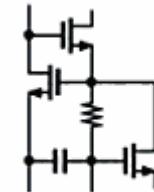
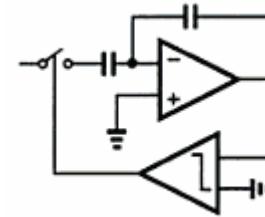
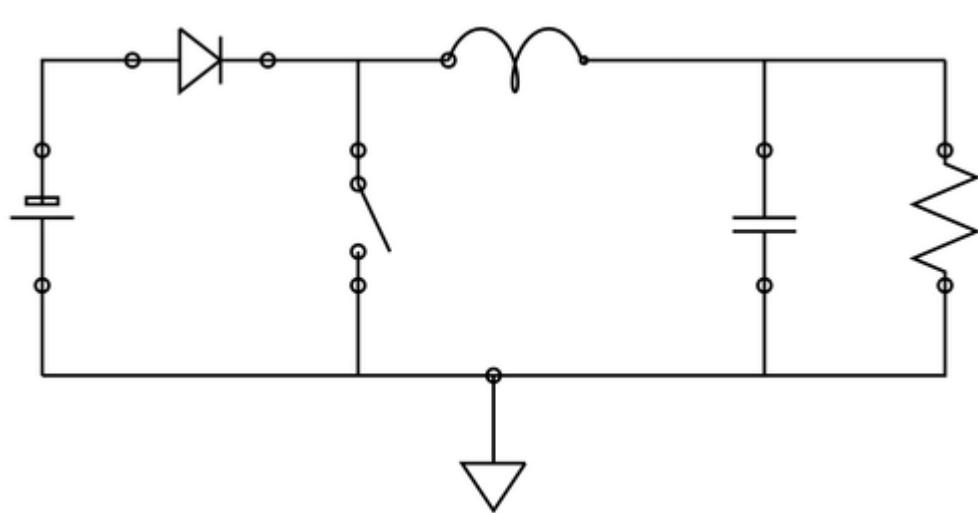
phthalocyanine



60	70	80	90	100
AGACCCCCAG	CAACCCCCGG	GGCCGTCGG	CCTCGGTGCGT	GTCGTGTGAT
160	170	180	190	200
AGACCCCGCG	TACGAATGCC	GGTCCACCAA	CAACCCGTGG	GCTTCGCAAGC
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	GGCATTTGGGC	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	GGTGTACCAAC
1160	1170	1180	1190	1200
ACTTGATCTA	TATACCCACCA	ATGTGTCATT	TATGGGGCCGC	ACATATCGTC
1260	1270	1280	1290	1300
CTGTCATGT	ACCTTTGTAT	CCTATCAGCC	TTGGTTCCCA	GGGGGTGTCT
1360	1370	1380	1390	1400
TGTTTGAGGG	GGTGGTGCCA	GATGAGGTGA	CCAGGATAGA	TCTCGACCG
1460	1470	1480	1490	1500
TCAGAGTCCTC	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

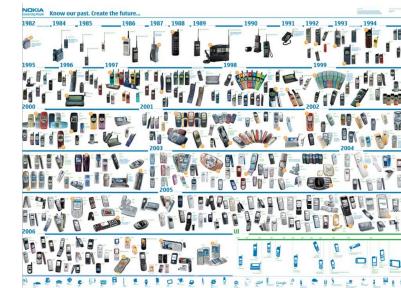
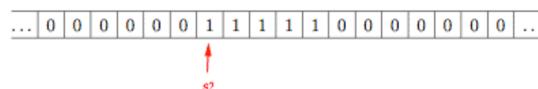
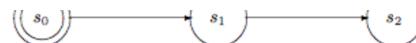
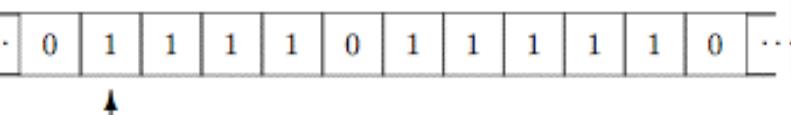
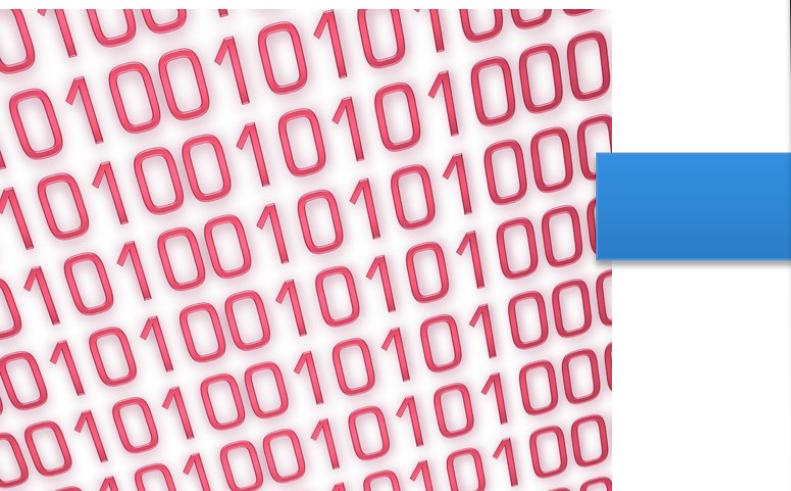
« Languages are the primary way in which system developers communicate, design and implement software systems »

General Purpose Languages

Assembly ?

COBOL ? LISP ? C ? C++ ?

Java? PHP ? C# ? Ruby ?

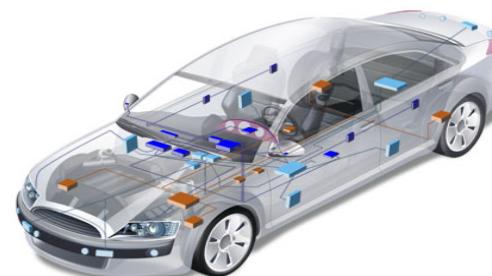


orange™



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.

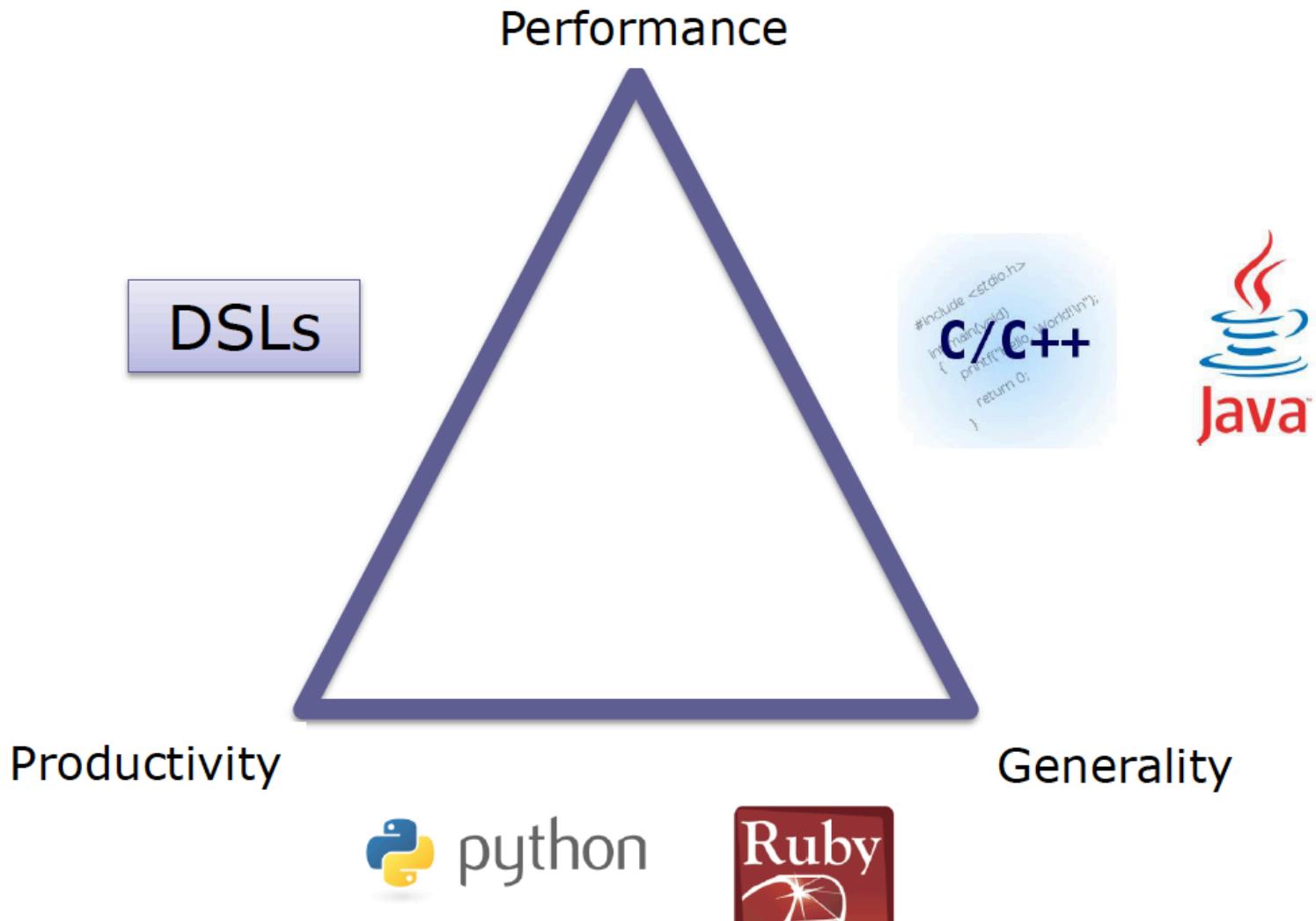


Domain Specific Languages

- 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



A discussable view (slide “OptiML...” Sujeeth et al., ICML’11)



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"

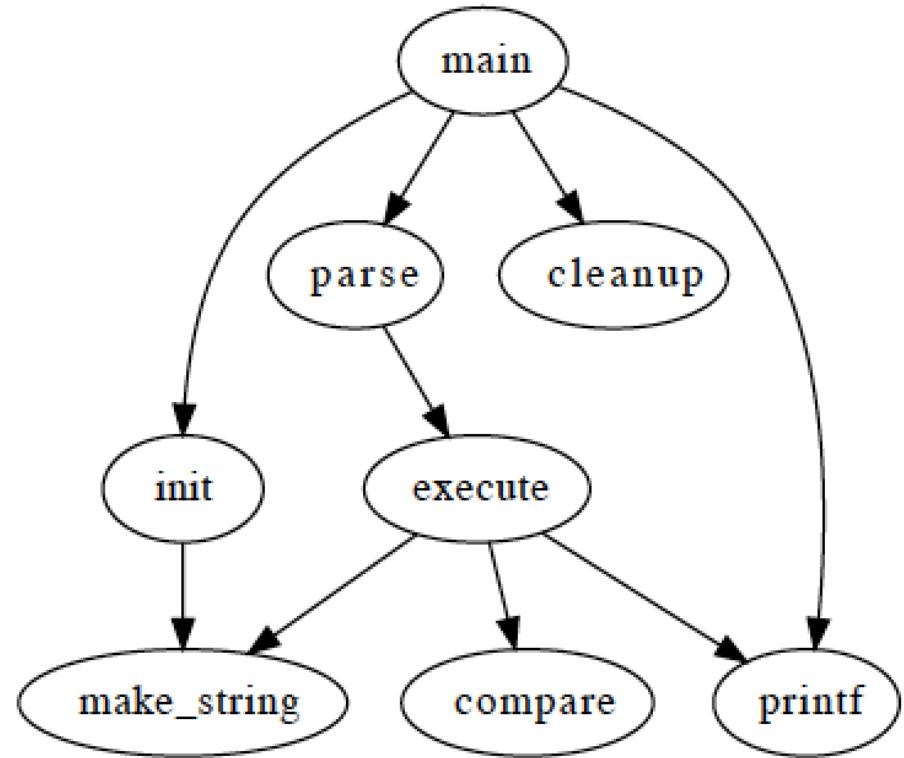
mimetype.assign = (
    ".html" => "text/html",
    ".txt" => "text/plain",
    ".jpg" => "image/jpeg",
    ".png" => "image/png"
)

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;  
}
```

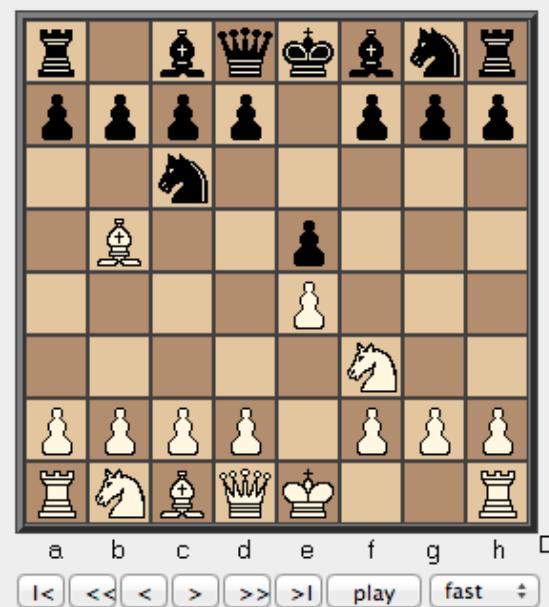


Domain: graph (drawing)

PGN (Portable Game Notation)

```
[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 {This opening is called the Ruy Lopez.} 3... a6
4. Ba4 Nf6 5. 0-0 Be7 6. Re1 b5 7. Bb3 d6 8. c3 0-0 9. h3 Nb8 10. d4 Nbd7
11. c4 c6 12. cxb5 axb5 13. Nc3 Bb7 14. Bg5 b4 15. Nb1 h6 16. Bh4 c5 17. dxe5
Nxe4 18. Bxe7 Qxe7 19. exd6 Qf6 20. Nbd2 Nxd6 21. Nc4 Nxc4 22. Bxc4 Nb6
23. Ne5 Rae8 24. Bxf7+ Rxf7 25. Nxf7 Rxel+ 26. Qxe1 Kxf7 27. Qe3 Qg5 28. Qxg5
hxg5 29. b3 Ke6 30. a3 Kd6 31. axb4 cxb4 32. Ra5 Nd5 33. f3 Bc8 34. Kf2 Bf5
35. Ra7 g6 36. Ra6+ Kc5 37. Ke1 Nf4 38. g3 Nxh3 39. Kd2 Kb5 40. Rd6 Kc5 41. Ra6
Nf2 42. g4 Bd3 43. Re6 1/2-1/2
```



Domain: chess (games)

Regular expression

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

Domain: strings (pattern matching)

Question to the audience

Give three examples of domain-specific languages (DSLs)

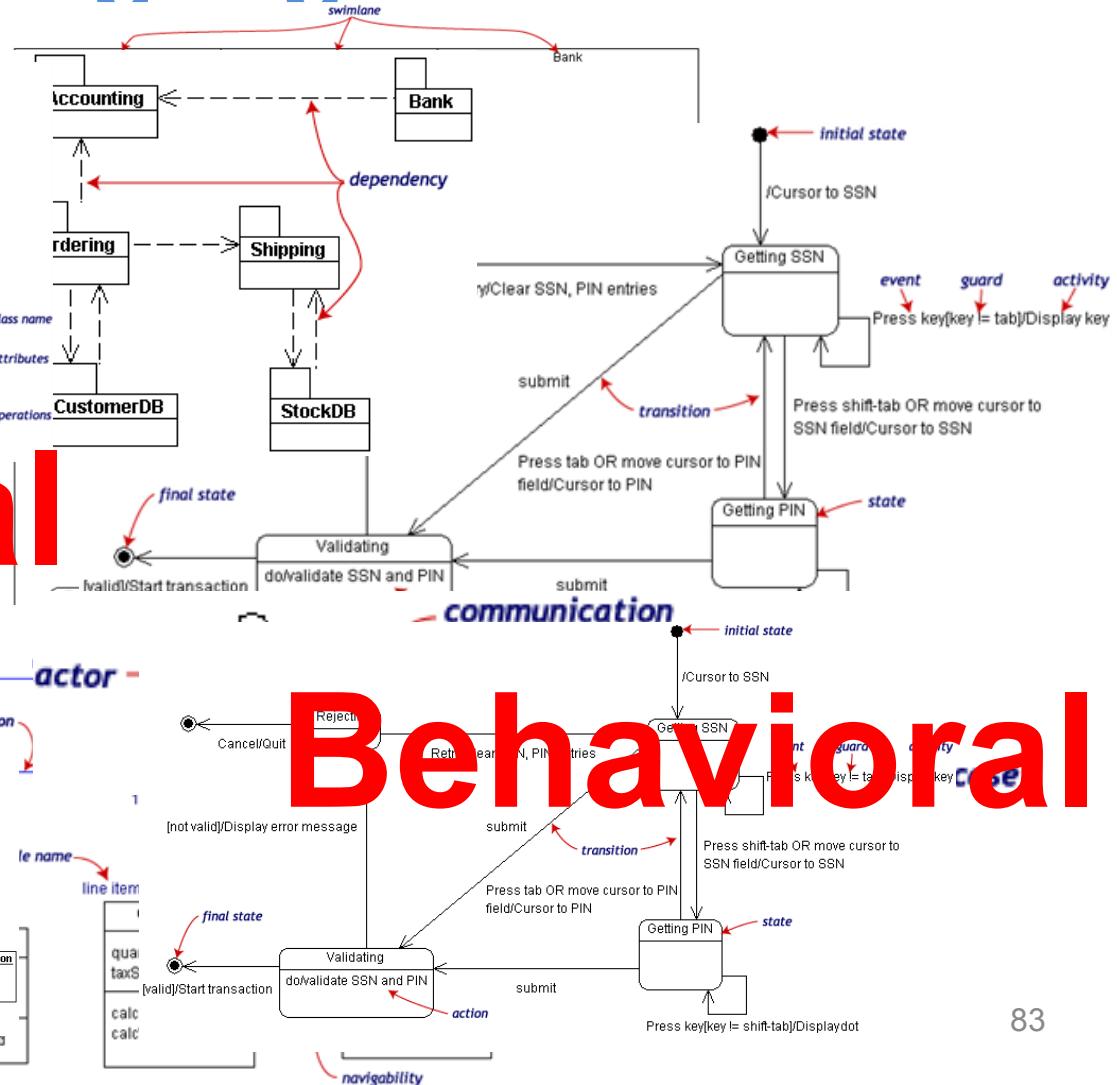
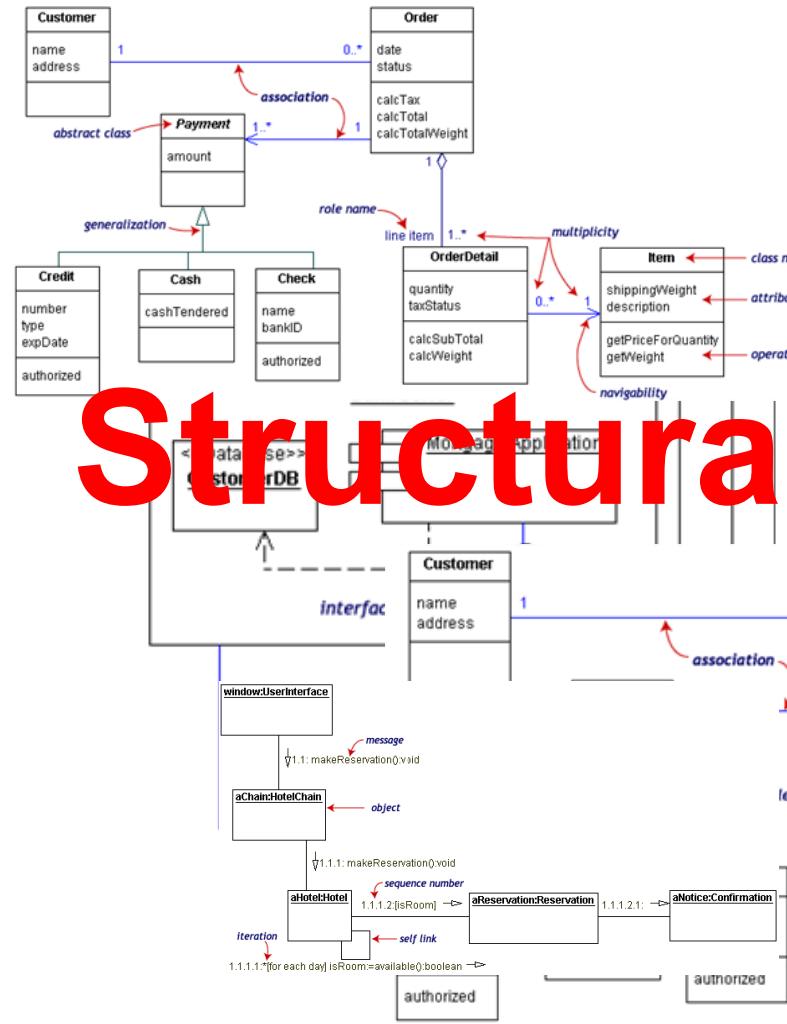
OCL

```
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

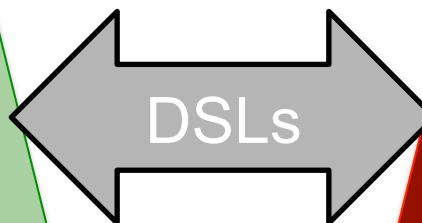
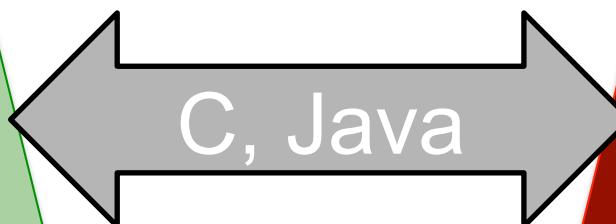
Structural Behavioral



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}

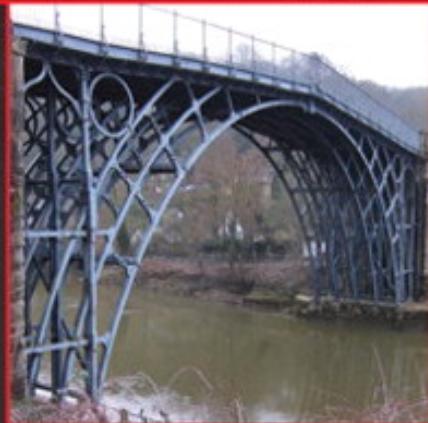
2011

**« 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

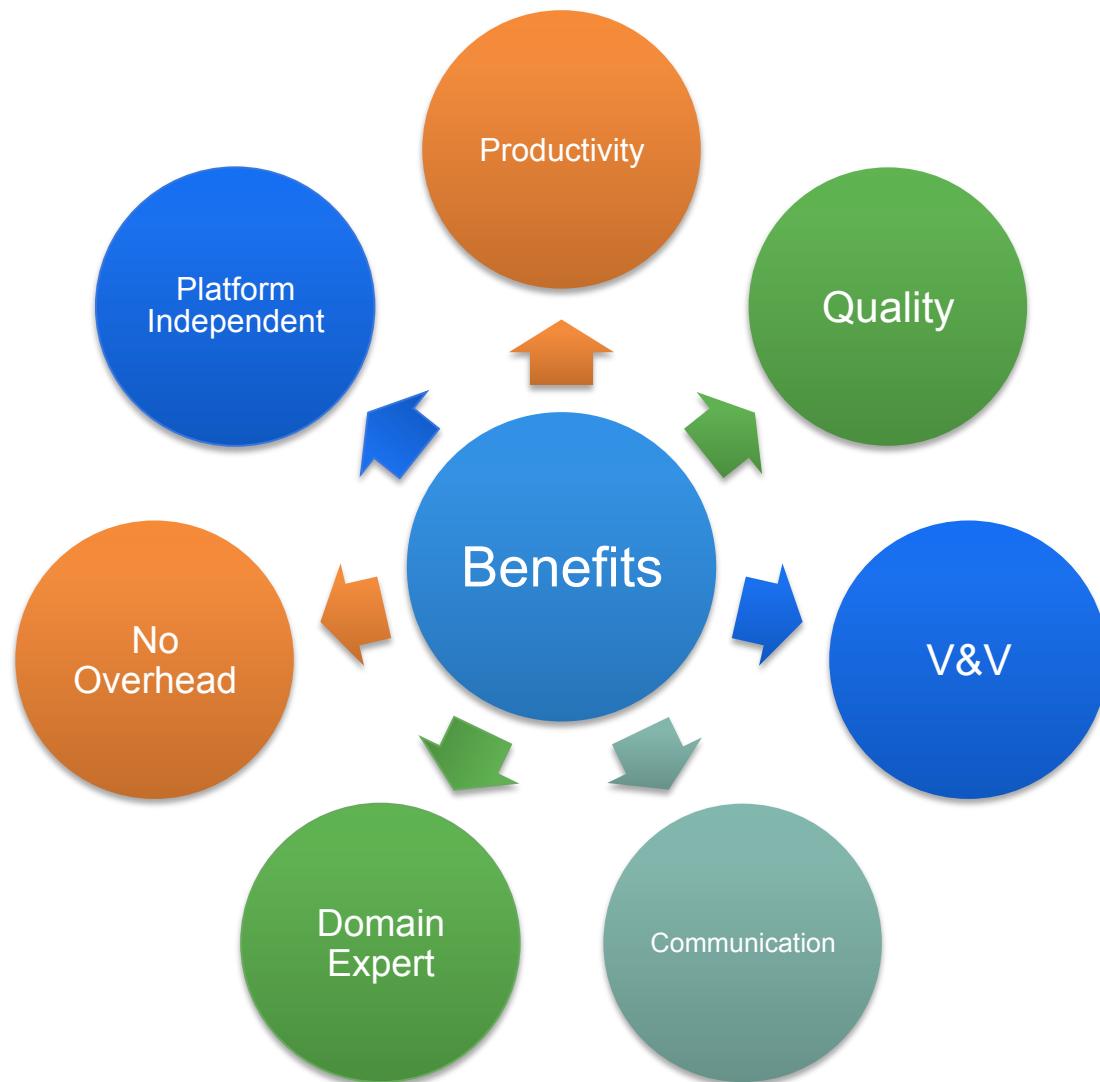
Higher abstractions

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

Specializing syntax and environment pays off?

- Promises of DSL « improvement » in terms of
 - usability, learnability, expressiveness, reusability, etc.
- Empirical study on the role of **syntax**
 - C-style syntax induces problems in terms of usability for novices; language more or less intuitive for (non-)programmers (Stefik et al. 2014)
 - Syntax issues with Java for students (Denny et al. 2011)
 - PL usability: method namings/placement, use of identifiers, API design (Ellis et al., Styllos et al., Clarke, Montperrus et al., etc.)
- More **specialized/sophicated tools/IDE** can be derived from a DSL
 - editors, compilers, debuggers

Question to the audience

- Take one DSL and formulate assumptions on their qualities (and superiority to a GPL-based solution)
- Imagine an experience for providing evidence that the DSL has such qualities

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

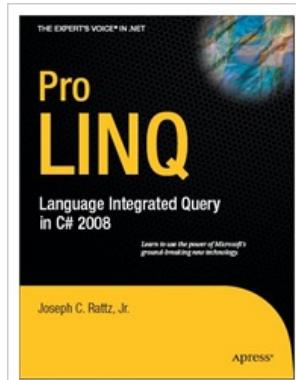
External vs Internal DSL (SQL example)

```
-- 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"));  
    }  
}
```

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)

Question to the audience

Find a DSL that is both
internal and external

HTML

- External DSL: <html>....
- Internal DSLs
 - LISP
 - Scala (XML support included in the language)

```
object XMLTest1 extends Application {  
    val page =  
        <html>  
            <head>  
                <title>Hello XHTML world</title>  
            </head>  
            <body>  
                <h1>Hello world</h1>  
                <p><a href="scala-lang.org">Scala</a> talks XHTML</p>  
            </body>  
        </html>;  
        println(page.toString())  
}
```

```

object XMLTest1 extends Application {
  val page =
<html>
  <head>
    <title>Hello XHTML world</title>
  </head>
  <body>
    <h1>Hello world</h1>
    <p><a href="scala-lang.org">Scala</a> talks XHTML</p>
  </body>
</html>;
  println(page.toString())
}

// Import the Glitter DSL
import glitter._

object Templates {
  // Define a reusable layout
  def layout(body: Xml) =
    html5dtd | 'html (
      'head :: 'title :: "Glitter is amazing!"
      | 'body :: body
    )

  // Define a template taking one String argument and using the Layout defined above
  def show(name: String) =
    layout (
      'h1 :: "Show user"
      | 'p :: ("Hello " | 'strong(name) | "!")
    )

  // Define a template taking a List of Strings, using the Layout defined above
  def index(users: List[String]) =
    layout (
      'h1 :: "User list"
      | 'ul % 'class~"user-list" :: (for (user <- users) yield ('li :: user))
    )
}

```

Scala

TCS Wyvern (Omar et al., OOPSLA'14) <https://github.com/julienrf/glitter>

```

1 let webpage : HTML = HTMLElement(Dict.empty(), [BodyElement(Dict.empty(),
2   [H1Element(Dict.empty(), [TextNode("Results for " + keyword)]),
3     ULElement((Dict.add Dict.empty() ("id","results")), to_list_items(query(db,
4       SelectStmt(["title", "snippet"], "products",
5         [WhereClause(InPredicate(StringLit(keyword), "title"))]))))))]
6
7 let webpage : HTML = <html><body><h1>Results for {keyword}</h1>
8   <ul id="results">{to_list_items(query(db,
9     SELECT title, snippet FROM products WHERE {keyword} in title))}
10  </ul></body></html>
11
12 let webpage : HTML = parse_html("<html><body><h1>Results for "+keyword+"</h1>
13   <ul id=\"results\">" + to_string(to_list_items(query(db, parse_sql(
14     "SELECT title, snippet FROM products WHERE '"+keyword+"' in title")))) +
15   "</ul></body></html>")

```

SQL

Plain SQL
(external DSL)

shape
#1

```
1 |-- SQL
2 SELECT * FROM journal
3   WHERE published_year = 2013
4     AND publisher = 'IEEE'
5 ORDER BY title
```

Java
(internal DSL)

shape
#2

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

Scala
(internal DSL)

shape
#3

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

Homework

JOOQ

```
-- 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();
```

JOOQ: some “open” questions

- How SQL queries are eventually executed?
- In addition to writing SQL queries, what is the effort needed? (e.g., database setup? configuration file? mapping object/database?)
- How the fluent API is implemented? (fluent API; design pattern?)
- What are the strange syntactical constructs of JOOQ (in comparison to pure SQL)?
- What are the typical complains of JOOQ users?
- Is it possible to translate any SQL query into JOOQ? Is full SQL supported? What subset of SQL is supported?
- Give executable examples of SQL queries and JOOQ programs

Homework

- Pick one question and answer!
- A collective work with individual contributions
 - Github
 - Wiki page: edit!
 - Close the issue (one issue per question)
- Preliminary deadline: end of this week
 - Issue assignment; comment that you have the intention to close the issue
 - The whole promotion should cover all issues
- Contribution deadline: 7th november

References

- Martin Fowler. Domain Specific Languages. Addison-Wesley Professional, 2010.
- Markus Voelter et al. “DSL Engineering: Designing, Implementing and Using Domain-Specific Languages.” dslbook.org, 2013.
- Kramer “Abstraction and Modelling - A Complementary Partnership” MODELS’08
- Tarr et al. “N Degrees of Separation: Multi-Dimensional Separation of Concerns” ICSE’99
- Benoit Combemale, Julien Deantoni, Benoit Baudry, Robert France, Jean-Marc Jézéquel, and Jeff Gray. « Globalizing Modeling Languages.” Computer, 2014.

References

- Leo A Meyerovich and Ariel S Rabkin. “Empirical analysis of programming language adoption” OOPSLA’13
- Felienne Hermans, Martin Pinzger, and Arie van Deursen. “Domain-Specific languages in practice: A user study on the success factors.” MODELS’09
- Paul Denny, Andrew Luxton-Reilly, Ewan Tempero, and Jacob Hendrickx. “Understanding the syntax barrier for novices.” ITiCSE ’11
- Tiark Rompf et al . “Optimizing Data Structures in High-Level Programs: New Directions for Extensible Compilers based on Staging” POPL’13

References

- Mathieu Acher, Benoît Combemale, Philippe Collet: “Metamorphic Domain-Specific Languages: A Journey into the Shapes of a Language.” Onward! 2014
- Jeffrey Stylos and Brad A. Myers. “The implications of method placement on api learnability” FSE’08
- Martin Monperrus, Michael Eichberg, Elif Tekes, and Mira Mezini. “What Should Developers Be Aware Of? An Empirical Study on the Directives of API Documentation”. *Empirical Software Engineering*, 17(6):703–737, 2012.

Plan

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

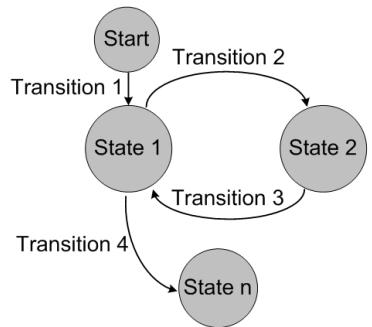
Contract

- Better understanding/source of inspiration of software languages and DSLs
 - Revisit of history and existing languages
- Foundations and practice of Xtext
 - State-of-the-art language workbench (Most Innovative Eclipse Project in 2010, mature and used in a variety of industries)
- Models and Languages
 - Perhaps a more concrete way to see models, metamodels and MDE (IDM in french)

BIBTEX



Graphviz



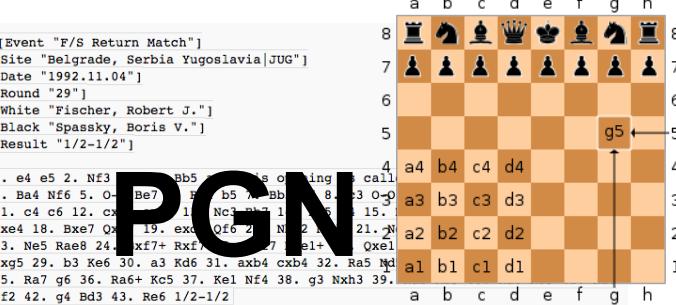
**Finite State
Machine**



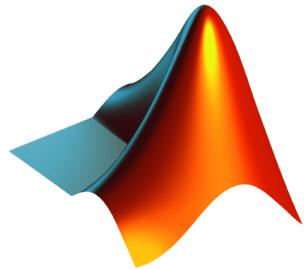
Domain-Specific Languages (DSLs)

```
[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 Bb5 3. c3 Nc6 4. Bb5 Nf6 5. O-O Be7 6. Bb7 Bb5 7. Nc3 Bb7 8. e5 Nf6 9. Bb7 Nc6 10. d4 Nf6 11. c4 c6 12. c5 Nf6 13. Nc3 Bb7 14. Nf3 Nc6 15. Nxe4 Nf6 16. Nc3 Nc6 17. Nf3 Nf6 18. Bxe7 Qe7 19. exd5 Qf6 20. Nf3 Nf6 21. Nxe4 Nf6 22. Nc3 Nf6 23. Ne5 Rae8 24. Bxf7+ Rxf7 25. Nf3 Nf6 26. Nxe4 Nf6 27. Nf3 Nf6 28. Nxe4 Nf6 29. b3 Ke6 30. a3 Kd6 31. axb4 cxb4 32. Ra5 Nd5 33. Nc3 Nf6 34. Nf3 Nf6 35. Ra7 g6 36. Ra6+ Kc5 37. Ke1 Nf4 38. g3 Nxh3 39. Nf2 g4 40. Ra6+ Kc6 41. Ra6 Nf6 42. g4 Bd3 43. Re6 1/2-1/2
```



Make



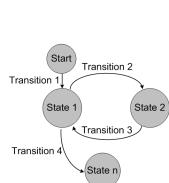
Matlab



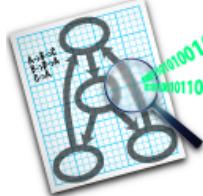
DSL = Syntax + Services

Specialized notation:

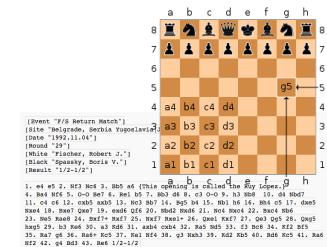
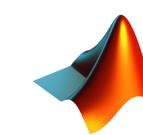
Textual or Graphical
Specific Vocabulary
Idiomatic constructs



BIBT_EX



SQL



Specialized tools/IDE:

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

Language workbenches

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

Language Workbenches

Erdweg et al. SLE'13

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

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

Java – strategoxt-sugar-papers/test/BookHandler.sugj – Eclipse – /Users/seba/tmp/ecli...

Transform ▾

BookSchema.sugj *BookHandler.sugj

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

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

Outli ▾

BookHandler appendBook book author editions isPublished getLanguage

Problems ▾

1 error, 1 warning

Description

- Errors (1 item)
 - expected element edition of namespace lib
- Warnings (1 item)
 - skipping validation of quoted attribute value

Resource Location

Resource	Location
BookHandler.sugj	line 18
BookHandler.sugj	line 14

Writable Smart Insert

The screenshot shows the Eclipse IDE interface with the Java perspective open. The central part of the window displays an XML schema file named BookHandler.sugj. The code includes imports for Sugar, Editor, and BookSchema, and defines a public class BookHandler with a method appendBook that takes a ContentHandler parameter. Inside the method, there is an annotation @Validate and several XML elements: a book element with an attribute title, an author element, and an editions element containing two edition elements. The edition elements have attributes year and publisher. The XML code uses a namespace prefix {lib} for the Sugar library.

The Problems view at the bottom left indicates there is 1 error and 1 warning. The Description view provides details about these issues:

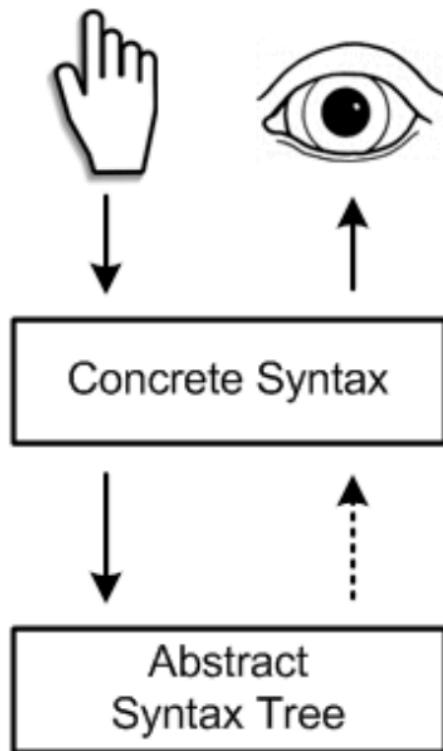
- Errors (1 item):** expected element edition of namespace lib (located at line 18)
- Warnings (1 item):** skipping validation of quoted attribute value (located at line 14)

The Outline view on the right shows the structure of the schema, including the BookHandler class, its methods appendBook, book, and the nested elements author, editions, edition, and edition.

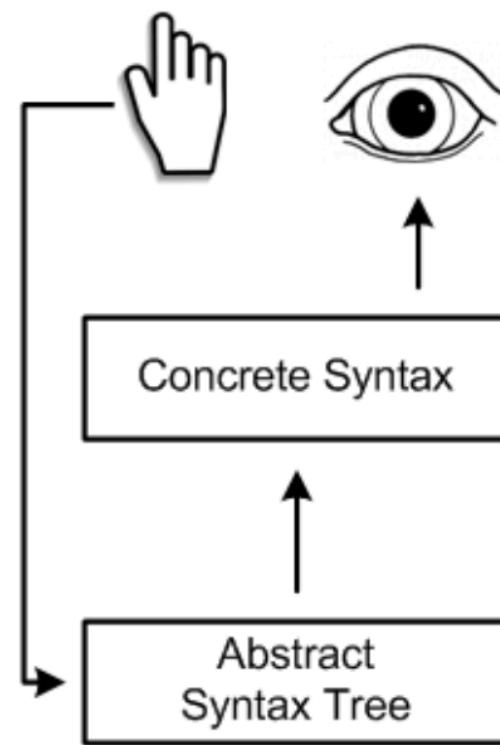
Sebastian Erdweg, Tillmann Rendel, Christian Kästner, and Klaus Ostermann. Sugarj: Library-based syntactic language extensibility. OOPSLA'11

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  where ...
21
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.

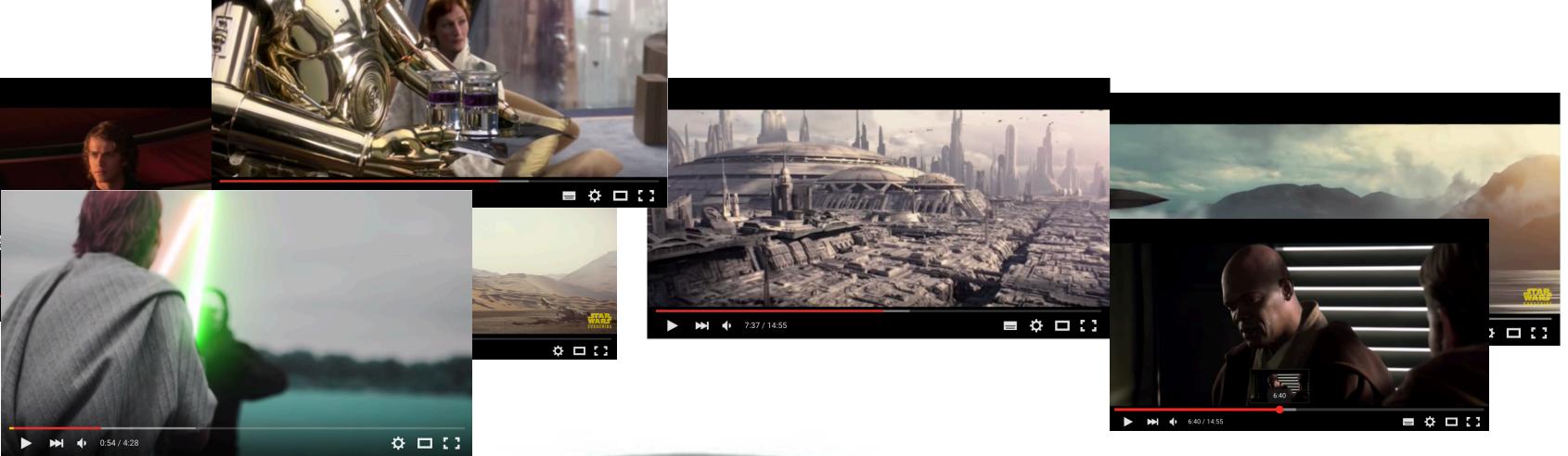
With the Spoofax language workbench, you can write the grammar of your language using the high-level SDF grammar formalism. Based on this grammar, basic editor services such as syntax highlighting and code folding are automatically provided. Using high-level descriptor languages, these services can be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

xtext





Generator
~ composition of
video sequences

**video
variants**





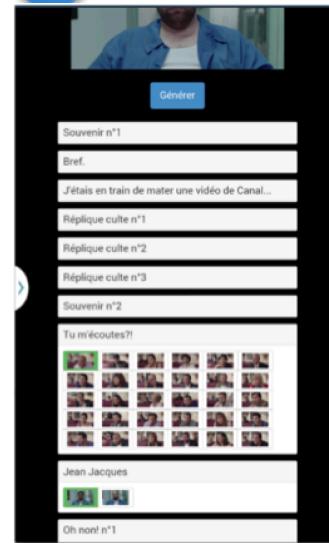
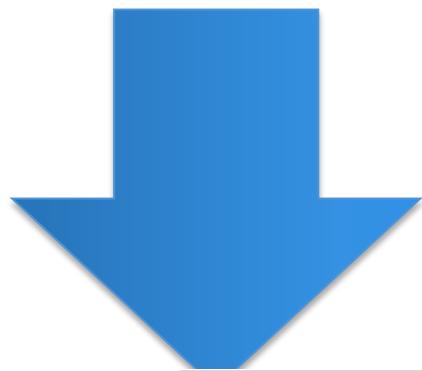
```

foo1.videogen ✎

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

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

```



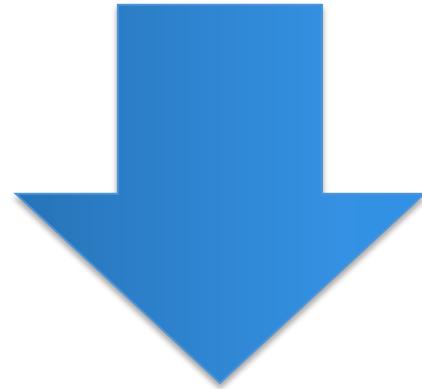
- ## Website/online
- Random generation
 - Configurator
 - Game
 - ...



```
foo1.videogen ✘

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

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



 FFmpeg

foo1.videoogen

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

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

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

#2 How to transform models/programs?



#3 How to manage variability/variants?

#4 How do frameworks internally work?

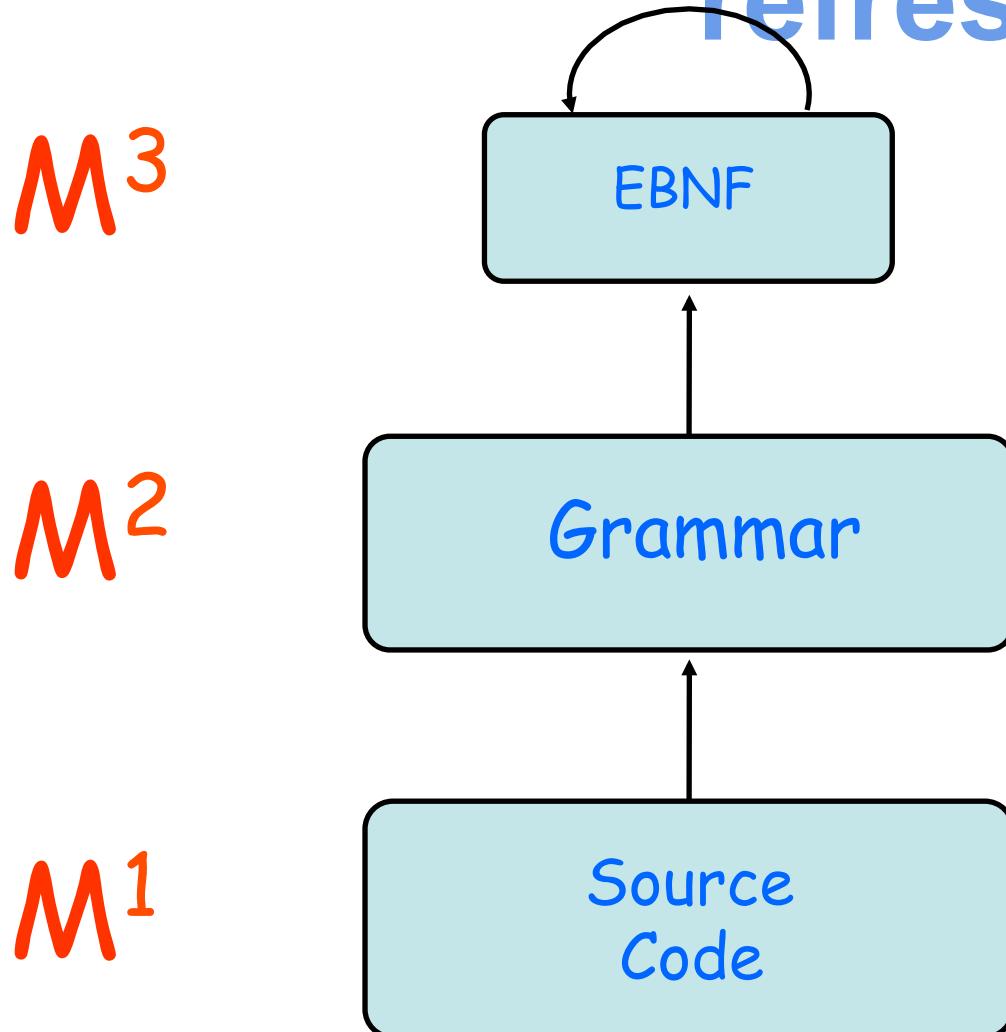
Xtext, a popular, easy-to-use model-based tool
for developping DSLs

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

Your DSL in 5'

Short Demonstration

Foundations (or some course refresh)



Java Grammar

```

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

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

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

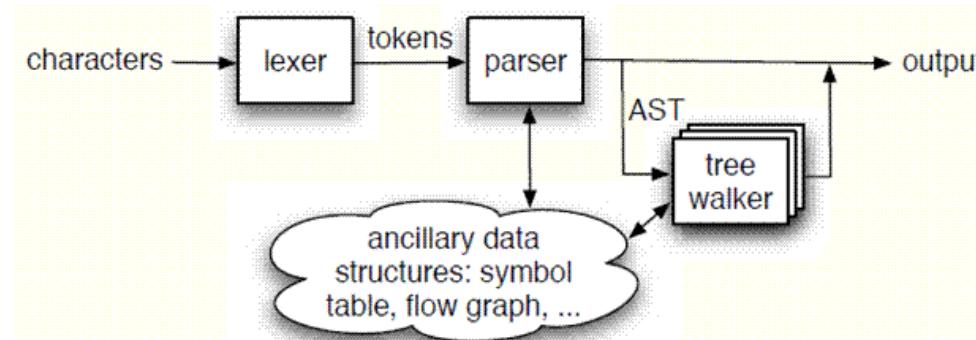
```

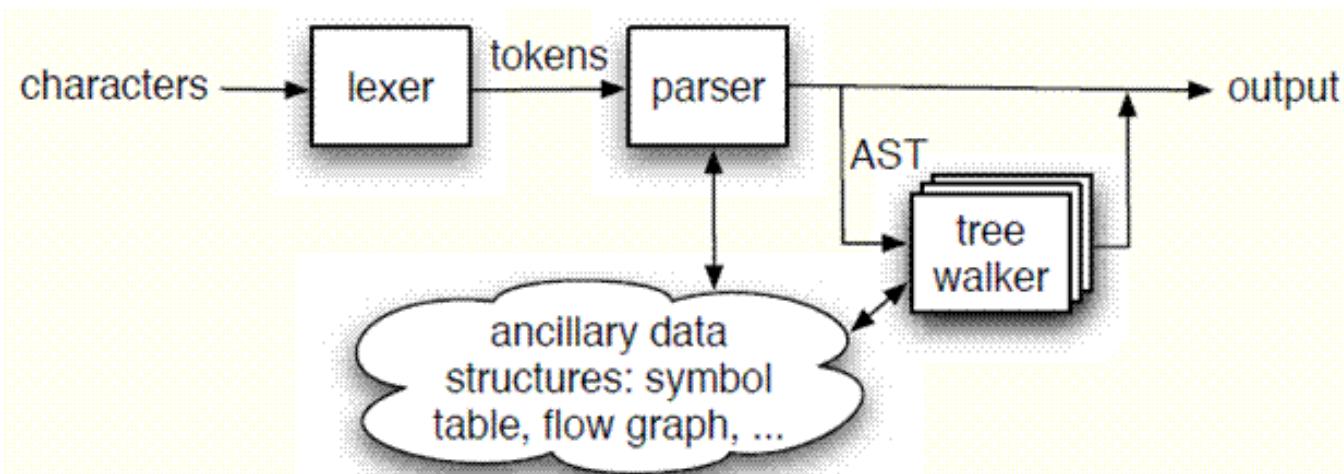
Java Program

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

Compilation Process

- Source code
 - Concrete syntax used for specifying a program
 - Conformant to a grammar
- Lexical analysis
 - 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'
|   '\"'
)
;
```

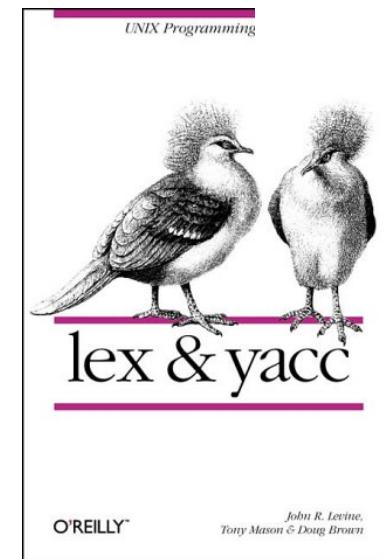
```

classOrInterfaceDeclaration
:   classDeclaration
|   interfaceDeclaration
;

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

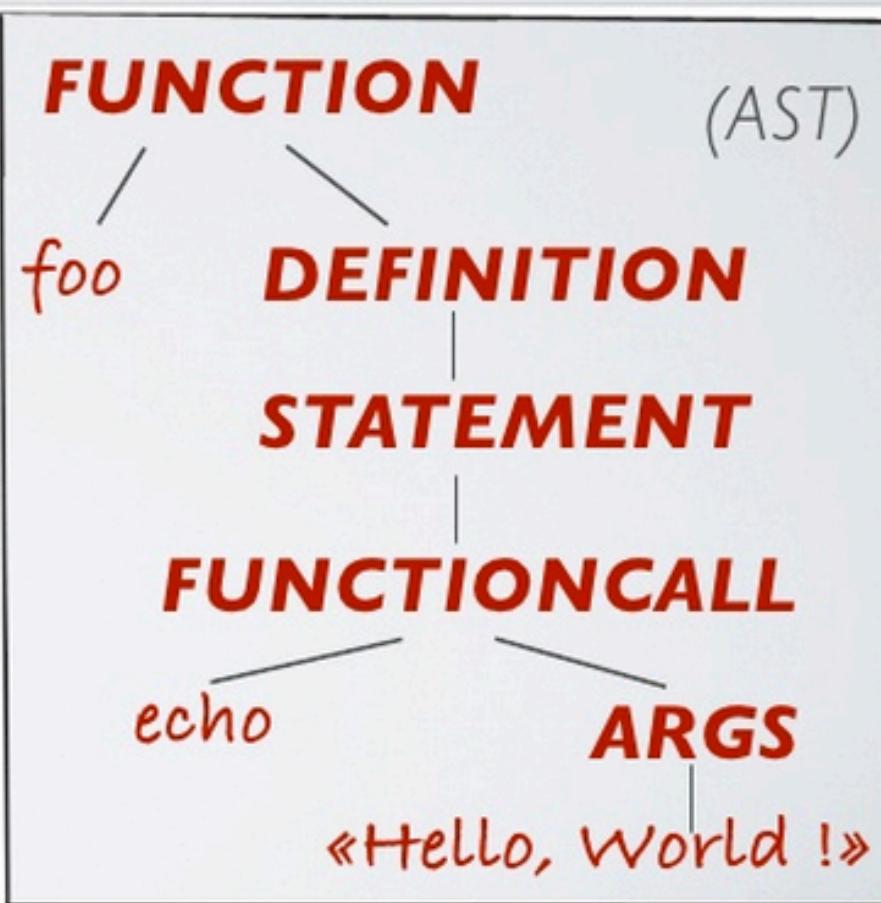
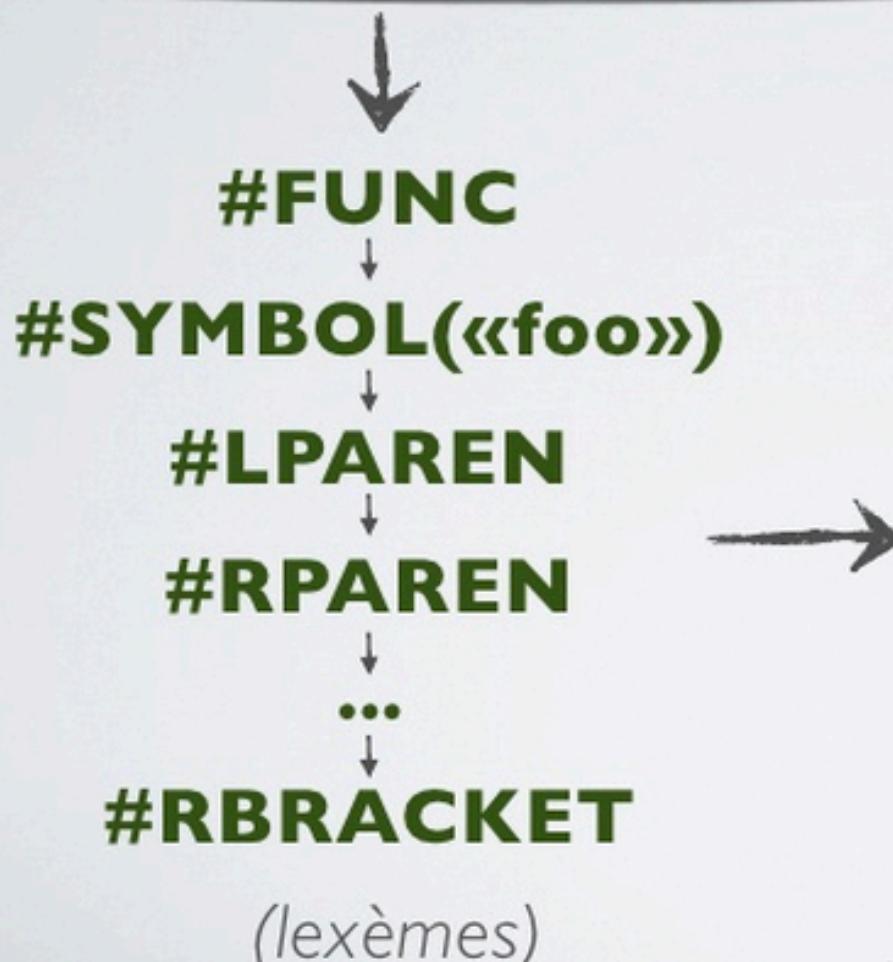
variableModifiers
:   (
    FINAL
|   annotation
)*
;

classDeclaration
:   normalClassDeclaration
|   enumDeclaration
;
```



EXEMPLE

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



```

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

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

```

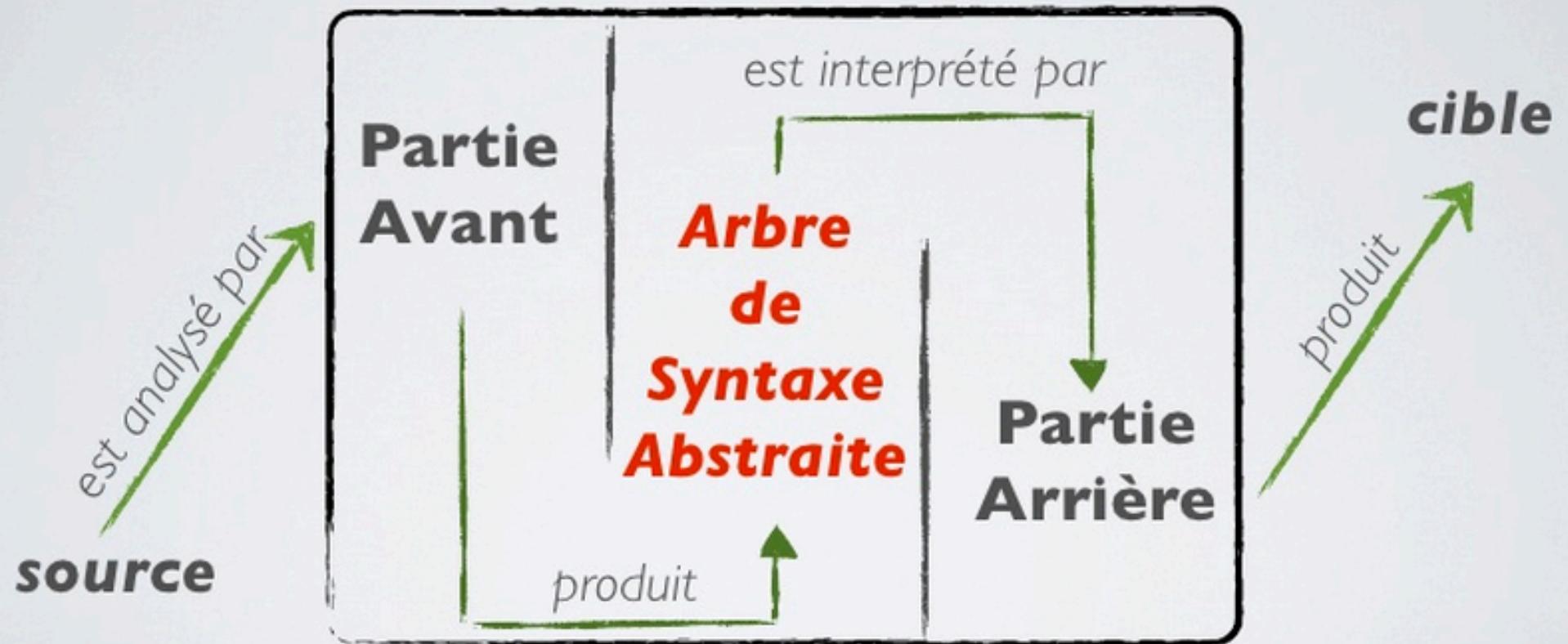
Scala AST (example)

```

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

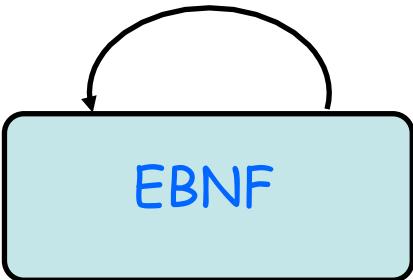
```

Compilation (en français)

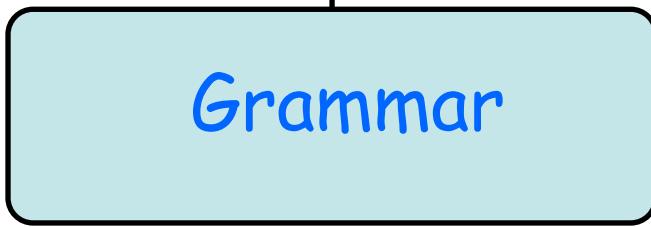


DSL? The same!

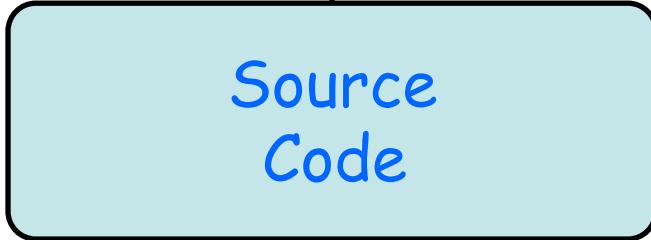
M³



M²



M¹



DSL Grammar

DSL specification/
program

UNIX Programming Tools



O'REILLY™

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

The
Pragmatic
Programmers

The Definitive
ANTLR
Reference

Building Domain-
Specific Languages

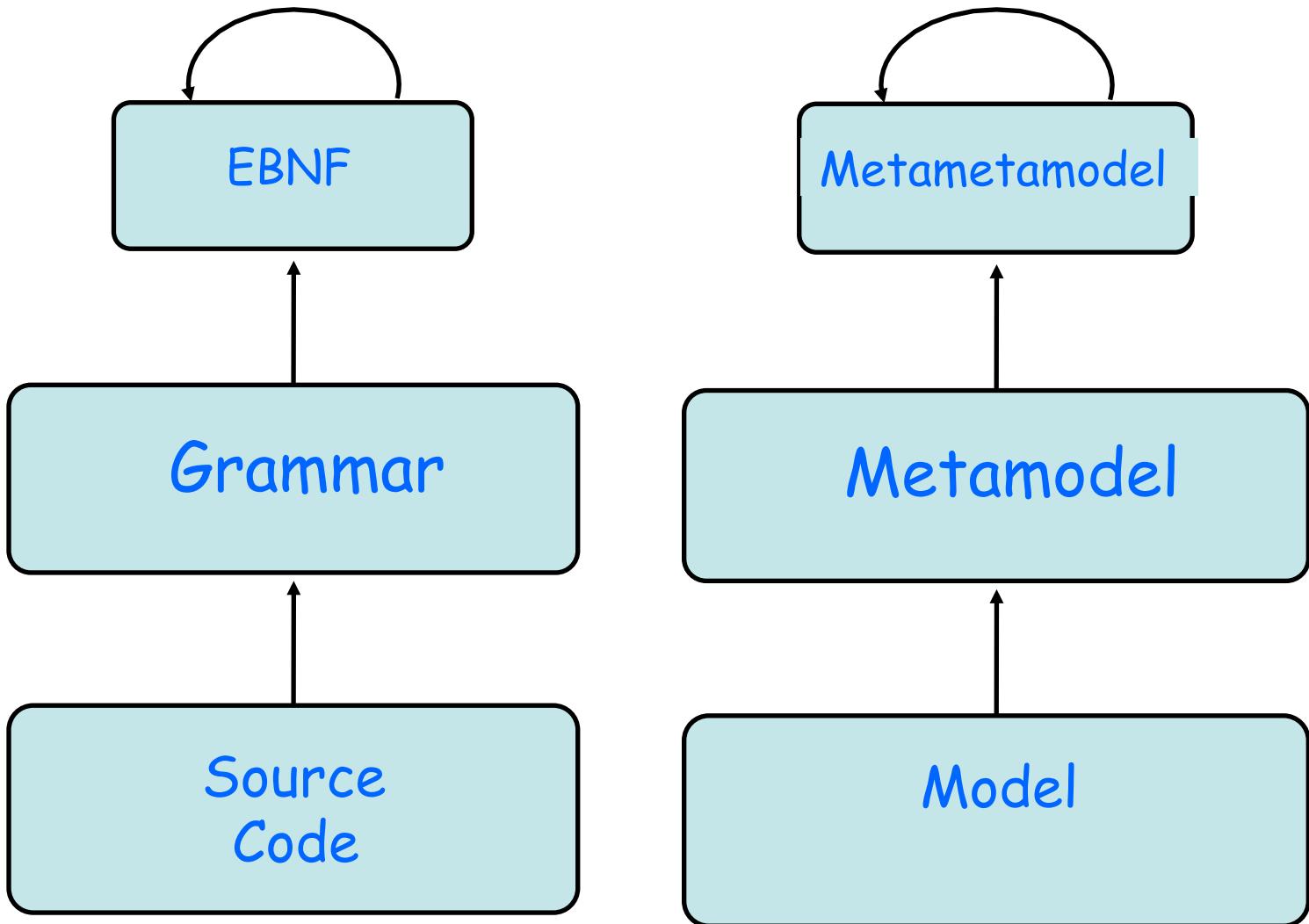


Terence Parr

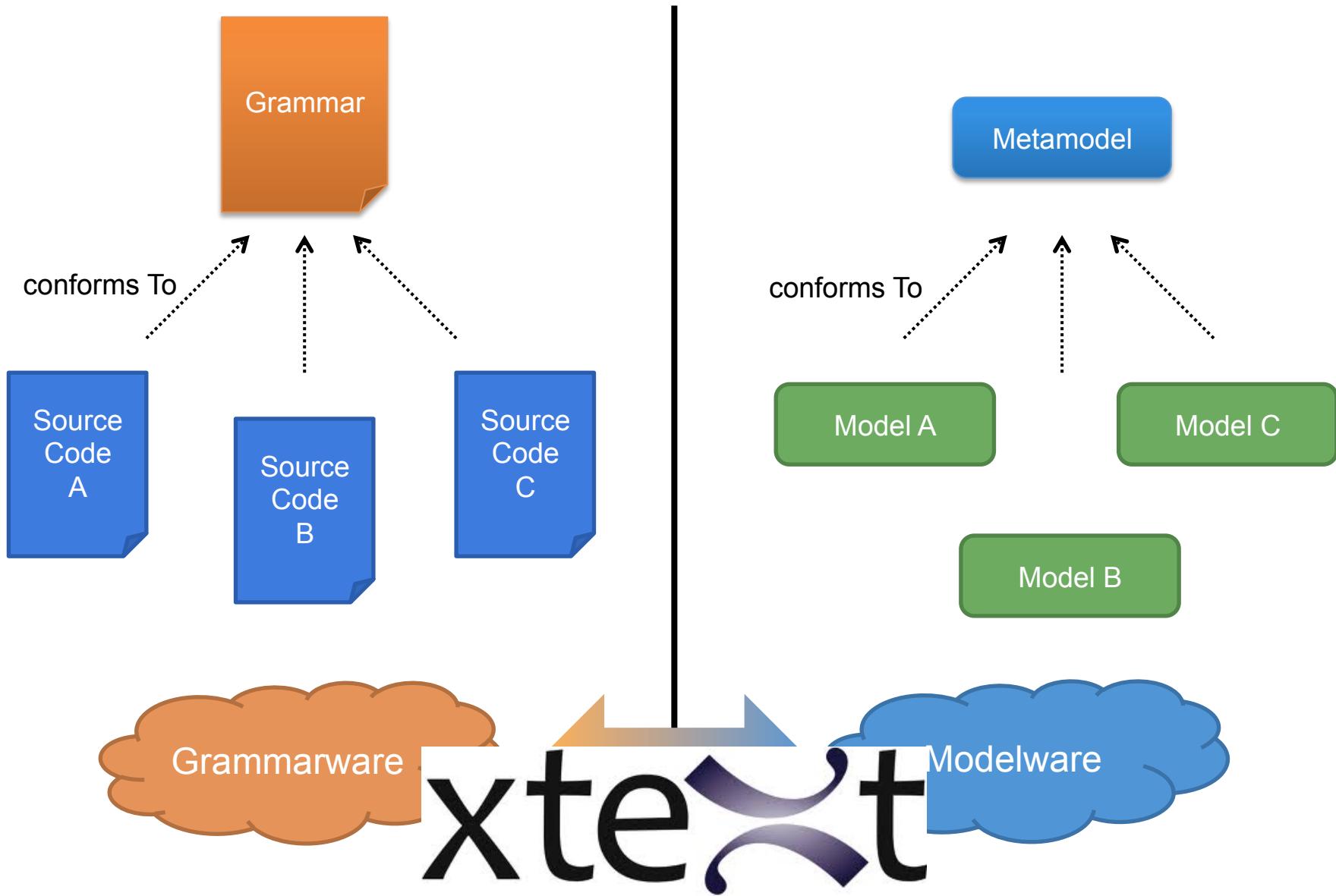
M³

M²

M¹



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)

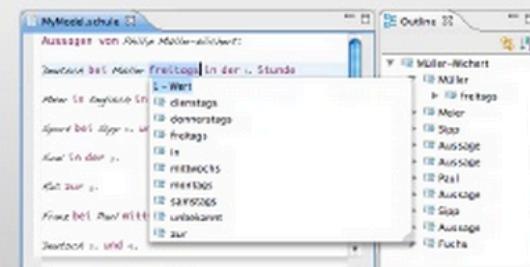
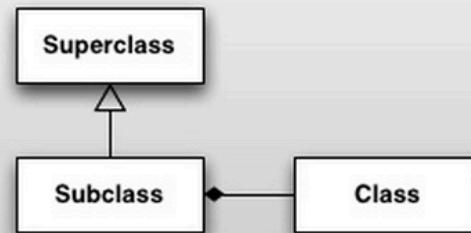
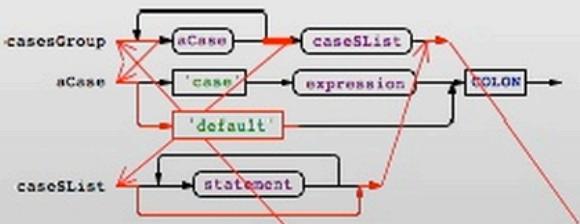
Model

Grammar

Xtext
Generator



Xtext Runtime

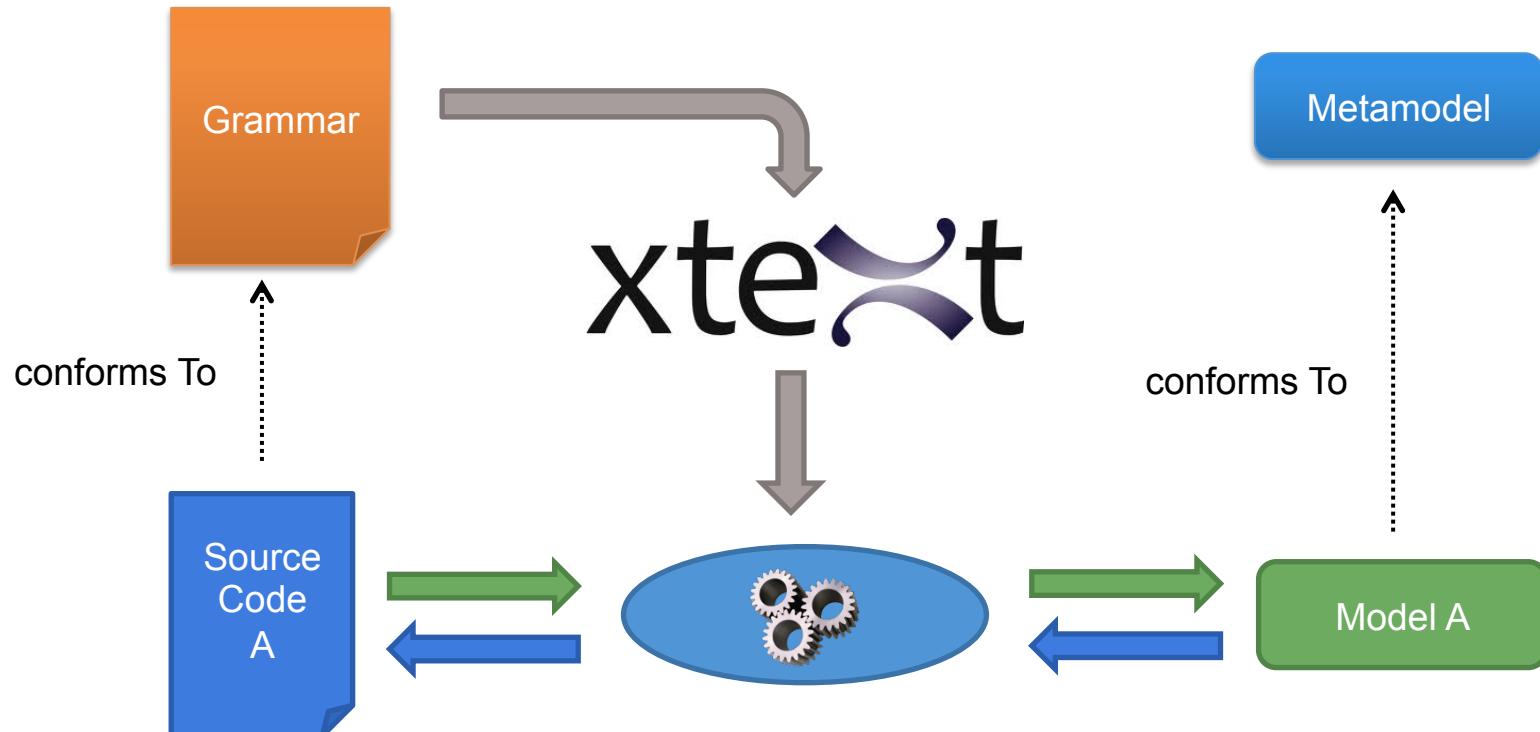


LL(*) Parser

ecore meta model

editor

Xtext, Grammar, Metamodel

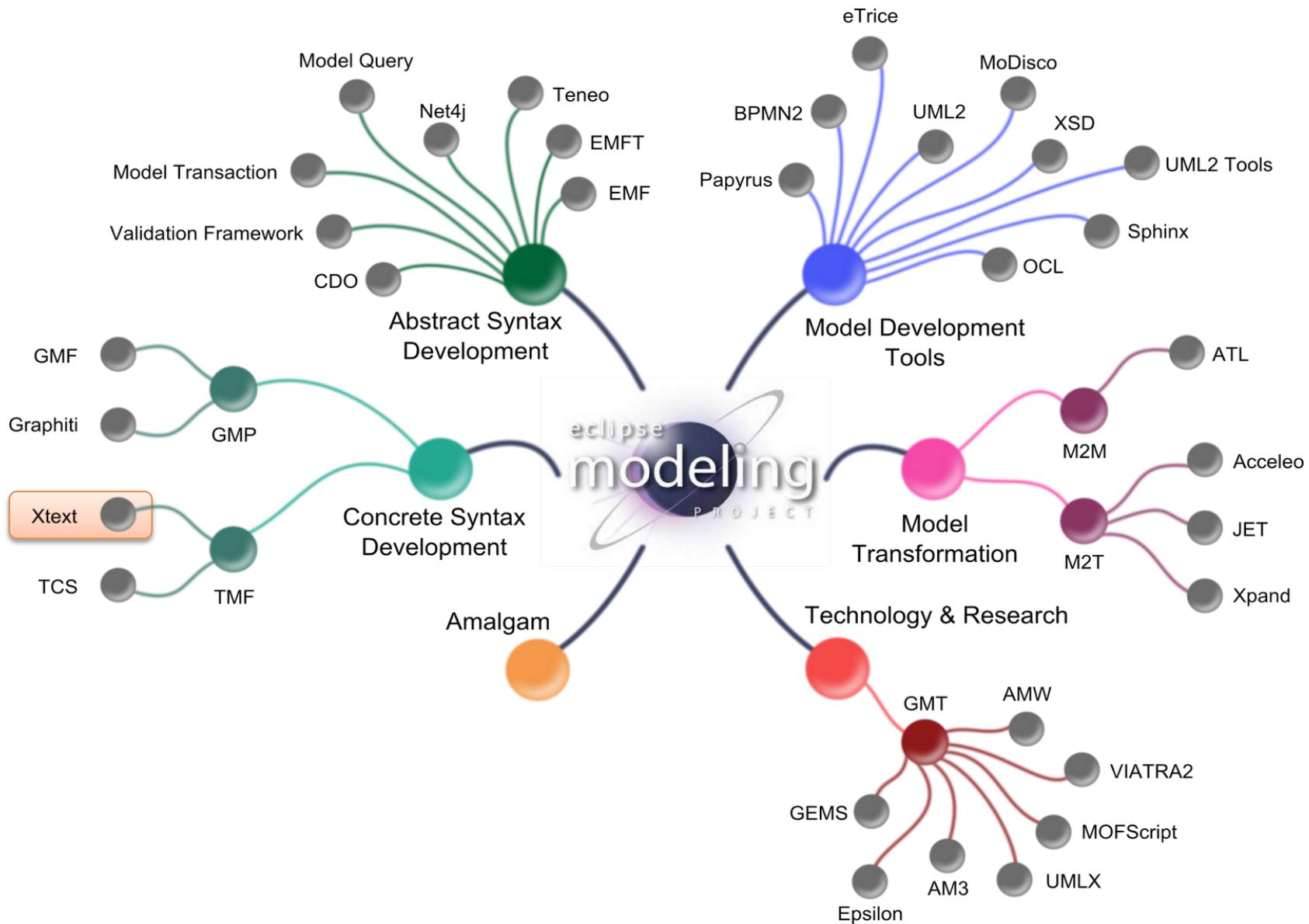


Xtext Project

- 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



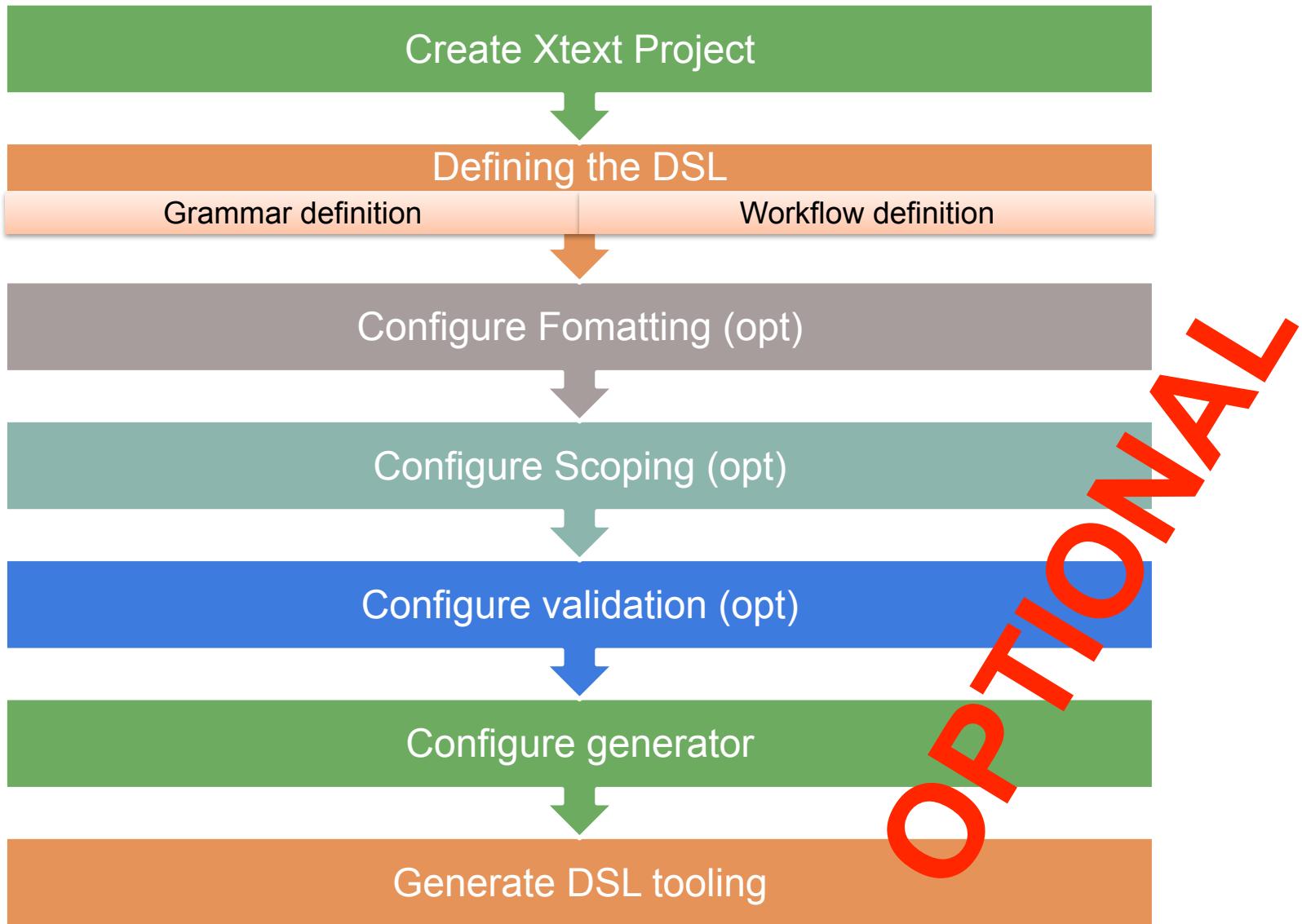
The Grammar Language of Xtext

- 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

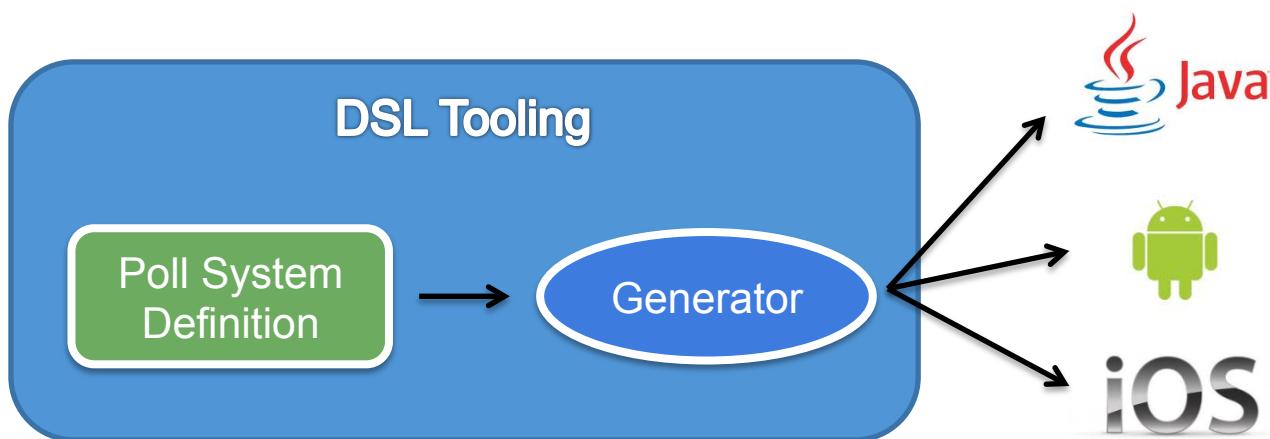
- Consistent look and feel
- Textual DSLs are a resource in Eclipse
- Open editors can be extended
- Complete framework to develop DSLs
- Easy to connect to any Java-based language

Development Process



A first example

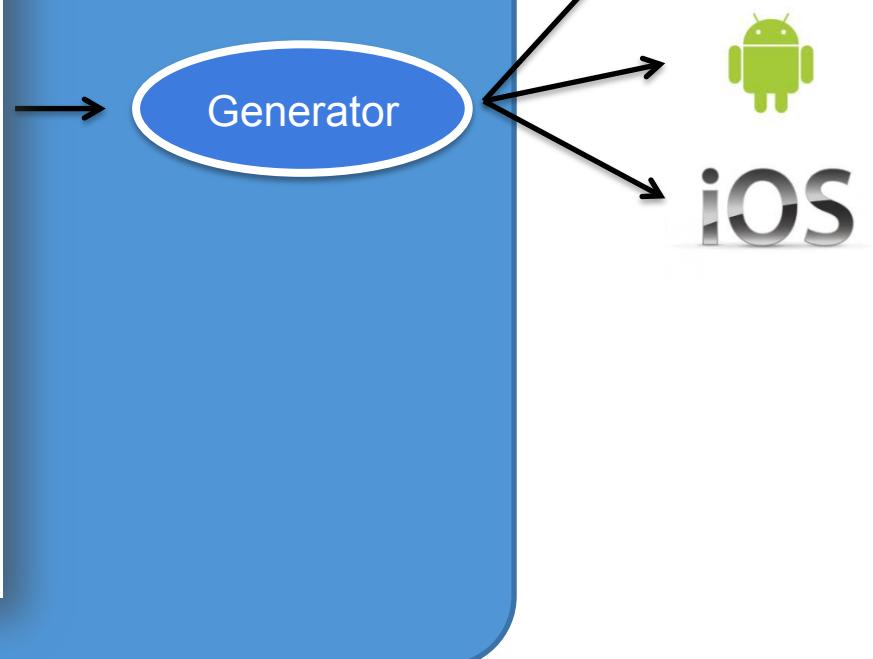
- Poll System application
 - Define a Poll with the corresponding questions
 - Each question has a text and a set of options
 - Each option has a text
- Generate the application in different platforms



Something like...

DSL Tooling

```
PollSystem {  
    Poll Quality {  
        Question q1 {  
            "Value the user experience"  
            options {  
                A : "Bad"  
                B : "Fair"  
                C : "Good"  
            }  
        }  
        Question q2 {  
            "Value the layout"  
            options {  
                A : "It was not easy to locate elements"  
                B : "I didn't realize"  
                C : "It was easy to locate elements"  
            }  
        }  
    }  
    Poll Performance {  
        Question q1 {  
            "Value the time response"  
            options {  
                A : "Bad"  
                B : "Fair"  
                C : "Good"  
            }  
        }  
    }  
}
```



Xtext Grammar

Grammar
definition →

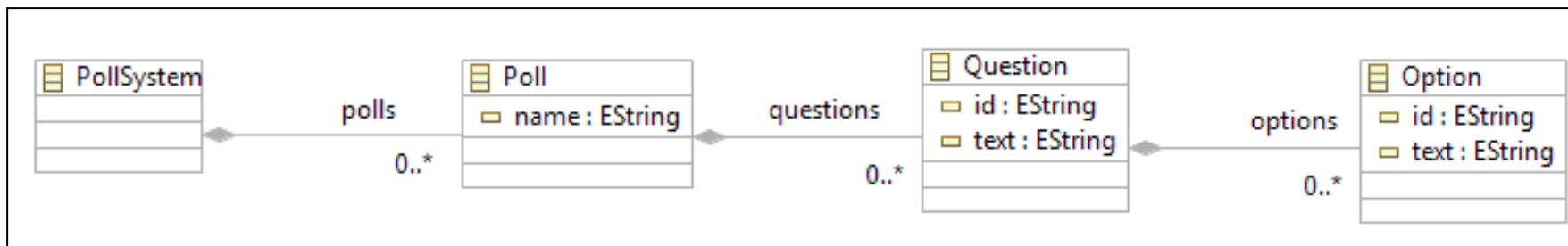
```
grammar fr.miage.xtext.Poll with org.eclipse.xtext.common.Terminals

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

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

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

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



Xtext Grammar

Grammar
reuse

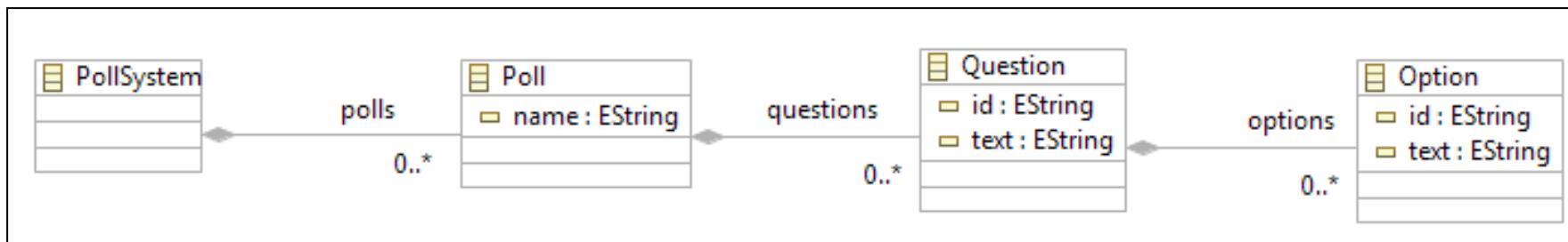
```
grammar fr.miage.xtext.Poll with org.eclipse.xtext.common.Terminals

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

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

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

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



Xtext Grammar

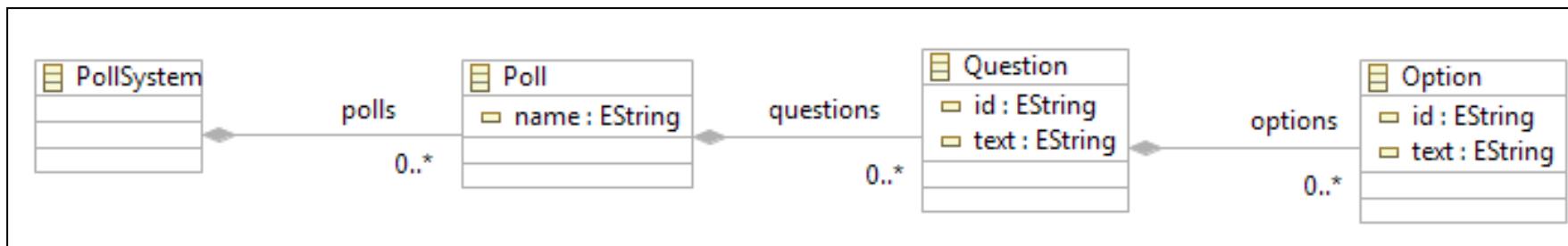
```
grammar fr.miage.xtext.Poll with org.eclipse.xtext.common.Terminals

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

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

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

Derived
metamodel



Xtext Grammar

Parser Rules

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

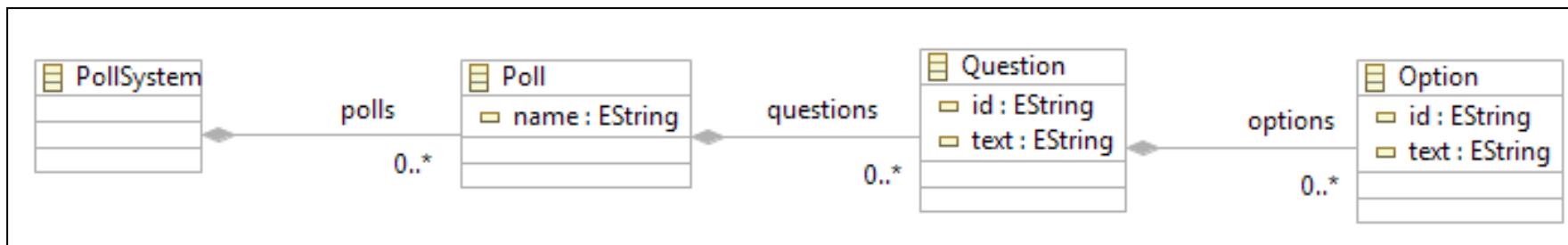
generate poll "http://www.miage.fr/xtext/Poll"

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

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

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

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



Xtext Grammar

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

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

PollSystem:

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

```

Poll:

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

```

Question:

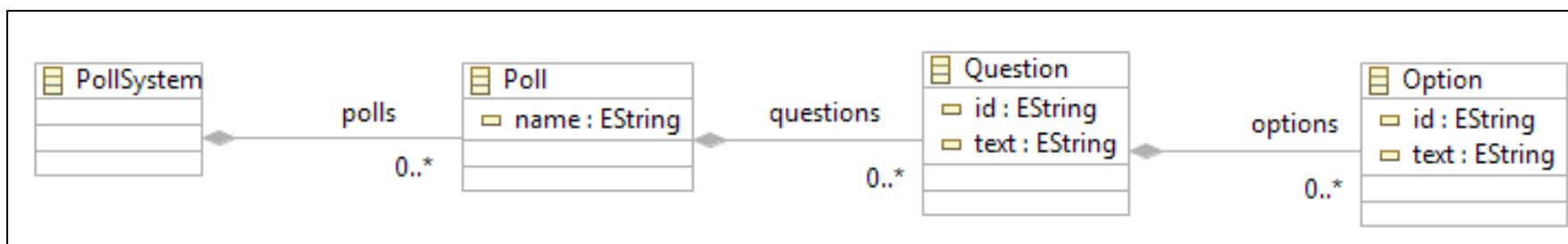
```
→ 'Question' id=ID '{' text=STRING 'options' '{' options+=Option+ '}'} '}';

```

Option:

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

Keywords



Xtext Grammar

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

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

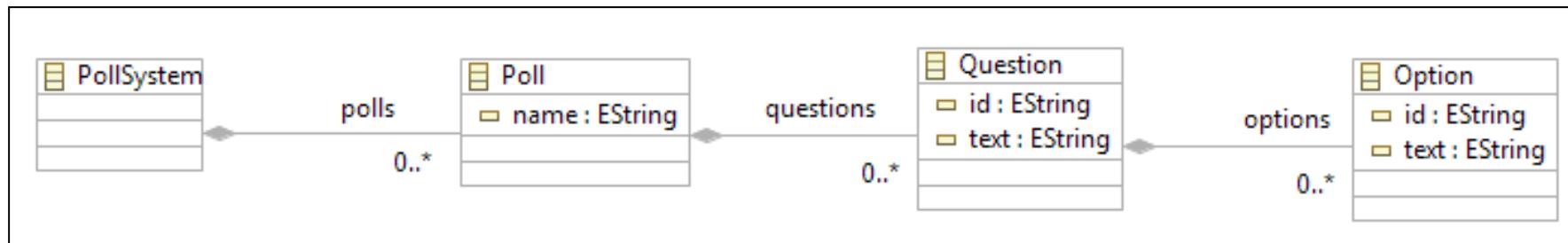
PollSystem:
    'PollSystem' '{' polls+=Poll+ '}';
    ^ Multivalue assignment

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

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

Option:
    id=ID ':' text=STRING;
    ^ Simple assignment
```

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



Xtext Grammar

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

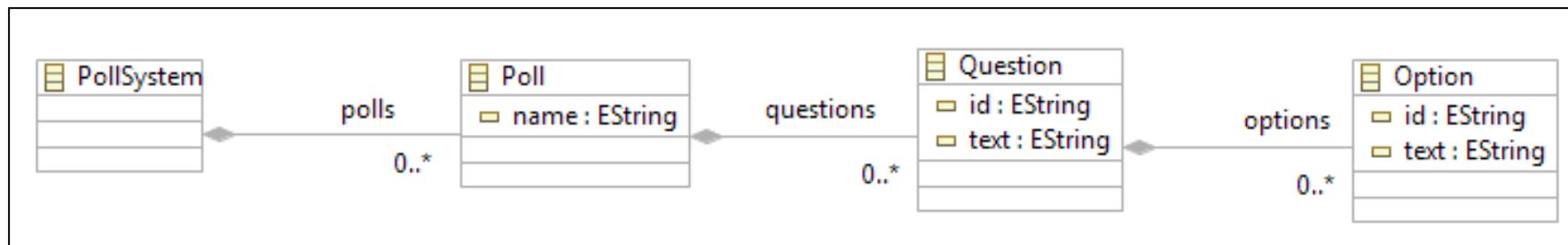
generate poll "http://www.miage.fr/xtext/Poll"

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

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

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

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



Xtext Grammar

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

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

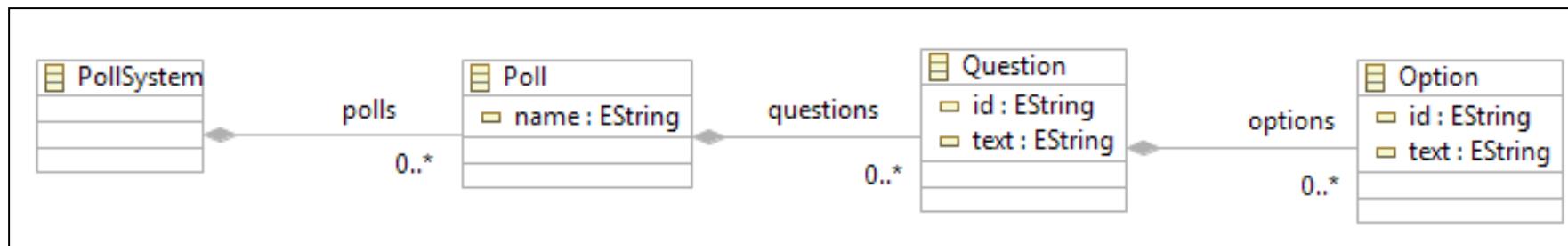
```
PollSystem:  
    'PollSystem' '{' polls+=Poll+ '}' ;
```

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

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

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

Containment



Grammar and Programs/Specifications/Models

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

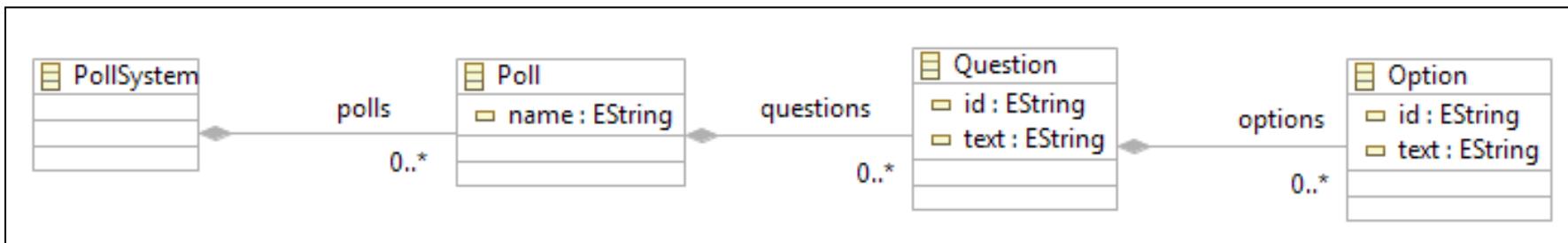
generate poll "http://www.miage.fr/xtext/Poll"

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

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

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

```
PollSystem {
    Poll Quality {
        Question q1 {
            "Value the user experience"
            options {
                A : "Bad"
                B : "Fair"
                C : "Good"
            }
        }
        Question q2 {
            "Value the layout"
            options {
                A : "It was not easy to locate elements"
                B : "I didn't realize"
                C : "It was easy to locate elements"
            }
        }
    }
    Poll Performance {
        Question q1 {
            "Value the time response"
            options {
                A : "Bad"
                B : "Fair"
                C : "Good"
            }
        }
    }
}
```



Grammar and Programs/Specifications/Models

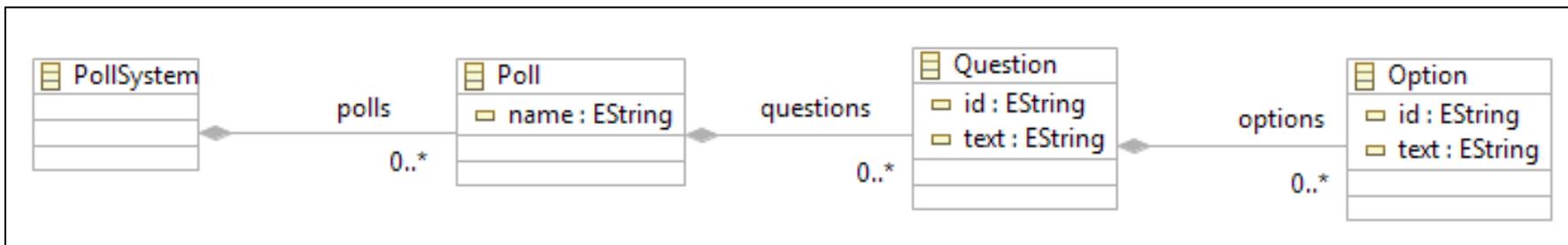
```
grammar fr.miage.xtext.Poll with org.eclipse.xtext.common.Terminals

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

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

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

```
PollSystem {
    Poll Quality {
        Question q1 {
            "Value the user experience"
            options {
                A : "Bad"
                B : "Fair"
                C : "Good"
            }
        }
        Question q2 {
            "Value the layout"
            options {
                A : "It was not easy to locate elements"
                B : "I didn't realize"
                C : "It was easy to locate elements"
            }
        }
    }
    Poll Performance {
        Question q1 {
            "Value the time response"
            options {
                A : "Bad"
                B : "Fair"
                C : "Good"
            }
        }
    }
}
```



Grammar and Programs/Specifications/Models

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

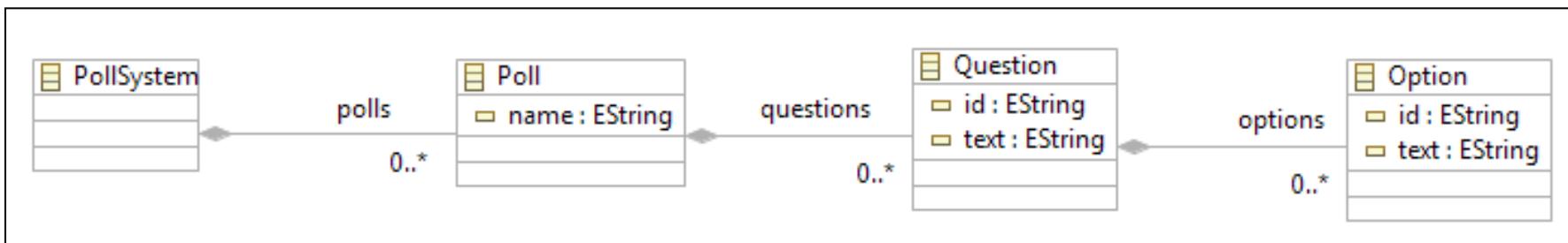
generate poll "http://www.miage.fr/xtext/Poll"

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

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

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

```
PollSystem {
    Poll Quality {
        Question q1 {
            "Value the user experience"
            options {
                A : "Bad"
                B : "Fair"
                C : "Good"
            }
        }
        Question q2 {
            "Value the layout"
            options {
                A : "It was not easy to locate elements"
                B : "I didn't realize"
                C : "It was easy to locate elements"
            }
        }
    }
    Poll Performance {
        Question q1 {
            "Value the time response"
            options {
                A : "Bad"
                B : "Fair"
                C : "Good"
            }
        }
    }
}
```



Grammar and Programs/Specifications/Models

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

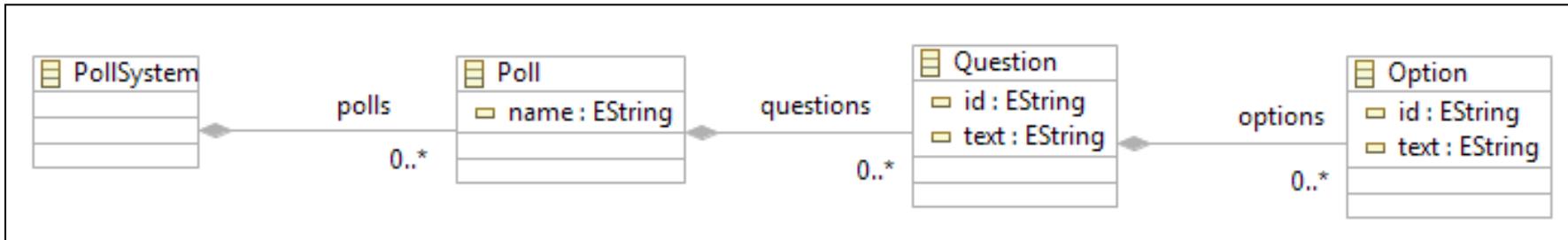
generate poll "http://www.miage.fr/xtext/Poll"

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

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

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

```
PollSystem {
    Poll Quality {
        Question q1 {
            "Value the user experience"
            options {
                A : "Bad"
                B : "Fair"
                C : "Good"
            }
        }
        Question q2 {
            "Value the layout"
            options {
                A : "It was not easy to locate elements"
                B : "I didn't realize"
                C : "It was easy to locate elements"
            }
        }
    }
    Poll Performance {
        Question q1 {
            "Value the time response"
            options {
                A : "Bad"
                B : "Fair"
                C : "Good"
            }
        }
    }
}
```



Quizz Time

Quetionnaire.xtext

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

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

Quizz Time

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

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

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

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

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

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

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

```
PollSystem [
}]
```

Quizz Time

Quetionnaire.xtext

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

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

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

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

Live Demonstration

The screenshot shows the Eclipse IDE interface with 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/dsl/Questionnaire"
4
```

New

Open

Open With

Show In

Copy

Copy Qualified Name

Paste

Delete

Build Path

Refactor

Import...

Export...

Refresh

Assign Working Sets...

Validate

Run As

Debug As

Replace With

Team

Compare With

Properties

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

name=ID '{' questions+=Questi

Question' id=ID '{' text=\$1

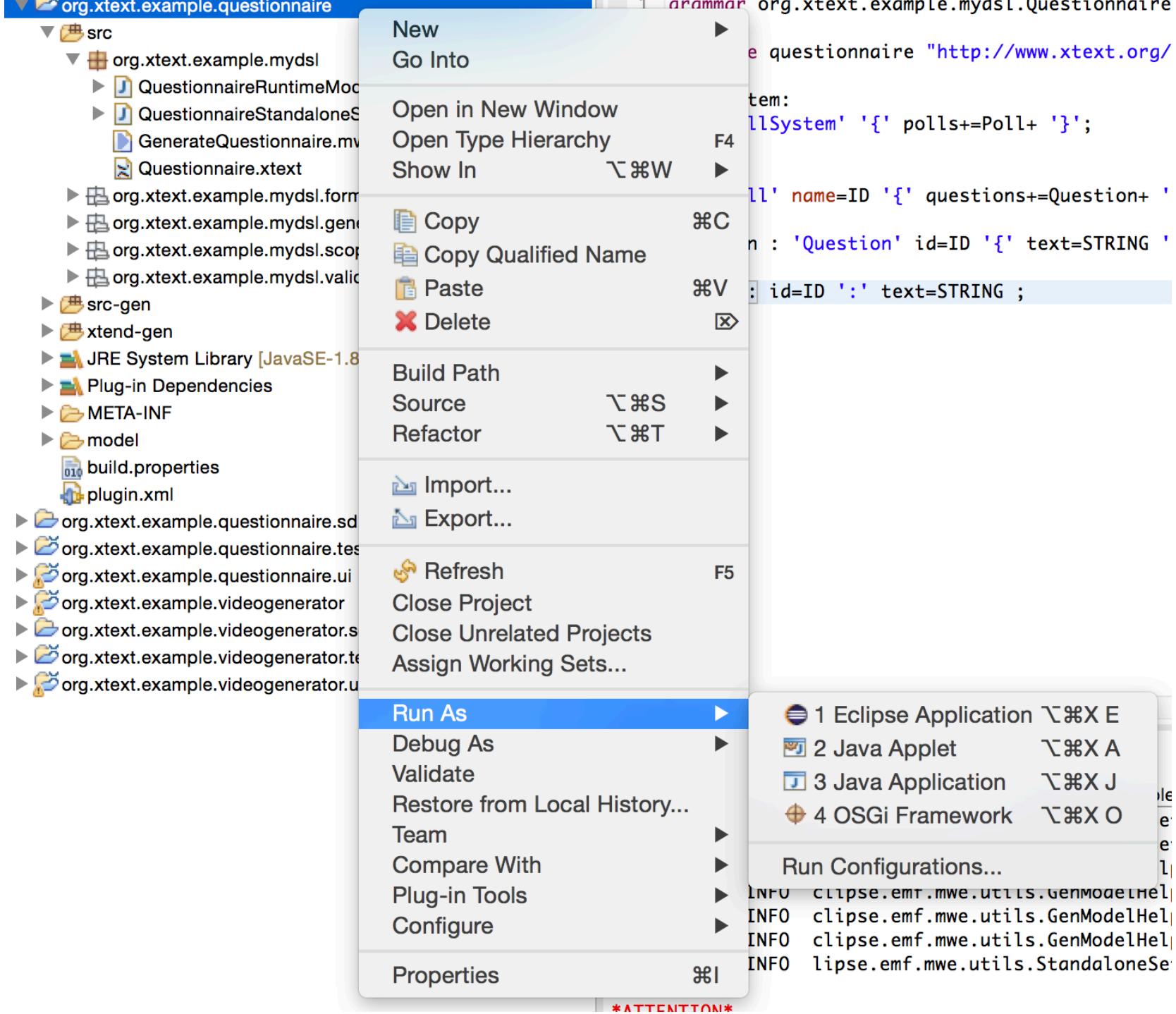
=ID ':' text=STRING ;

1 MWE2 Workflow

Run Configurations...

The screenshot shows the Eclipse IDE interface with the 'Console' tab selected. The log output is as follows:

```
<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/  
413 [main] INFO clipse.emf.mwe.utils.GenModelHelper - Registered GenModel 'http://www.eclipse.org/xtext/xbase/Xtype' from 'platform:/resource/  
436 [main] INFO clipse.emf.mwe.utils.GenModelHelper - Registered GenModel 'http://www.eclipse.org/xtext/xbase/Xbase' from 'platform:/resource/  
436 [main] INFO clipse.emf.mwe.utils.GenModelHelper - Registered GenModel 'http://www.eclipse.org/xtext/common/JavaVMTypes' from 'platform:/resource/  
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)  
11812 [main] INFO erator.parser.antlr.AntlrToolFacade - downloading file from 'http://download.itemis.com/antlr-generator-3.2.0-patch.jar'  
108842 [main] INFO erator.parser.antlr.AntlrToolFacade - finished downloading.  
108848 [main] INFO ipse.emf.mwe.utils.DirectoryCleaner - Cleaning /Users/macher1/Documents/workspaceIDM1516/org.xtext.example.questionnaire/  
108849 [main] INFO ipse.emf.mwe.utils.DirectoryCleaner - Cleaning /Users/macher1/Documents/workspaceIDM1516/org.xtext.example.questionnaire/  
108849 [main] INFO ipse.emf.mwe.utils.DirectoryCleaner - Cleaning /Users/macher1/Documents/workspaceIDM1516/org.xtext.example.questionnaire/  
110353 [main] INFO clipse.emf.mwe.utils.GenModelHelper - Registered GenModel 'http://www.xtext.org/example/mydsl/Questionnaire' from 'platfo  
113410 [main] INFO text.generator.junit.Junit4Fragment - generating Junit4 Test support classes  
113428 [main] INFO text.generator.junit.Junit4Fragment - generating Compare Framework infrastructure  
113584 [main] INFO .emf.mwe2.runtime.workflow.Workflow - Done.
```



**File**

Create a new file resource.



Enter or select the parent folder:

FooQuestionnaire



FooQuestionnaire

VideoGen1

File name:

Advanced >>



Cancel

Finish

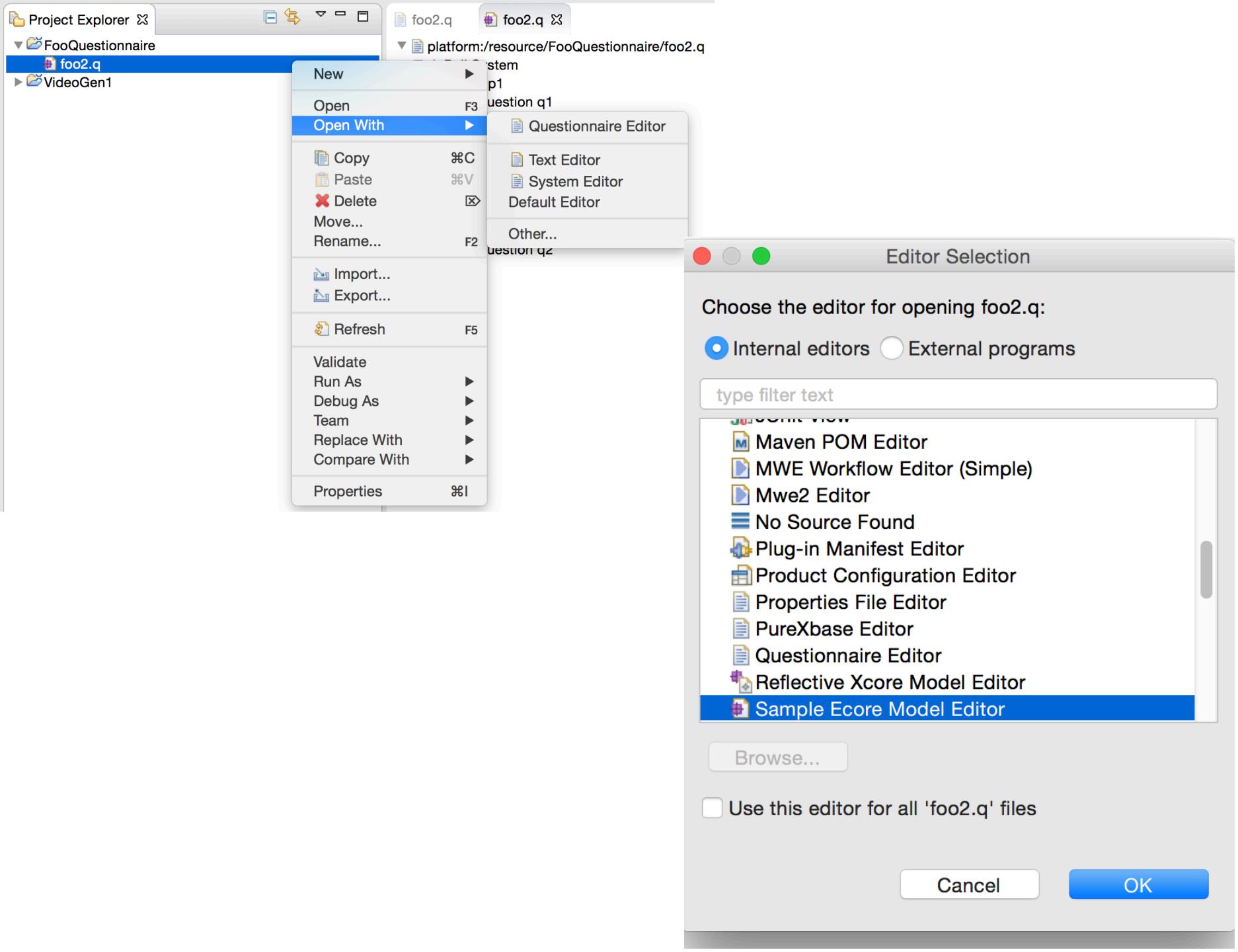
```
PollSystem {

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

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

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

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



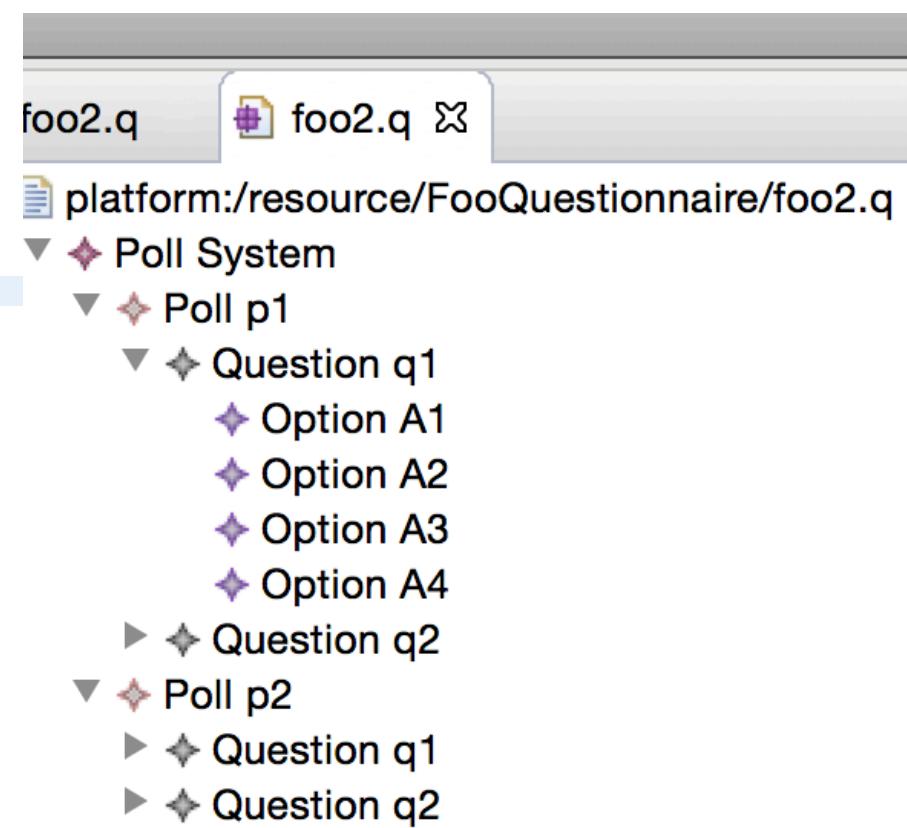
```
2.q ✎
ollSystem {

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

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

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

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



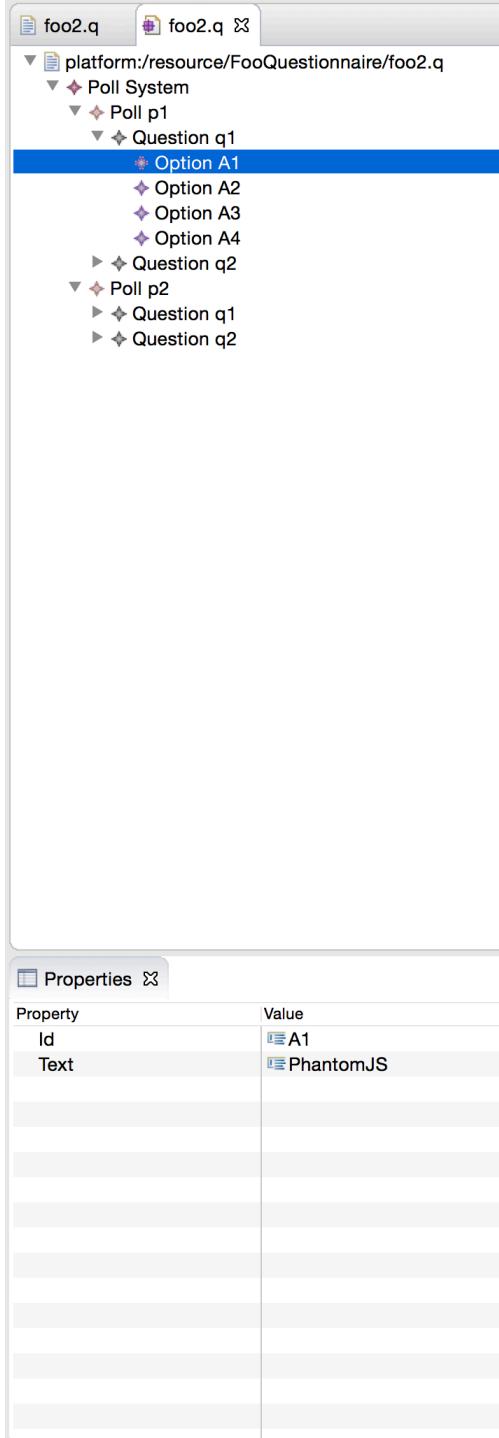
```
2.q ✎
ollSystem {

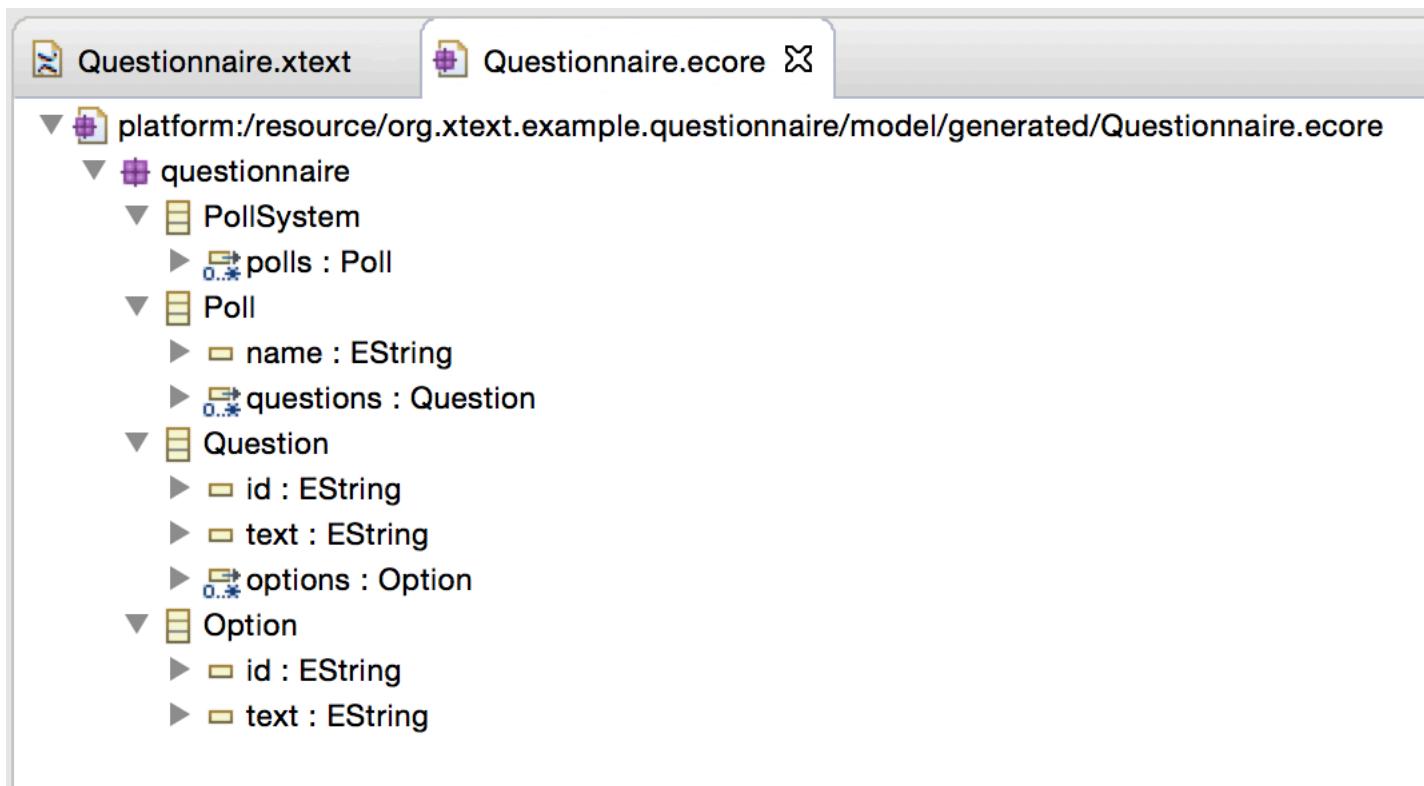
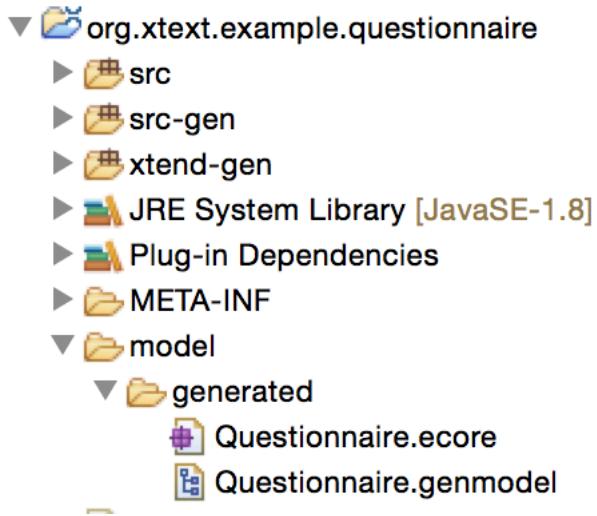
    Poll p1 {
        Question q1 {
            "What is the best JavaScript framework for testing?"
            options [
                A1: "PhantomJS"
                A2: "Jasmine"
                A3: "Mocha"
                A4: "I prefer to develop my own framework"
            ]
        }

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

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

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





Another example:

Chess

“Queen to c7.
Check.”



“Rd2-c2,
rook at d2 moves to c2”

Moves in Chess:

Rook at a1 moves to a5.

Piece Square Action Destination

Bishop at c8 captures knight at h3.

Piece Square Action Destination

N b1 x c3

Piece Square Action Destination

g2 - g4

Square Action Destination

Bishop at c8 captures knight at h3

$\mathbb{B} \text{ c8 x h3}$



P e2 – e4

p g7 – g5

Knight at b2 moves to c3

pawn at f7 moves to f5

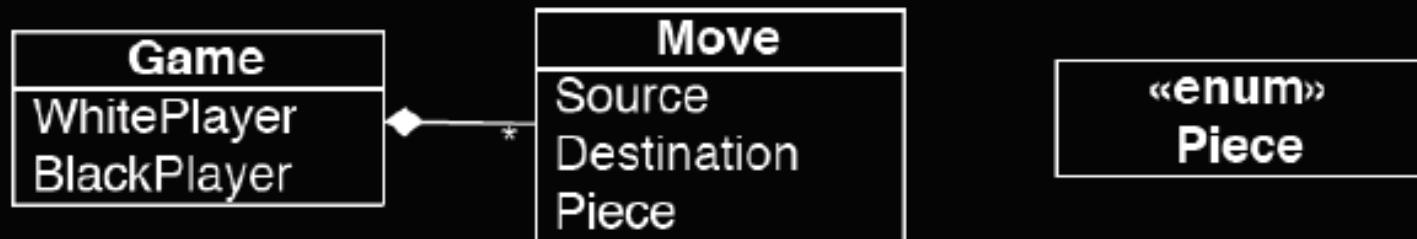
Q d1 – h5

1-0

Concrete Syntax

Constraints !!!

Abstract Syntax



Chess Example - Grammar

Game:

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

Move:

```
AlgebraicMove | SpokenMove;
```

AlgebraicMove:

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

SpokenMove:

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

terminal Square:

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

enum Piece:

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

Chess Example - Model

White: "Mayfield"

Black: "Trinks"

pawn at e2 moves to e4

pawn at f7 moves to g5

K b1 - c3

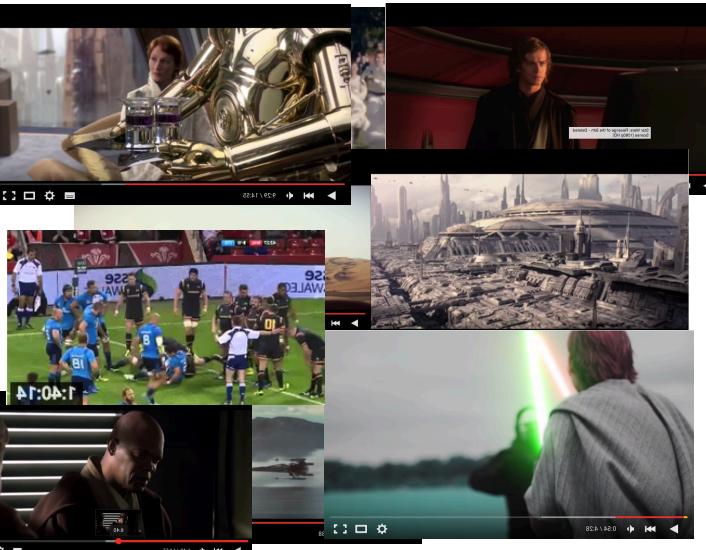
f7 - f5

queen at d1 moves to h5

// 1-0

Back to Video

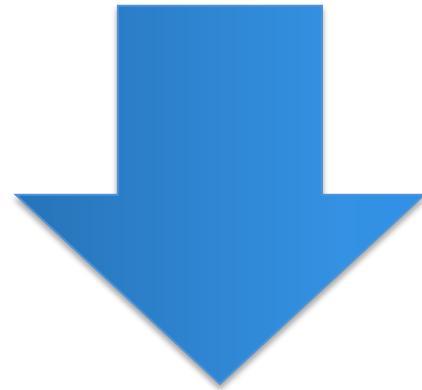
(VideoGen)



```
foo1.videogen ✘

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

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



 FFmpeg

foo1.videoogen

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

Quizz Time

Write a Xtext grammar so that the specification below is conformant

The image shows a code editor window with a tab labeled "foo1.videogen". The content of the file is a Xtext grammar for a video specification language. The grammar defines several video sequences (v1 through v5) with their URLs or local paths. It also defines two sets of alternatives (v3 and v4), each containing three video sequences. The code uses Xtext's built-in types like "videoseq" and "alternatives".

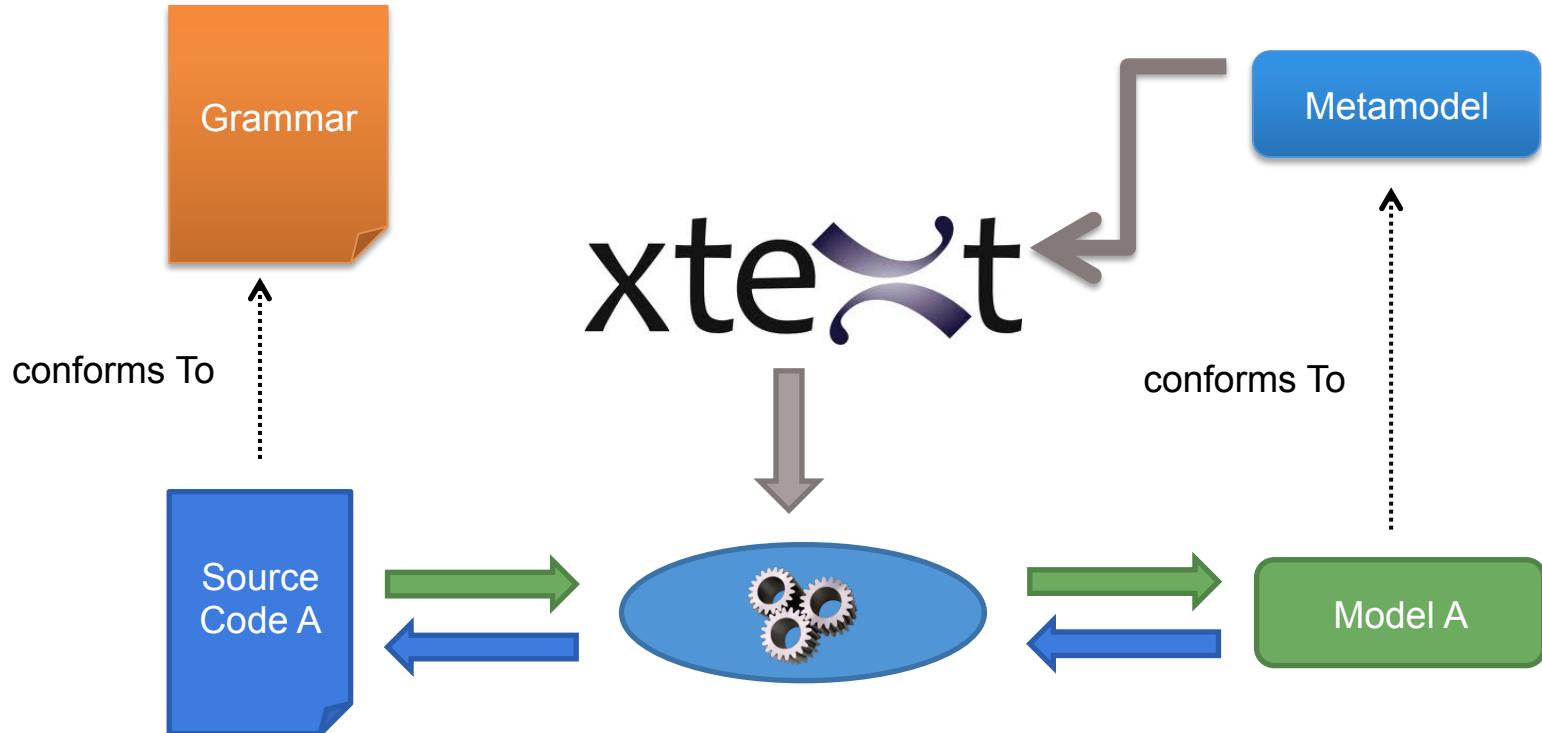
```
mandatory videoseq v1 "https://www.youtube.com/watch?v=PJNi1uYhV5w"
optional videoseq v2 "v2Folder/v2.mp4"
@alternatives v3 {
    videoseq v31 "v3/seq1.mp4"
    videoseq v32 "v3/seq1.mp4"
    videoseq v33 "v3/seq1.mp4"
}
@alternatives v4 {
    videoseq v41 "v4/seq1.mp4"
    videoseq v42 "v4/seq1.mp4"
}
mandatory videoseq v5 "https://www.youtube.com/watch?v=ezKx-S0LiNQ"
```

From Metamodel

To

Grammar (other side)

From Metamodel to Grammar





Give me a **metamodel**,

I'll give you (for free)

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



Give me a **metamodel**,

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

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

Live
Demonstration

New

Select a wizard

Create an Xtext project from existing Ecore models

Wizards:

Xtext

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

?

< Back Next > Cancel Finish

New Xtext Project From Ecore

Select EPackages

Select the EPackages to generate an Xtext grammar for.

EPackages:

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

Add... Set Default Remove

Entry rule:

PollSystem - questionnaire

?

< Back Next > Cancel Finish

Questionnaire.xtext Questionnaire.ecore Questionnaire.xtext Questionnaire.ecore Questionnaire2.xtext

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

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

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

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

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

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

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

Quizz Time

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

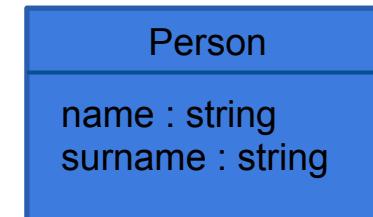
Graphical DSL (vs Textual DSL)

Graphical vs Textual DSLs

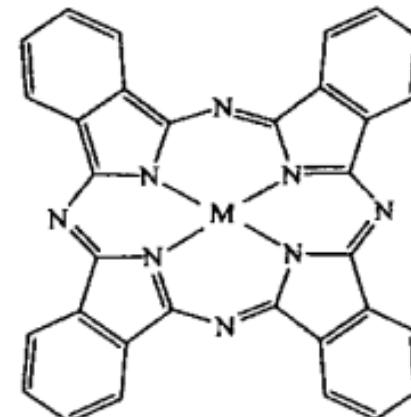
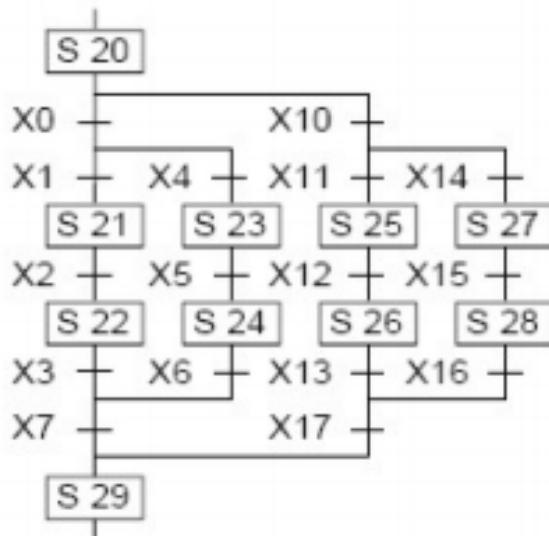
- Success depends on how the notation fits the domain

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

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

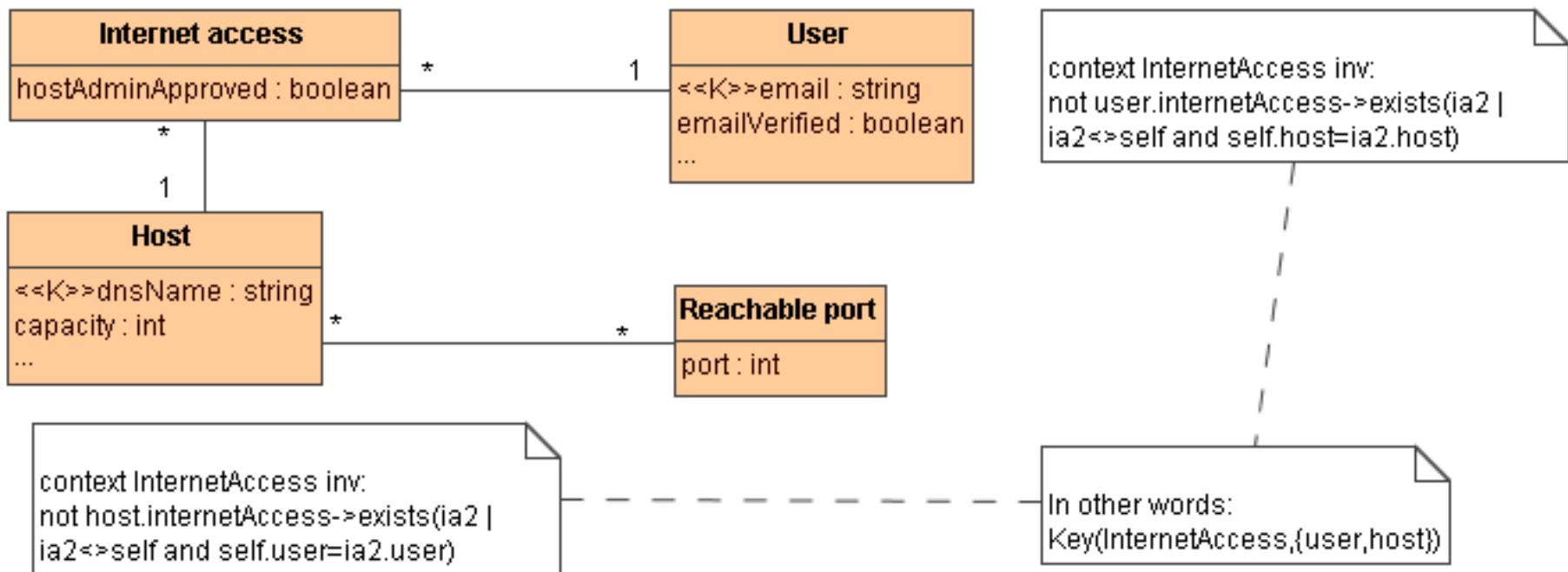


- Graphical DSLs are not always easier to understand



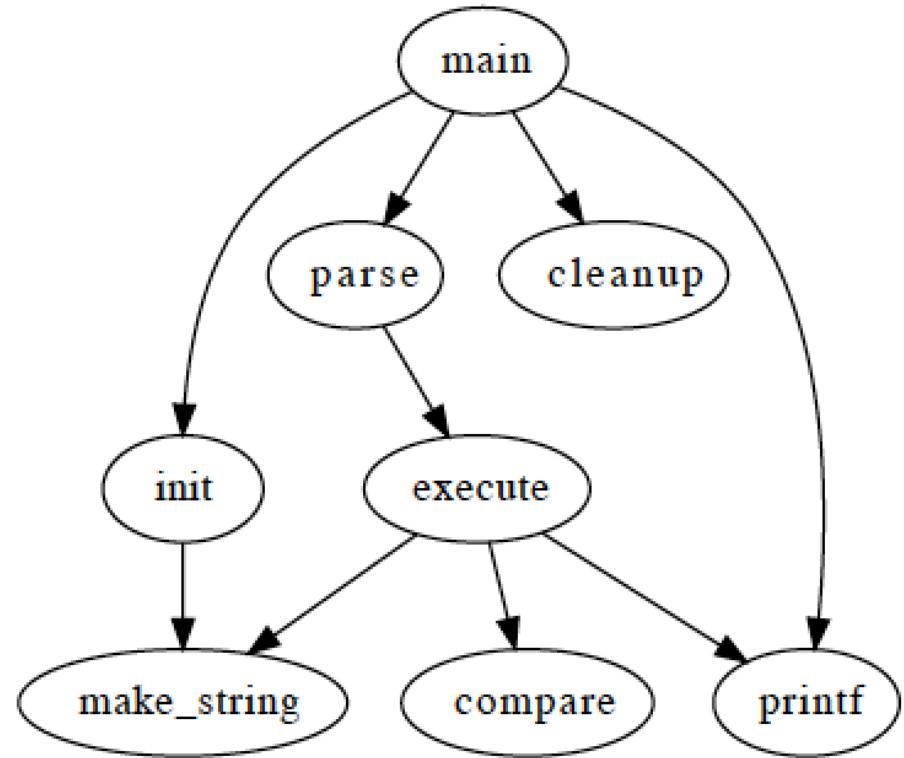
phthalocyanine

A language can be graphical and textual

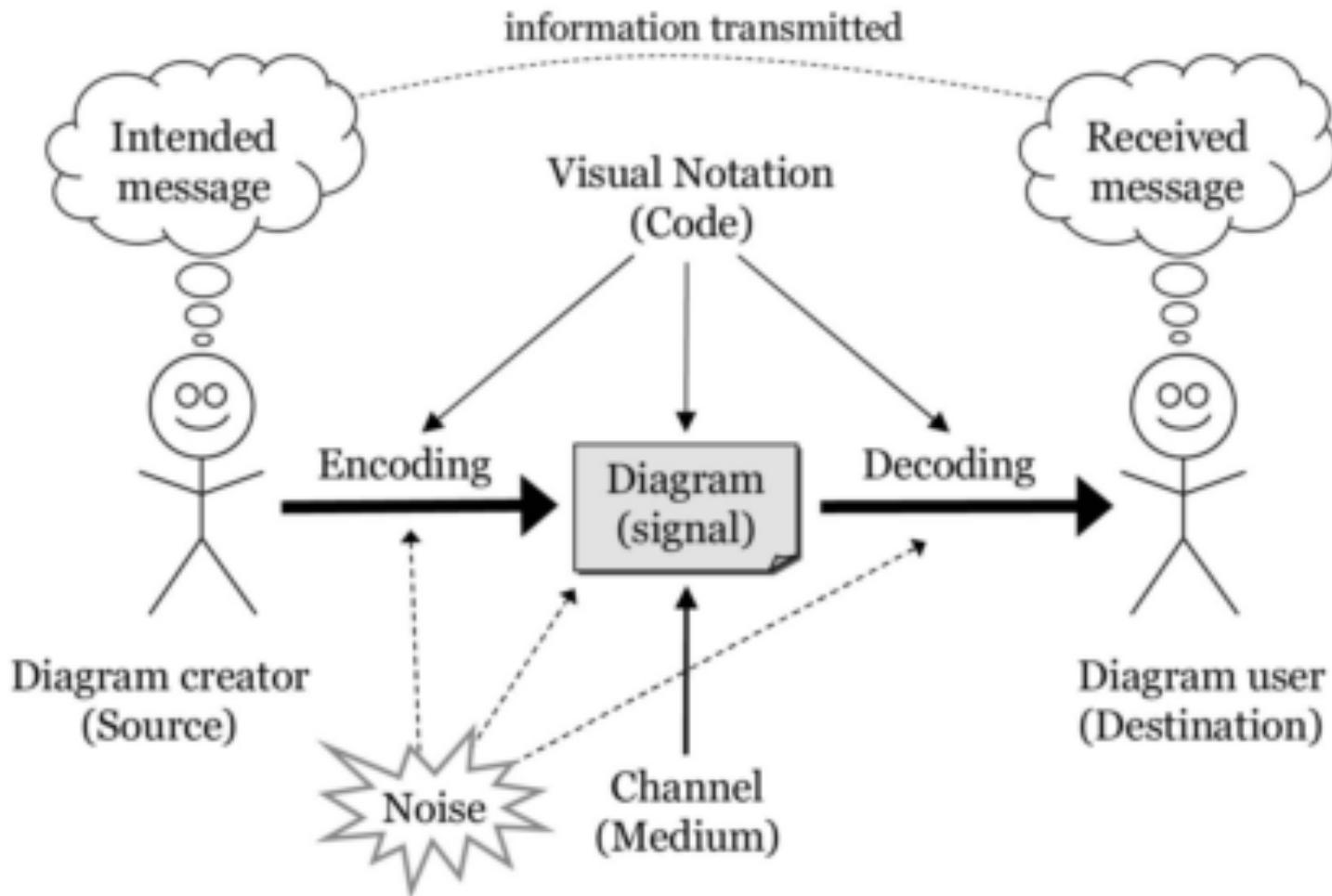


Alternative representation

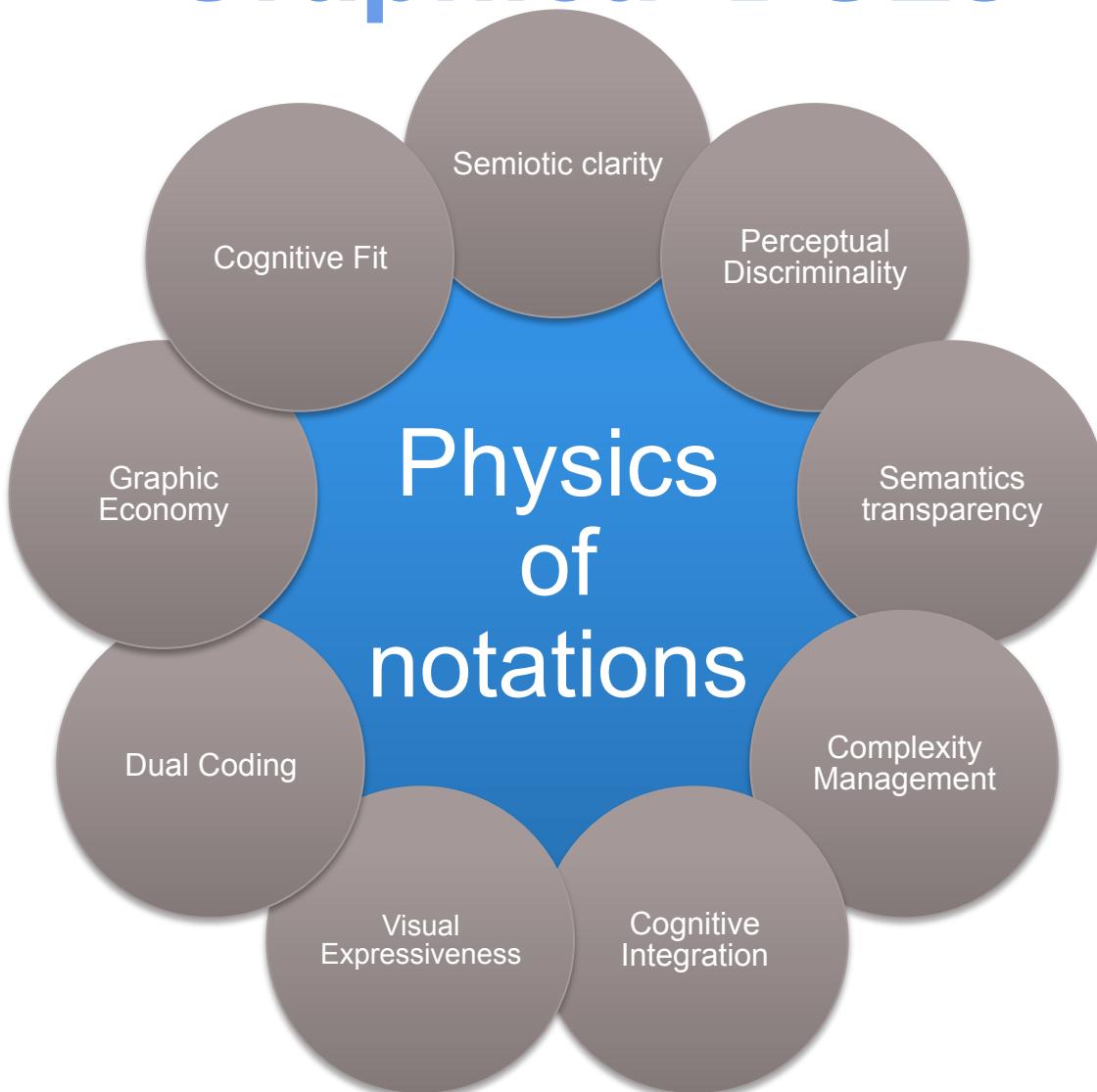
```
digraph G {  
    main -> parse -> execute;  
    main -> init;  
    main -> cleanup;  
    execute -> make_string;  
    execute -> printf;  
    init -> make_string;  
    main -> printf;  
    execute -> compare;  
}
```



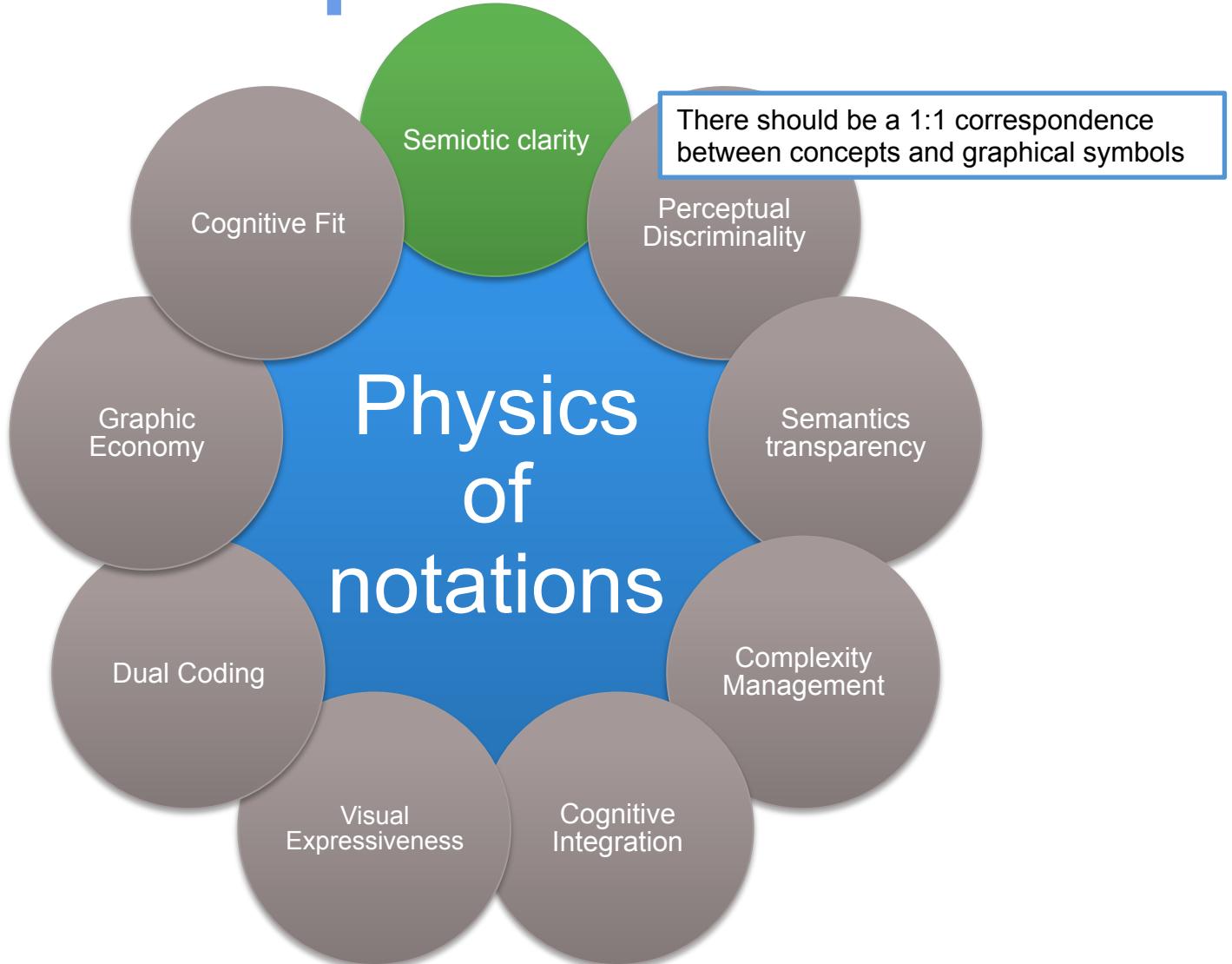
Recommendations for Graphical DSLs



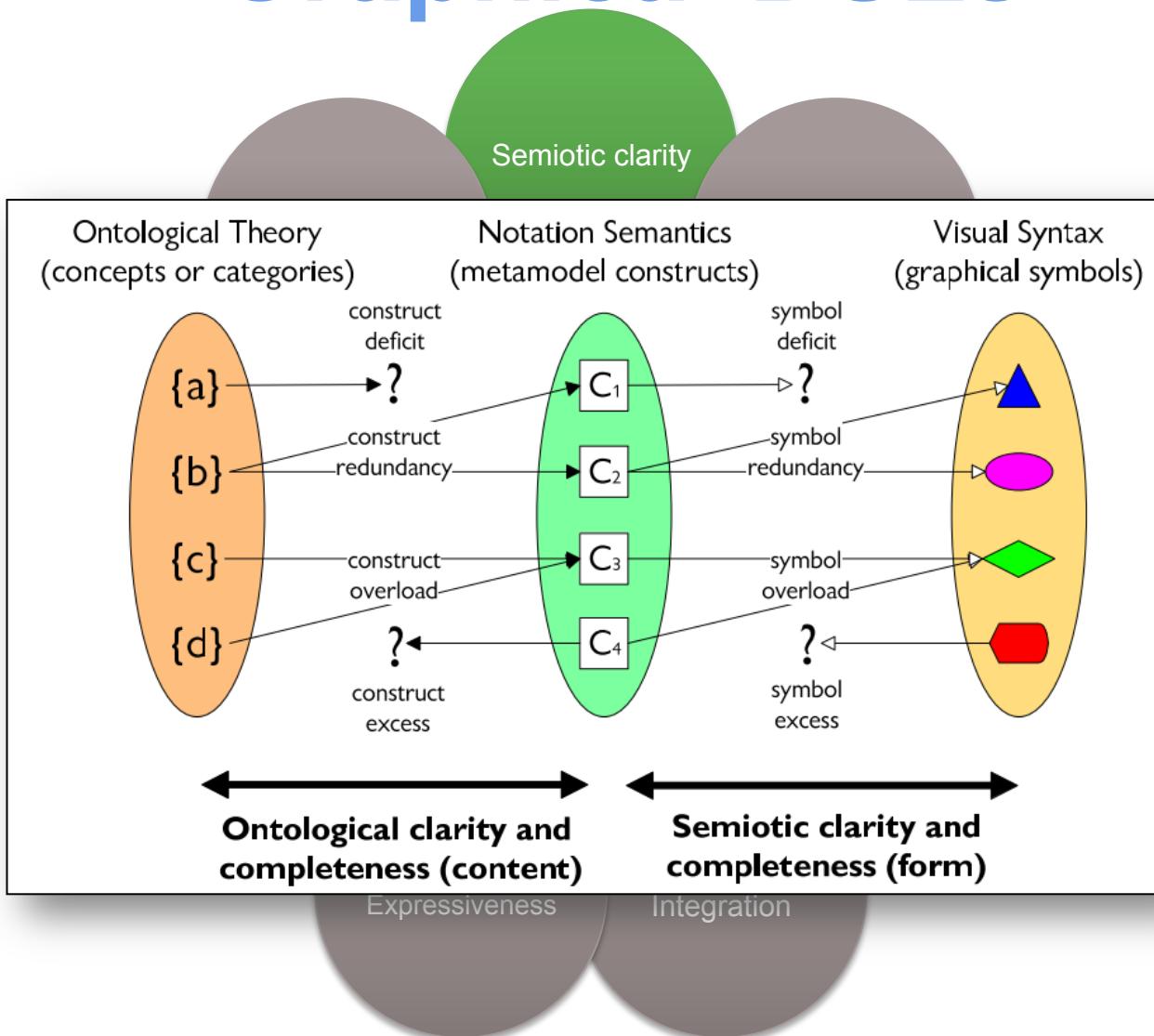
Recommendations for Graphical DSLs



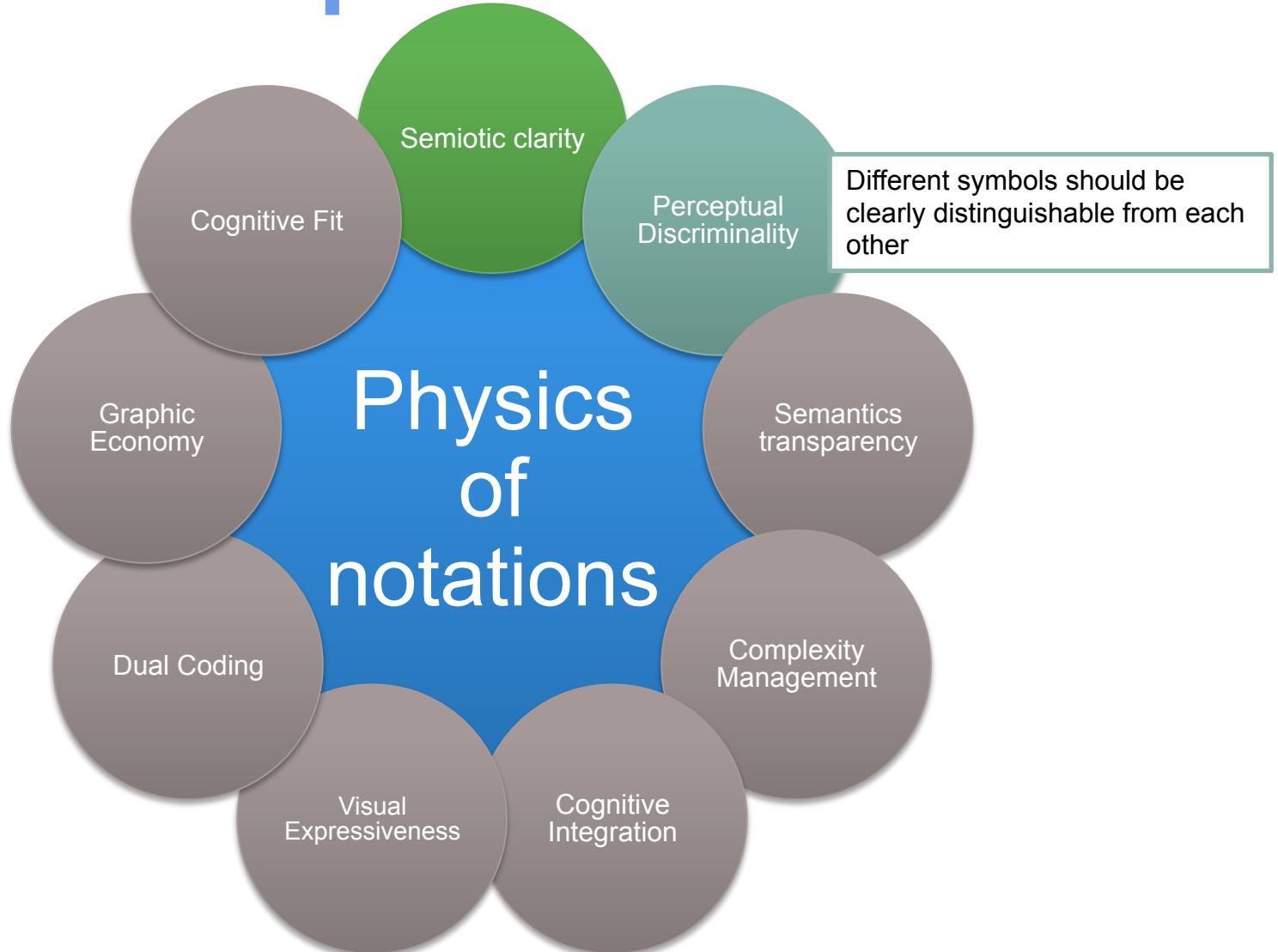
Recommendations for Graphical DSLs



Recommendations for Graphical DSLs



Recommendations for Graphical DSLs



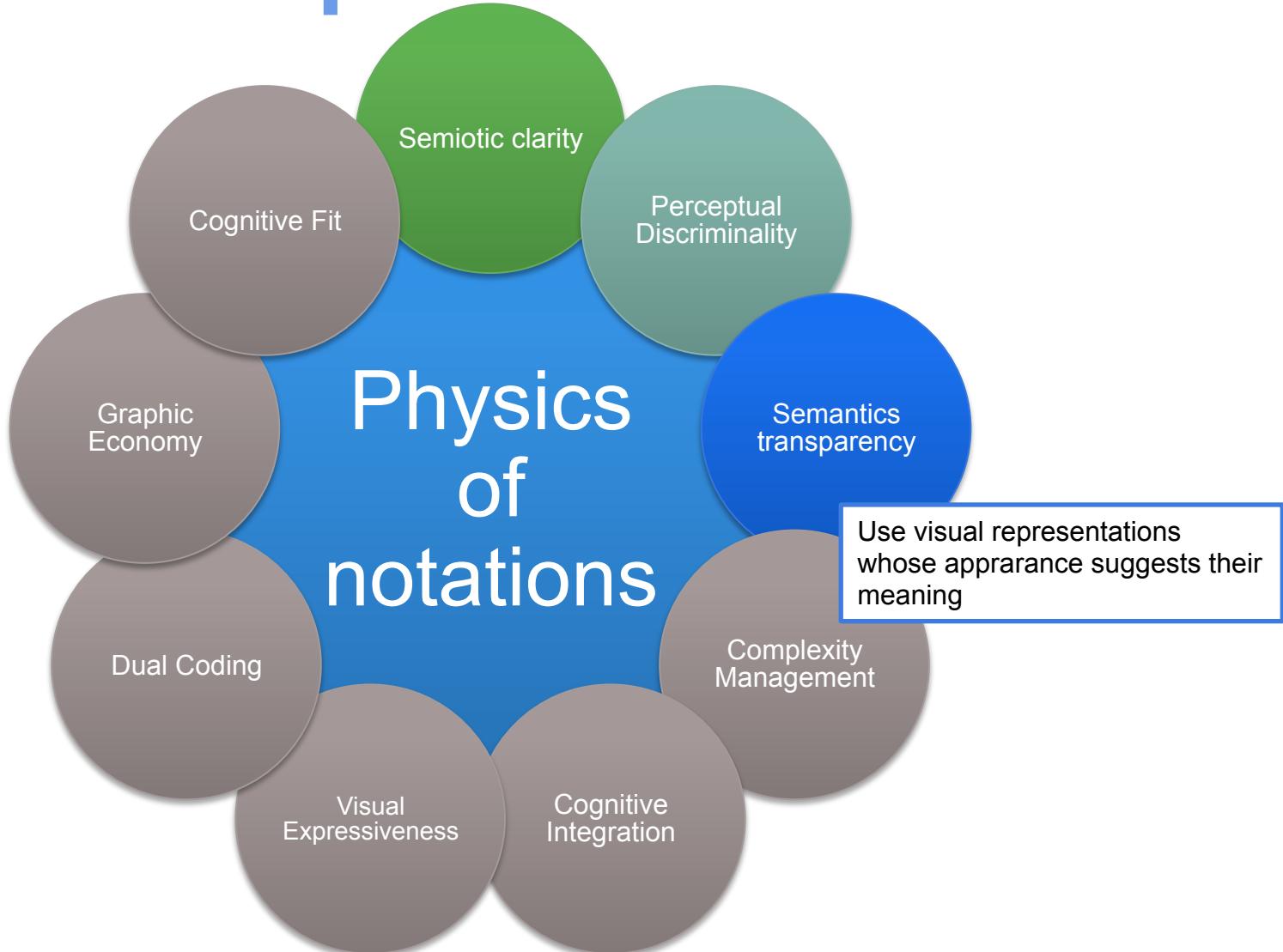
Recommendations for Graphical DSLs

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

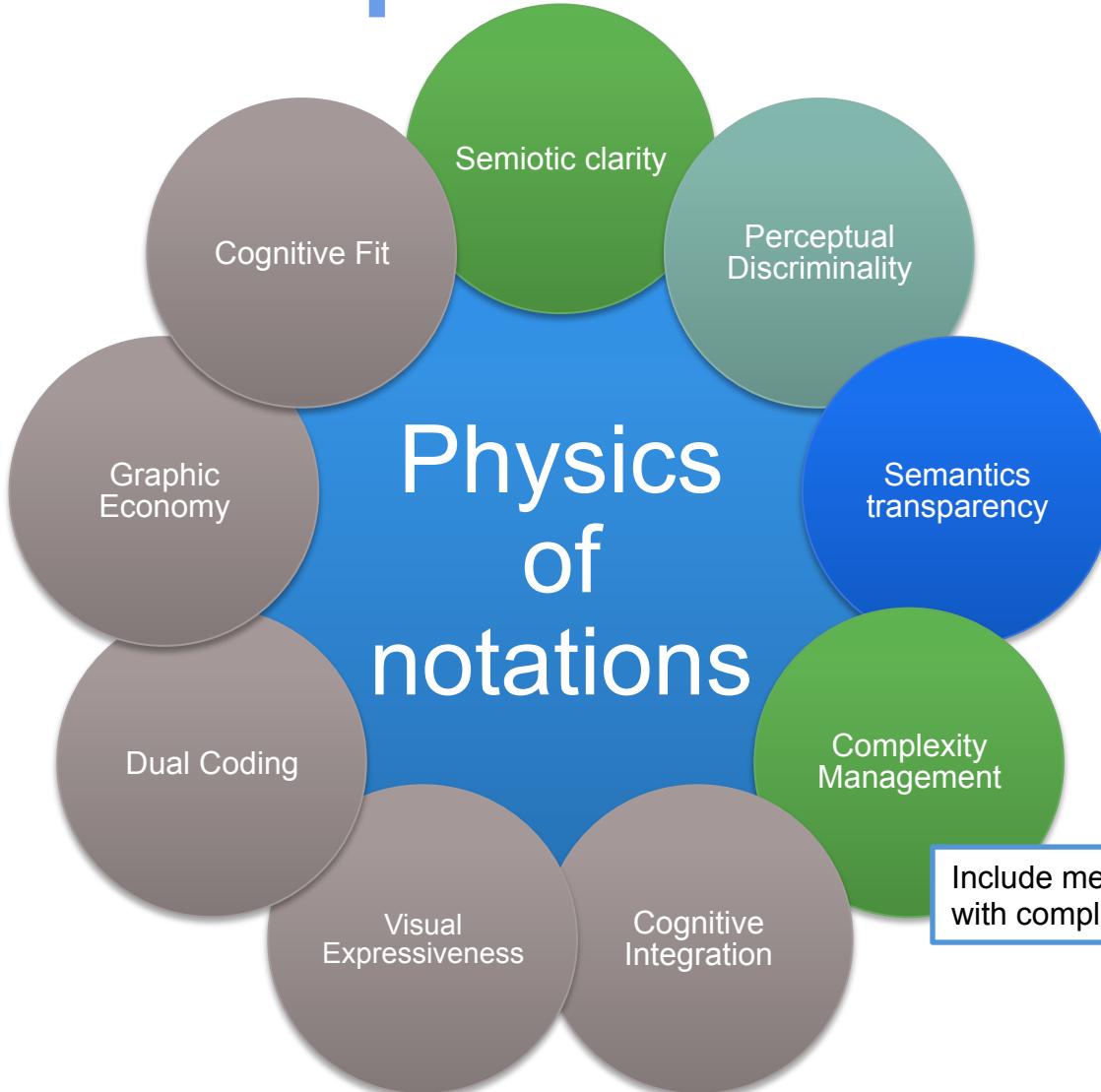
Visual Expressiveness

Cognitive Integration

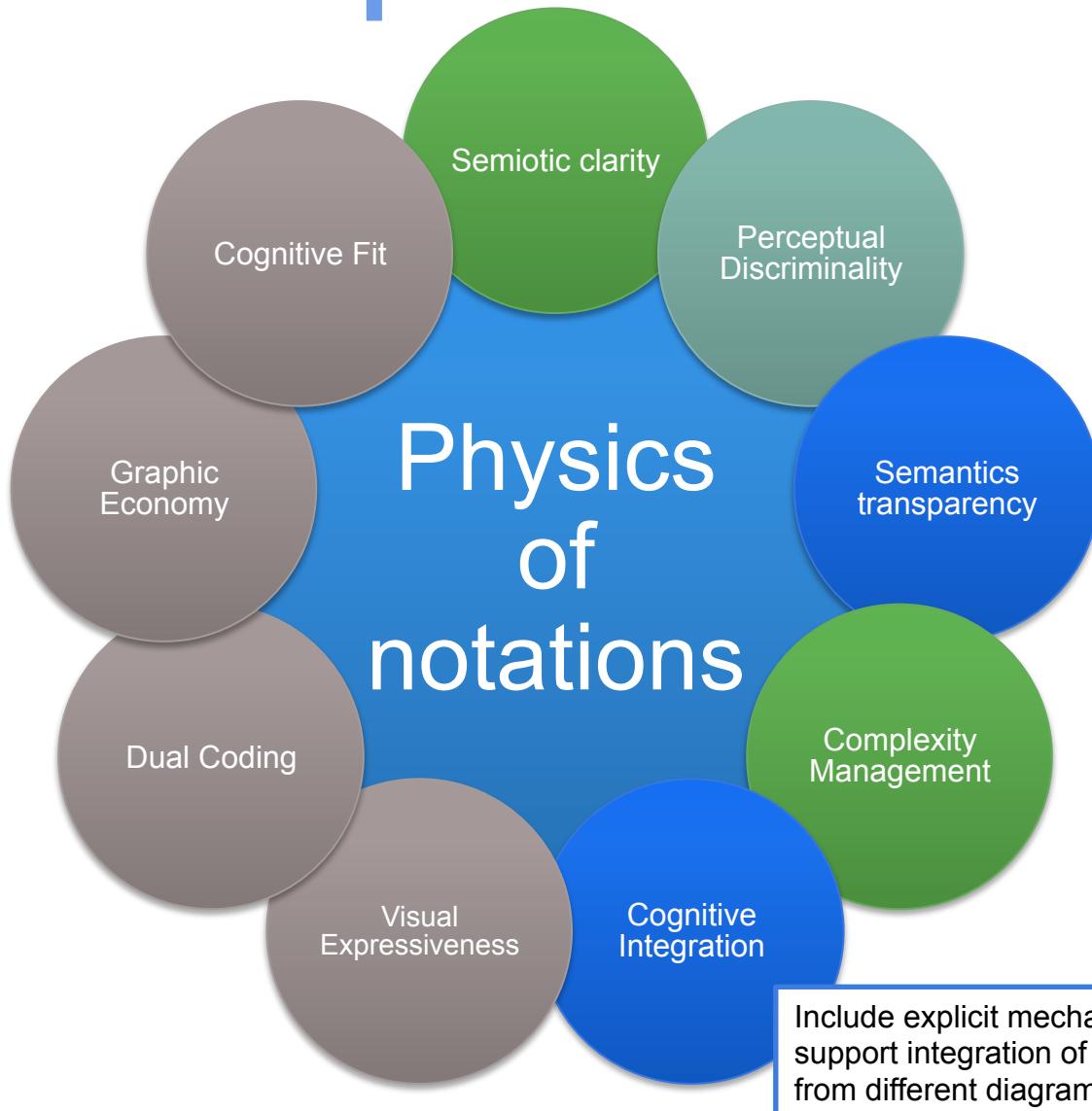
Recommendations for Graphical DSLs



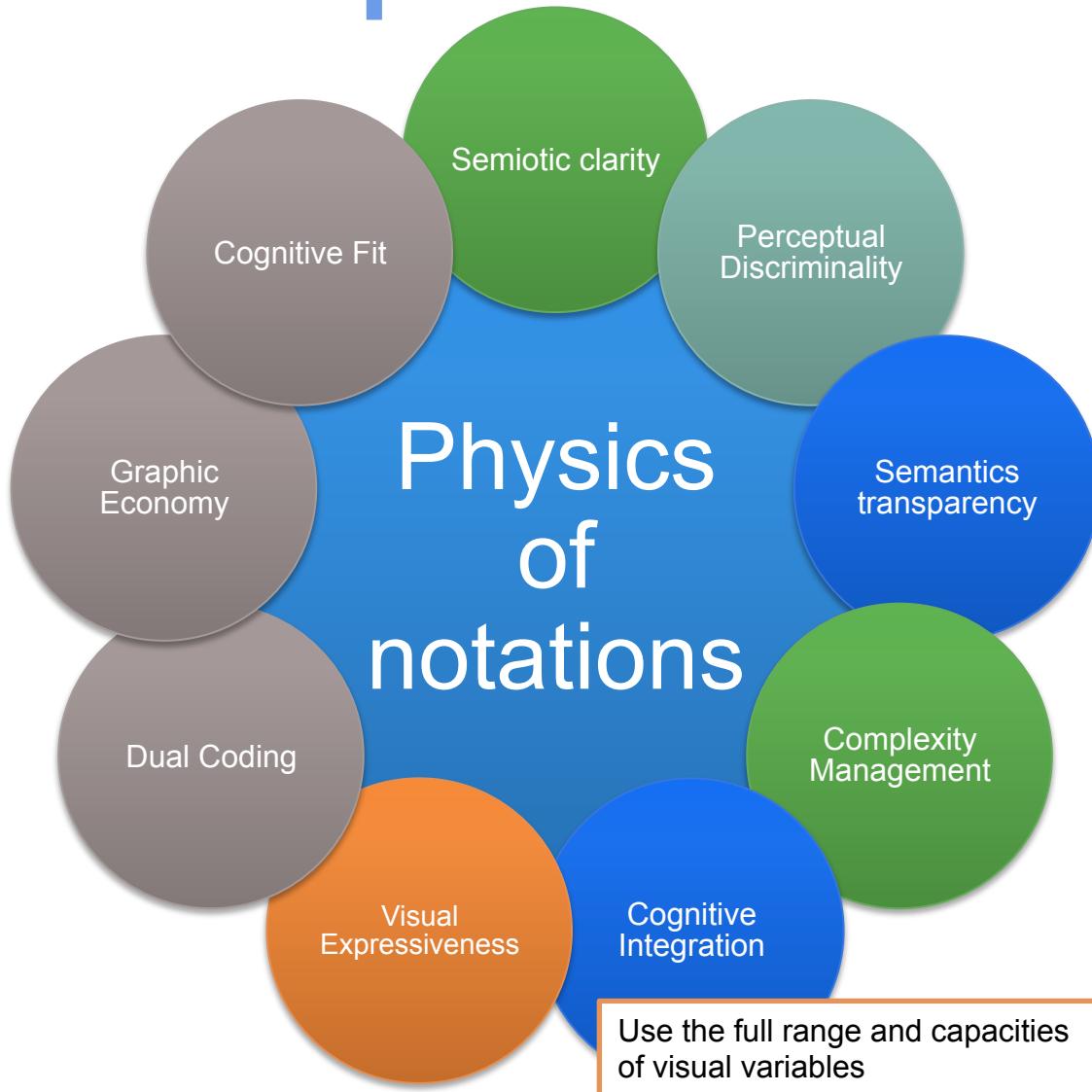
Recommendations for Graphical DSLs



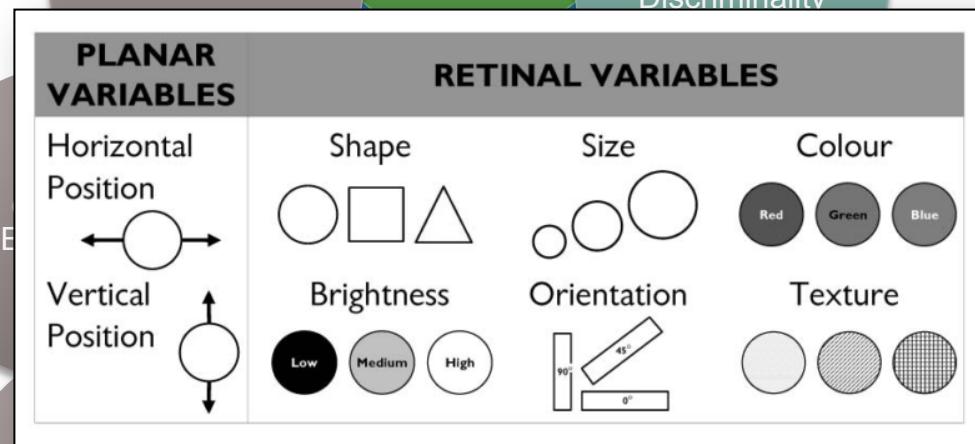
Recommendations for Graphical DSLs



Recommendations for Graphical DSLs



Recommendations for Graphical DSLs

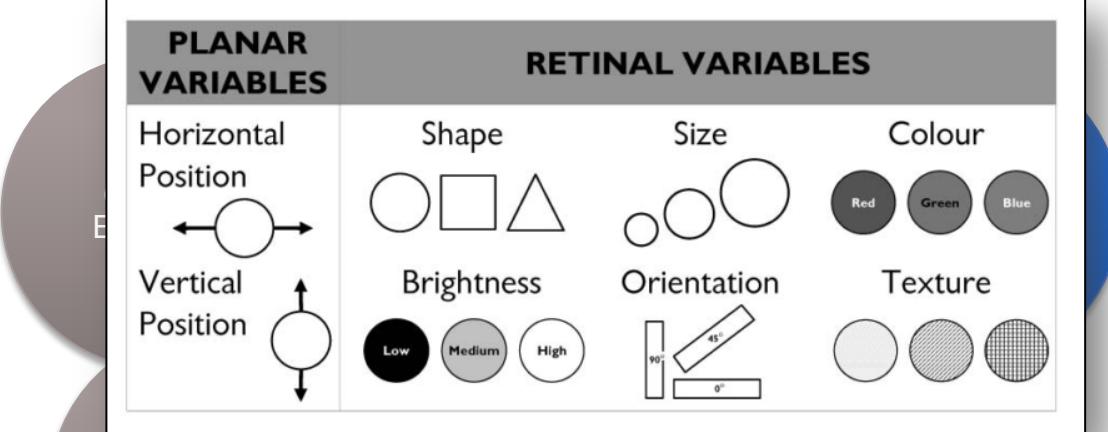


Dual Coding

Visual Expressiveness

Cognitive Integration

Complexity Management



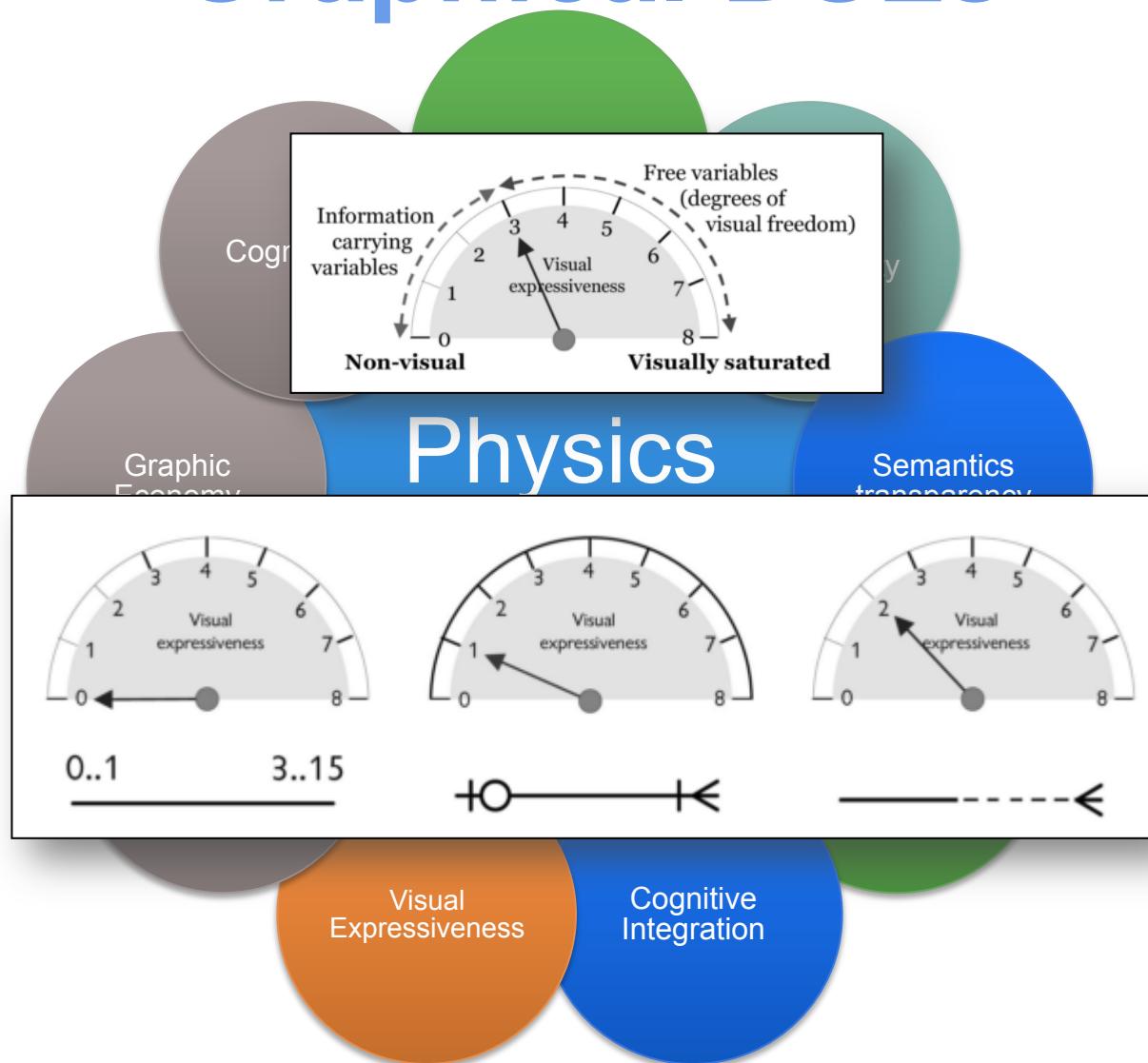
Dual Coding

Visual Expressiveness

Cognitive Integration

Complexity Management

Recommendations for Graphical DSLs



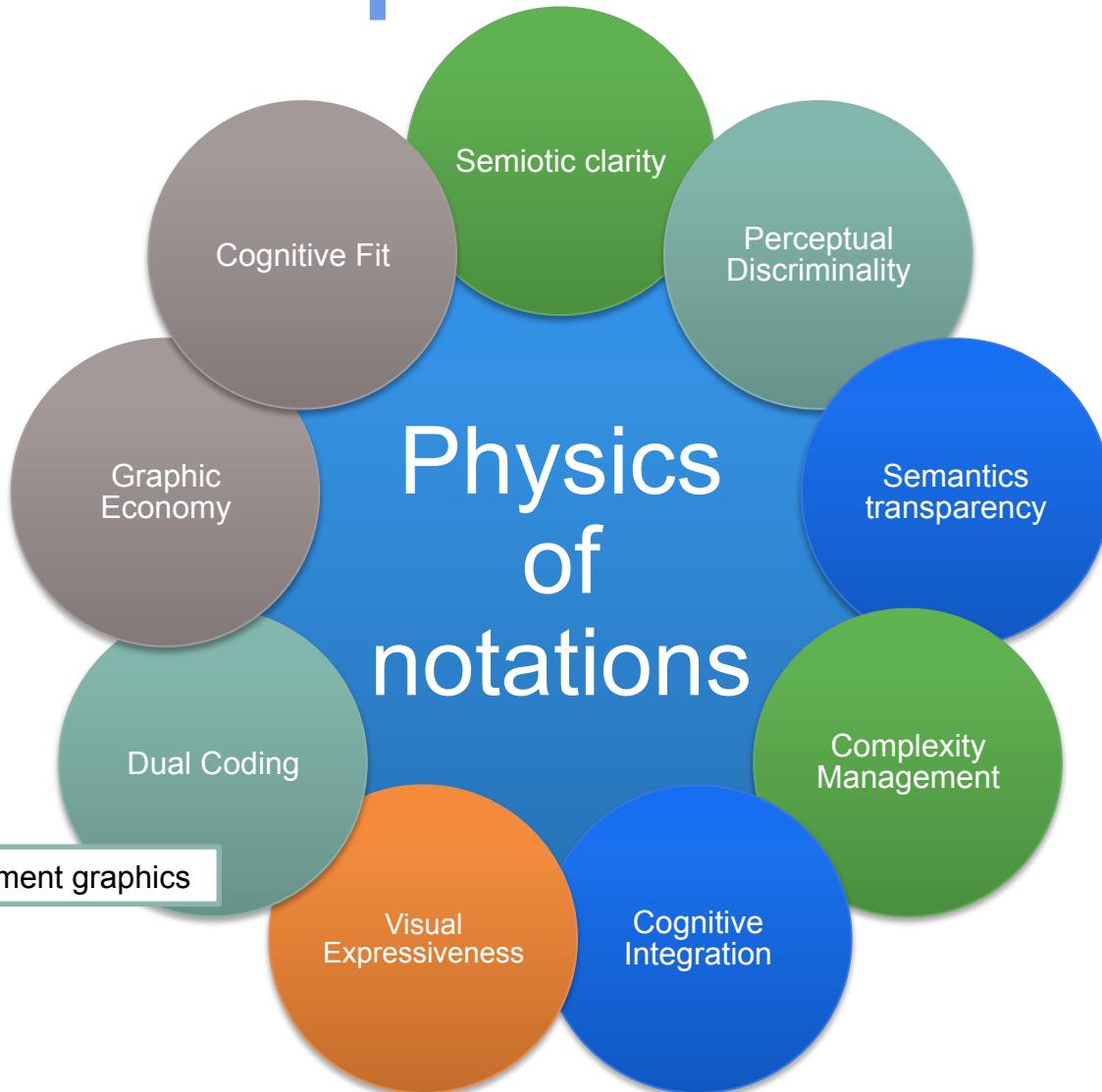
Recommendations for Graphical DSLs

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

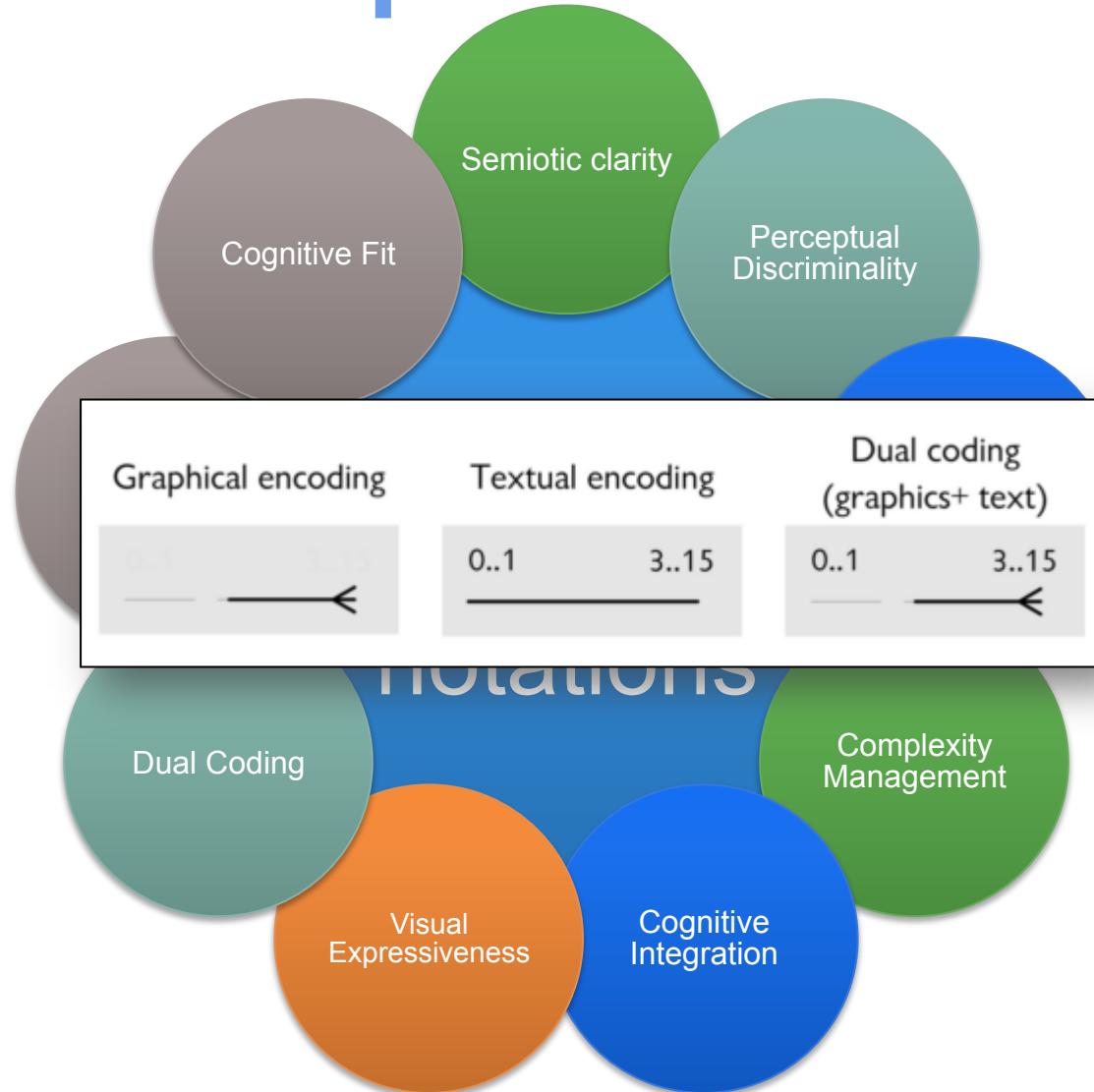
Visual Expressiveness

Cognitive Integration

Recommendations for Graphical DSLs



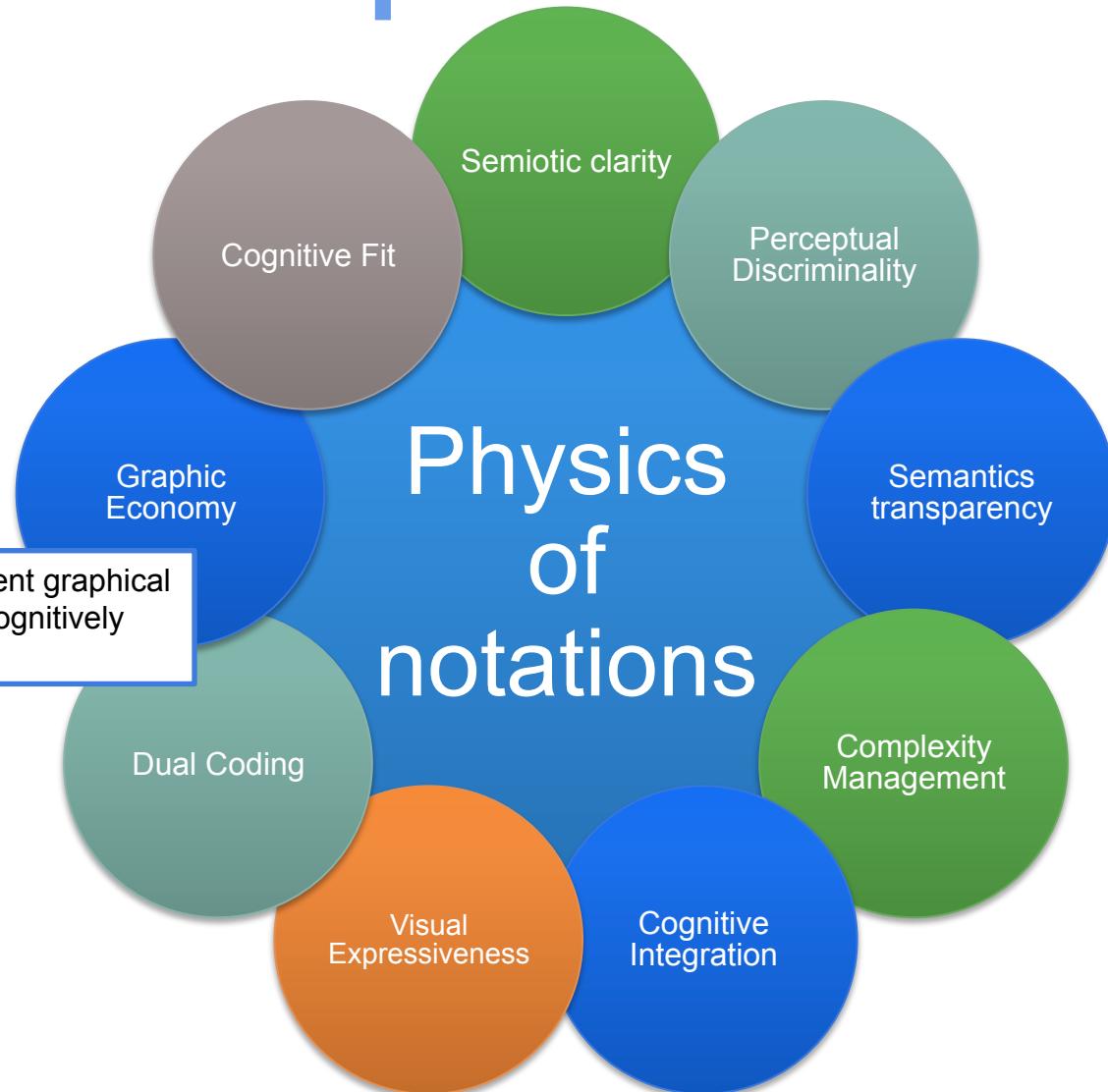
Recommendations for Graphical DSLs



Recommendations for Graphical DSLs

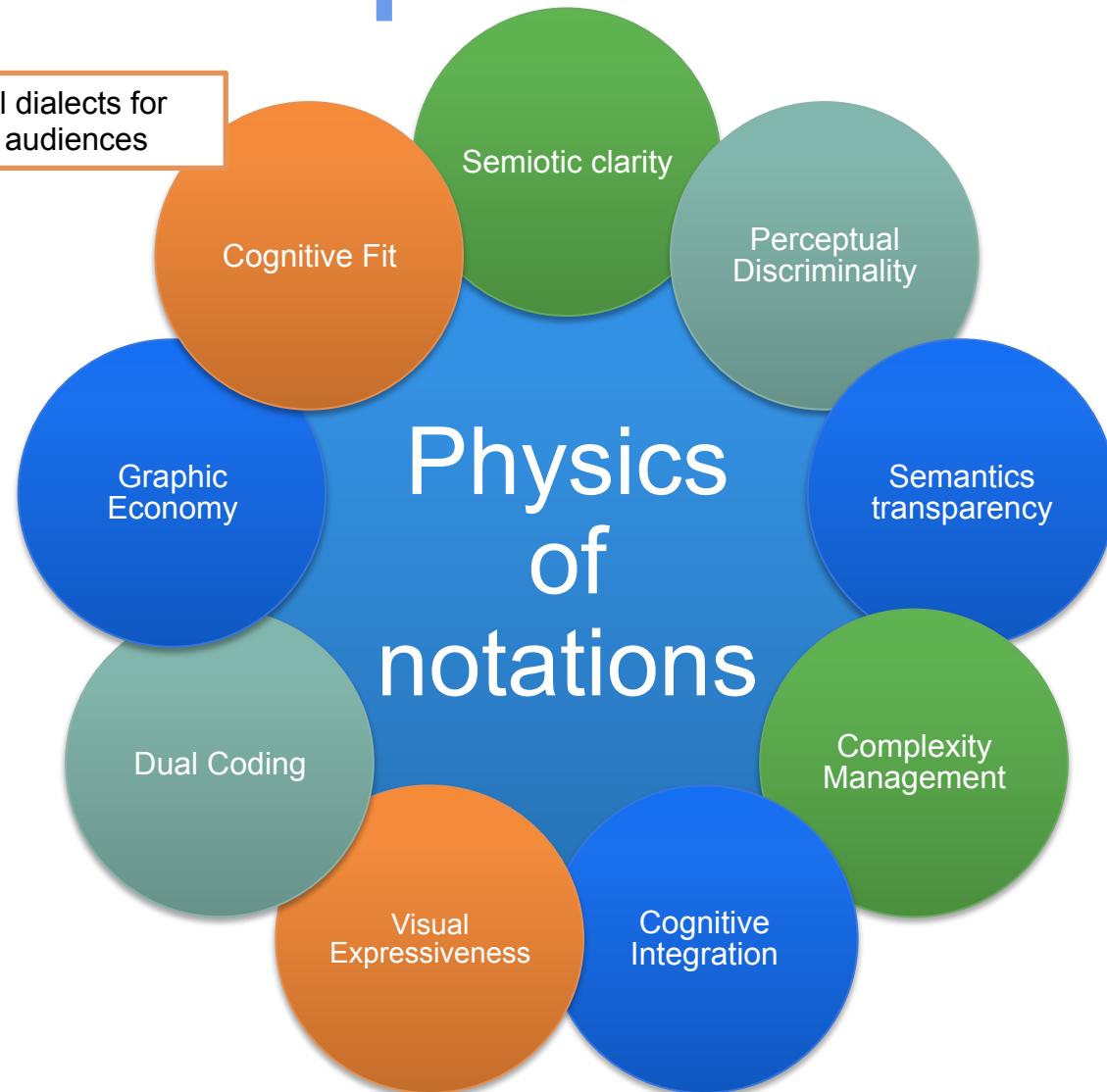
Physics of notations

The number of different graphical symbols should be cognitively manageable



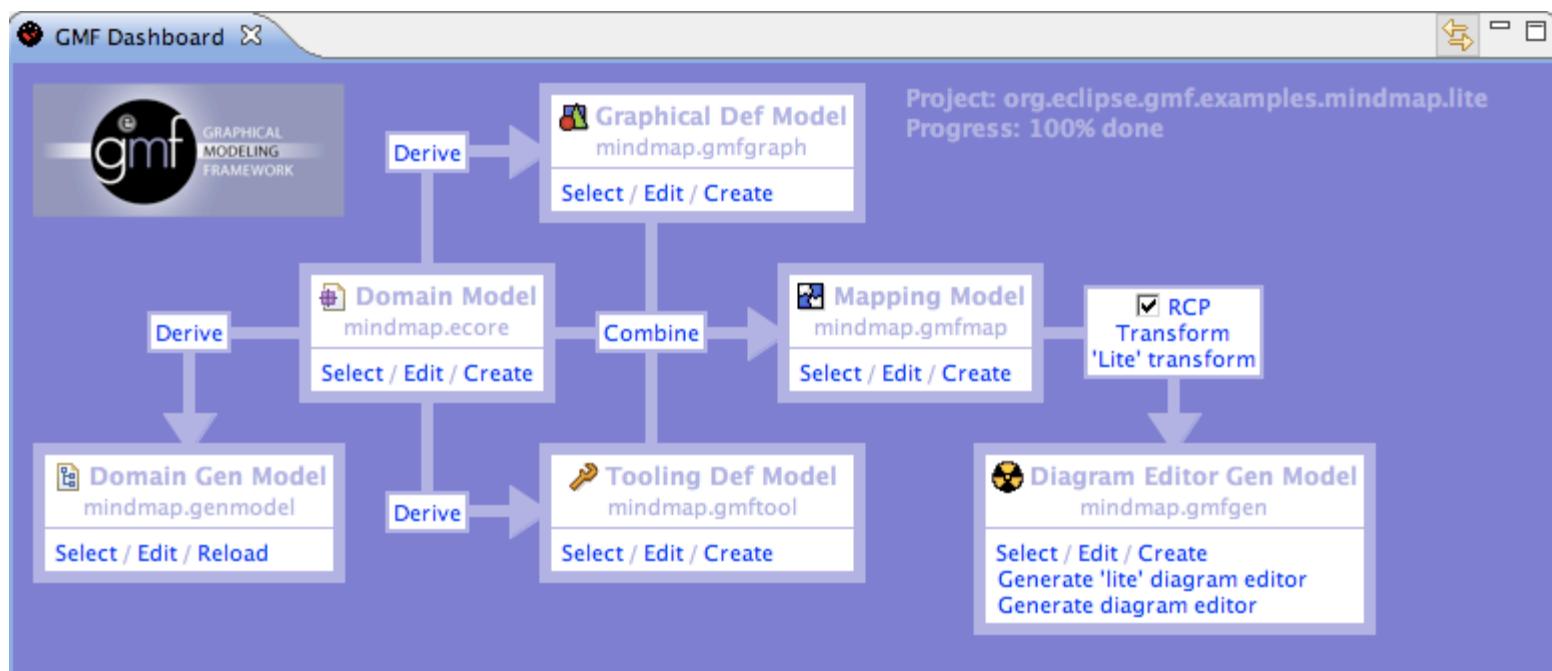
Recommendations for Graphical DSLs

Use different visual dialects for different tasks and audiences



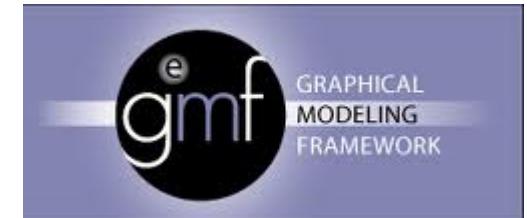
Graphical Modeling Framework (GMF)

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

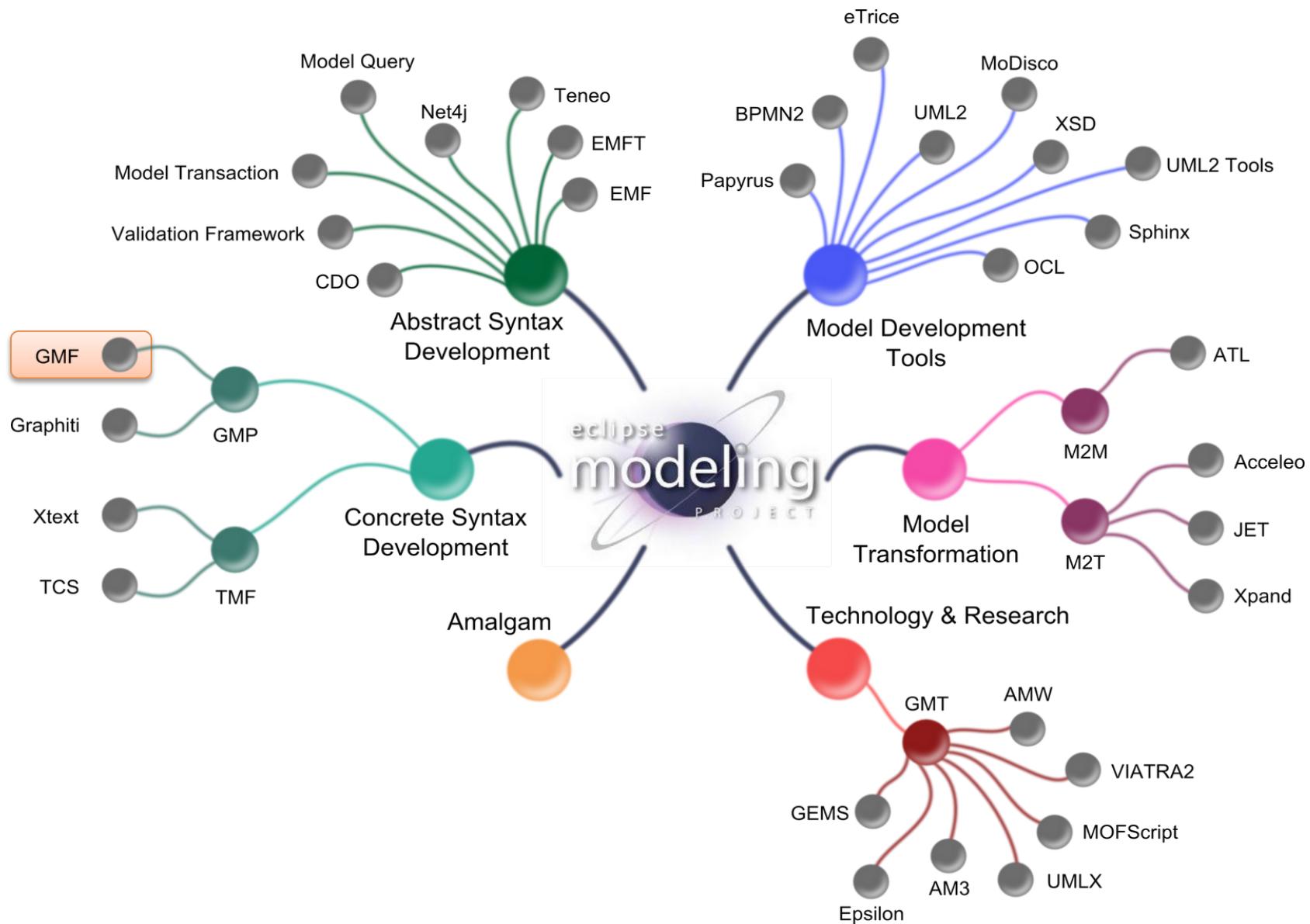


GMF

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



Eclipse Modeling Project



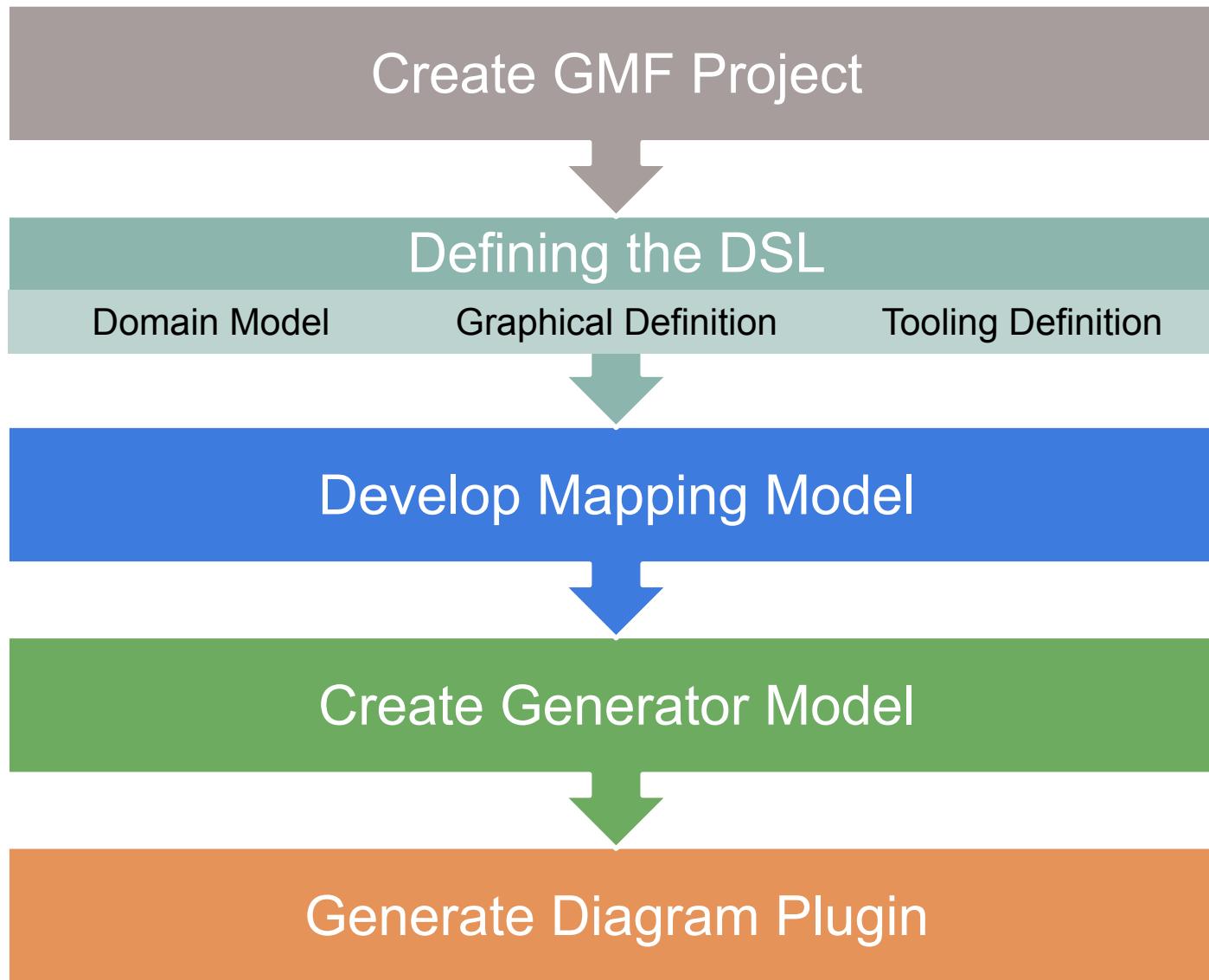
GMF features

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

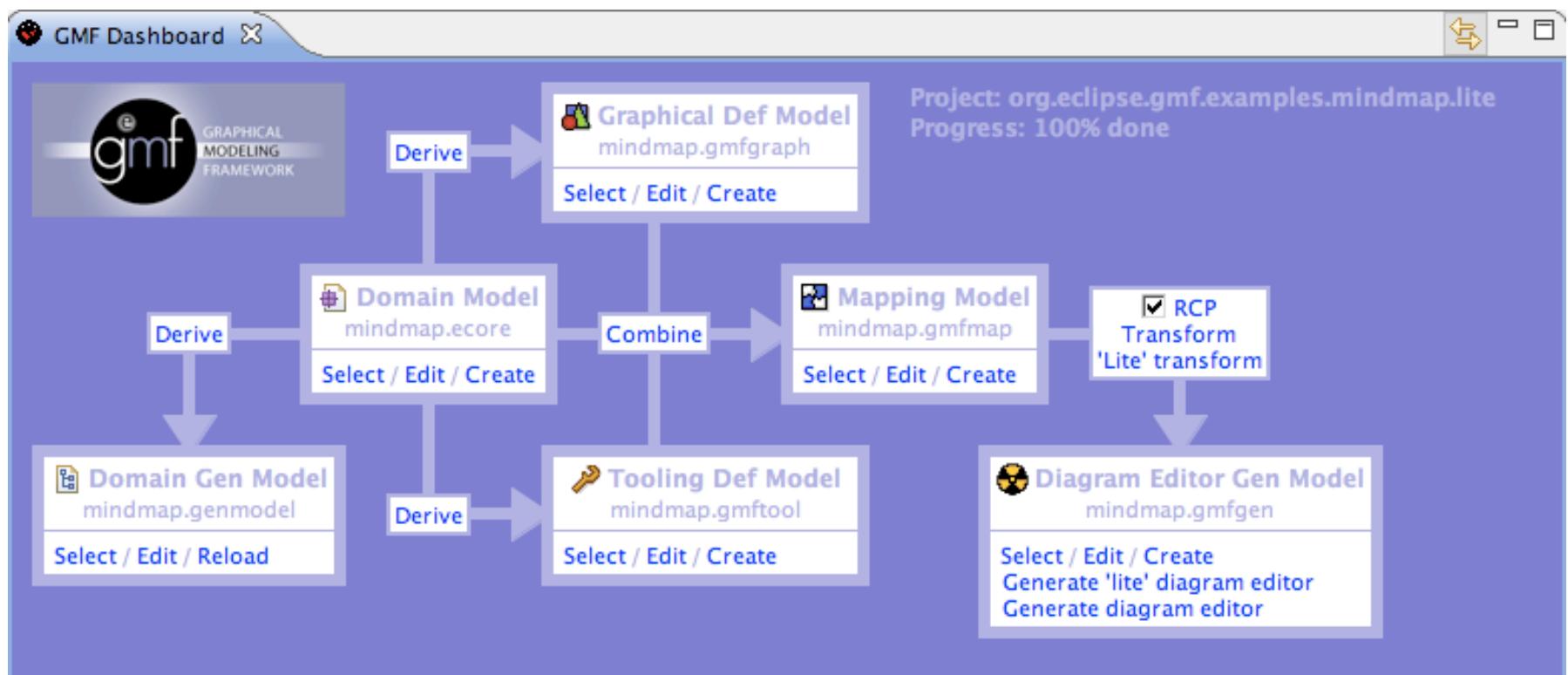
Main Advantages

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

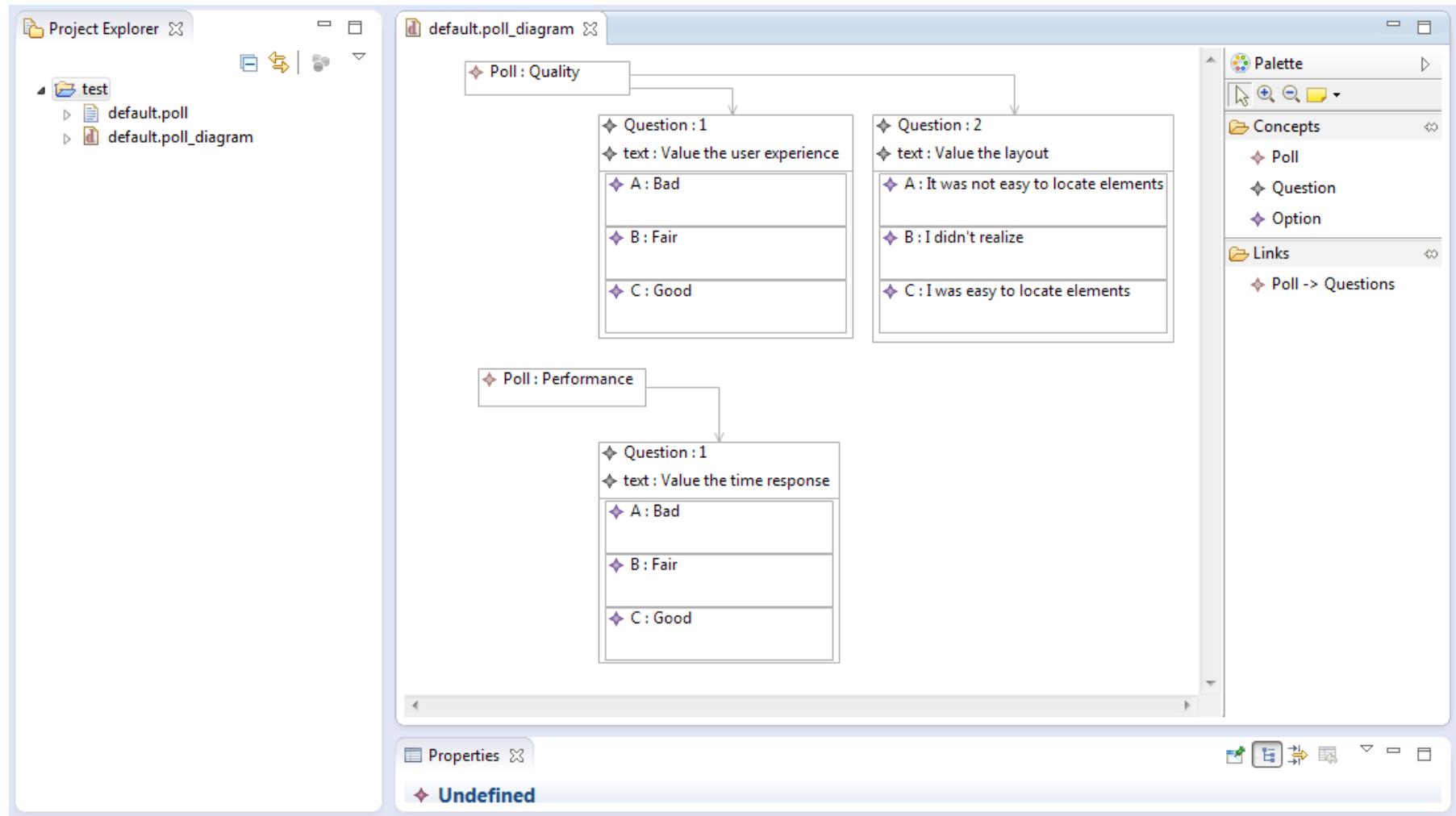
Development Process



Development Process

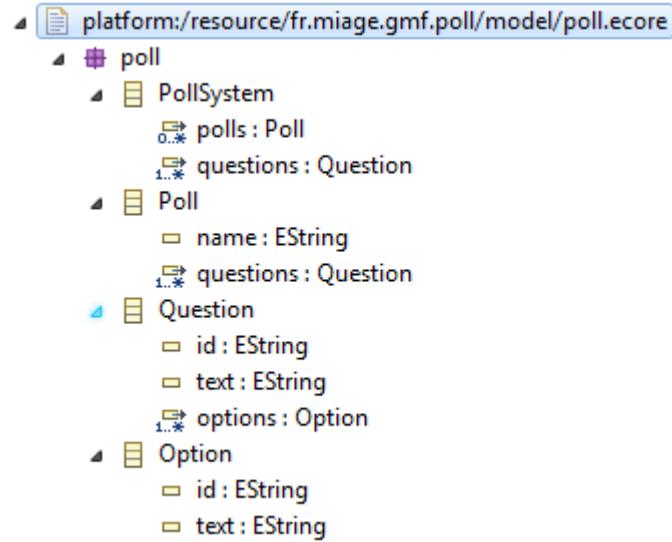


Example (Graphical Notation)



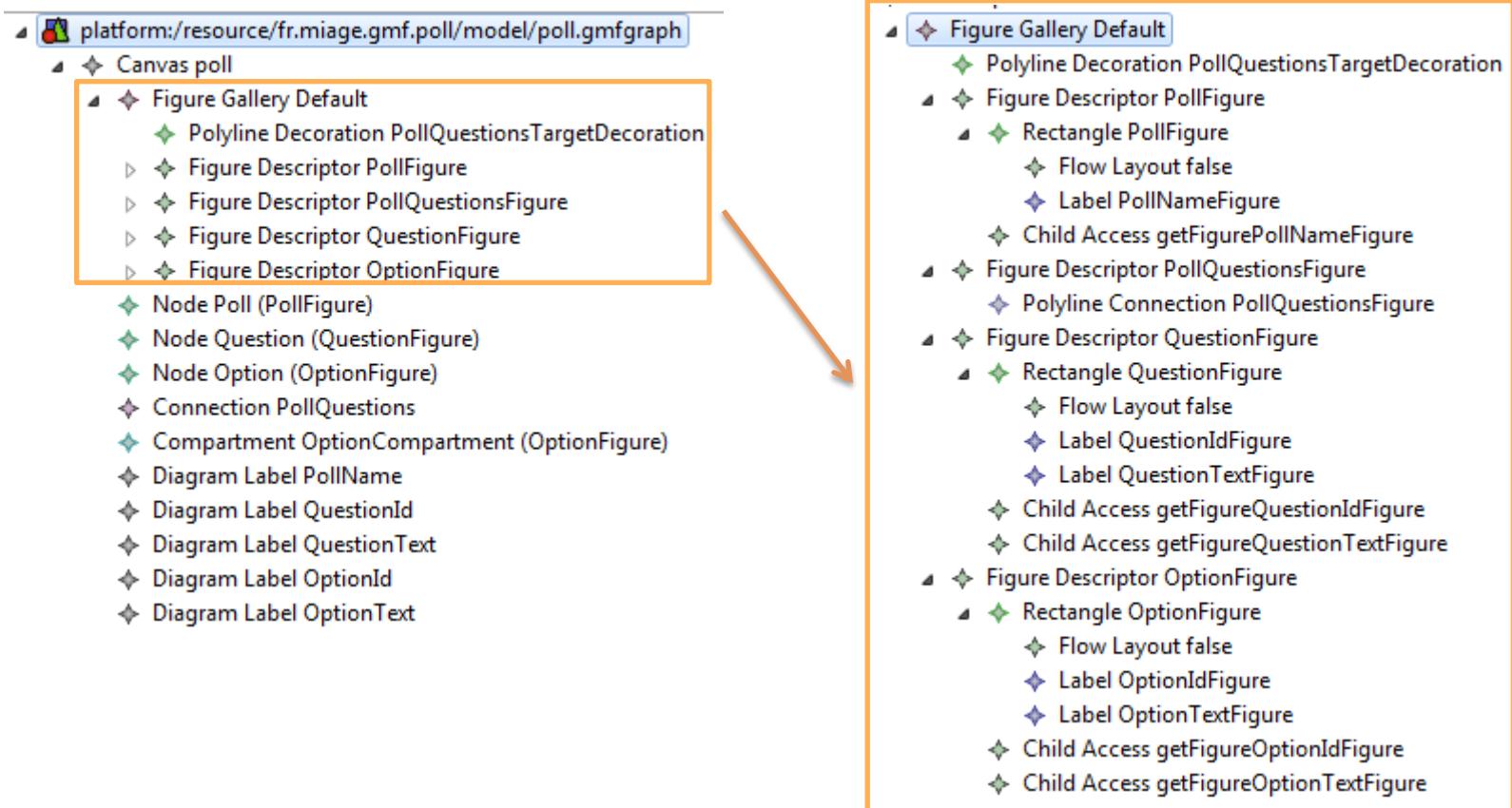
Poll System Metamodel

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



Graphical Definition

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



Plan

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

Contract

- Better understanding/source of inspiration of software languages and DSLs
 - Revisit of history and existing languages
- Foundations and practice of Xtext
 - State-of-the-art language workbench (Most Innovative Eclipse Project in 2010, mature and used in a variety of industries)
- Models and Languages
 - Perhaps a more concrete way to see models, metamodels and MDE (IDM in french)

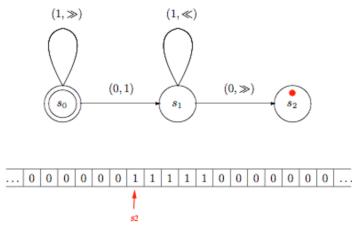
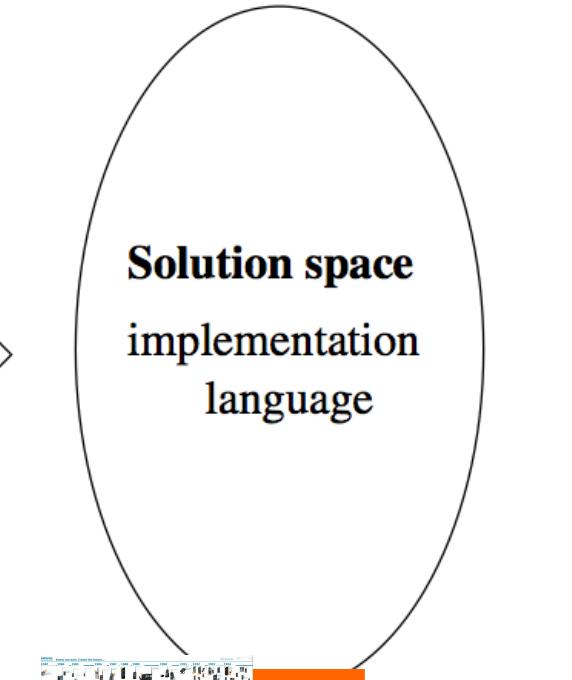
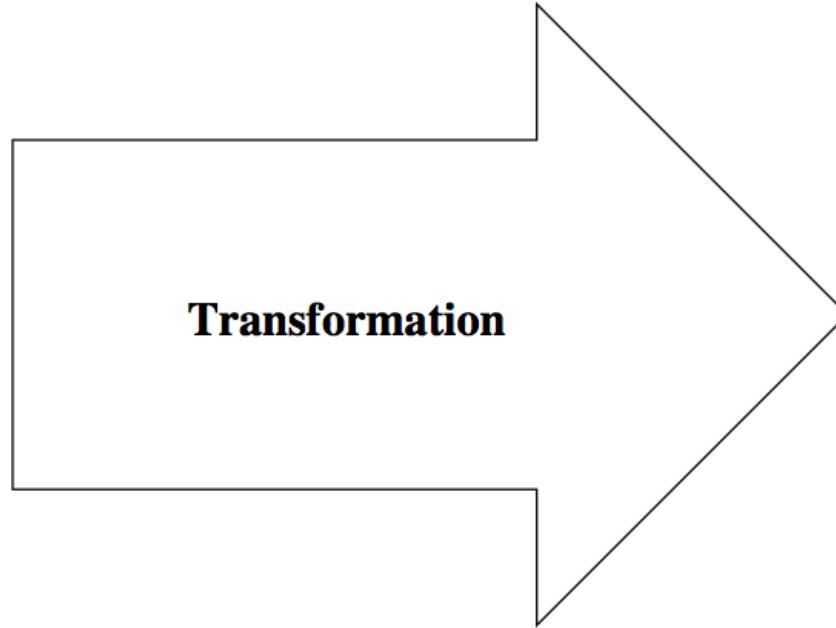
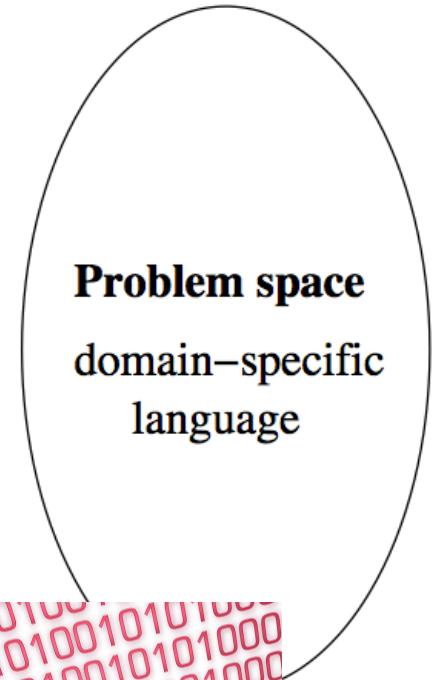
DSL,

Model,

Metamodel,

Summary

Abstraction Gap



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

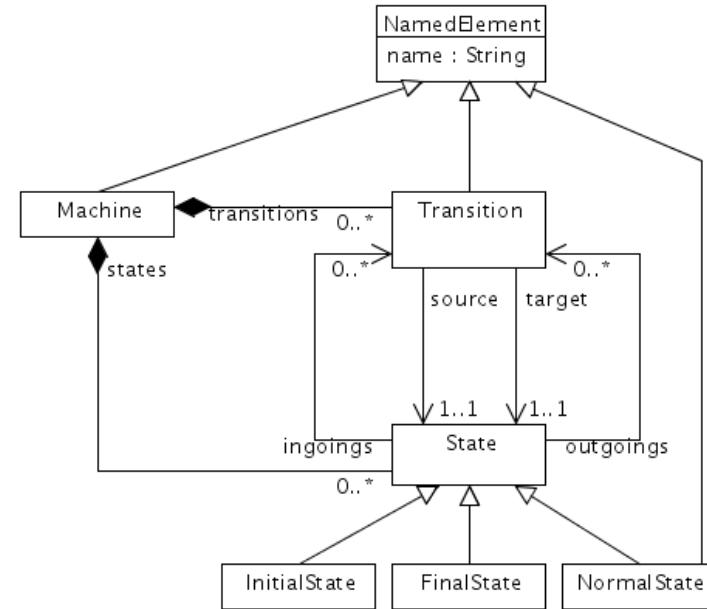
Generative approach

- 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



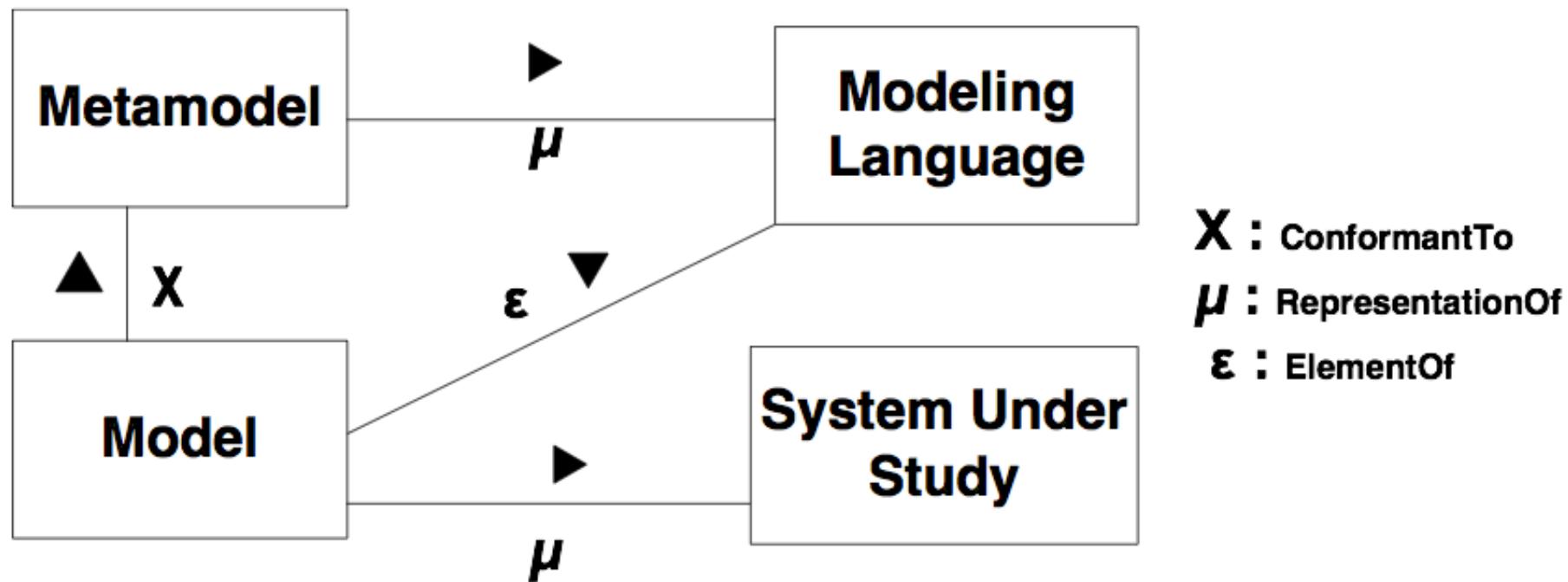
conforms To

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

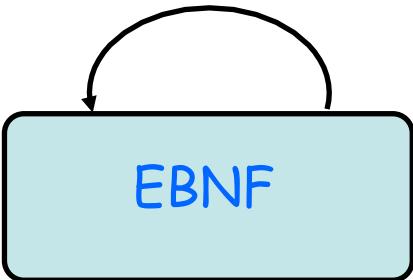
Source Code/Model

conforms To

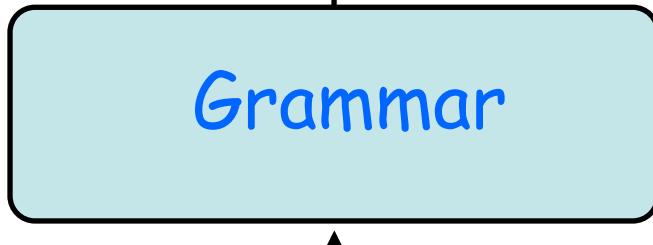
Model, Metamodel, Metametamodel, DSML



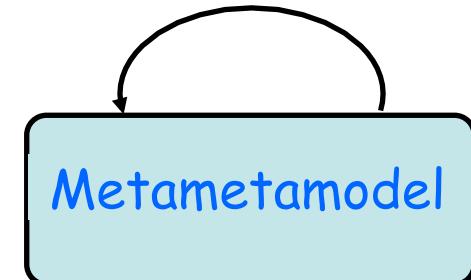
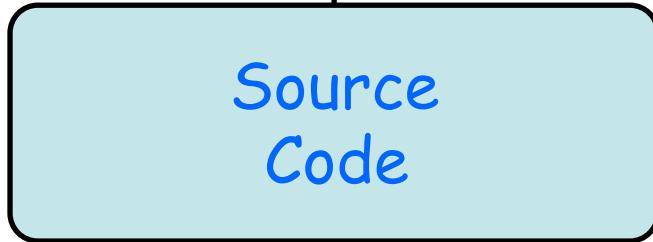
M³



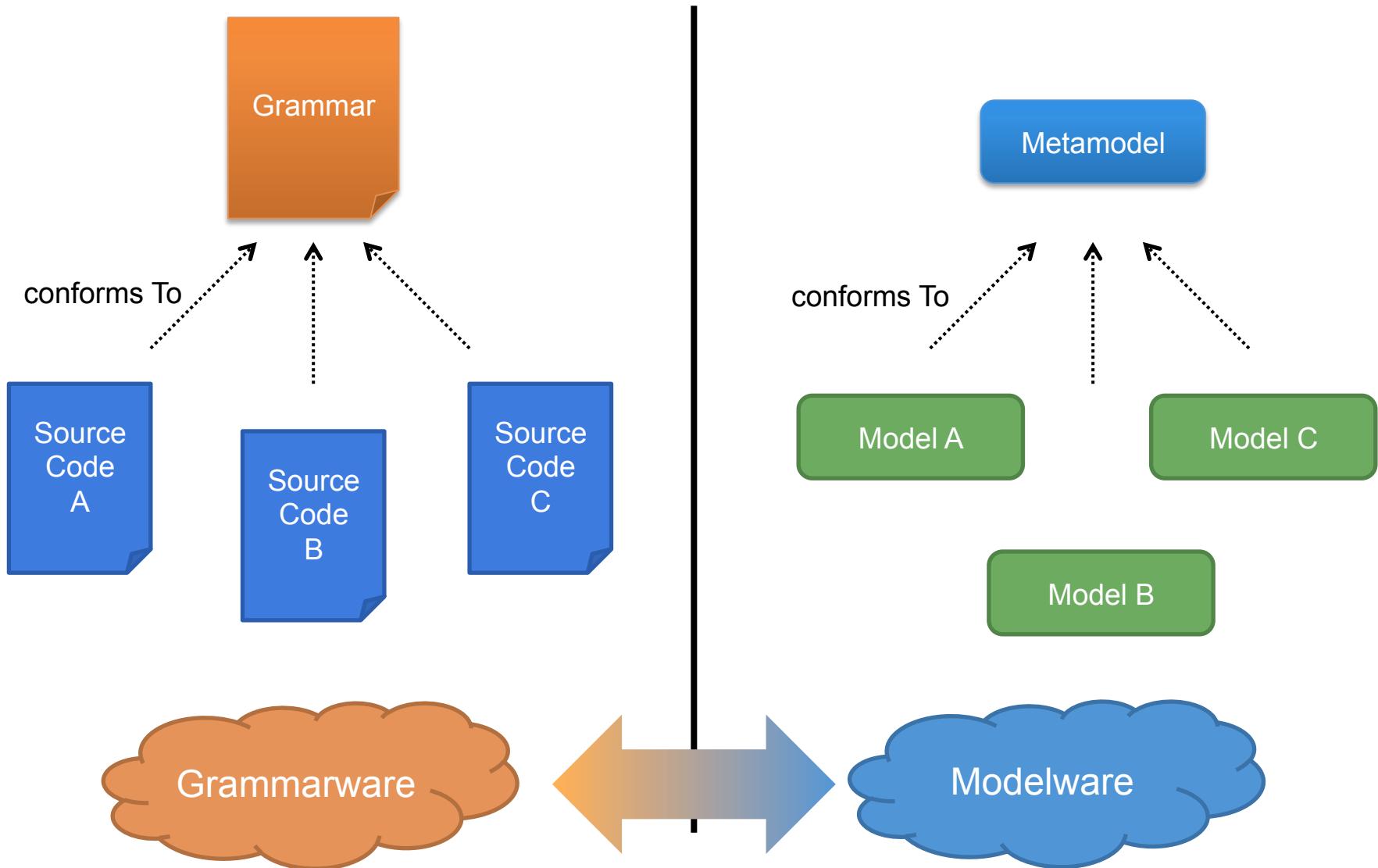
M²



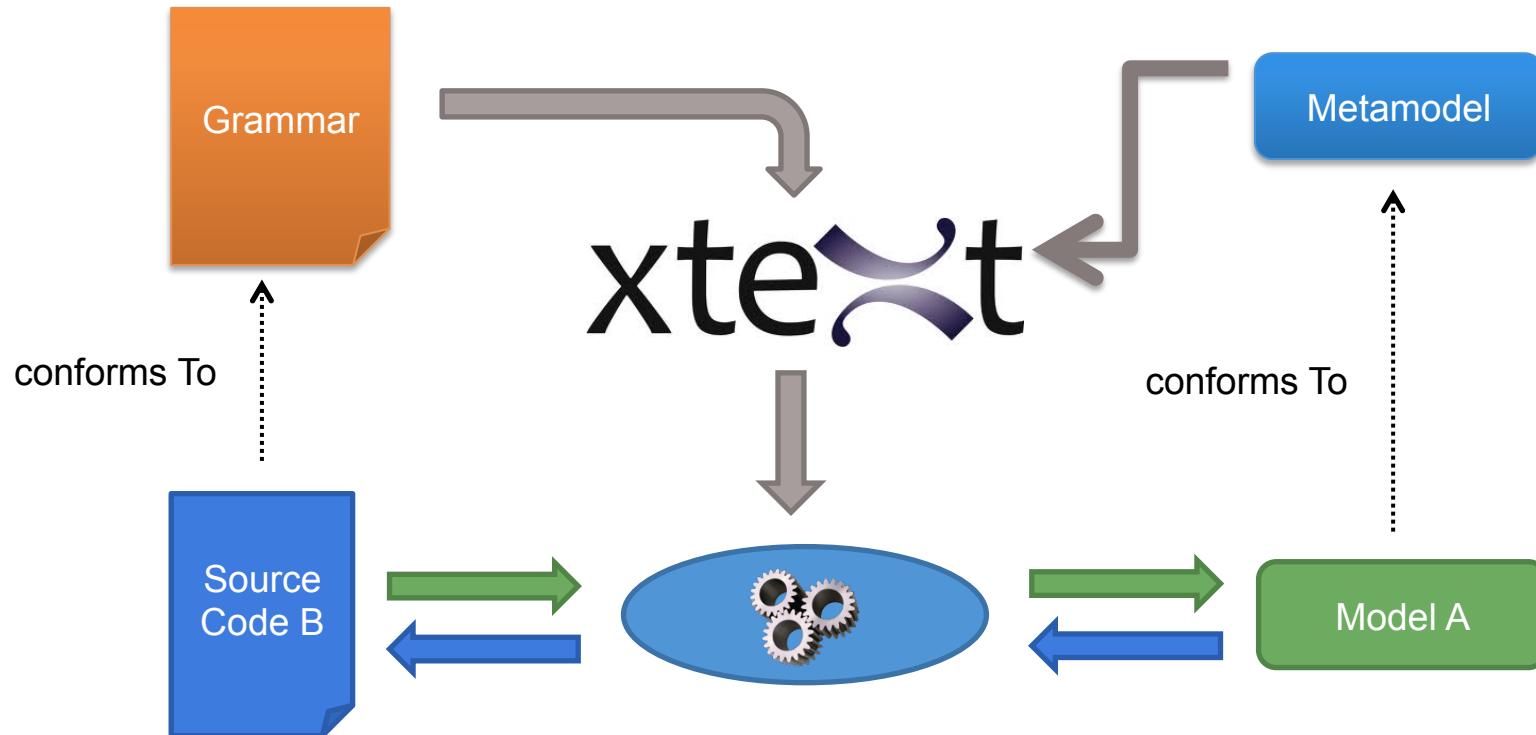
M¹



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}

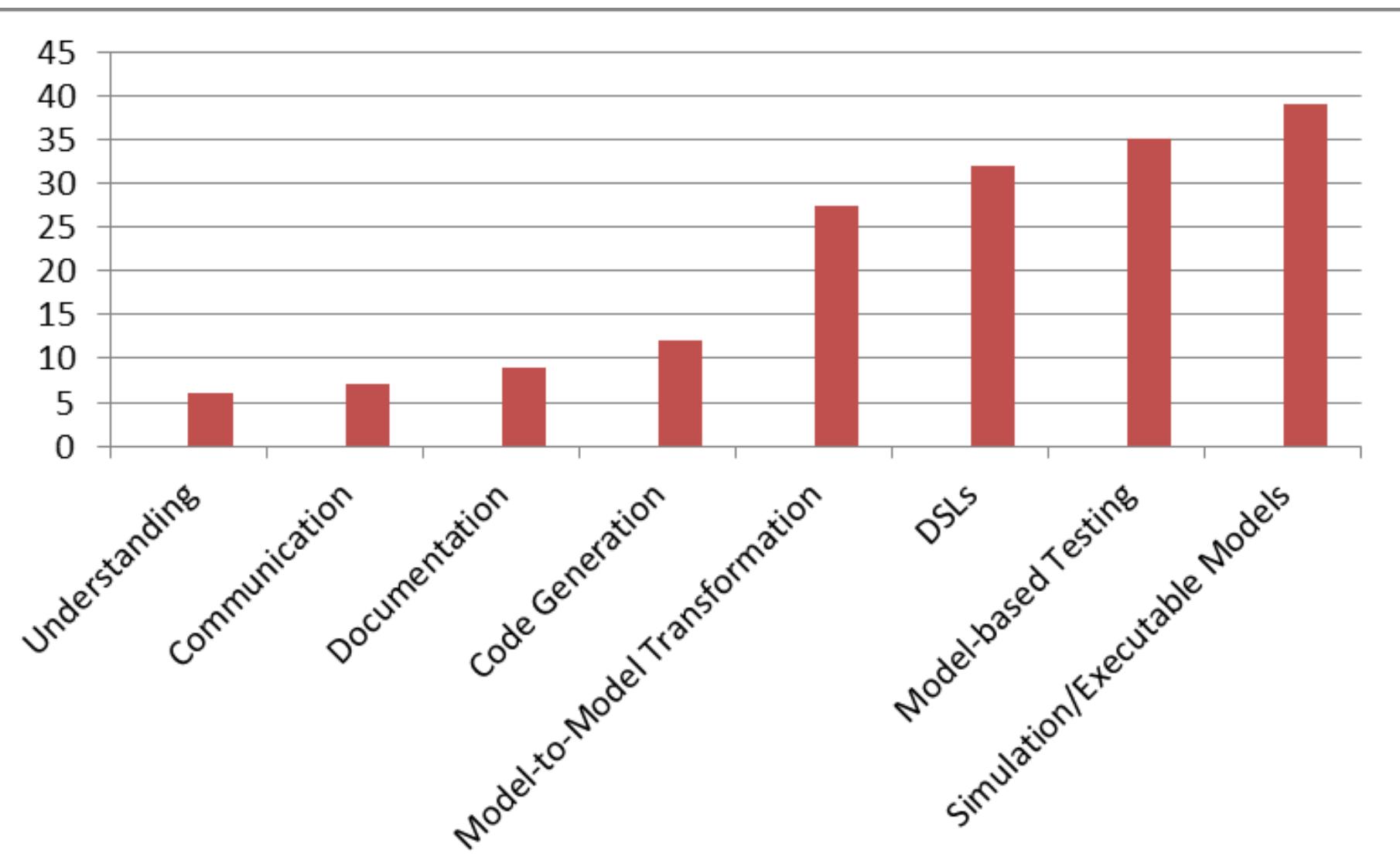
Jon Whittle
School of Computing and
Communications
Lancaster University, UK
+44 1524 510492

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

2011

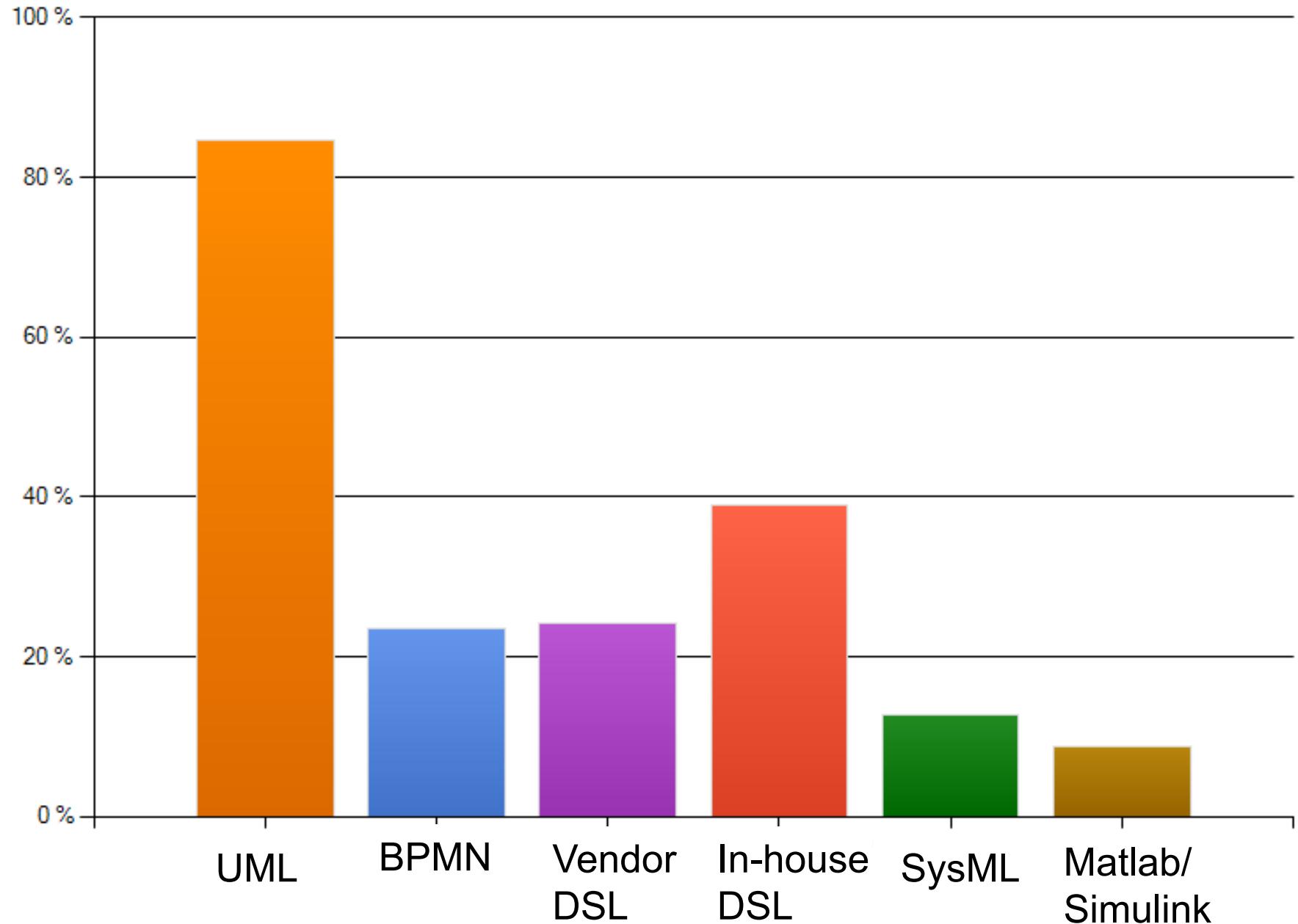
**« Domain-specific
languages are far more
prevalent than
anticipated »**

What are models used for?

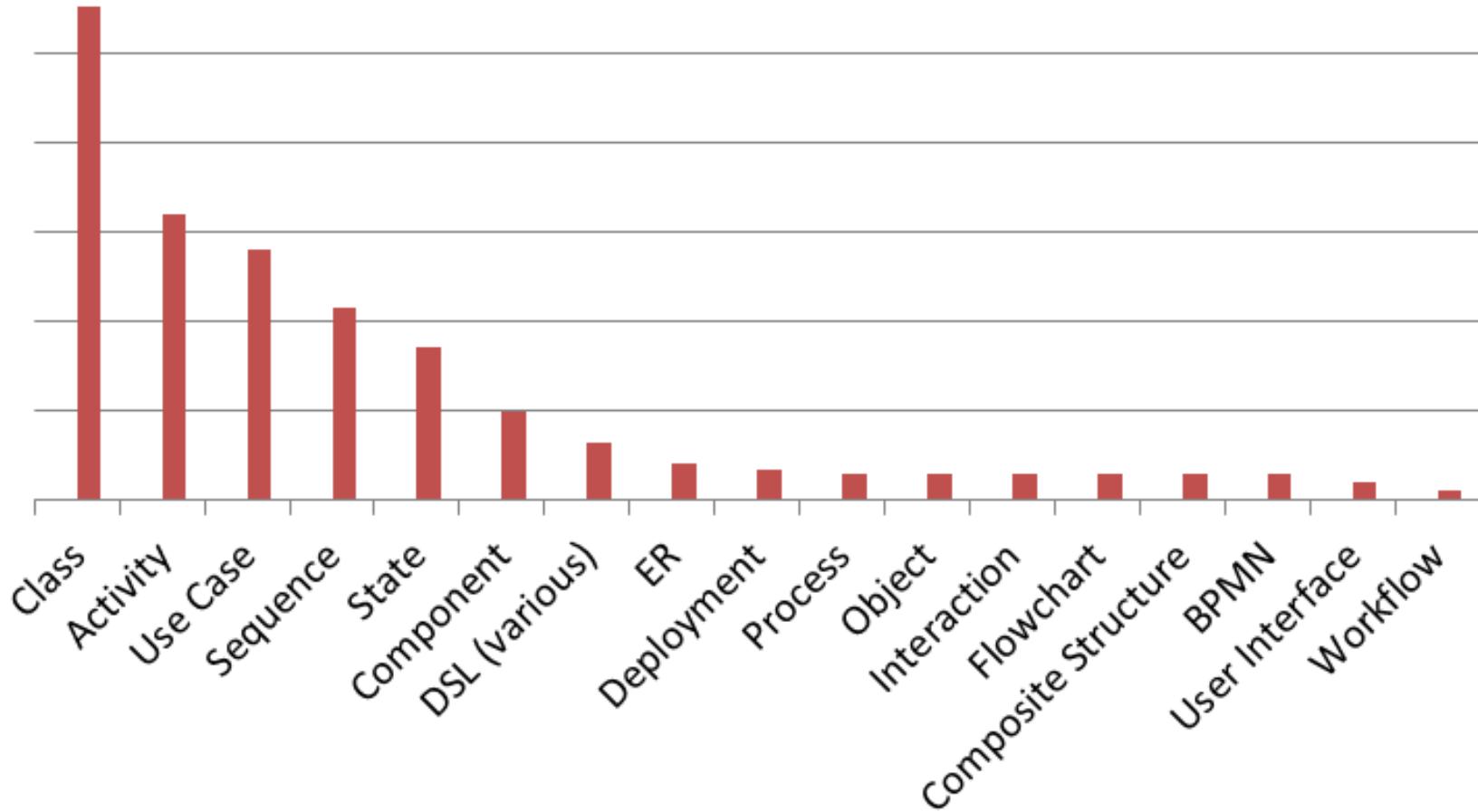


"Do not use" percentages for MDE activities

Which modeling languages do you use?



Which diagrams are used?



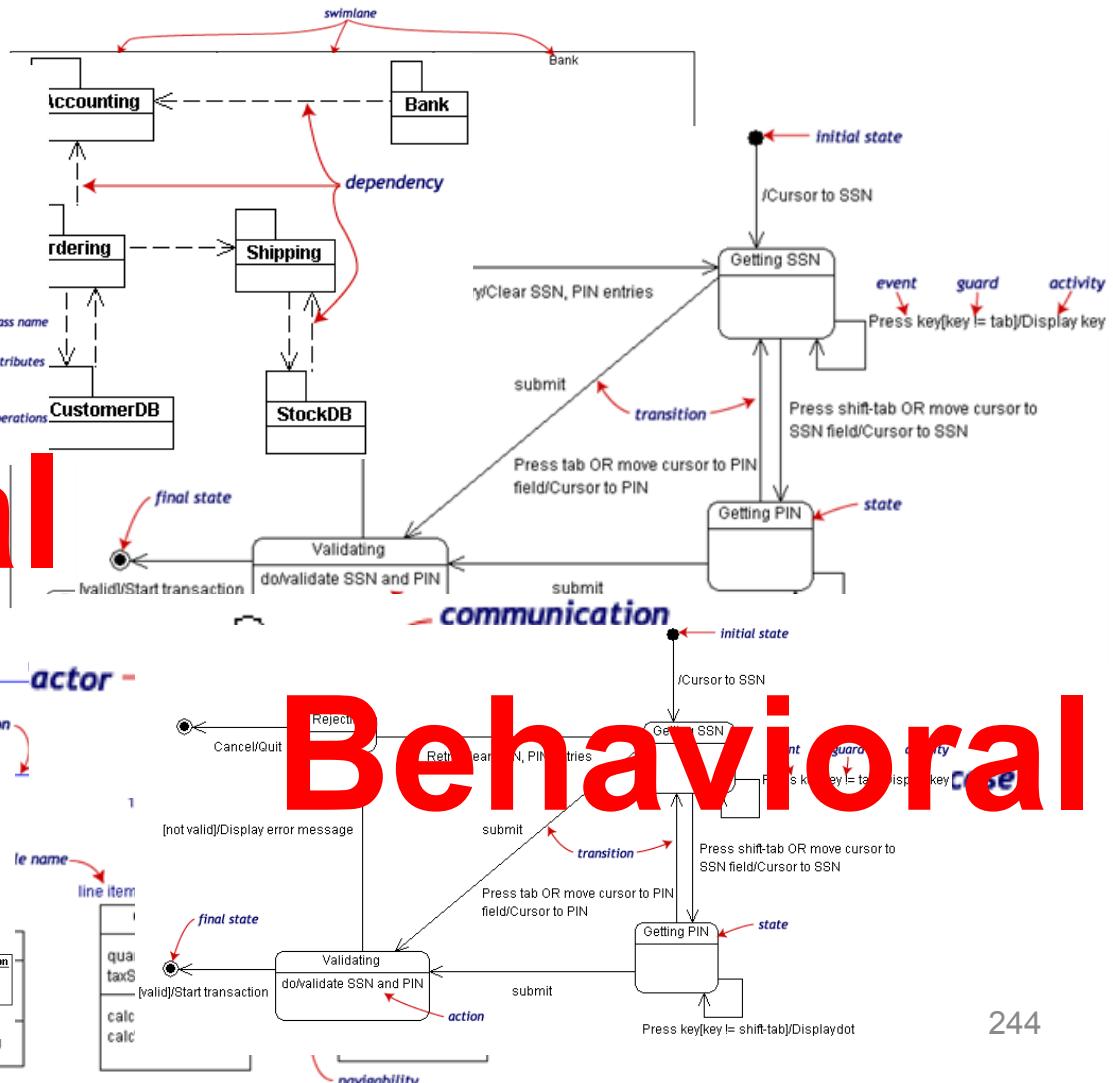
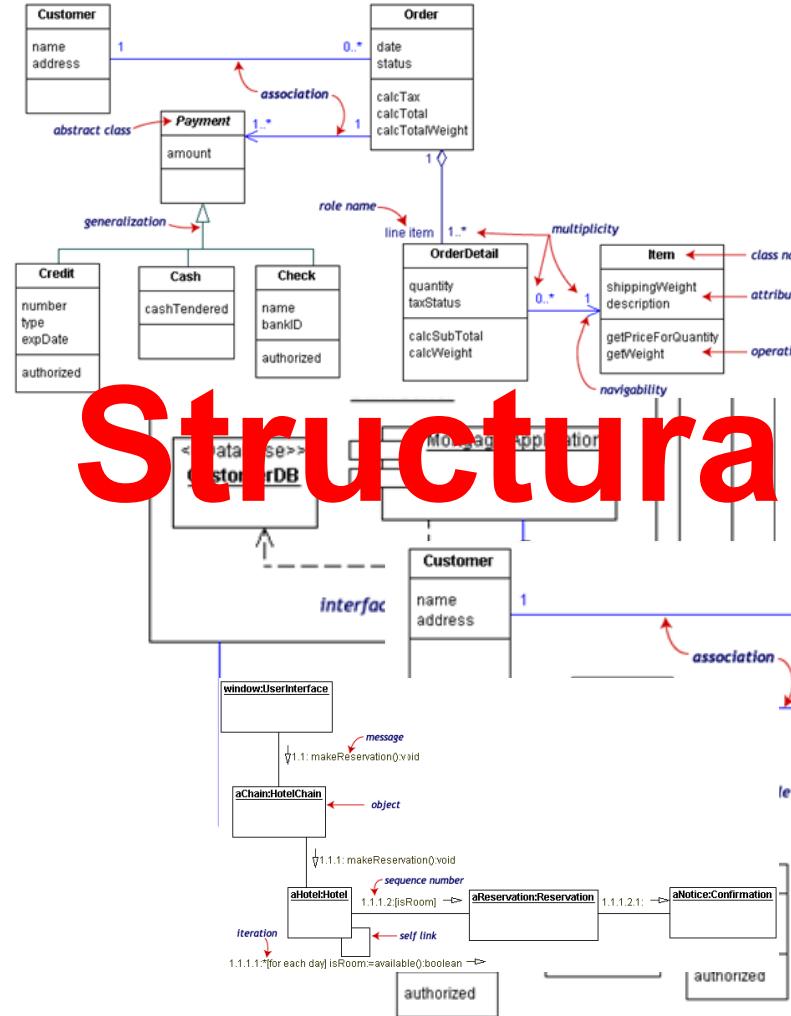
19 different diagram types are used regularly

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



The Definitive
ANTLR
Reference



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

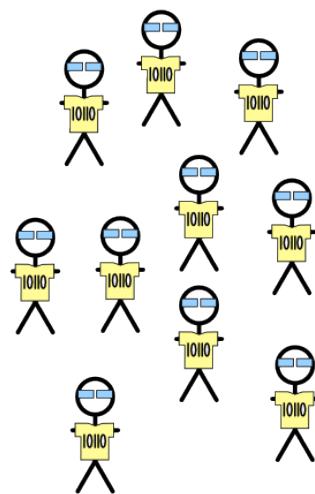
But my take away
message is NOT

That DSLs should be used
systematically, in every
situations

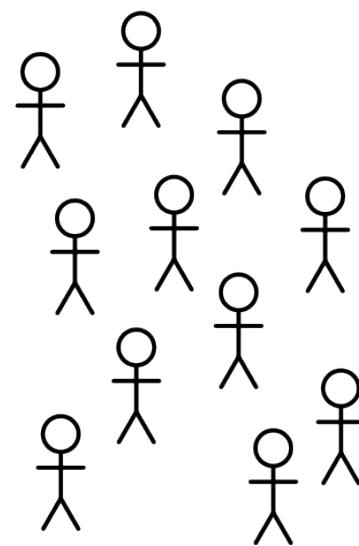
When Developing DSLs?

- 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

Actors

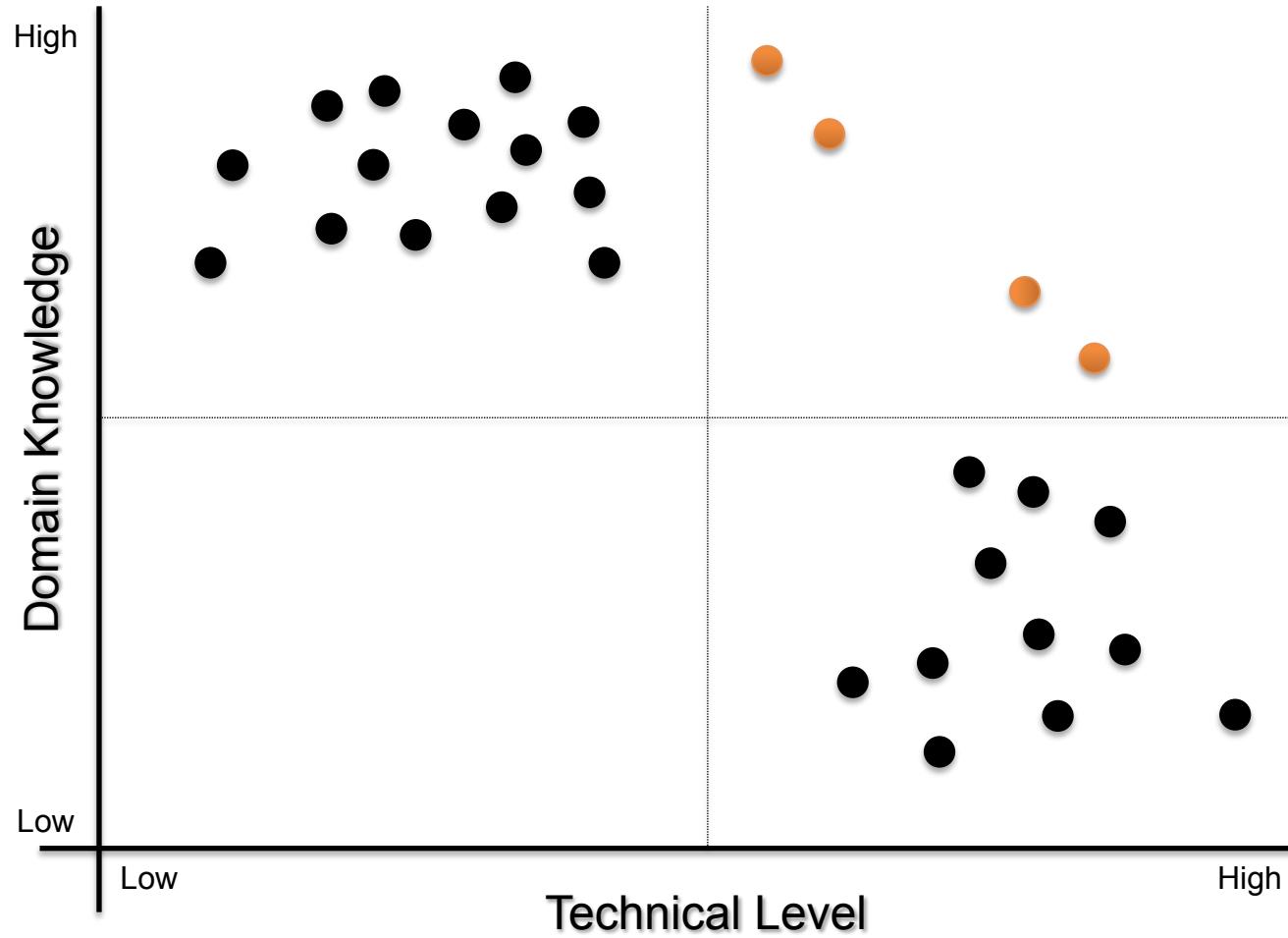


Developers



End-Users

Actors



Best Practices

Limit
Expressiveness

Viewpoints

Evolution

Learn from
GPLs

Support

Tooling

Worst Practices

- Initial conditions
 - Only Gurus allowed
 - Believe that only gurus can build languages 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

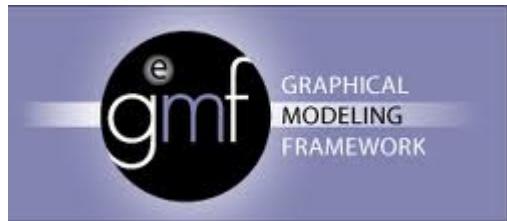
Worst Practices

- 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 ?

(see also resources and
lab sessions)

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

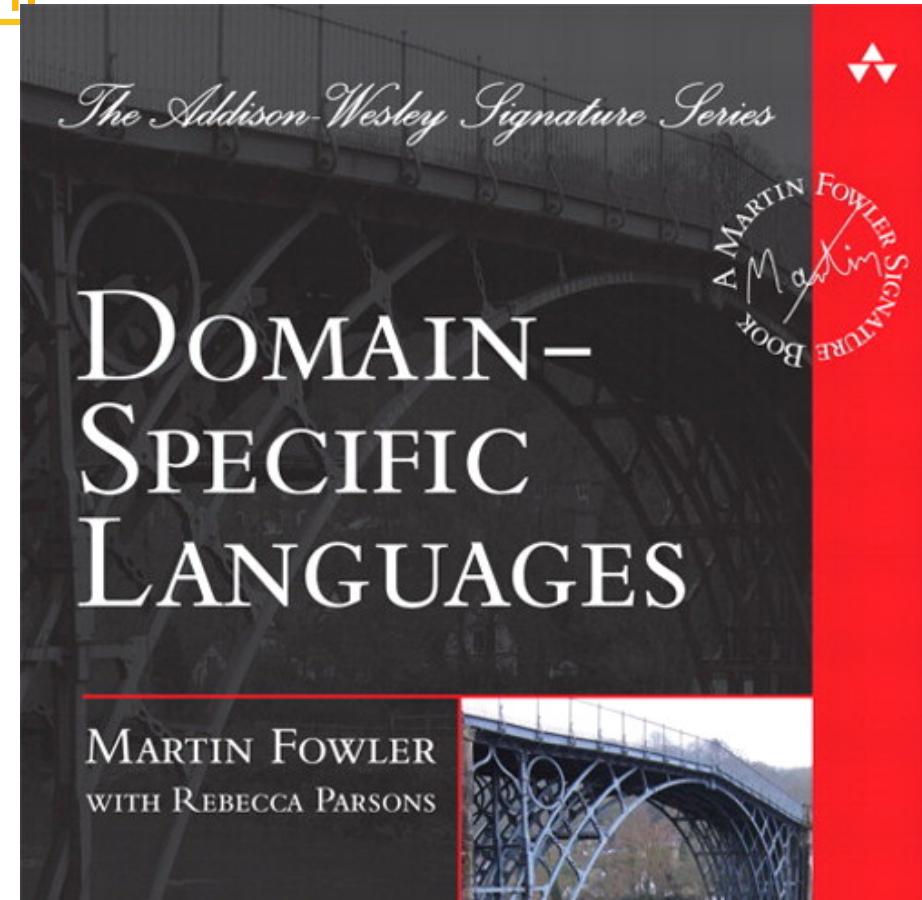


Empirical Assessment of MDE in Industry

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

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

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



References

- Martin Fowler. Domain Specific Languages. Addison-Wesley Professional, 2010.
- Markus Voelter et al. “DSL Engineering: Designing, Implementing and Using Domain-Specific Languages.” dslbook.org, 2013.
- Sven Efftinge, Moritz Eysholdt, Jan Köhnlein, Sebastian Zarnekow, Robert von Massow, Wilhelm Hasselbring, and Michael Hanus. Xbase: Implementing domain-specific languages for java. GPCE ’12
- Steven Kelly and Risto Pohjonen. Worst practices for domain-specific modeling. IEEE Software, 26(4):22–29, 2009.
- Lennart C.L. Kats and Eelco Visser. The spoofax language workbench: Rules for declarative specification of languages and ides OOPSLA’10

References

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