Domain-Specific Languages

Mathieu Acher
Maître de Conférences
mathieu.acher@irisa.fr

UNIVERSITÉ DE RENNES 1

istic Informatique Électronique

Material

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

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)

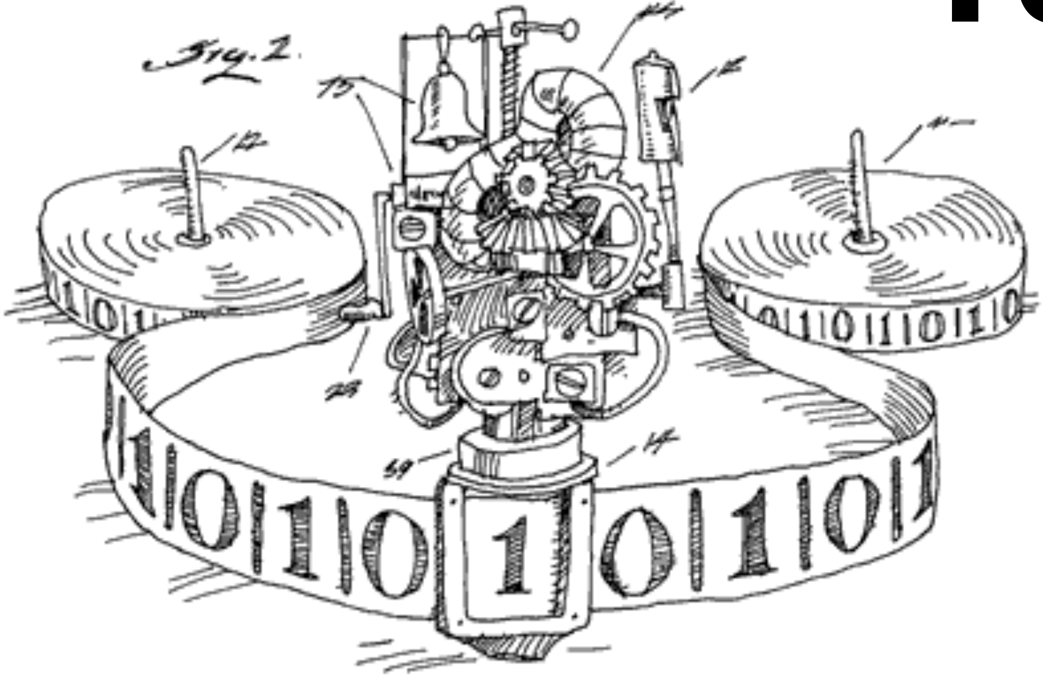
- Models and Languages
  - Perhaps a more concrete way to see models, metamodels and MDE (IDM in french)

What are DSLs

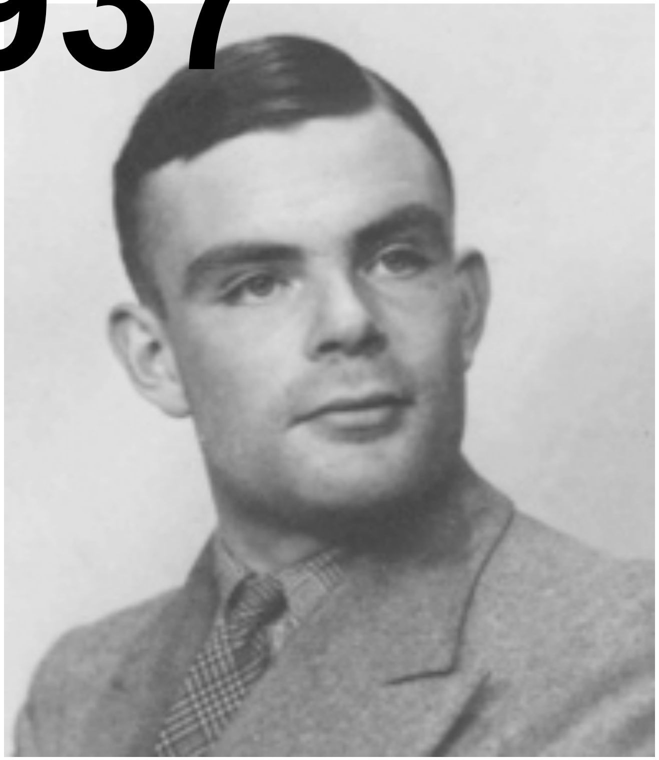
Where are DSLs

Why DSLs (will) matter

The (Hi)Story of Software Engineering / Computer Science



1937

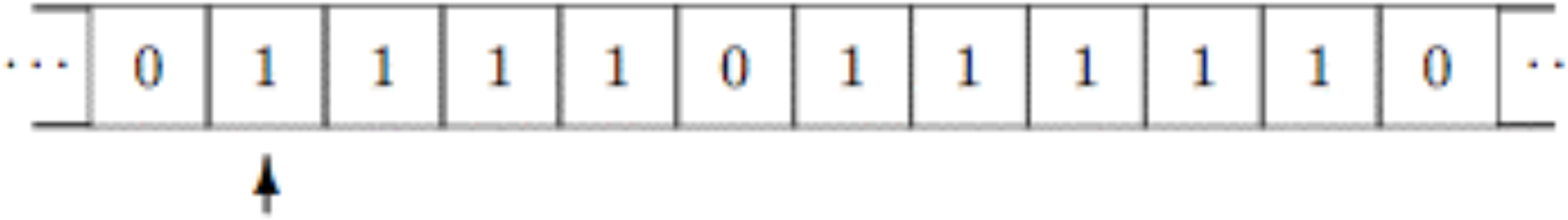


Turing Machine

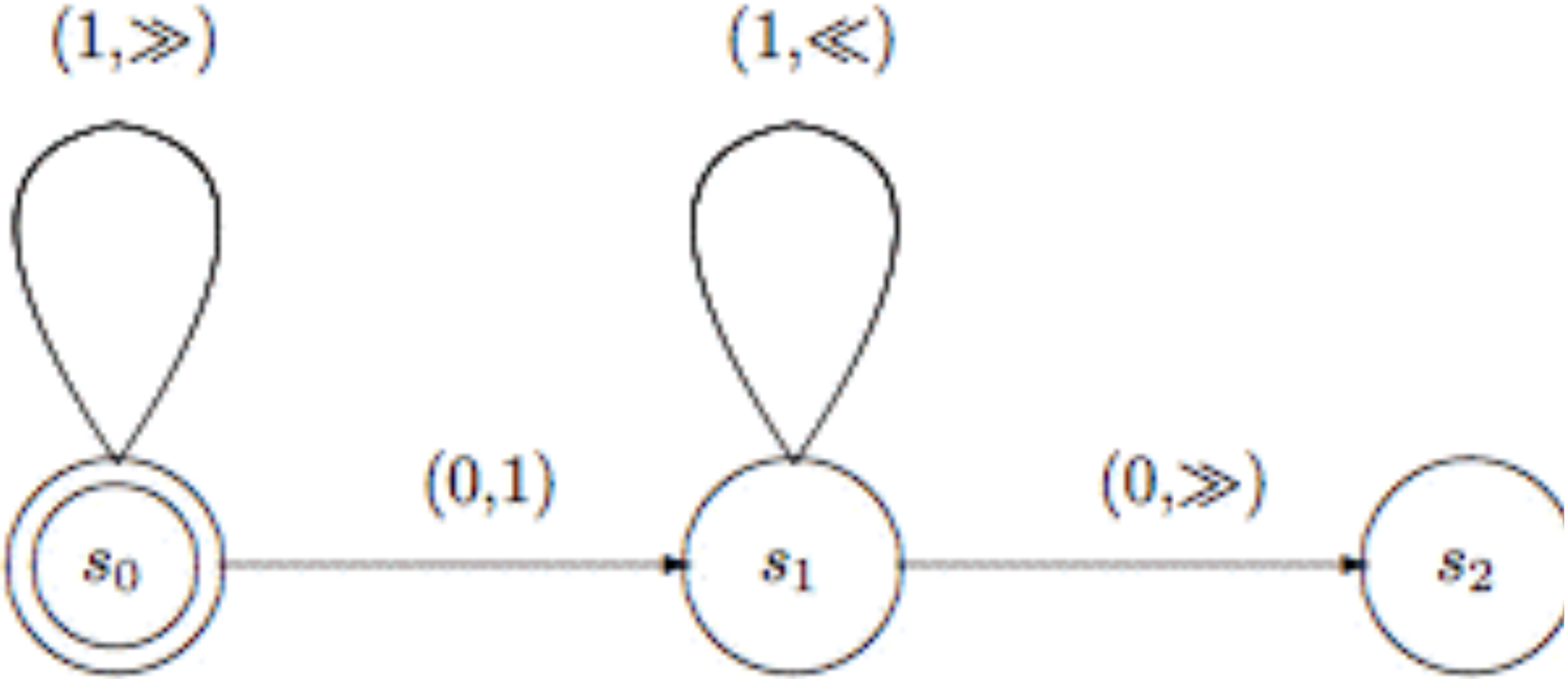
- Infinite tape divided into Cells (0 or 1)
- Read-Write Head
- Transitition rules

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

$$\langle\, State_{\text{current}},\, Symbol,\, State_{\text{next}},\, Action \,\rangle$$

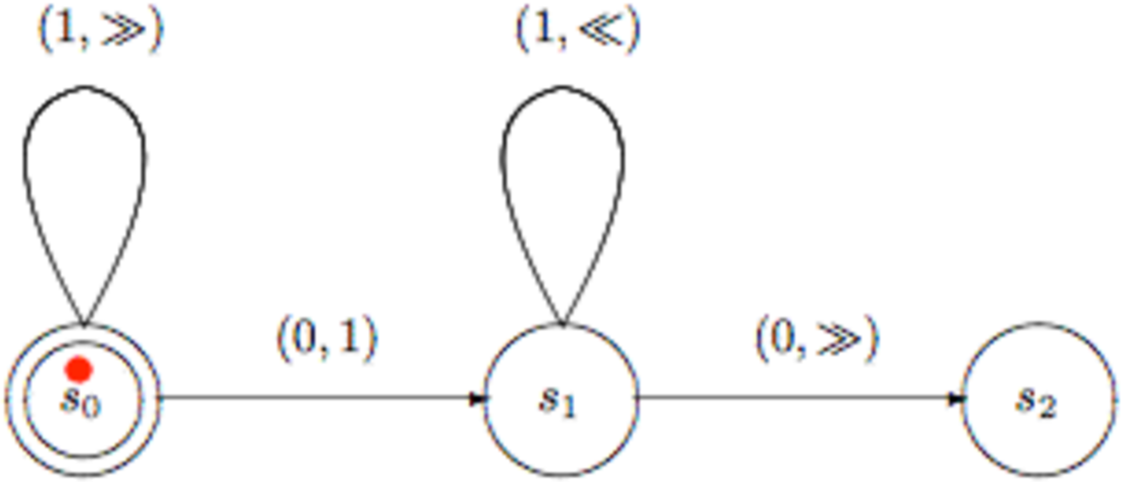


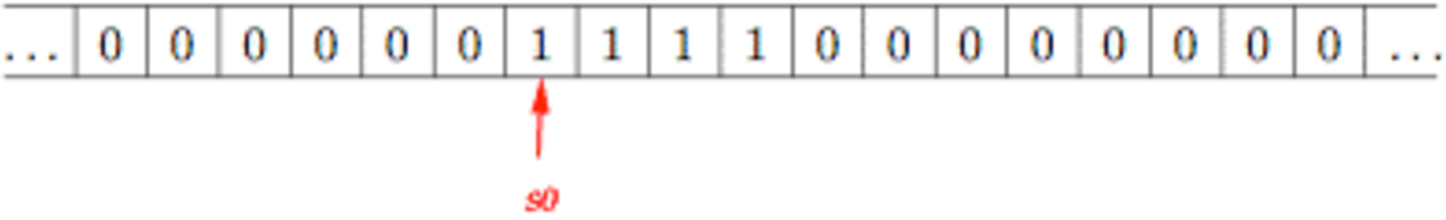
Turing Machine
~ kind of state machine

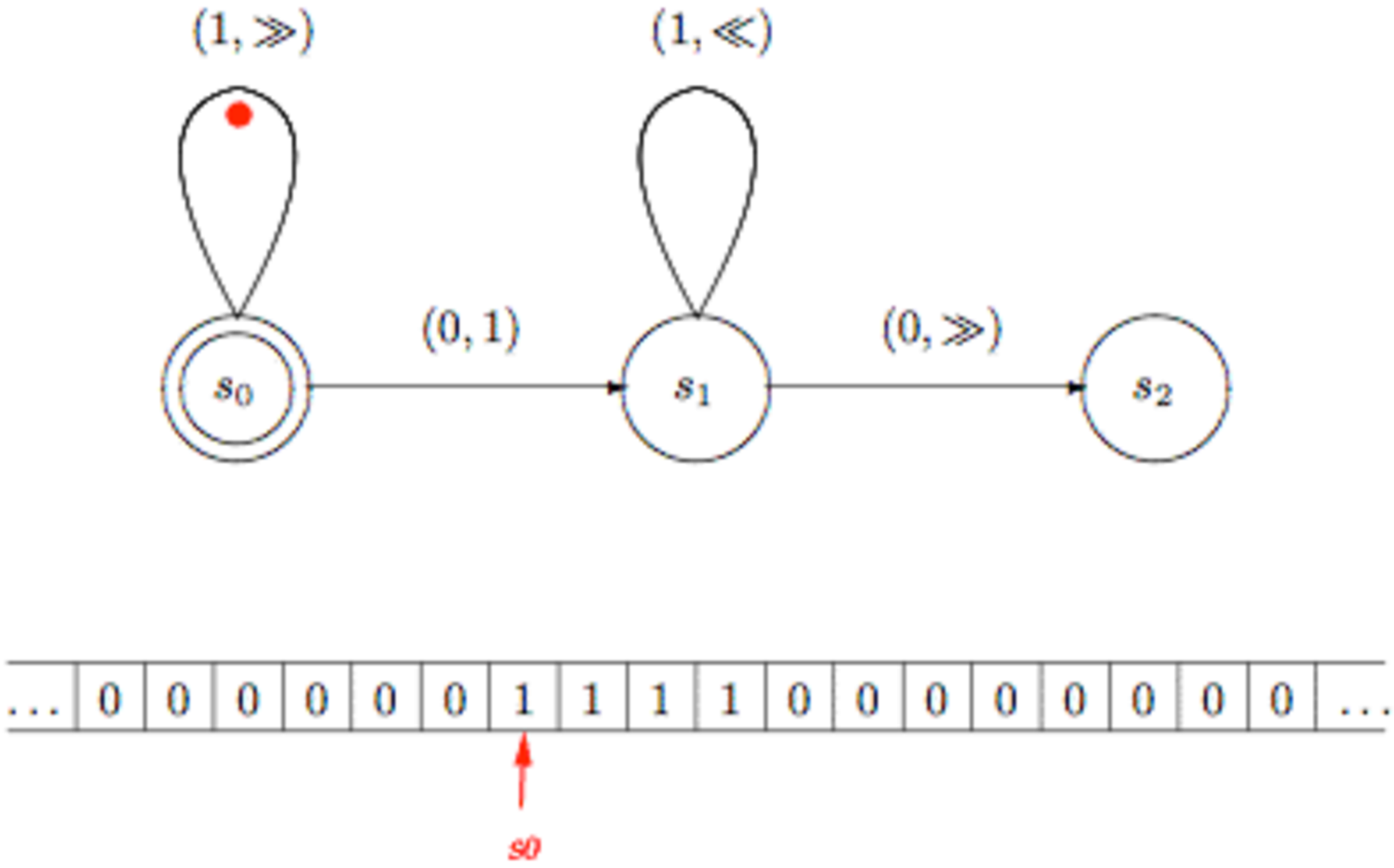


Successor (add-one) function

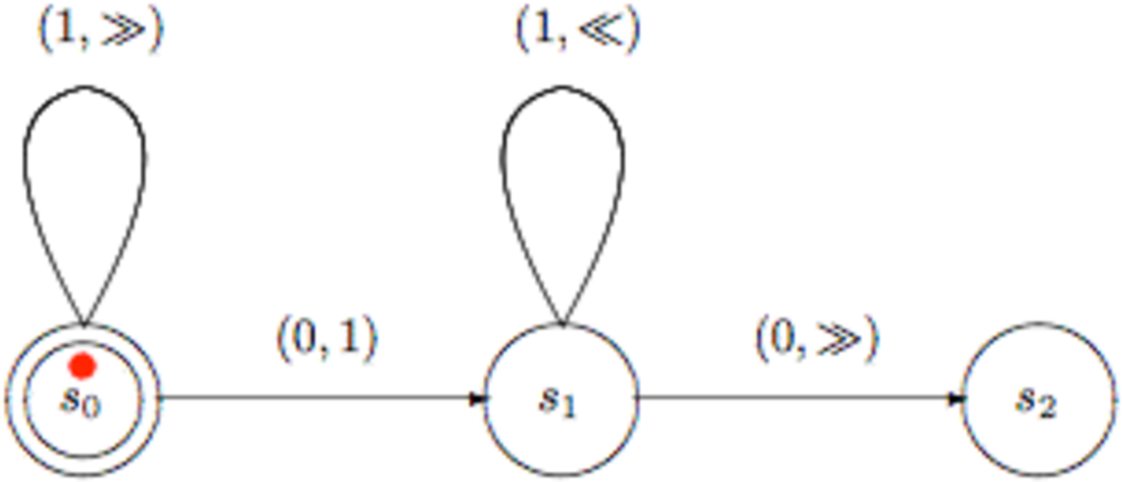
assuming that number n as a block of n+1 copies of the symbol ‘1’ on the tape (here, n=3)





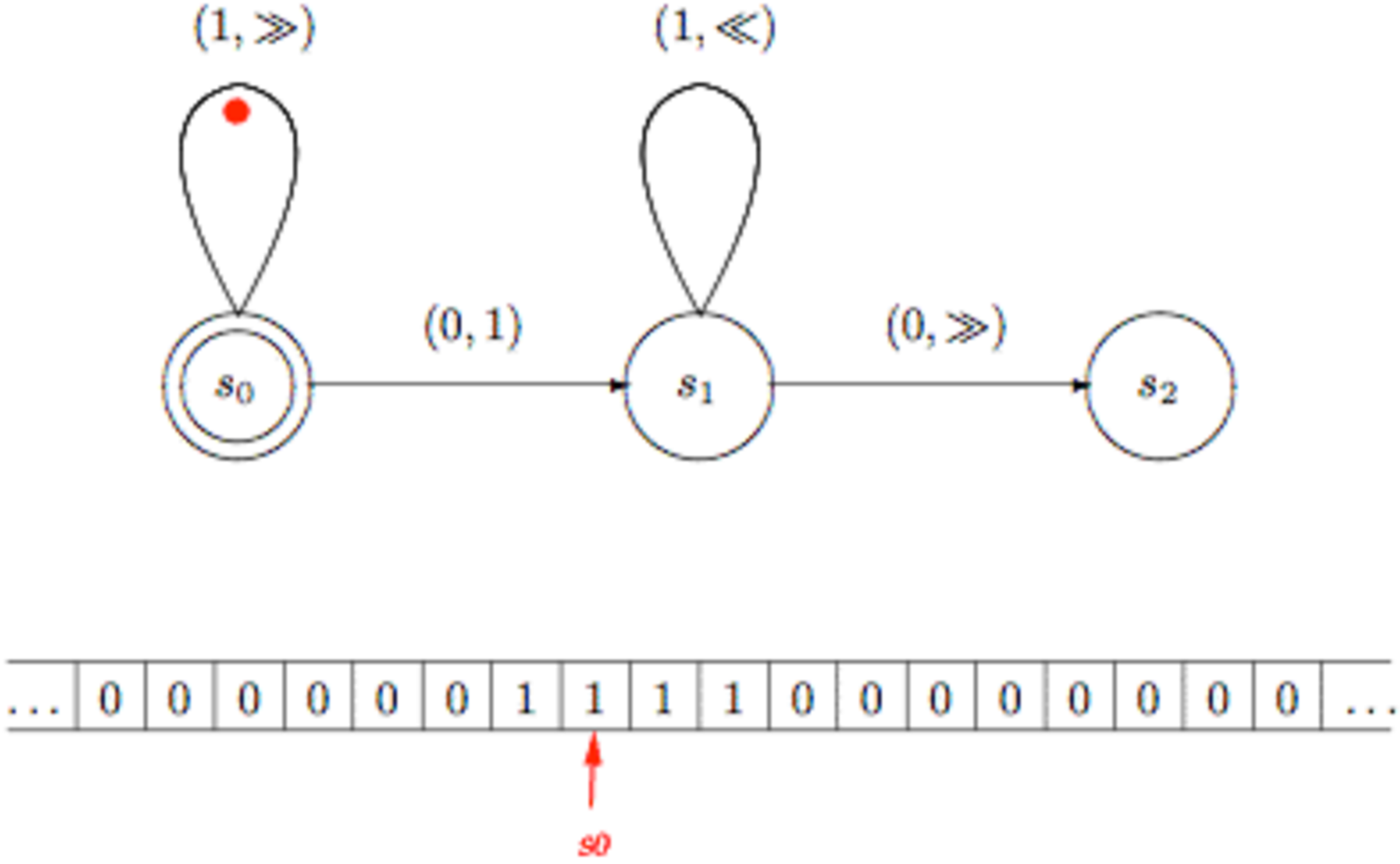


…	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	…

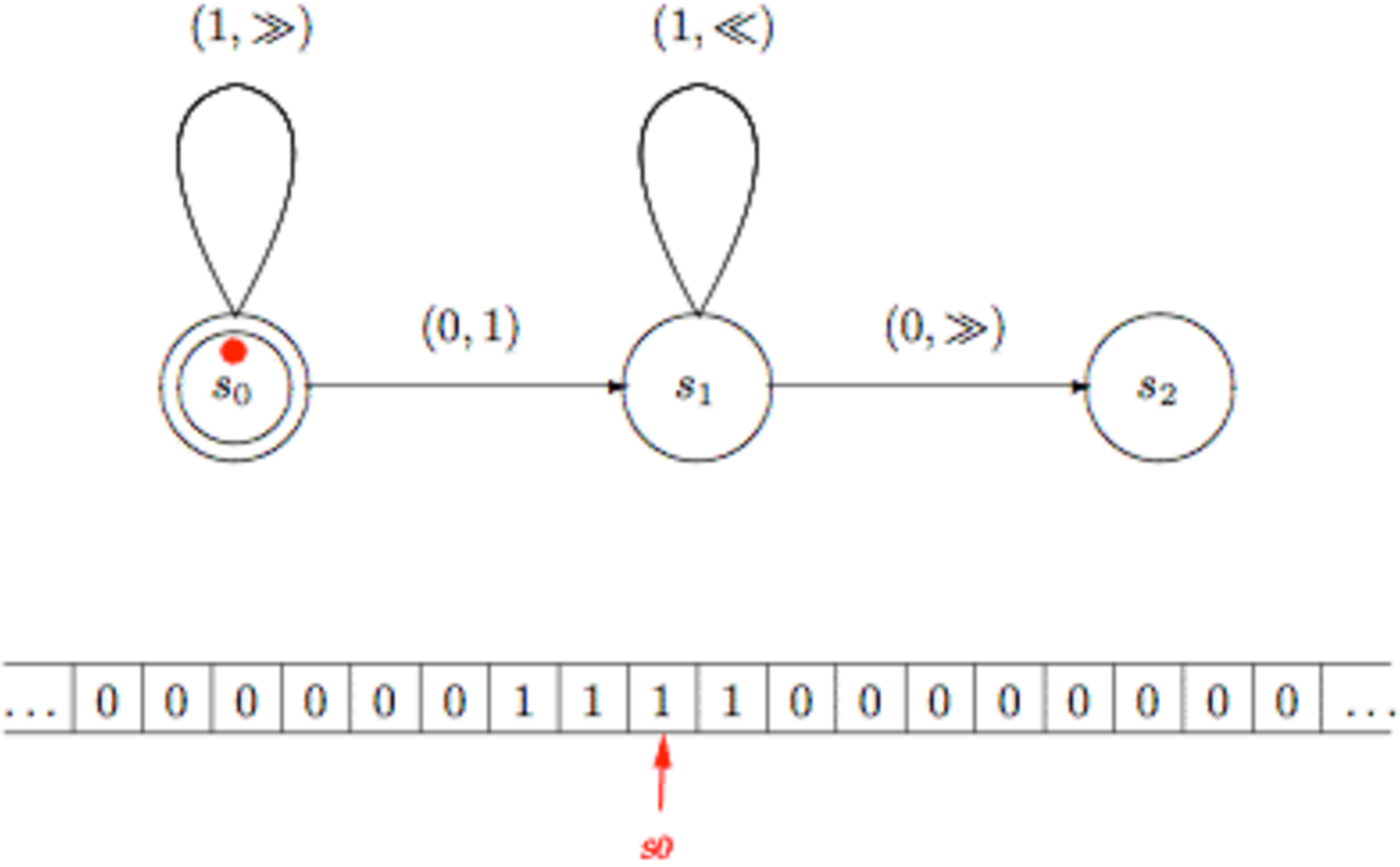


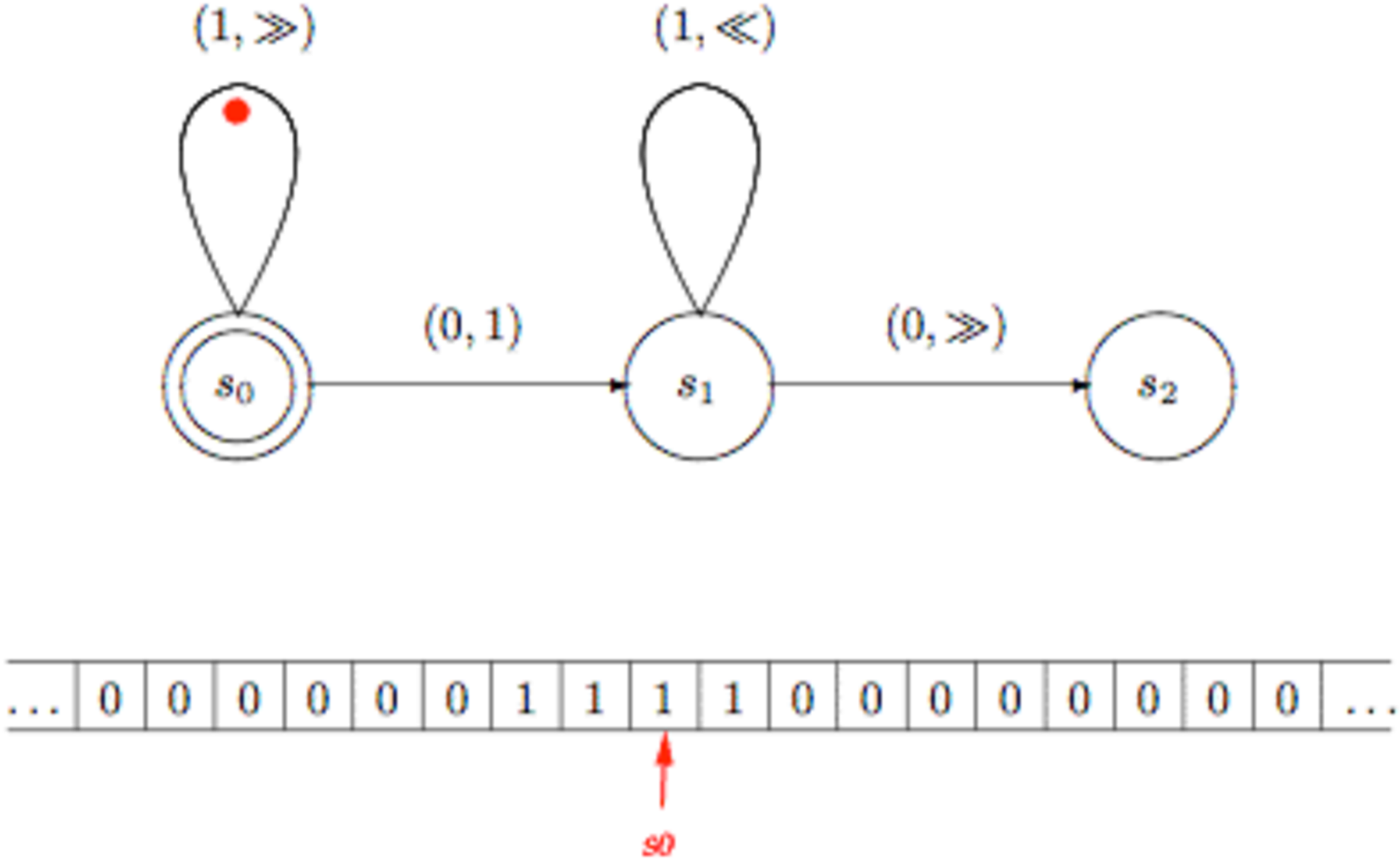
...	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	...

s0

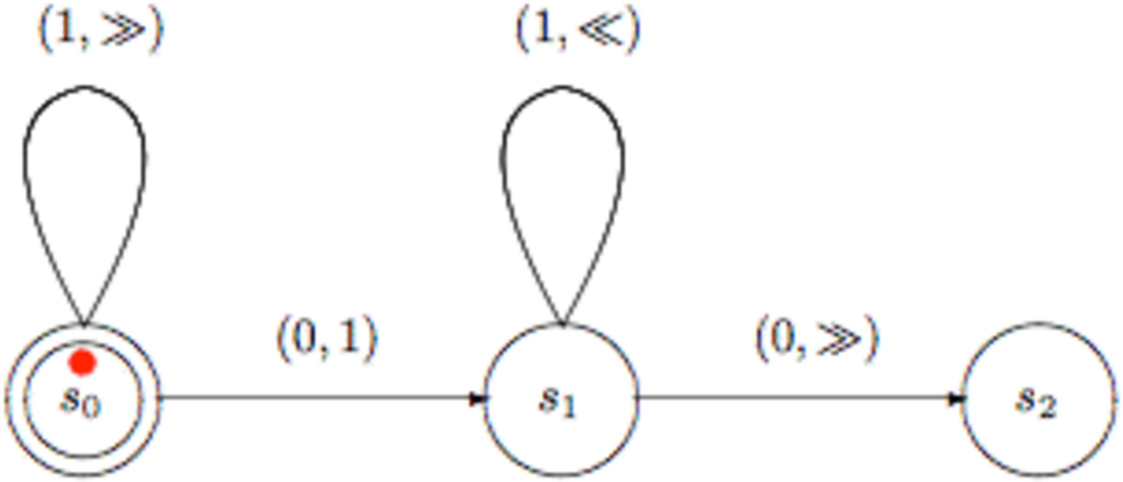


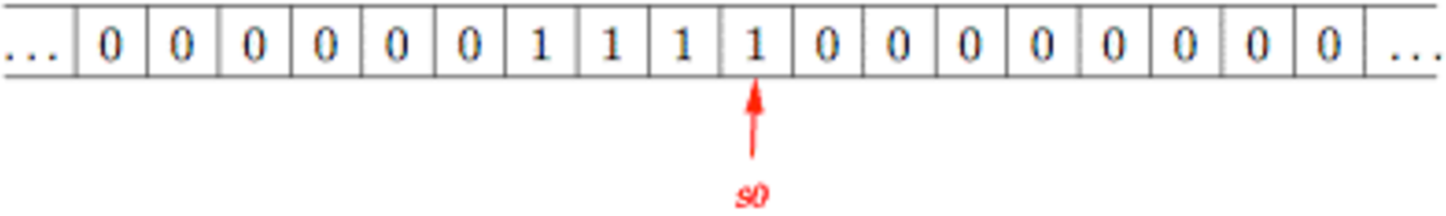
…	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	…

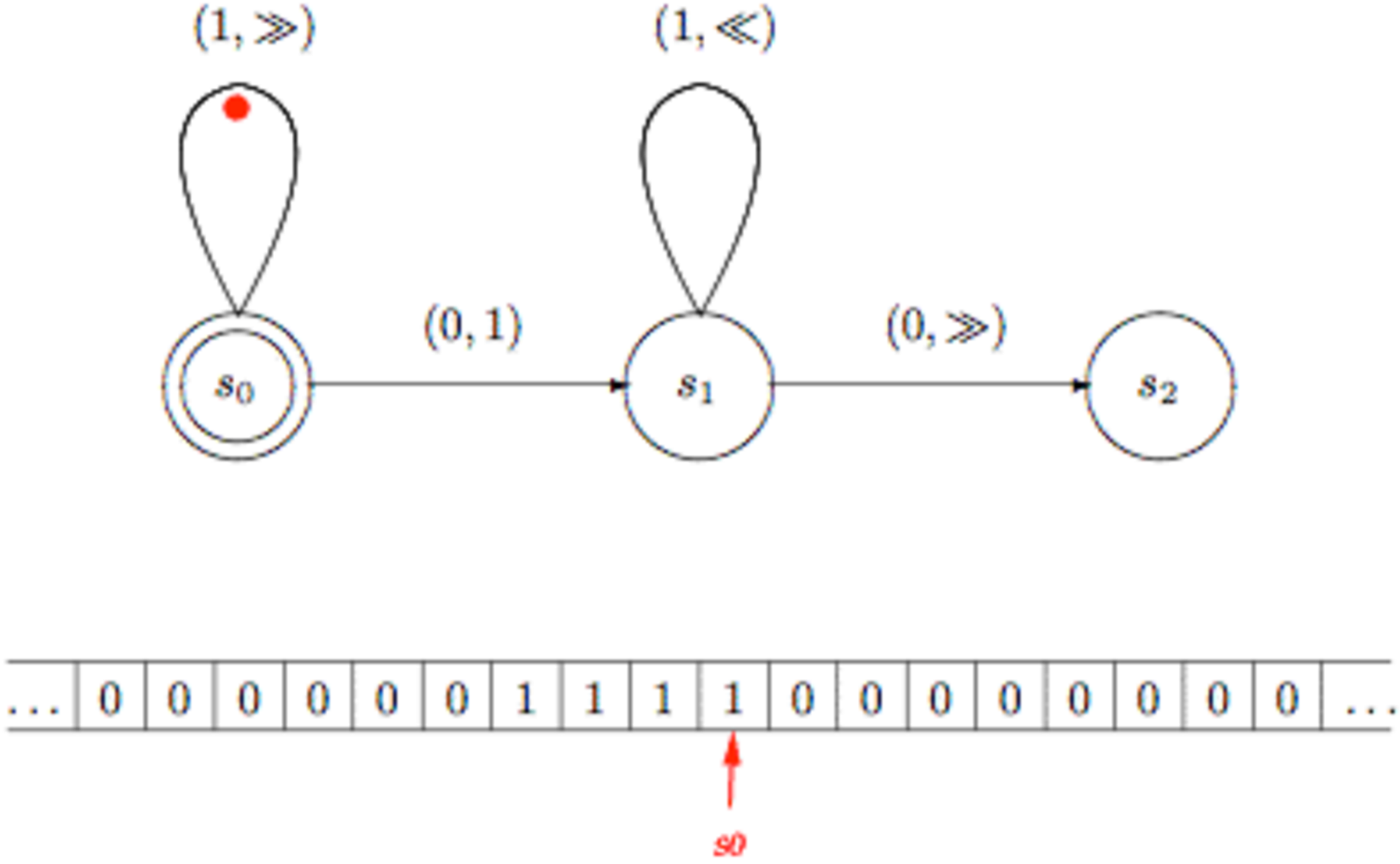

(1, ≫)
(1, ≪)
(0, 1)
(0, ≫)
s_0
s_1
s_2
… 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 …
s0



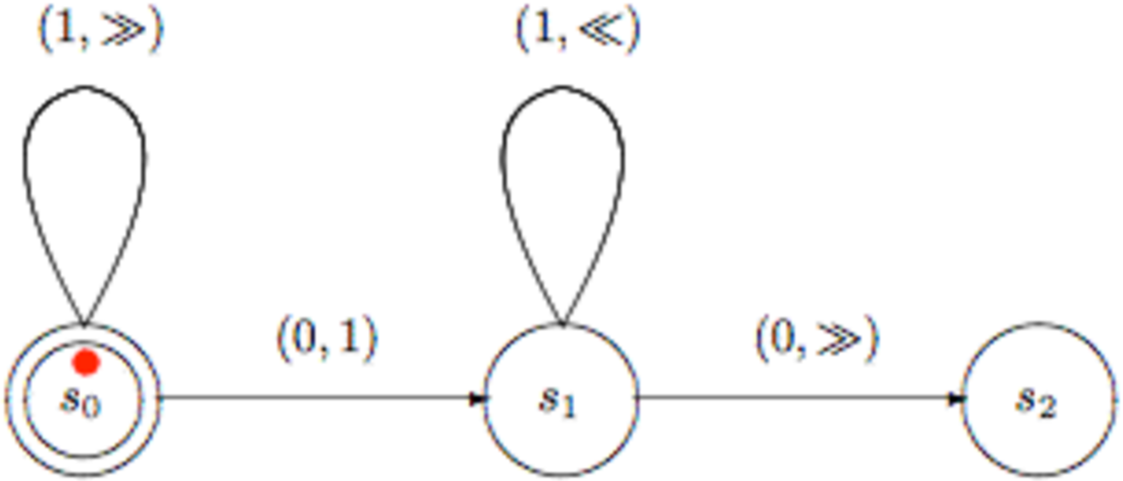
…	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	…


(1, ≫)
(1, ≪)
(0, 1)
(0, ≫)
s_0
s_1
s_2


…
0
0
0
0
0
0
1
1
1
1
0
0
0
0
0
0
0
0
…
s0

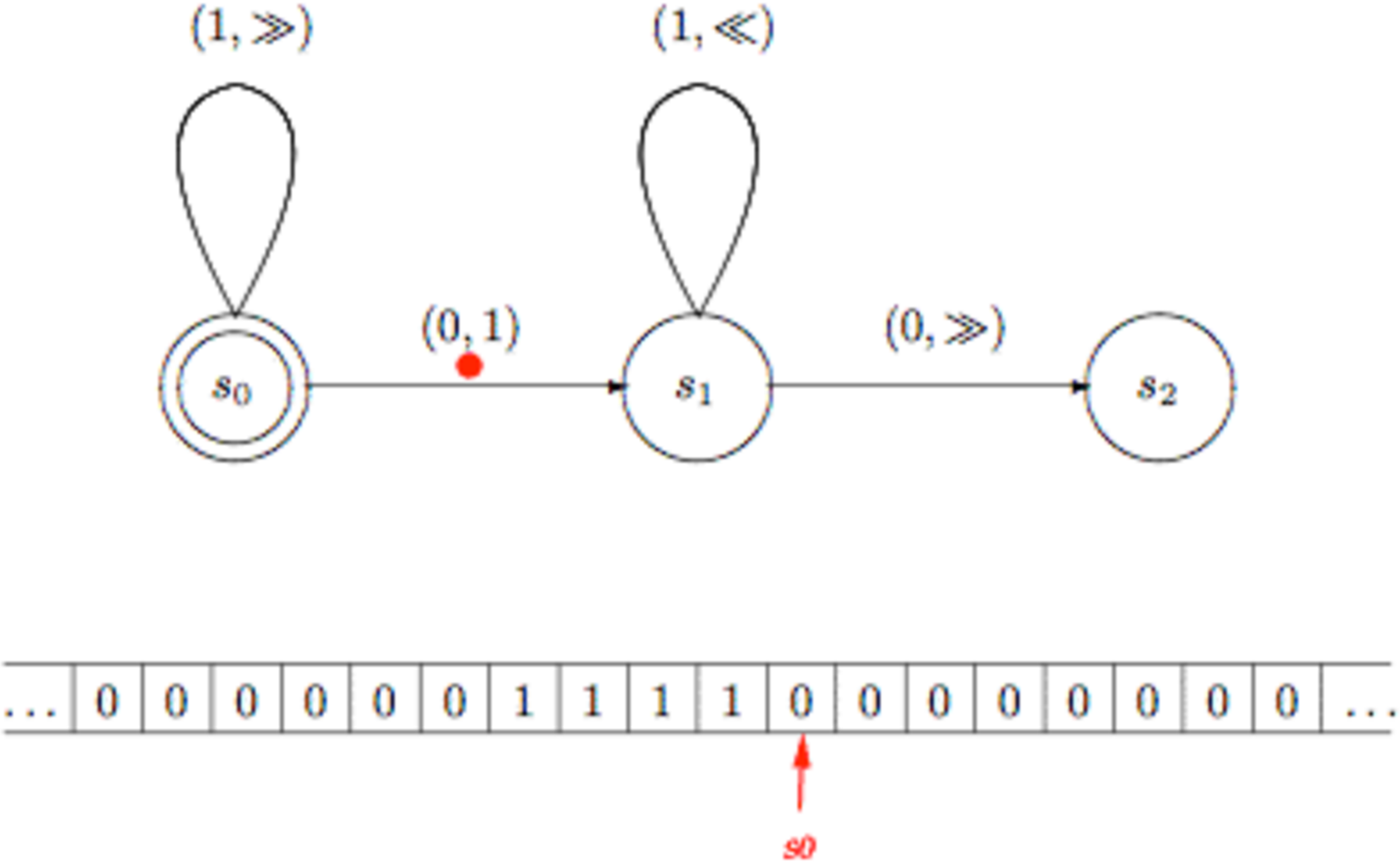


…	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	…

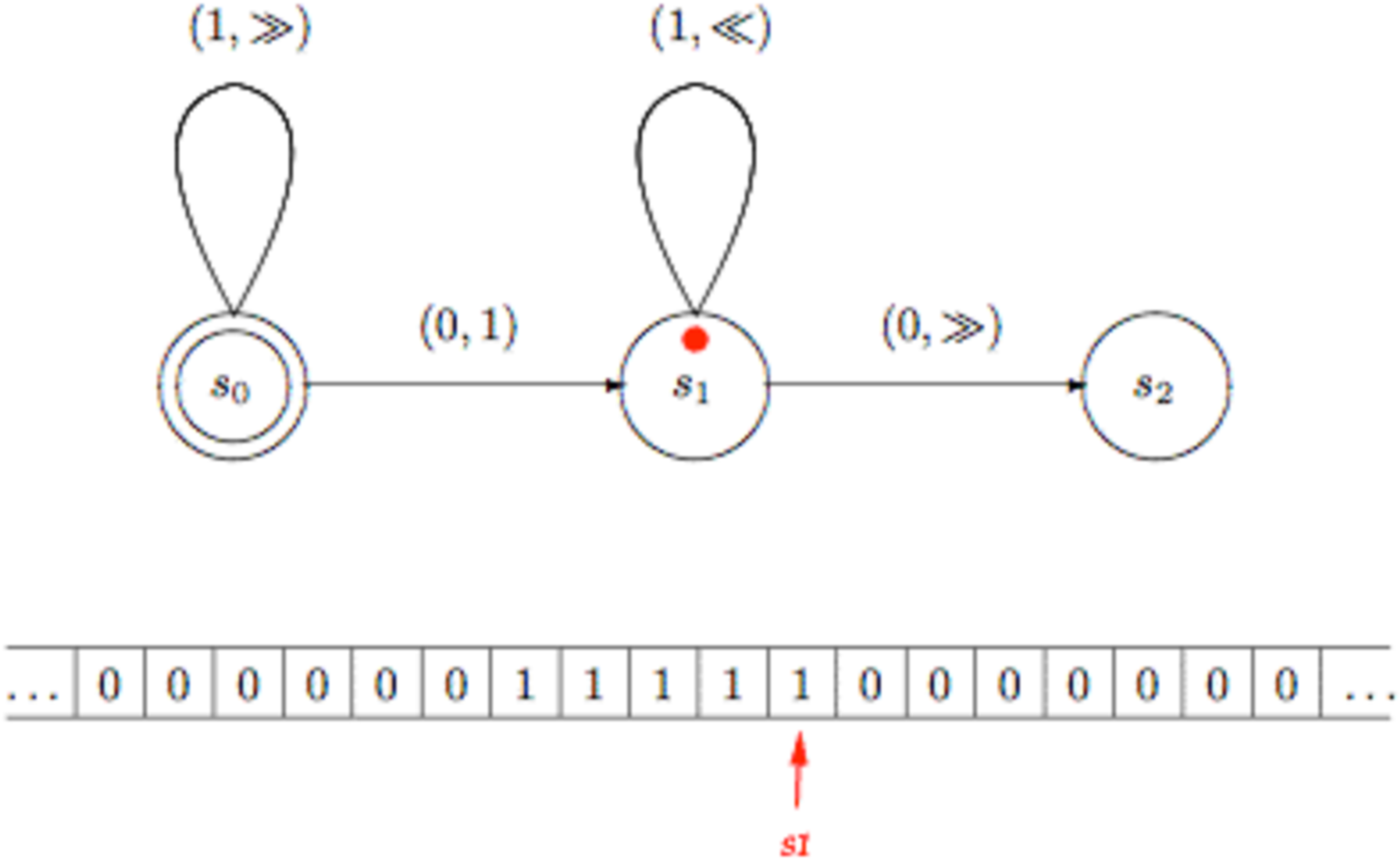


…	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	…

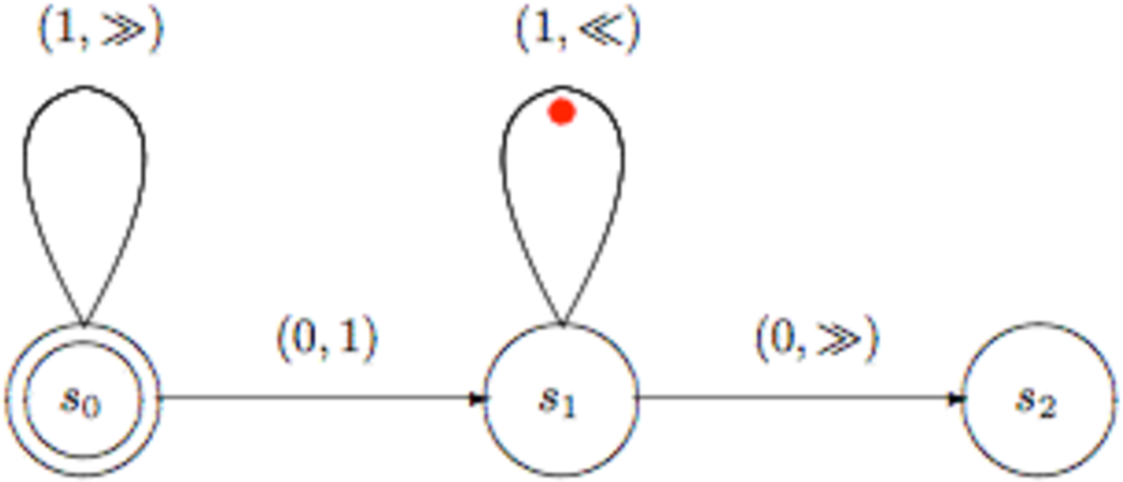
s0



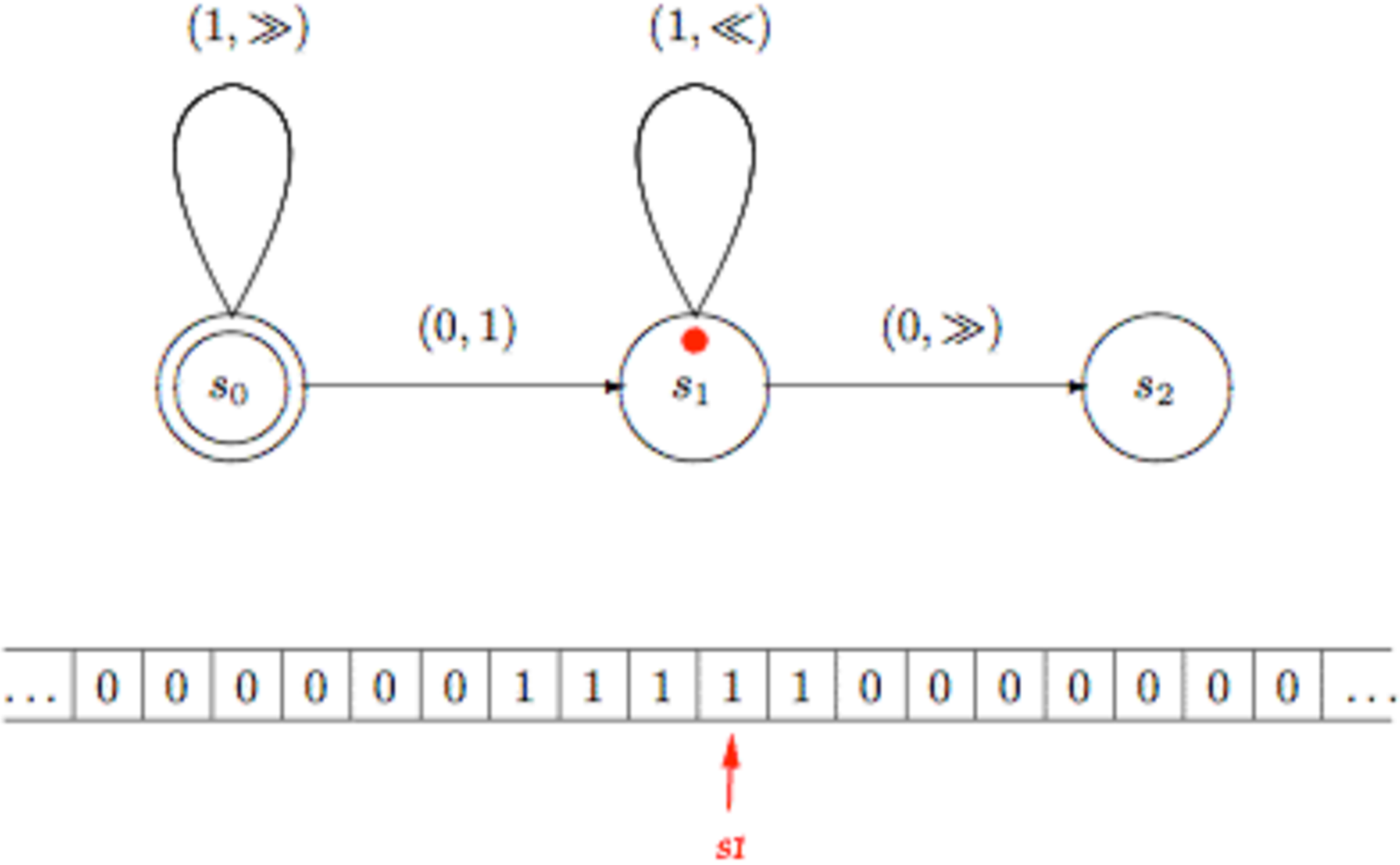
…	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	…



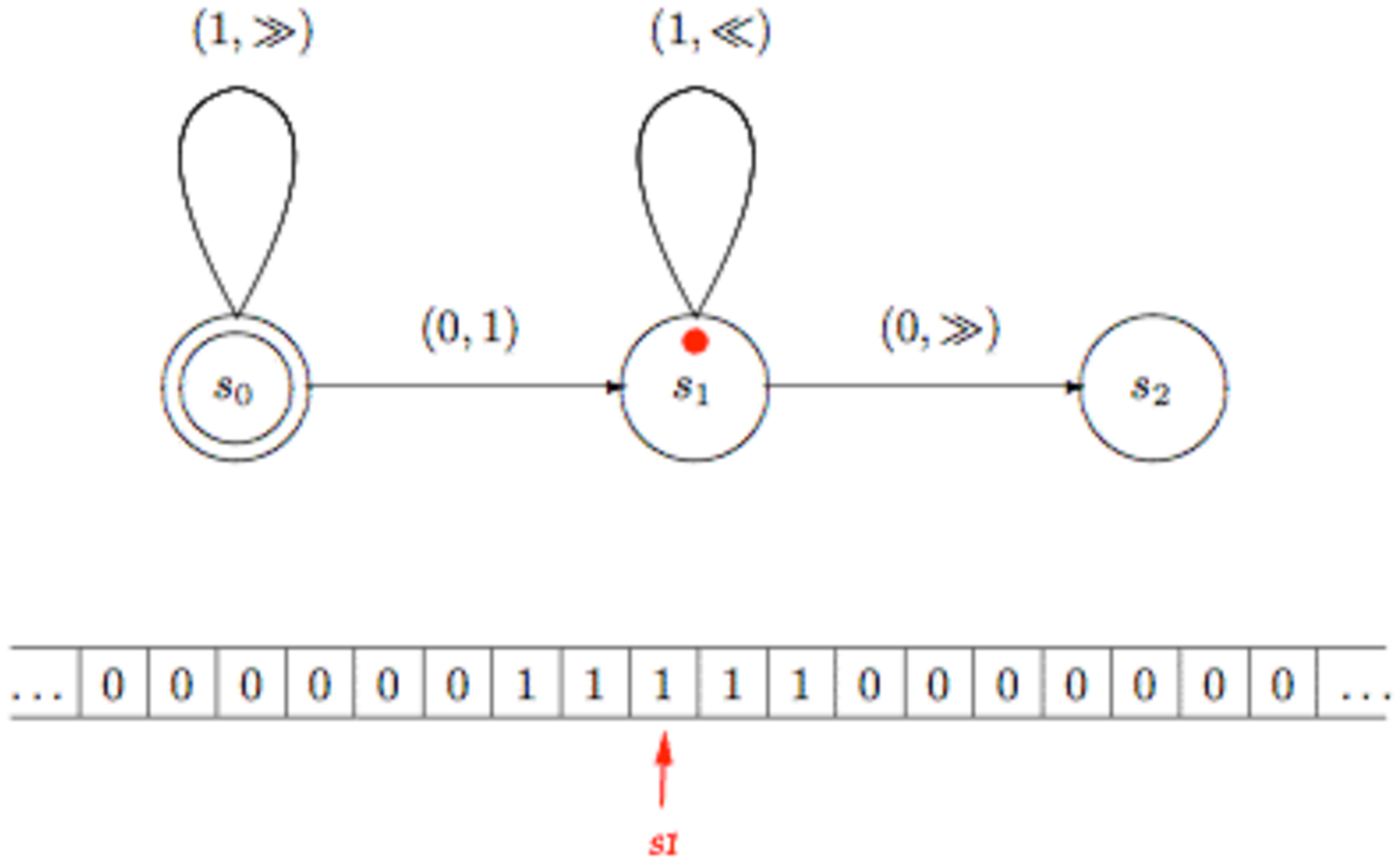
...	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	...



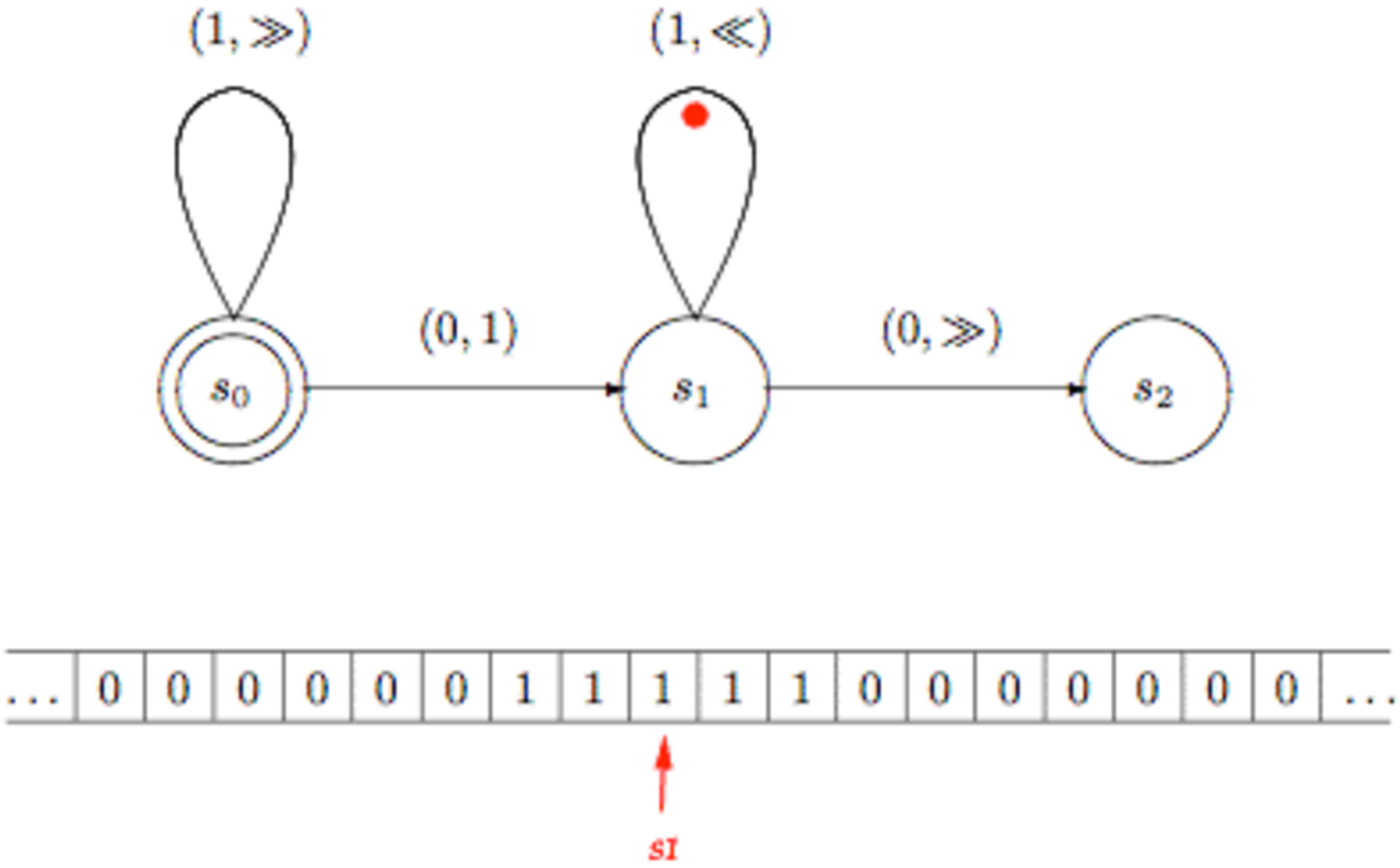
...	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	...
											↑ s_1								



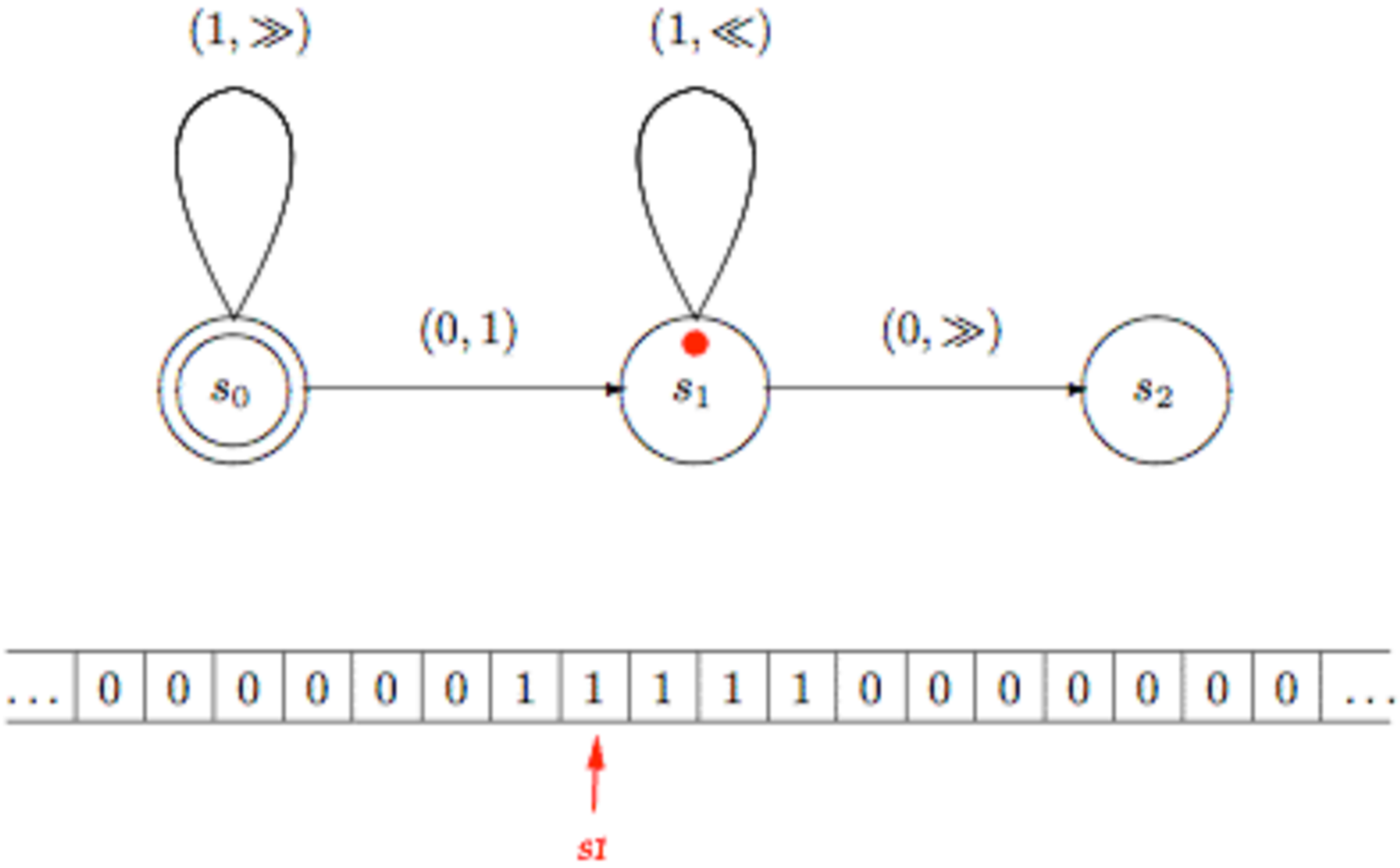
...	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	...

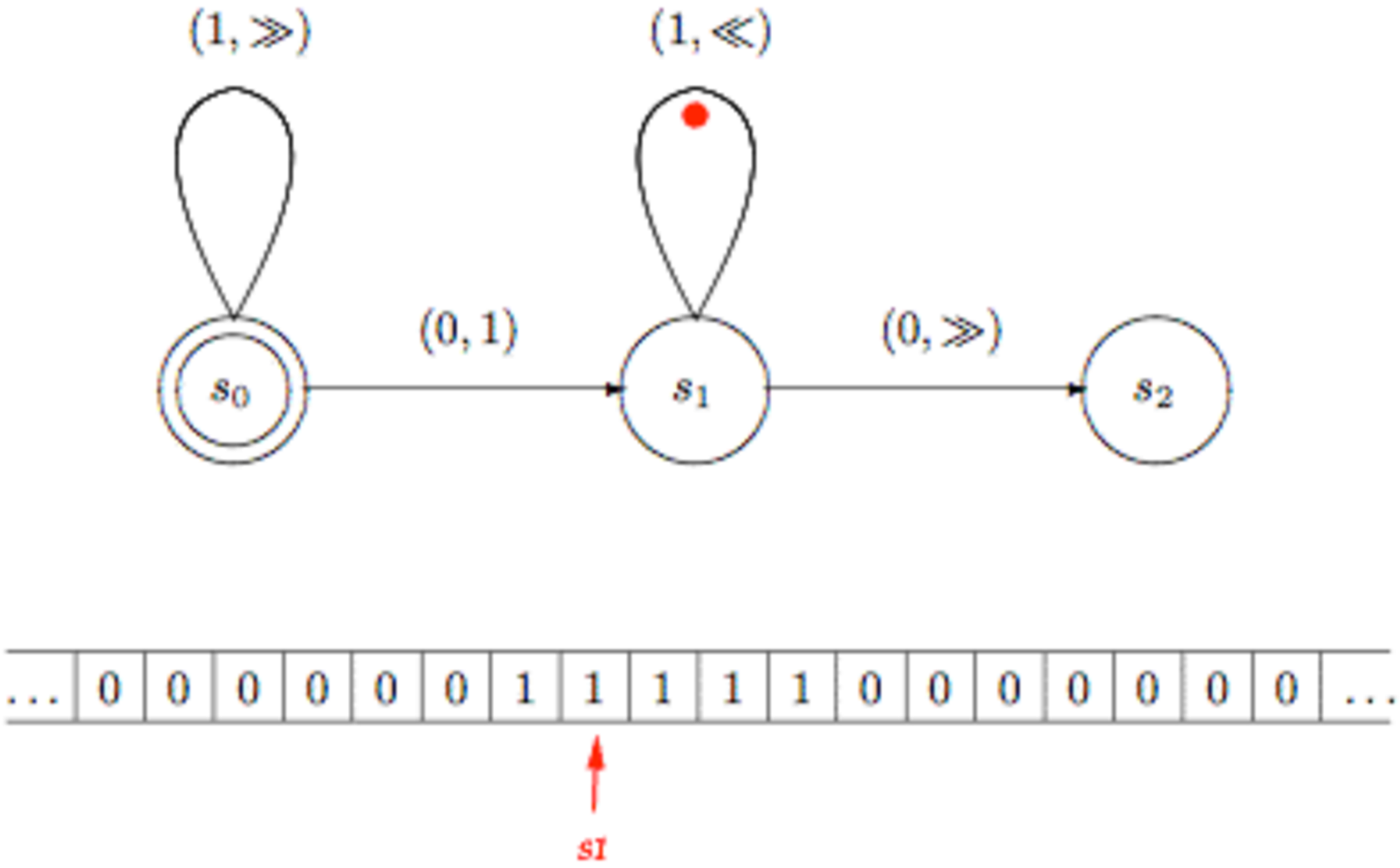


...	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	...

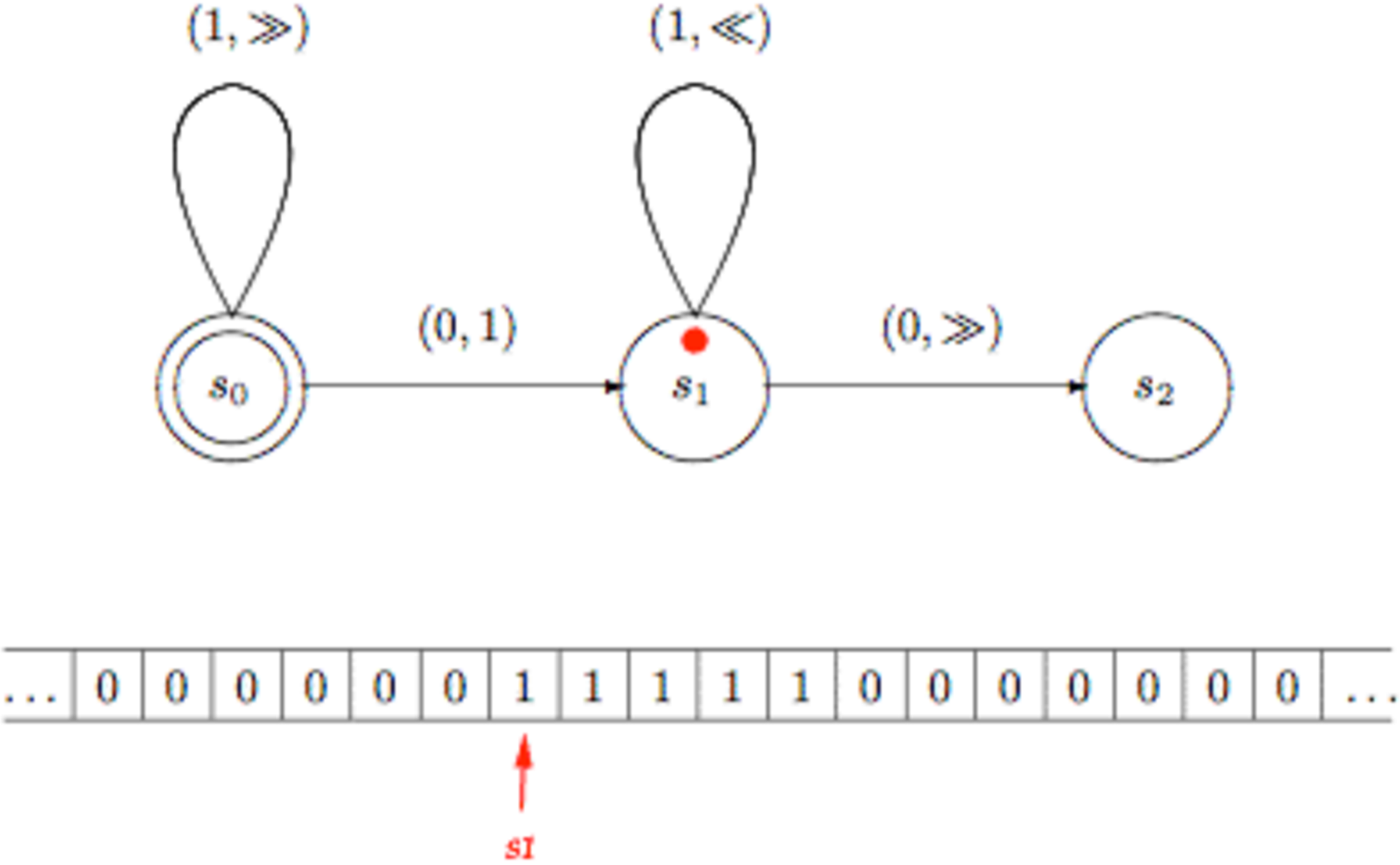


...	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	...

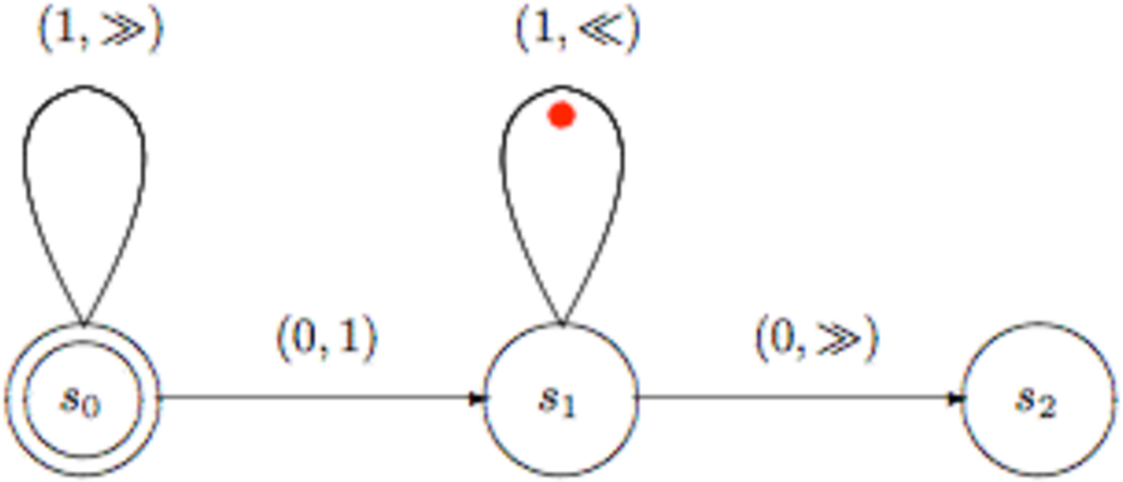

$(1, \gg)$
$(1, \ll)$
$(0, 1)$
$(0, \gg)$
s_0
s_1
s_2
… 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 …
s_1

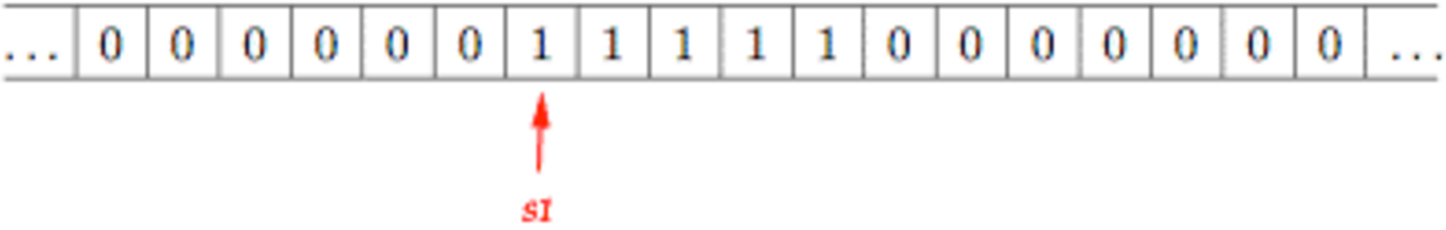


...	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	...

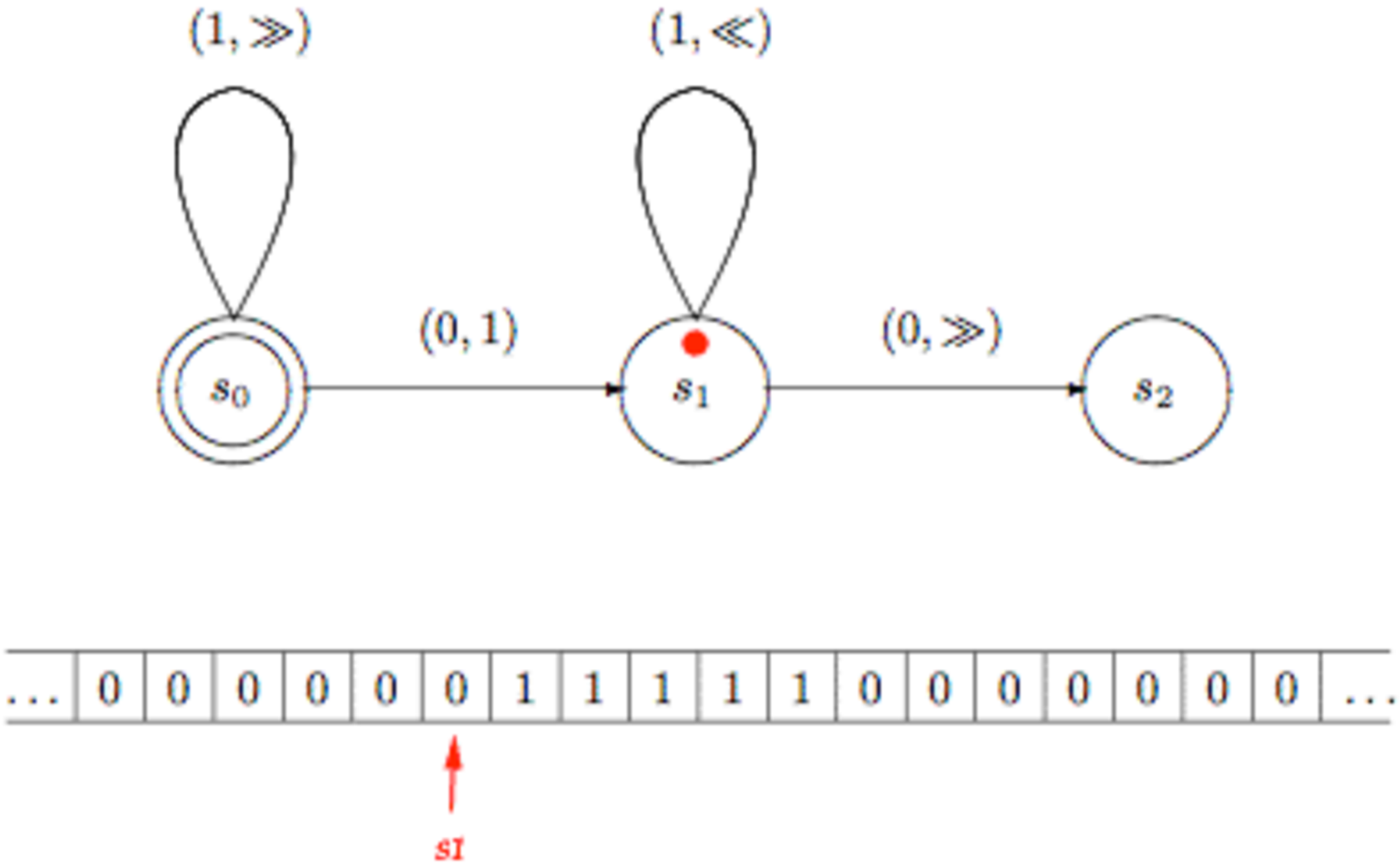


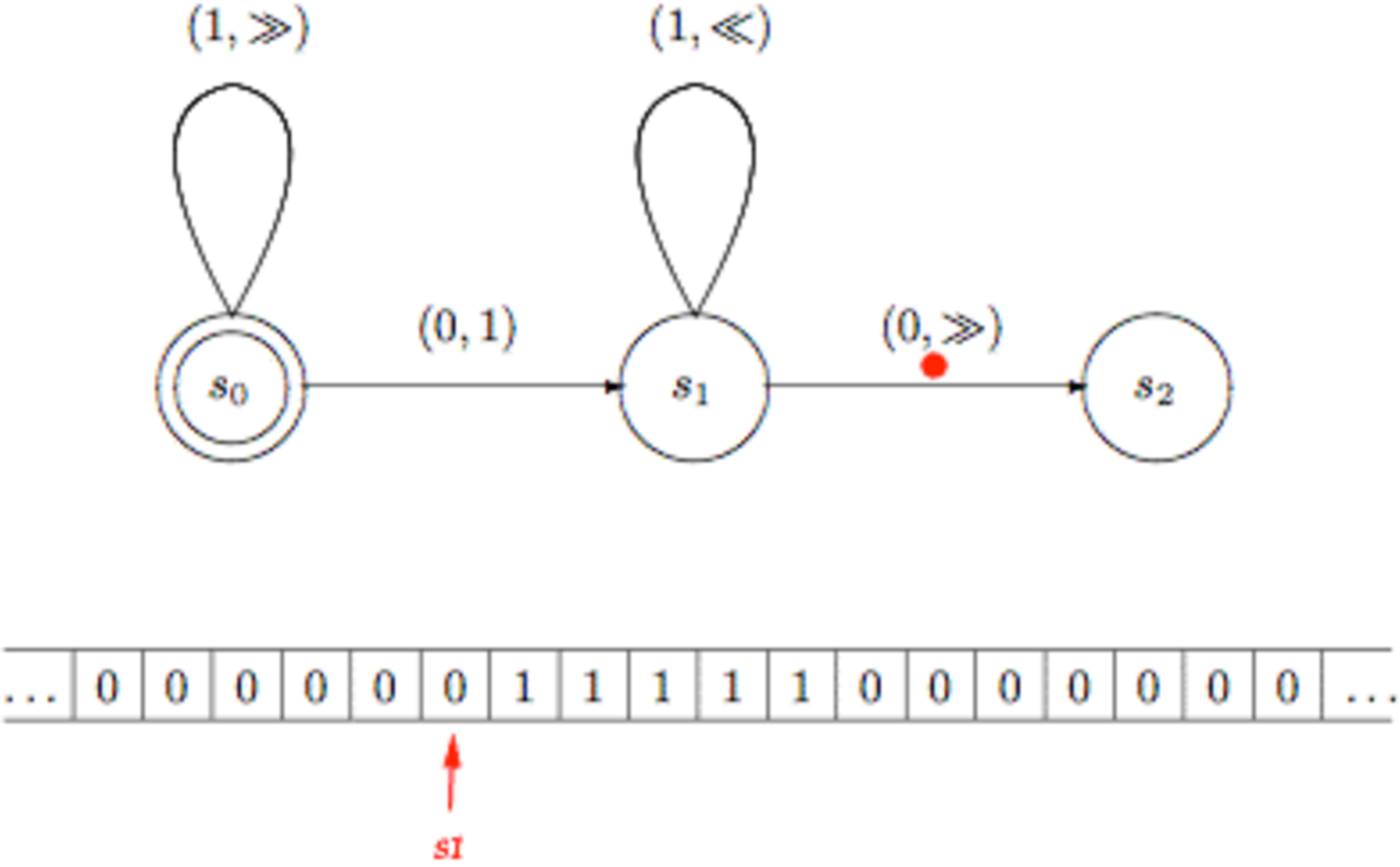
…	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	…



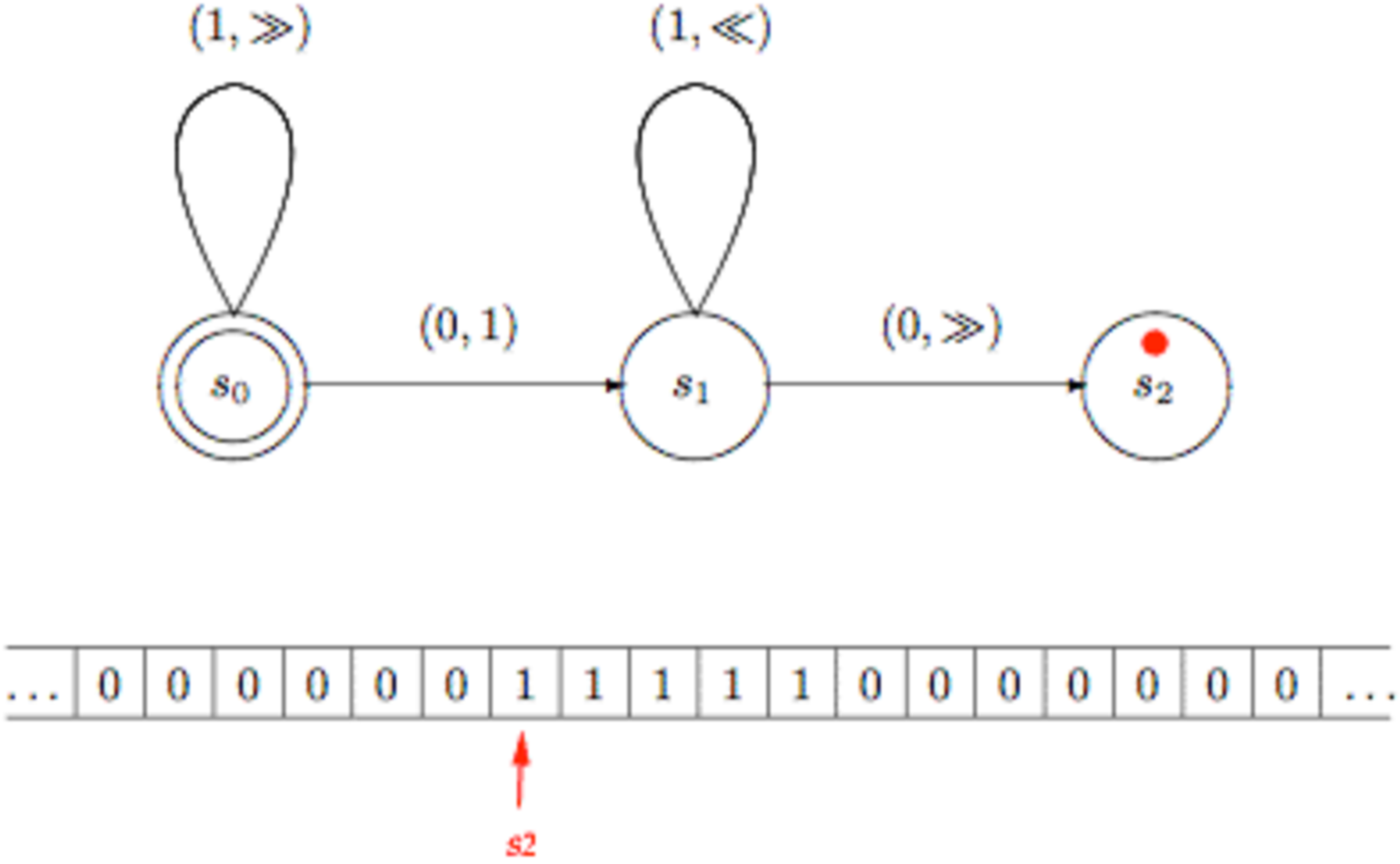


...	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	...


$(1, \gg)$
$(1, \ll)$
$(0, 1)$
$(0, \gg)$
s_0
s_1
s_2
… 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 …
s1

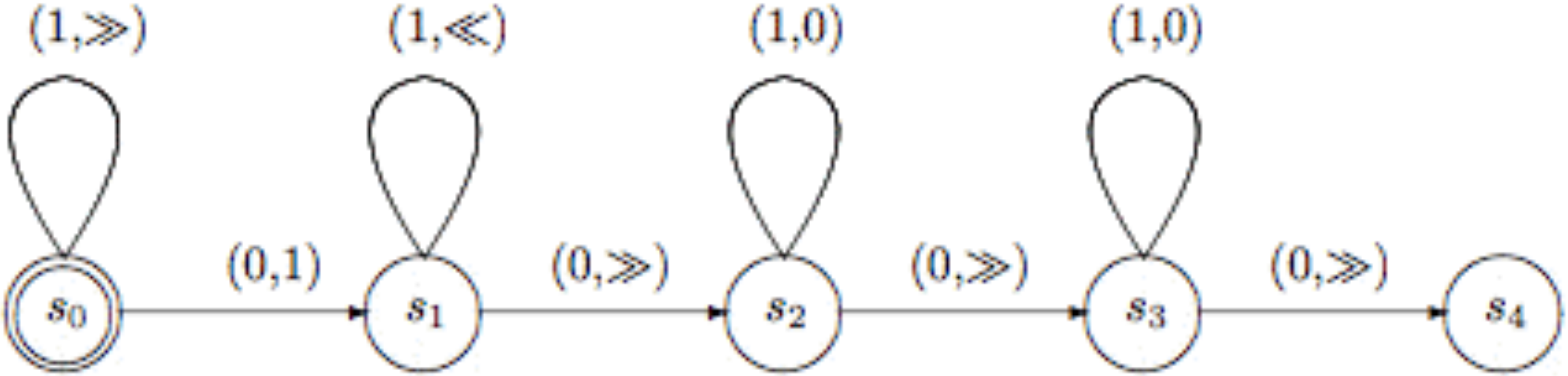


...	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	...

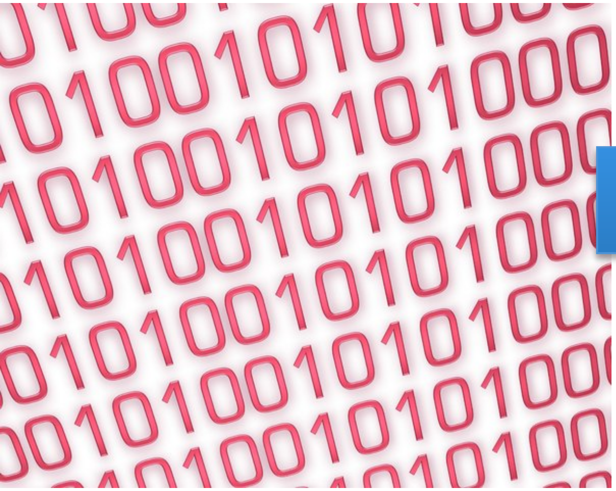


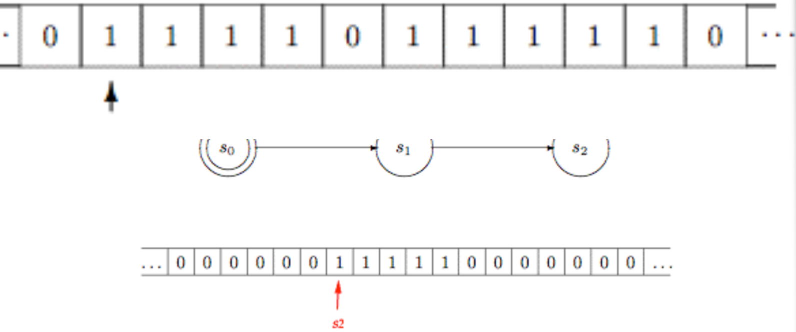
...	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	...

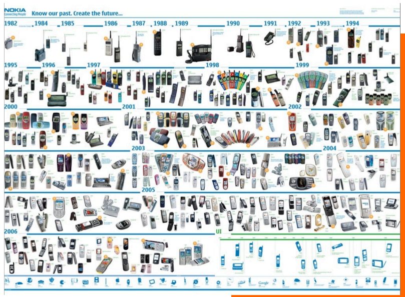
Addition of n+m



The (Hi)S[redacted]ory of Software Engineering [redacted] Computer Science





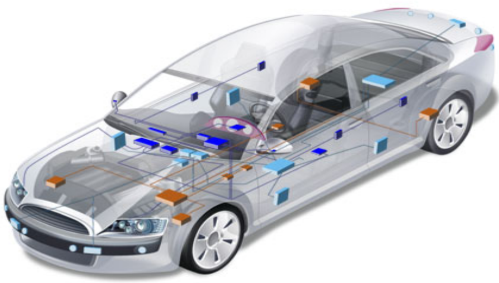


orange™

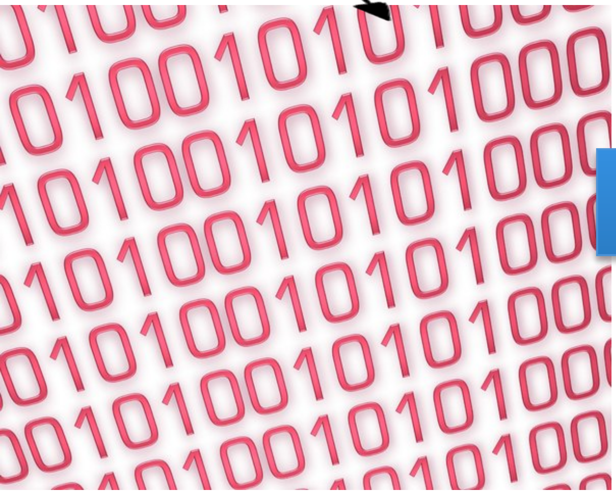
ANDRO

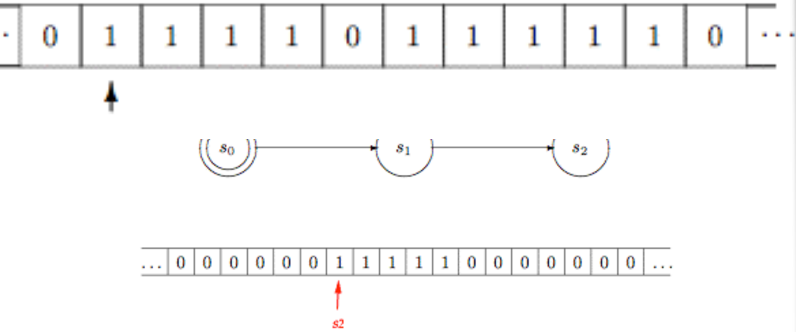
Google

twitter



Software Languages

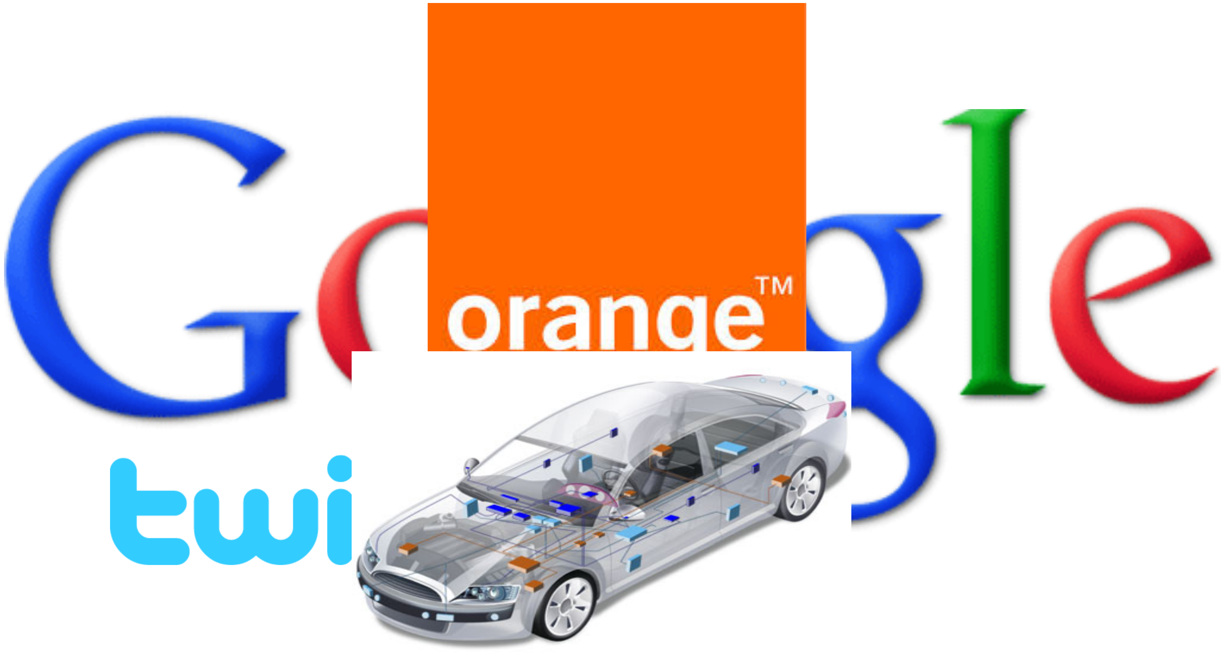




Programming the Turing Machine

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

Complex Systems



Distributed systems

Thousands of engineers/expertise

Web dev.

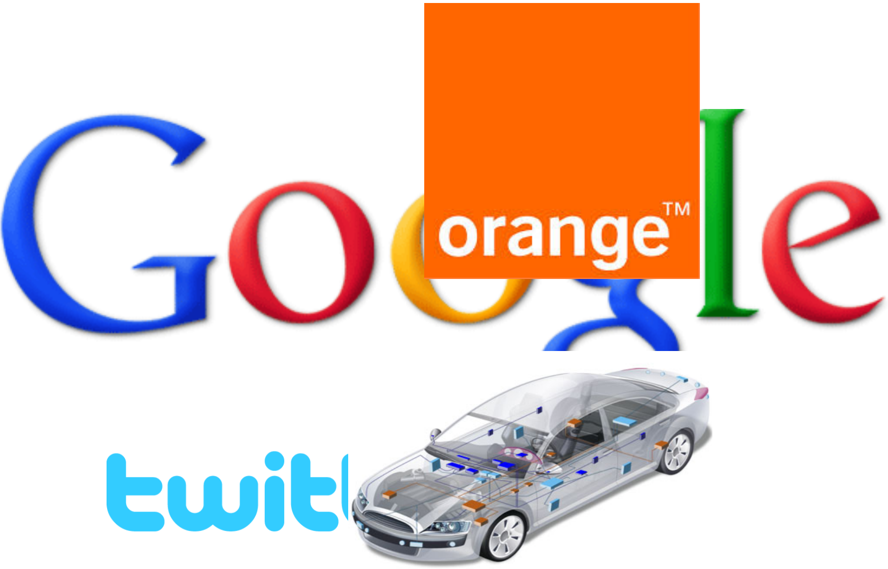
Large-scale systems

Critical Systems

Programming the Turing Machine

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

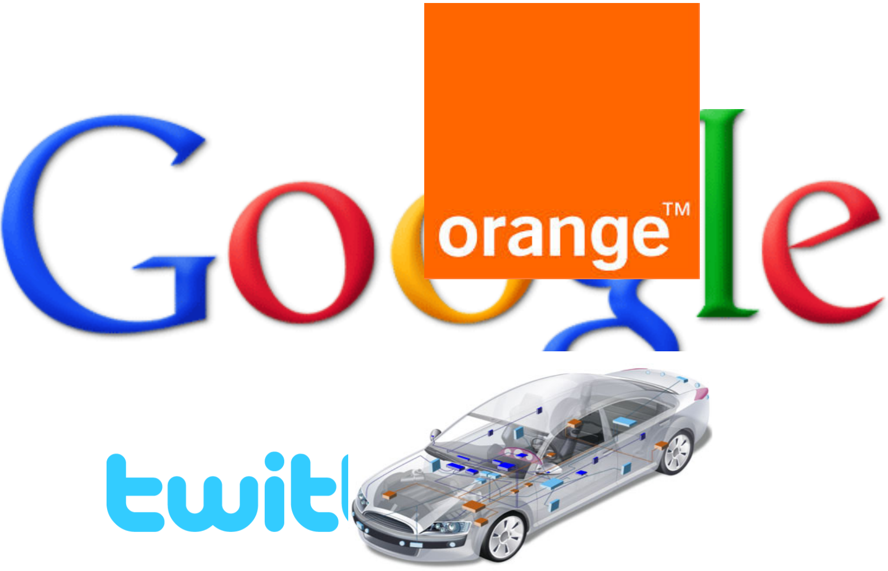
You cannot be serious



Programming the Turing Machine

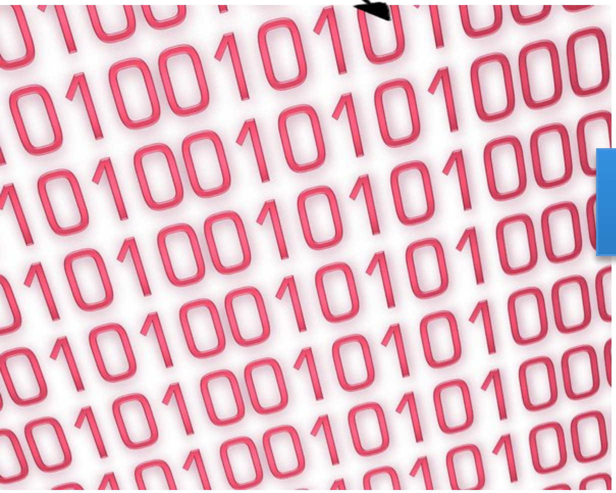
Why aren't we using tapes, states and transitions after all ?

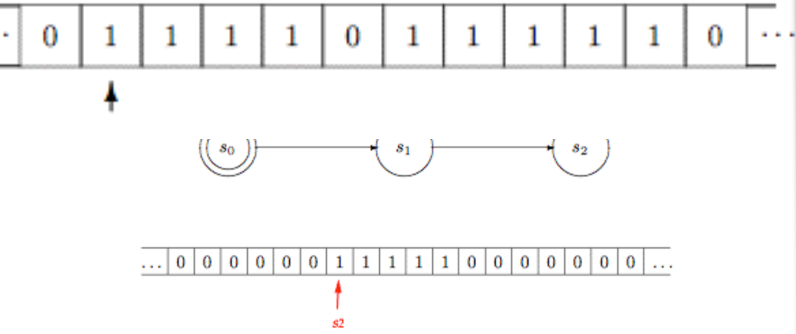
Software Languages



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

Languages





Complex Systems

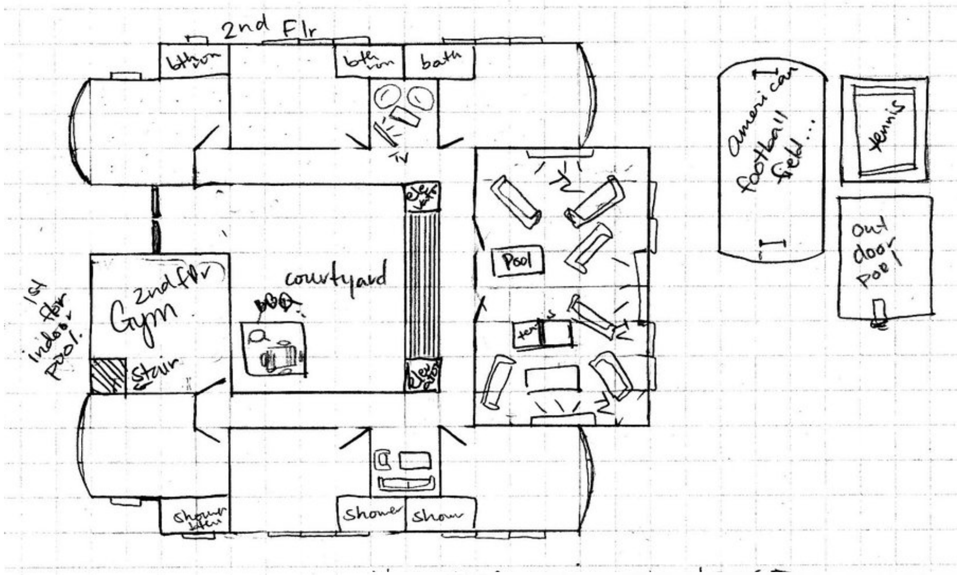
What is a language?

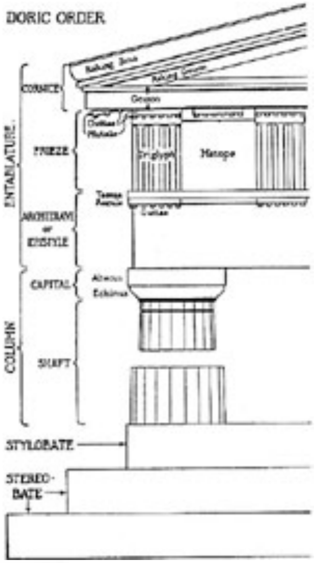
- « A system of signs, symbols, gestures, or rules used in **communicating** »

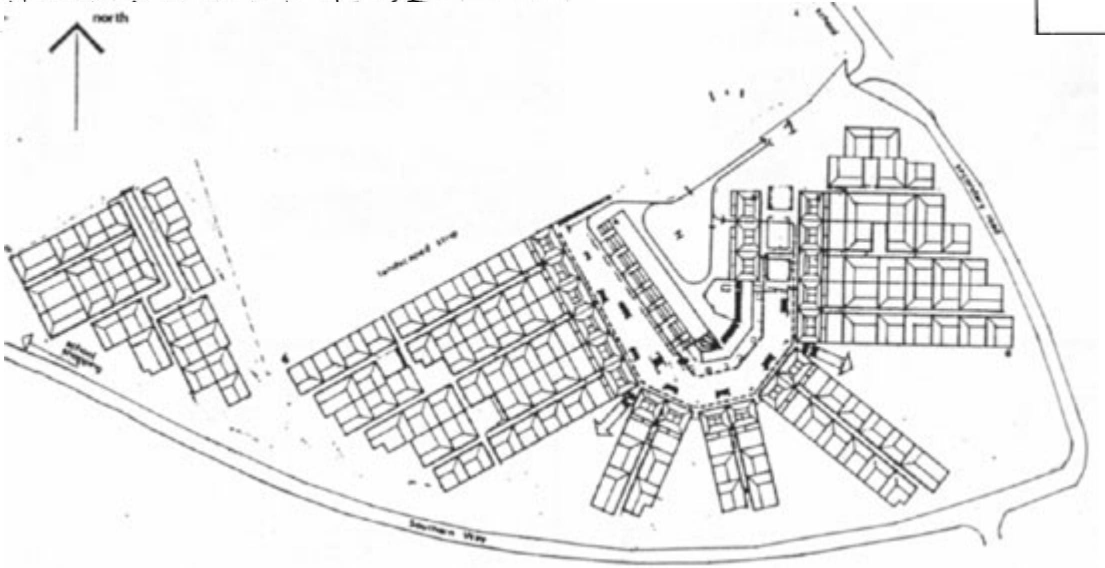
- « The **special** vocabulary and usages of a scientific, professional, or other group »

- « A system of symbols and rules used for communication with or between computers. »

Architecture

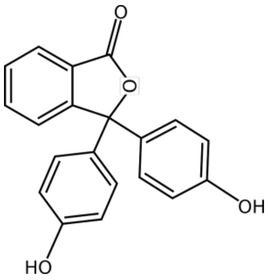


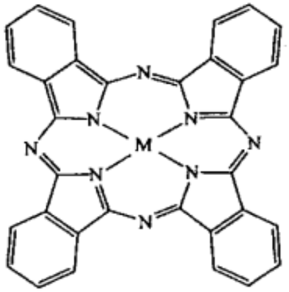




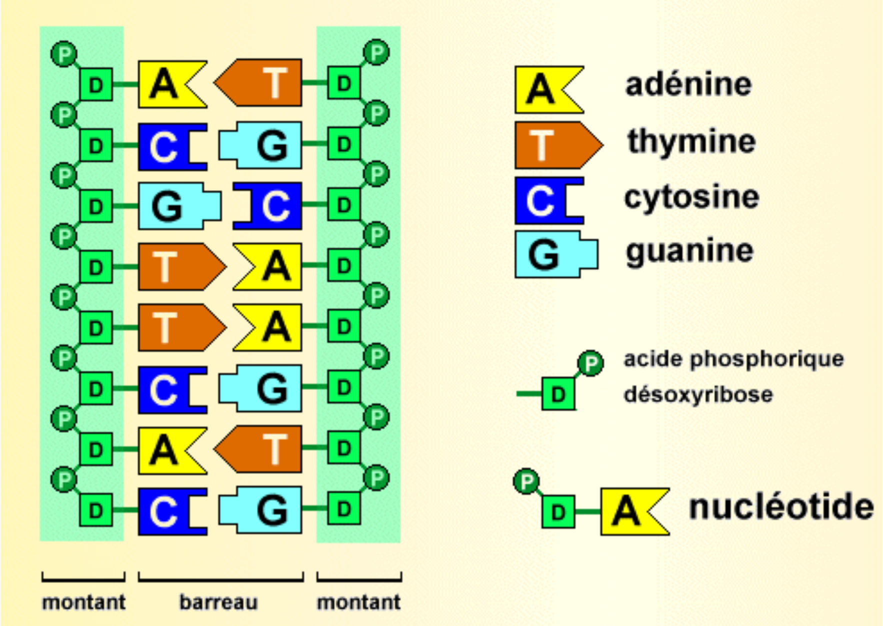
Cartography

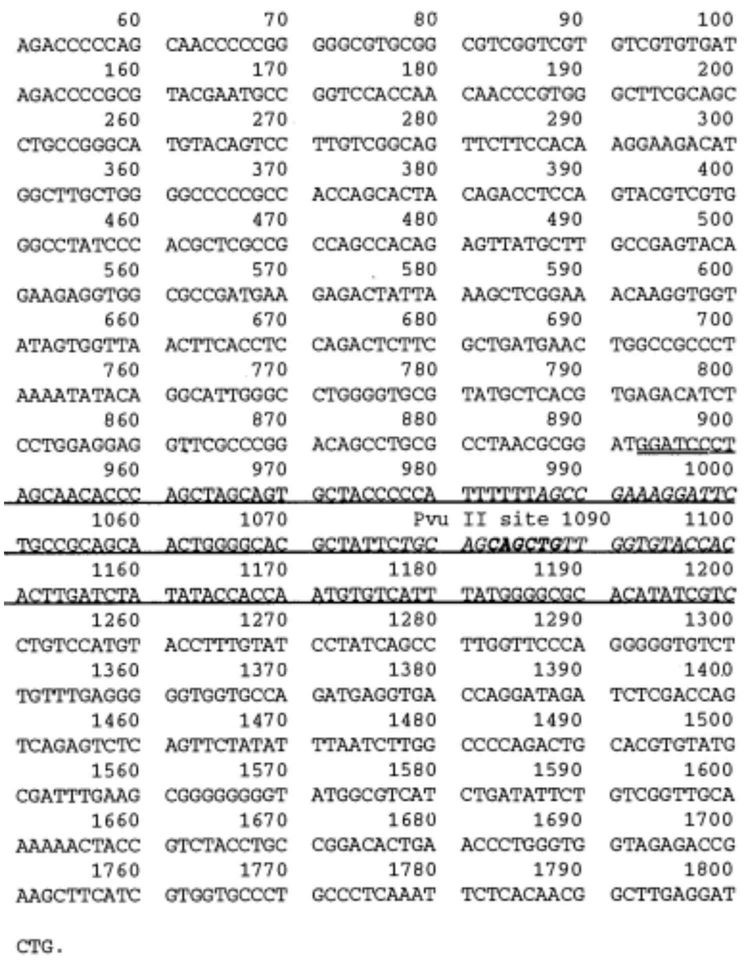
Biology





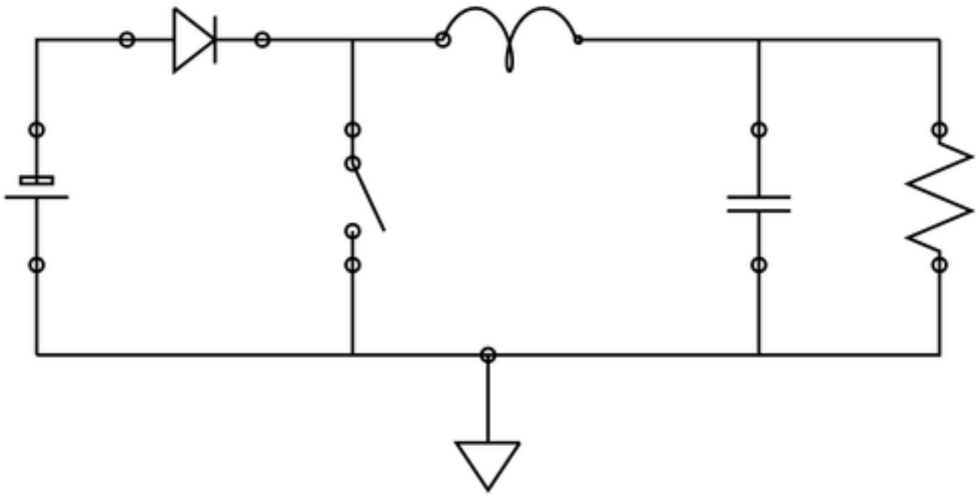
phthalocyanine

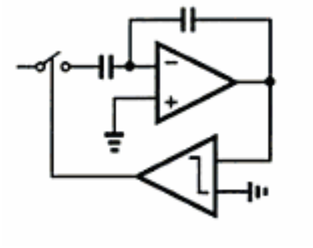


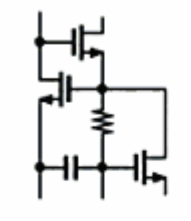


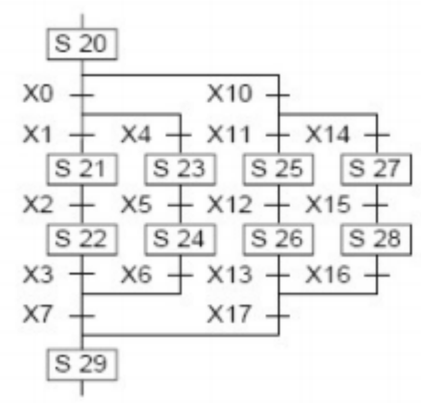
```
        60         70         80         90        100
AGACCCCCAG CAACCCCCGG GGGCGTGCGG CGTCGGTCGT GTCGTGTGAT
       160        170        180        190        200
AGACCCCGCG TACGAATGCC GGTCCACCAA CAACCCGTGG GCTTCGCAGC
       260        270        280        290        300
CTGCCGGGCA TGTACAGTCC 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
GAAGAGGTGG CGCCGATGAA GAGACTATTA AAGCTCGGAA ACAAGGTGGT
       660        670        680        690        700
ATAGTGGTTA ACTTCACCTC CAGACTCTTC GCTGATGAAC TGGCCGCCCT
       760        770        780        790        800
AAAATATACA GGCATTGGGC CTGGGGTGCG TATGCTCACG TGAGACATCT
       860        870        880        890        900
CCTGGAGGAG GTTCGCCCGG ACAGCCTGCG CCTAACGCGG ATGGATCCCT
       960        970        980        990       1000
AGCAACACCC AGCTAGCAGT GCTACCCCCA TTTTTTAGCC GAAAGGATTC
      1060       1070          Pvu II site 1090       1100
TGCCGCAGCA ACTGGGGCAC GCTATTCTGC AGCAGCTGTT GGTGTACCAC
      1160       1170       1180       1190       1200
ACTTGATCTA TATACCACCA ATGTGTCATT TATGGGGCGC ACATATCGTC
      1260       1270       1280       1290       1300
CTGTCCATGT ACCTTTGTAT CCTATCAGCC TTGGTTCCCA GGGGGTGTCT
      1360       1370       1380       1390       1400
TGTTTGAGGG GGTGGTGCCA GATGAGGTGA CCAGGATAGA TCTCGACCAG
      1460       1470       1480       1490       1500
TCAGAGTCTC AGTTCTATAT TTAATCTTGG CCCCAGACTG CACGTGTATG
      1560       1570       1580       1590       1600
CGATTTGAAG CGGGGGGGGT ATGGCGTCAT CTGATATTCT GTCGGTTGCA
      1660       1670       1680       1690       1700
AAAAACTACC GTCTACCTGC CGGACACTGA ACCCTGGGTG GTAGAGACCG
      1760       1770       1780       1790       1800
AAGCTTCATC GTGGTGCCCT GCCCTCAAAT TCTCACAACG GCTTGAGGAT

CTG.
```

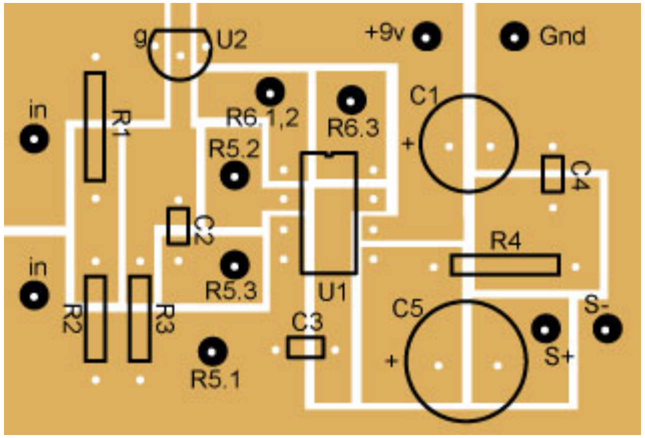
Electronics











In Software Engineering

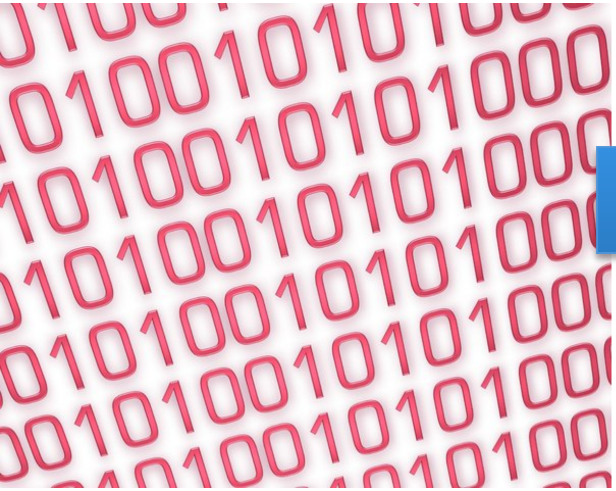
« Languages are the primary way in which system developers communicate, design and implement software systems »

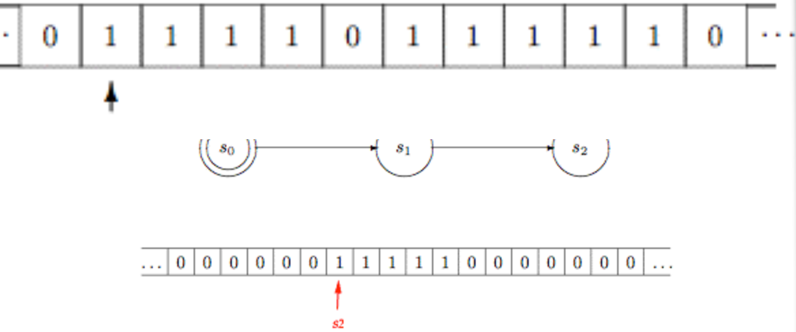
General Purpose Languages

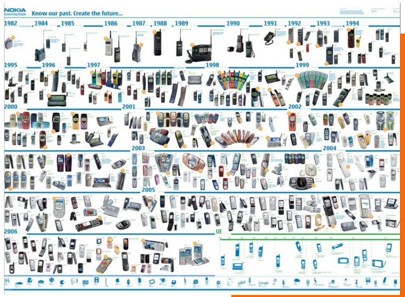
Assembly ?

COBOL ? LISP ? C ? C++ ?

Java? PHP ? C# ? Ruby ?





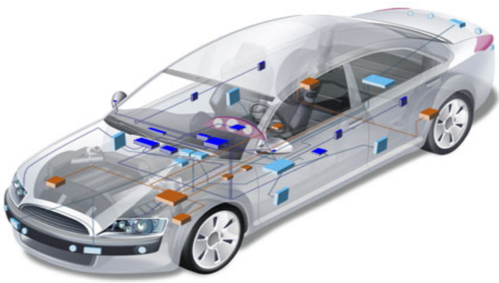


orange™

andro

Google

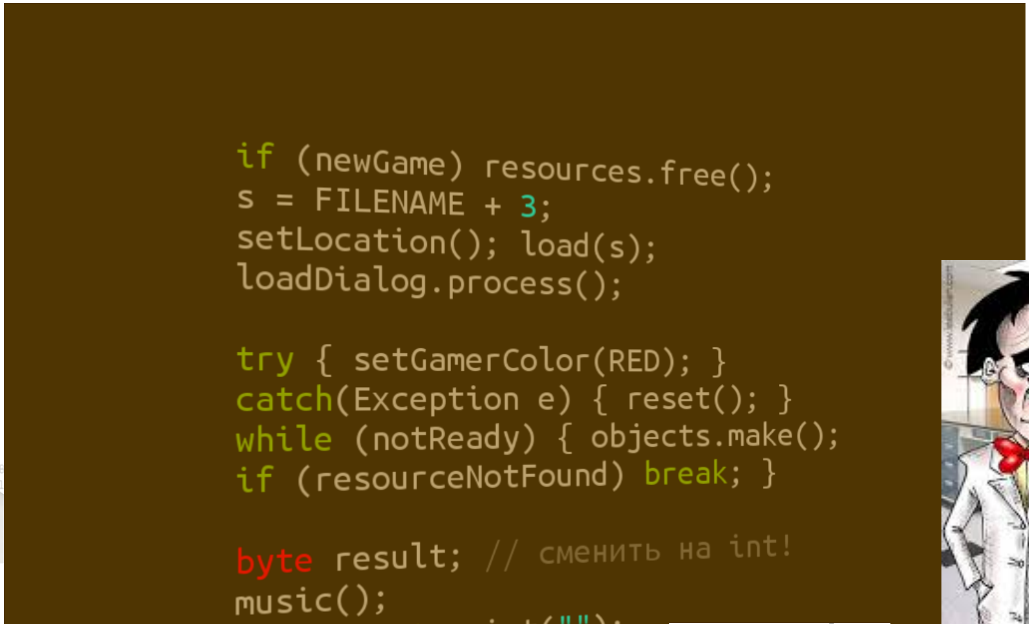
twitter



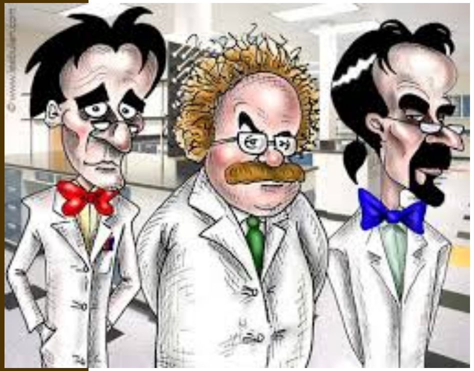
Limits of General Purpose Languages (1)

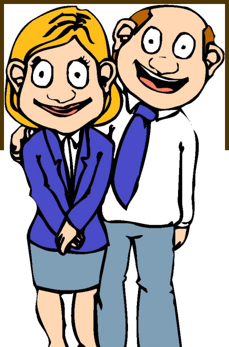
- **Abstractions** and **notations** used are not natural/suitable for the stakeholders

User Interface Designer with a passion for designing beautiful and functional user experiences. Minimalist who believes that less is more.



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

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

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





Limits of General Purpose Languages (2)

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

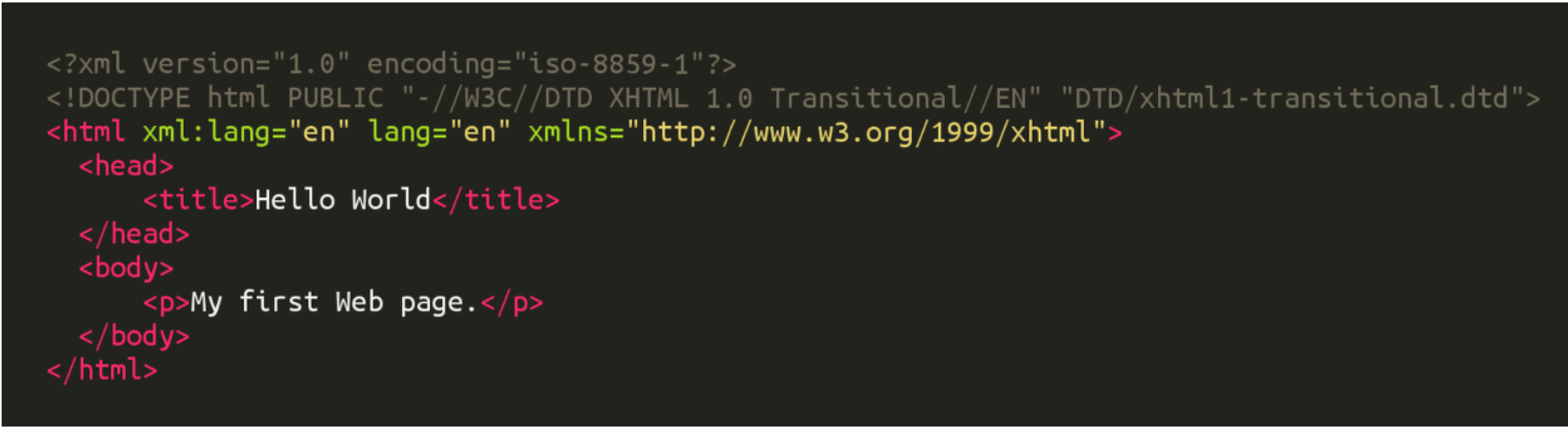
Domain Specific Languages

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

Domain Specific Languages (DSLs)

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

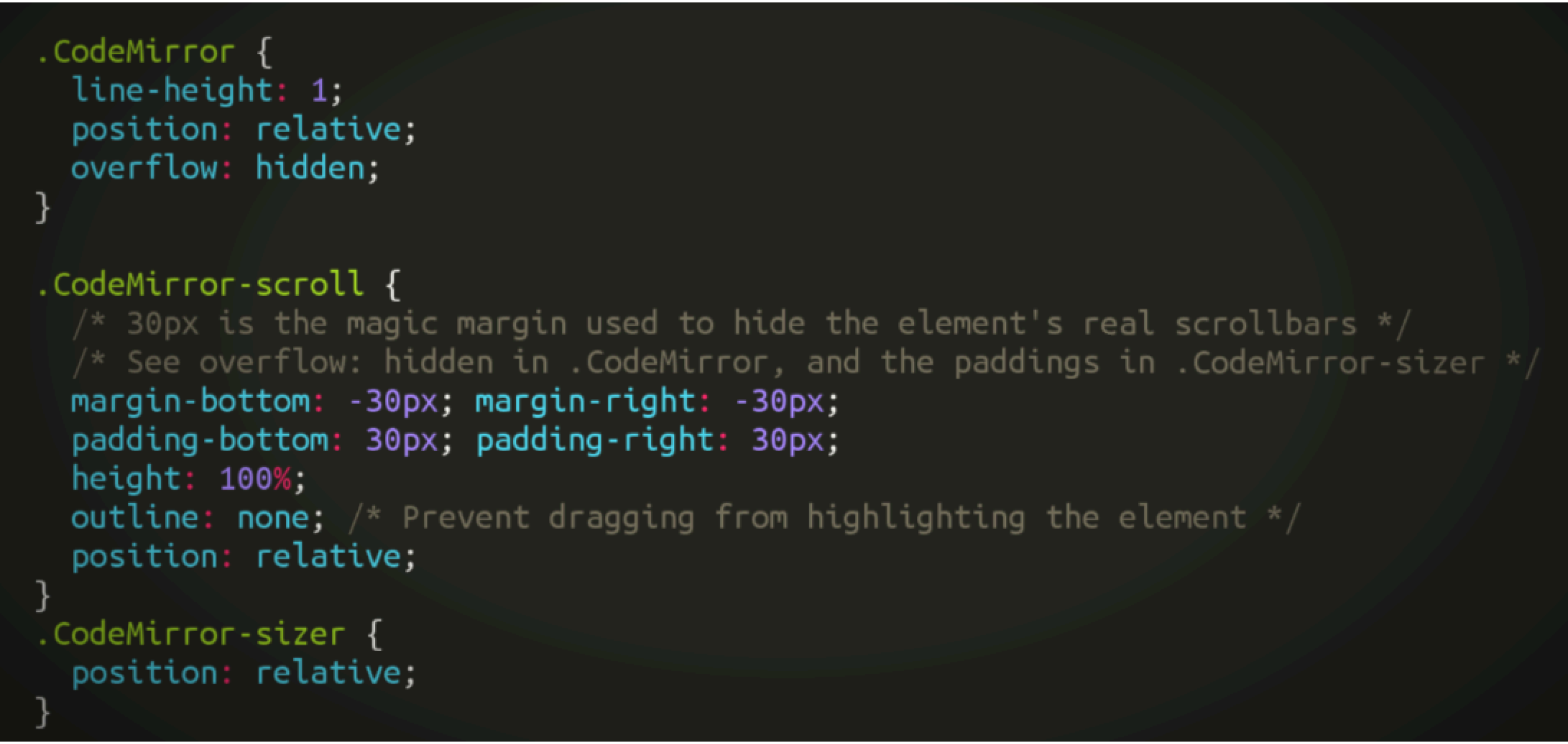
HTML



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

Domain: web (markup)

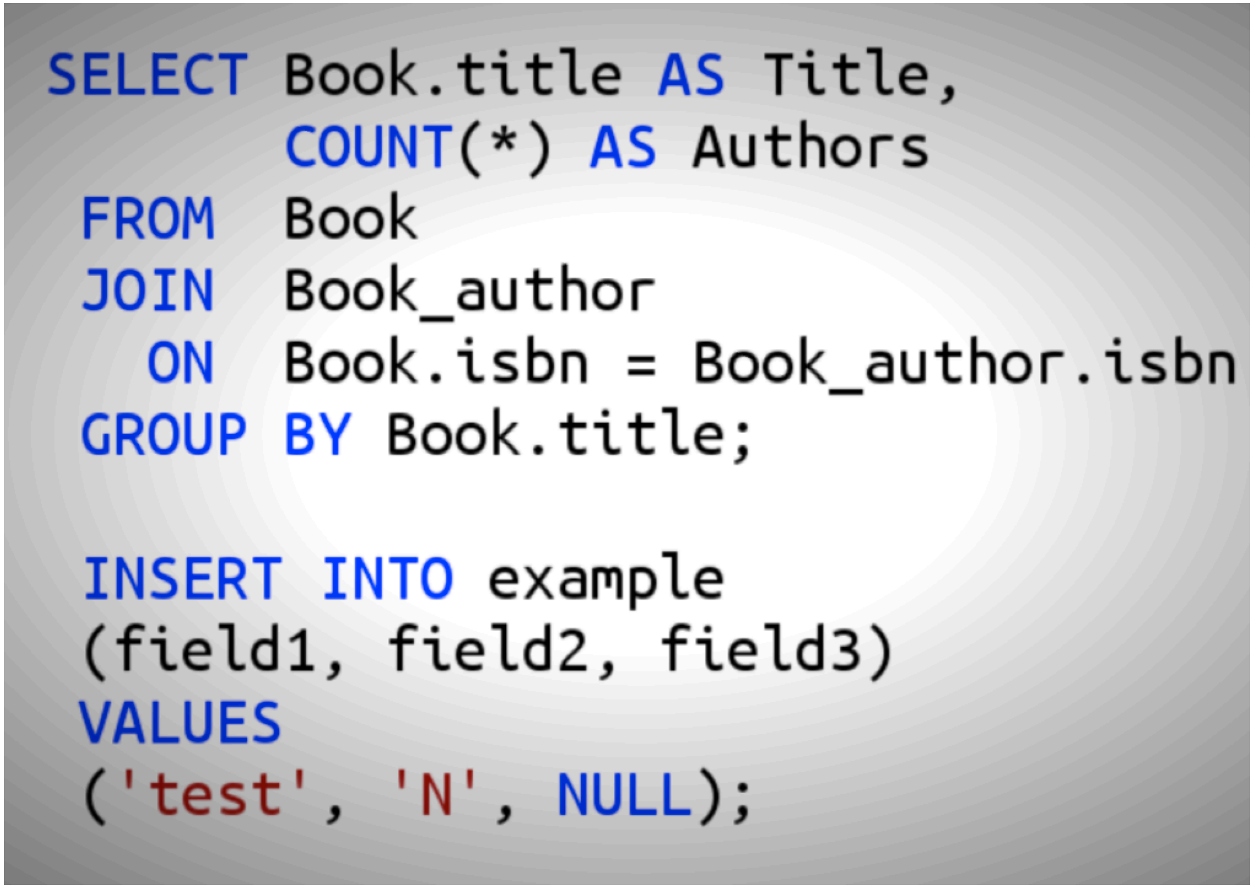
CSS



```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)

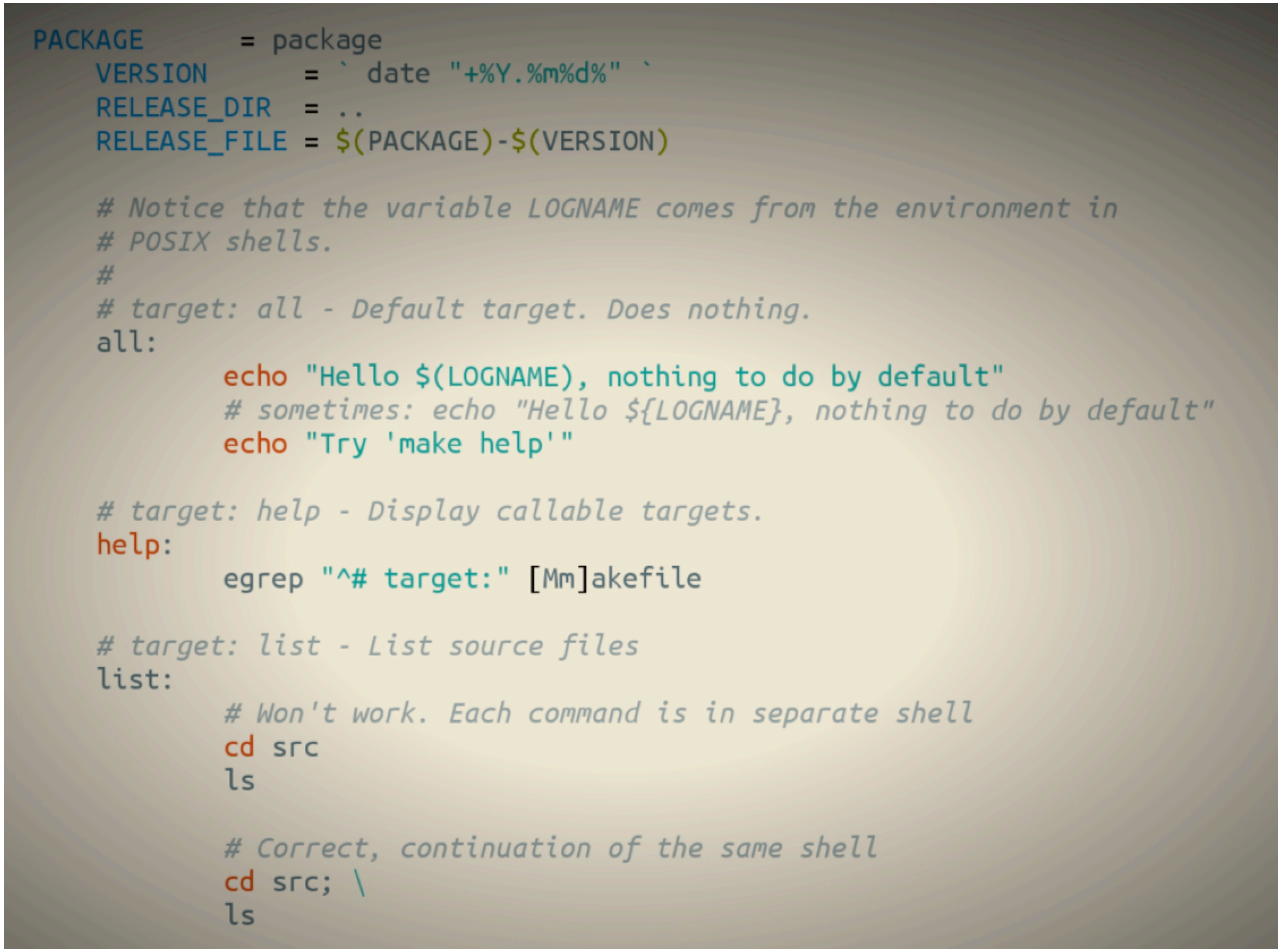
SQL



```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)

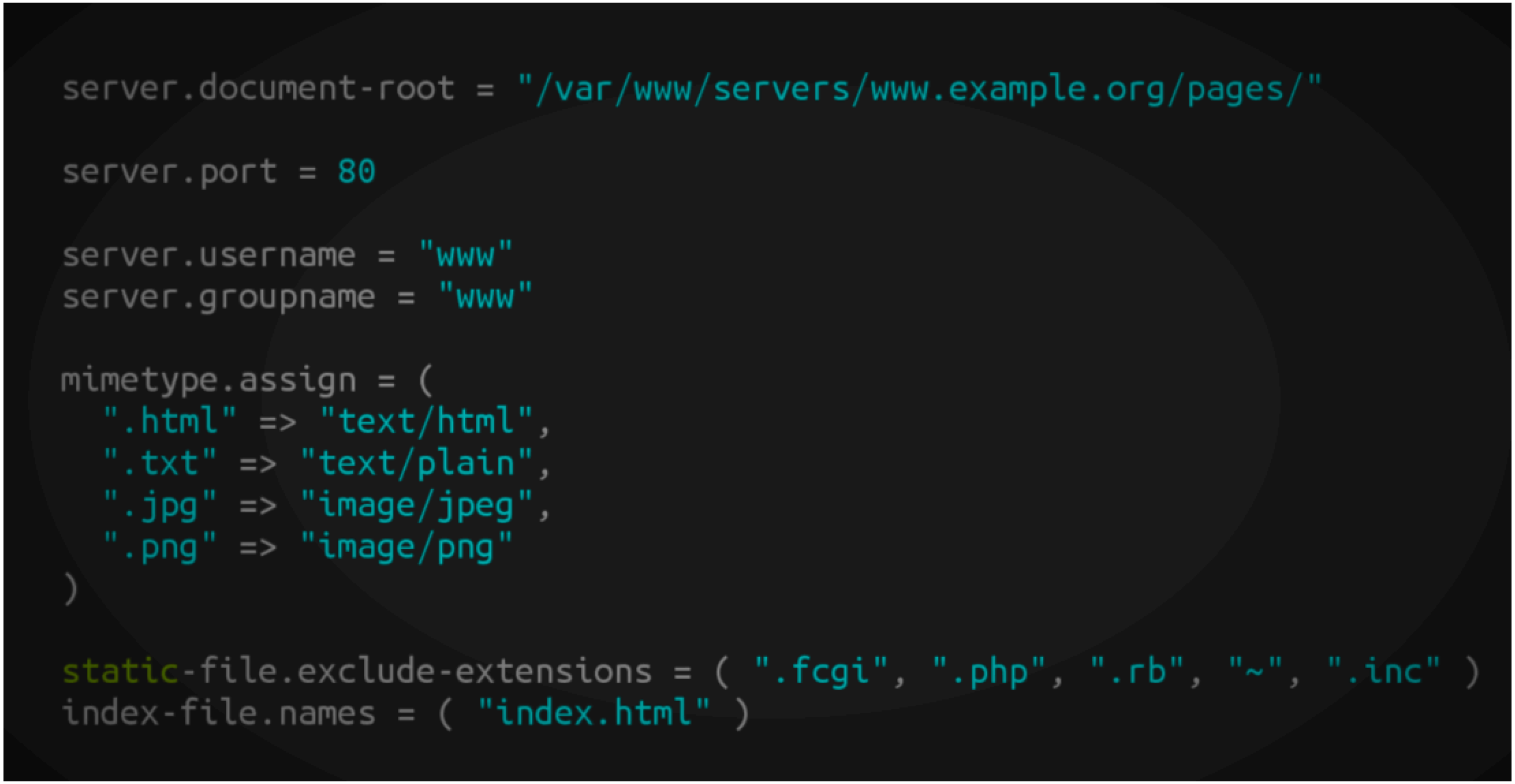
Makefile



```
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

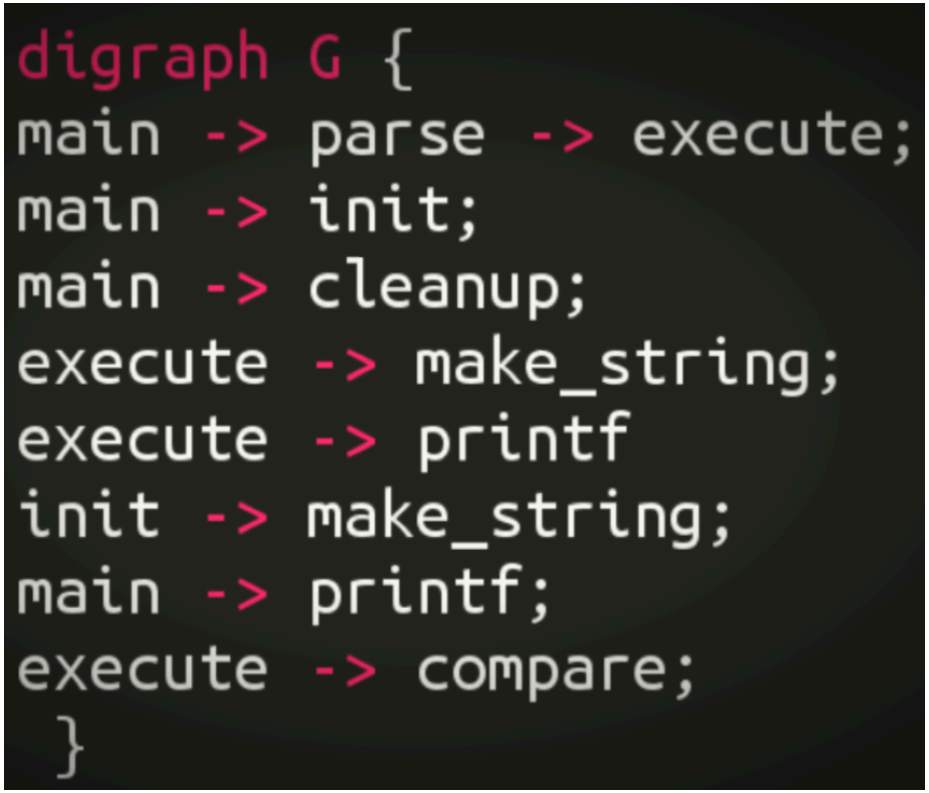
Lighthttpd configuration file



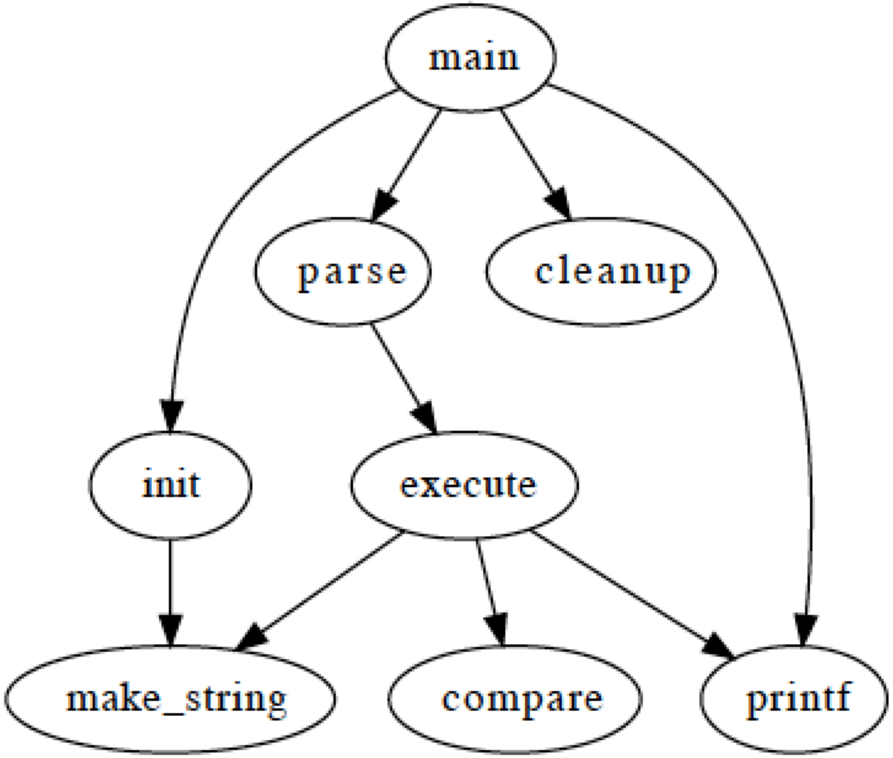
```
server.document-root = "/var/www/servers/www.example.org/pages/"

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

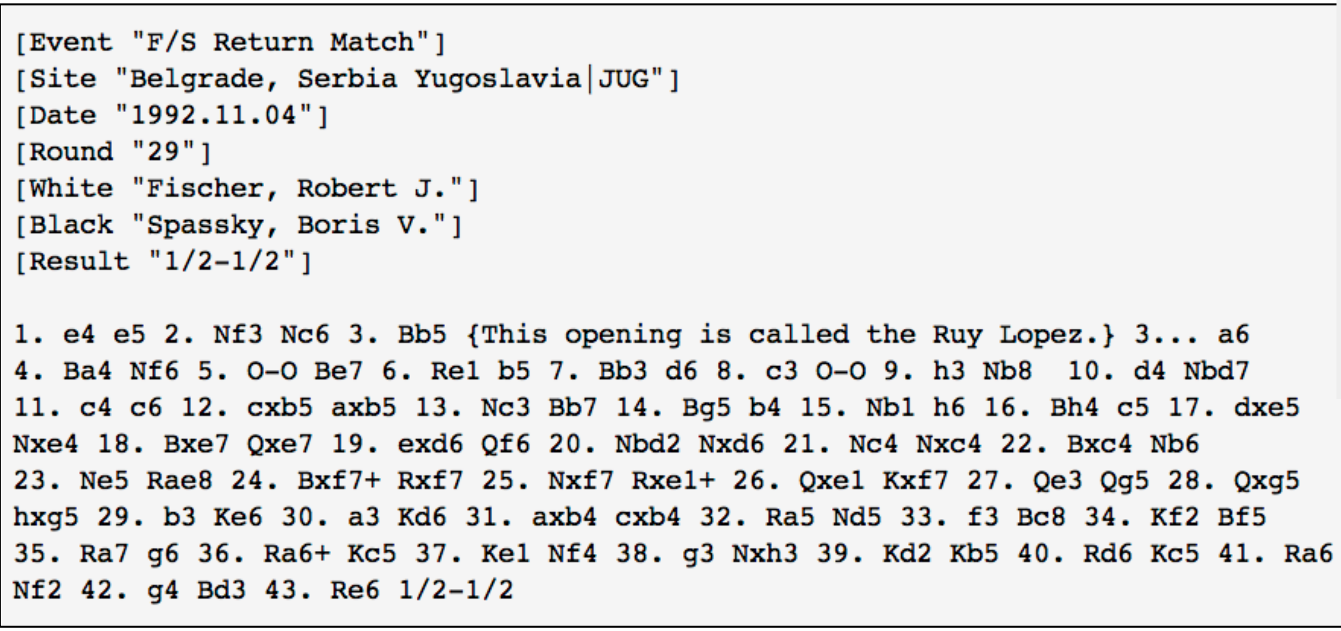


```
digraph G {
main -> parse -> execute;
main -> init;
main -> cleanup;
execute -> make_string;
execute -> printf
init -> make_string;
main -> printf;
execute -> compare;
 }
```

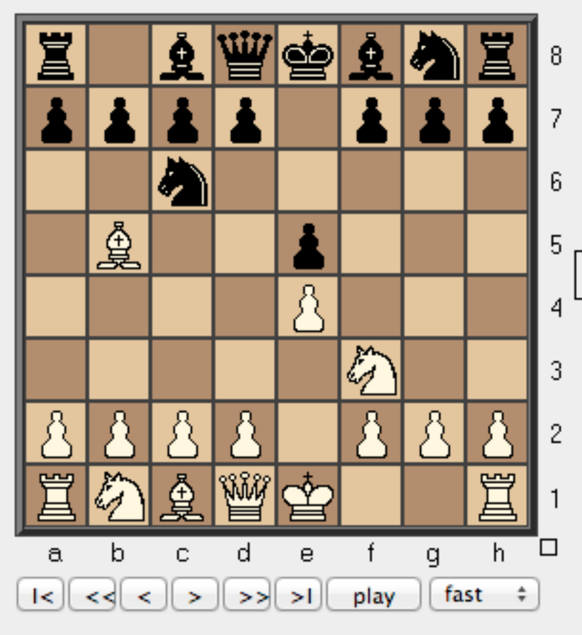


Domain: graph (drawing)

PGN (Portable Game Notation)

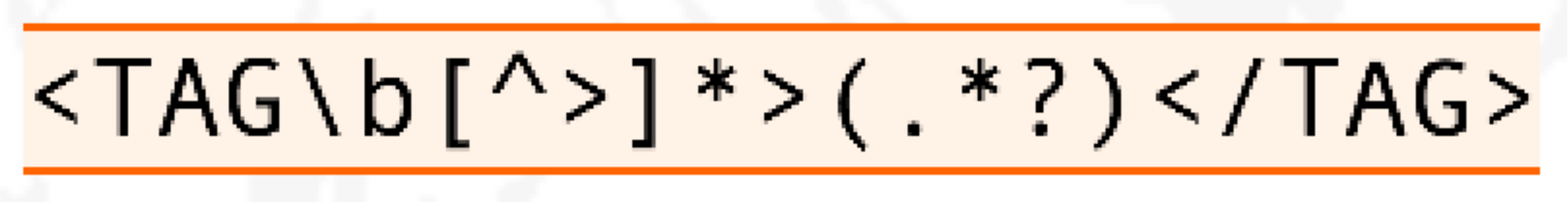


```
[Event "F/S Return Match"]
[Site "Belgrade, Serbia Yugoslavia|JUG"]
[Date "1992.11.04"]
[Round "29"]
[White "Fischer, Robert J."]
[Black "Spassky, Boris V."]
[Result "1/2-1/2"]

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



Domain: chess (games)

Regular expression



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

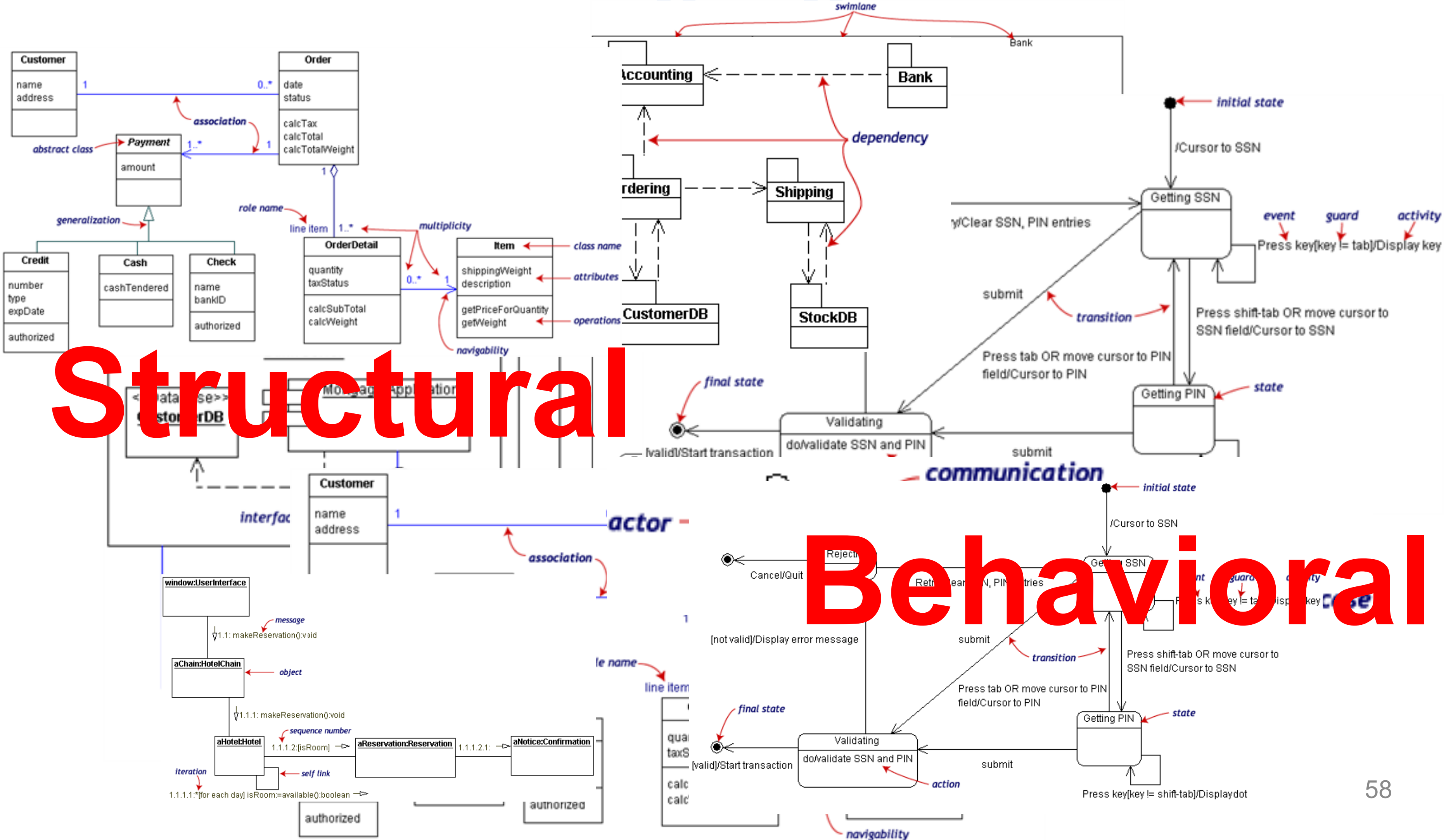
Domain: strings (pattern matching)

OCL

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

Domain: model management

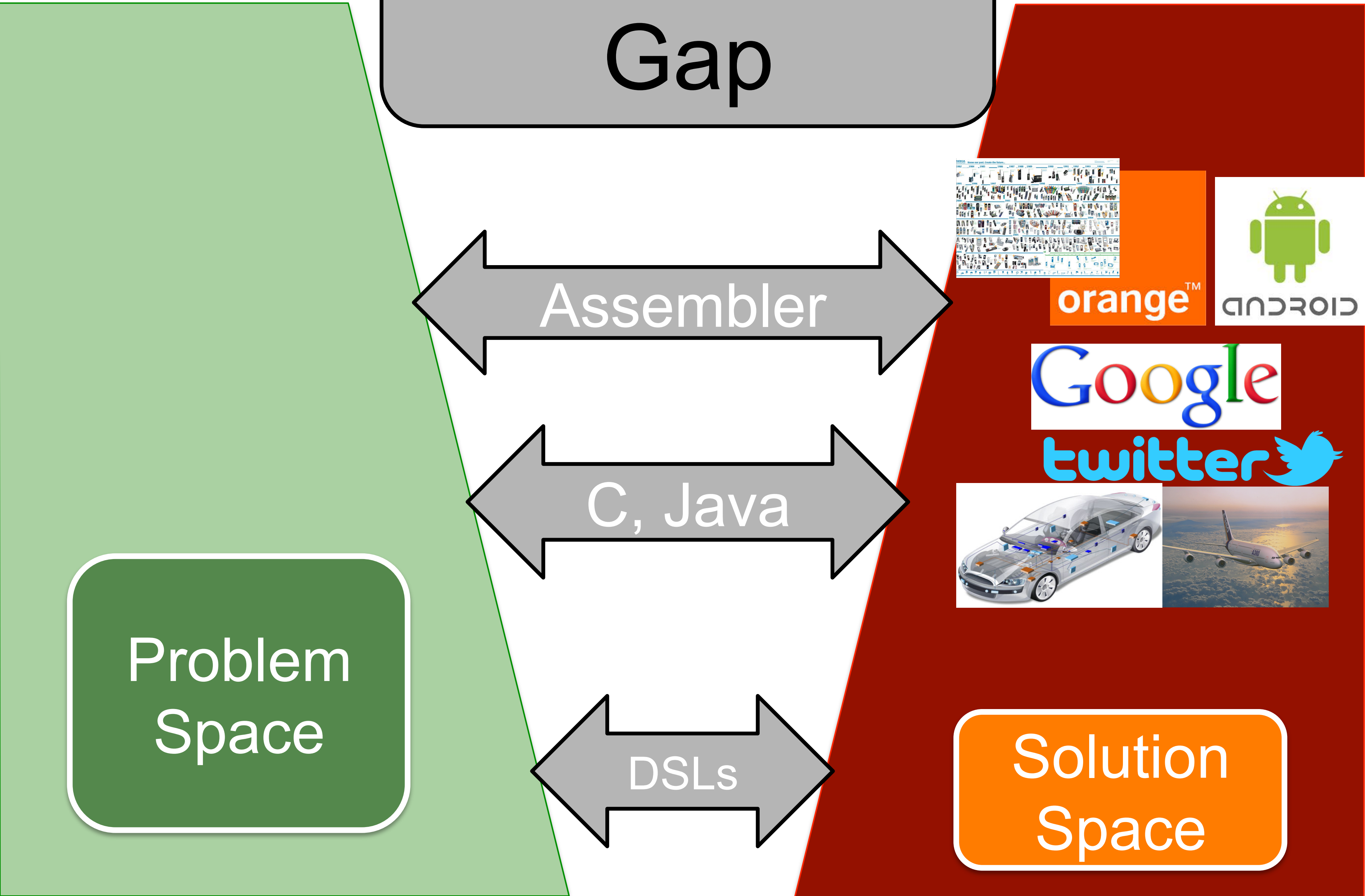
UML can be seen as a collection of domain-specific modeling languages



Structural

Behavioral

Abstraction Gap



Assembler

C, Java

DSLs

Problem Space

Solution Space

« 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.

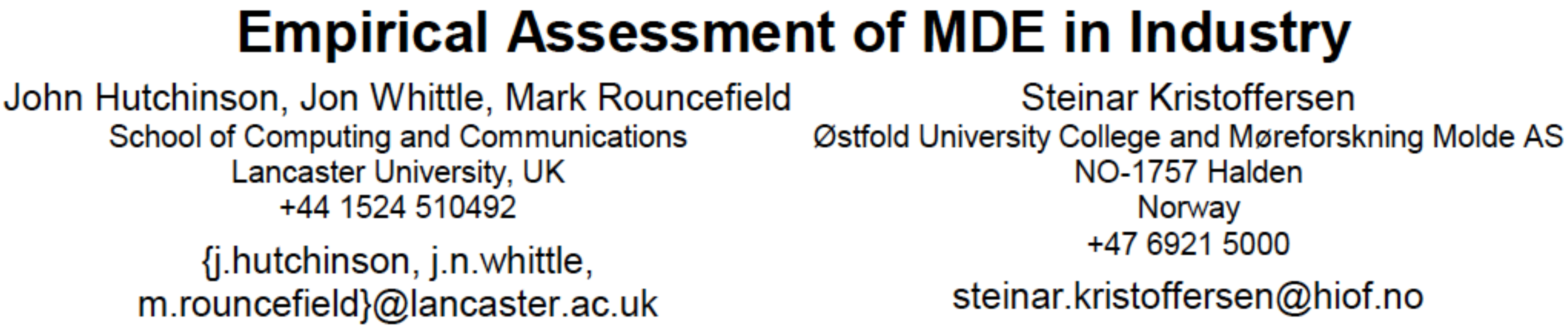
aka **General-Purpose Languages**

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

aka **Domain-Specific Languages**

1972

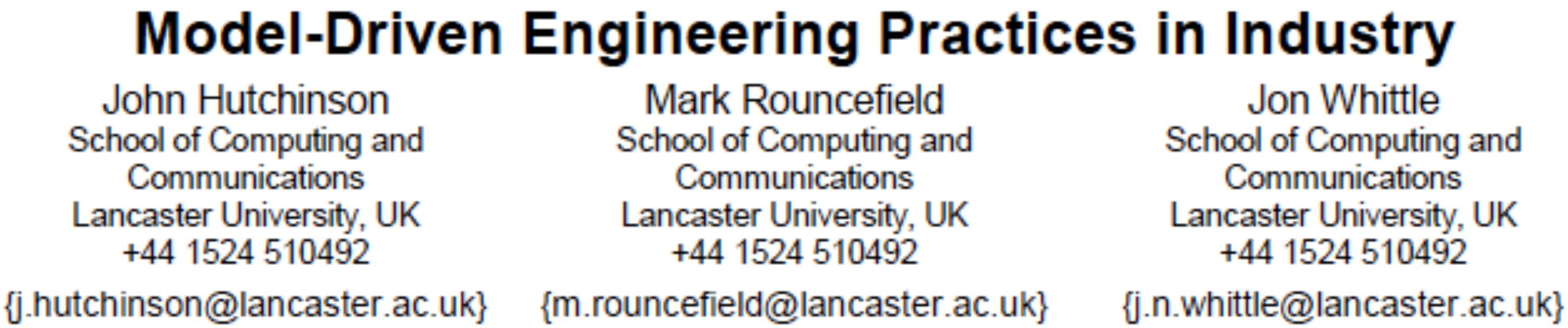
ACM Turing Lecture, « The Humble Programmer »
Edsger W. Dijkstra



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

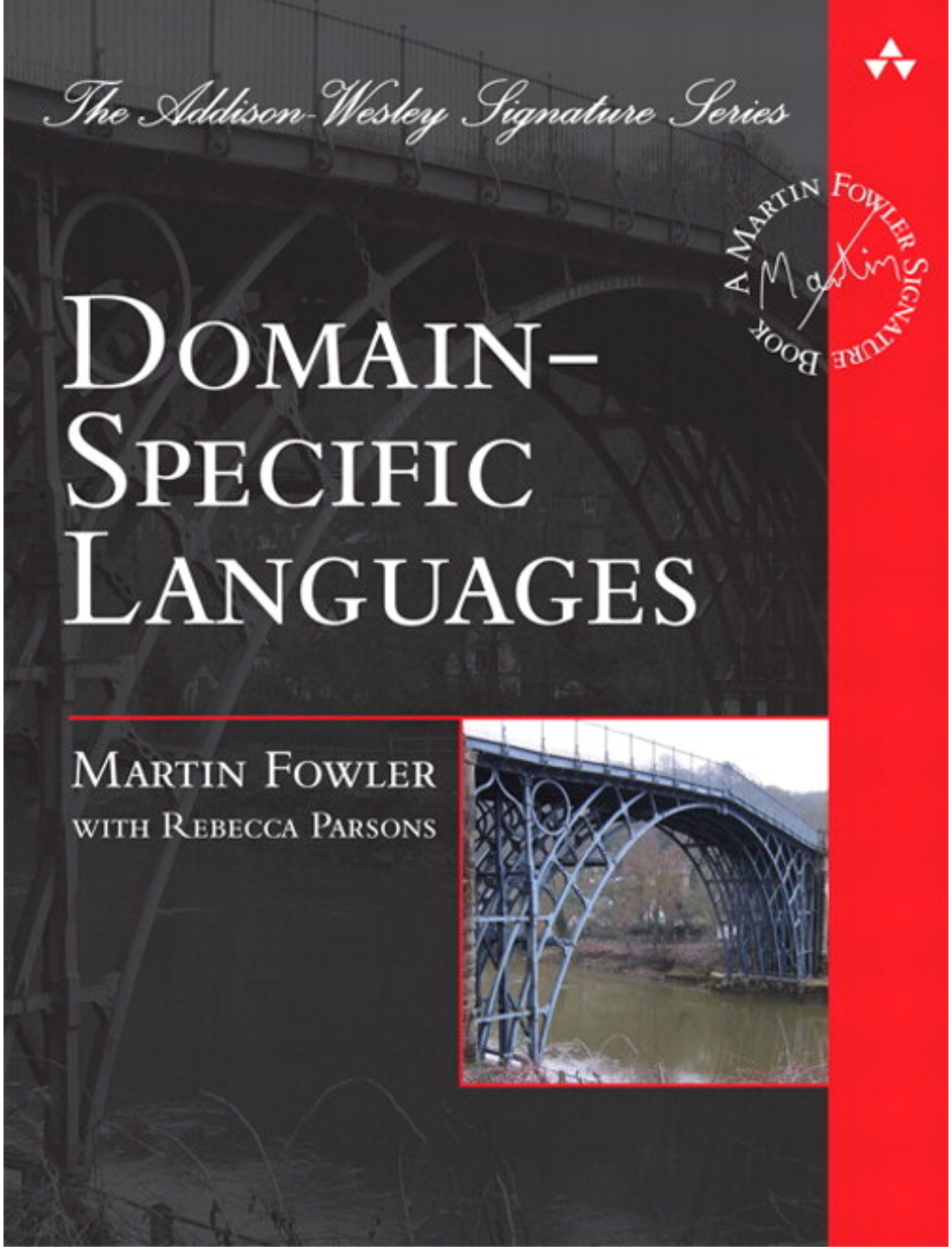
John Hutchinson
School of Computing and Communications
Lancaster University, UK
+44 1524 510492
{j.hutchinson@lancaster.ac.uk}

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
A Martin Fowler Signature Book
Domain-Specific Languages
Martin Fowler
with Rebecca Parsons

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, **<u>rather than a general purpose language</u>** 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.

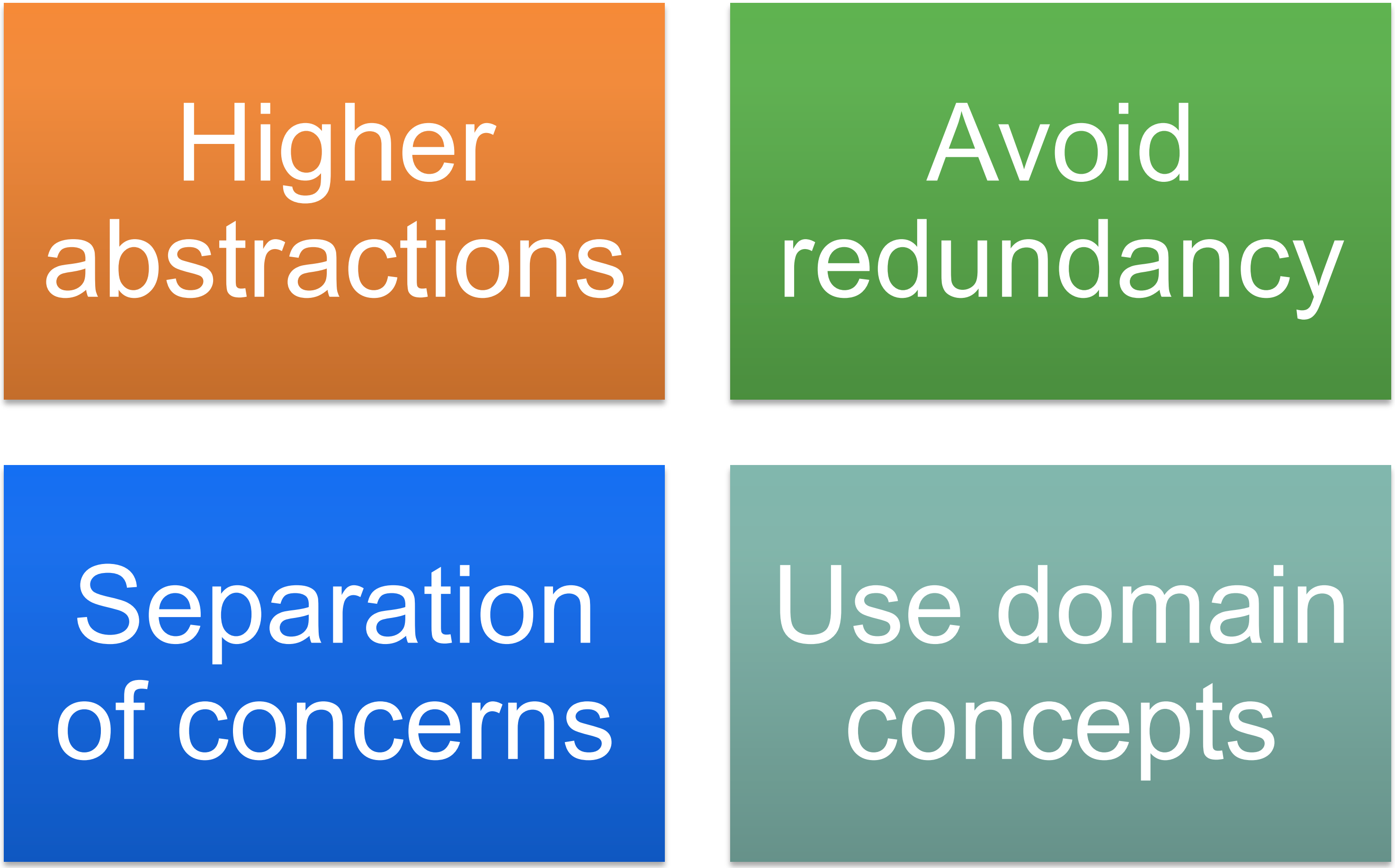
A GPL is a formal notation that can be used to describe problem solutions in a precise manner.

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



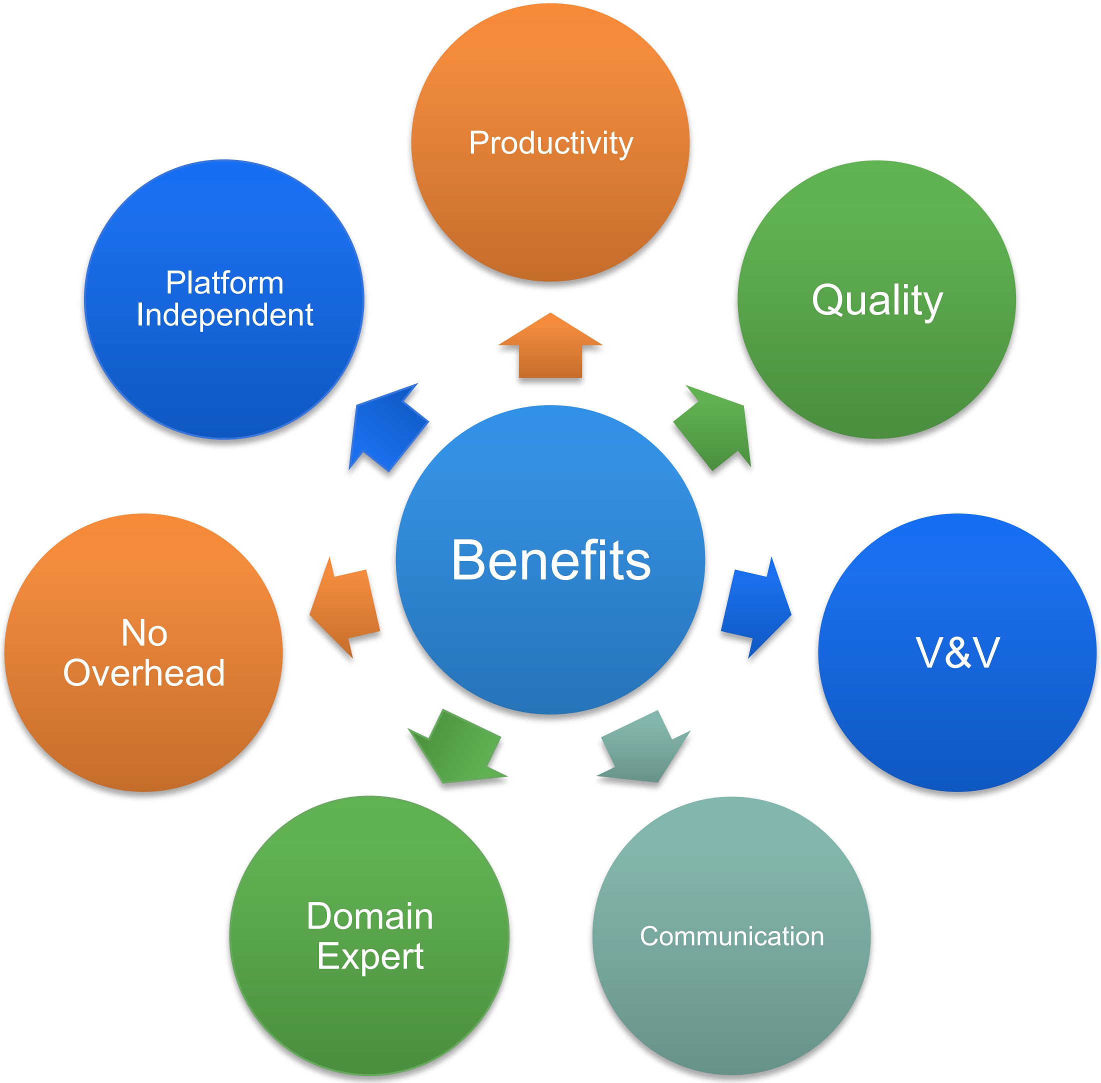
Higher abstractions

Avoid redundancy

Separation of concerns

Use domain concepts

Promises of domain-specific languages



Benefits

Productivity

Quality

V&V

Communication

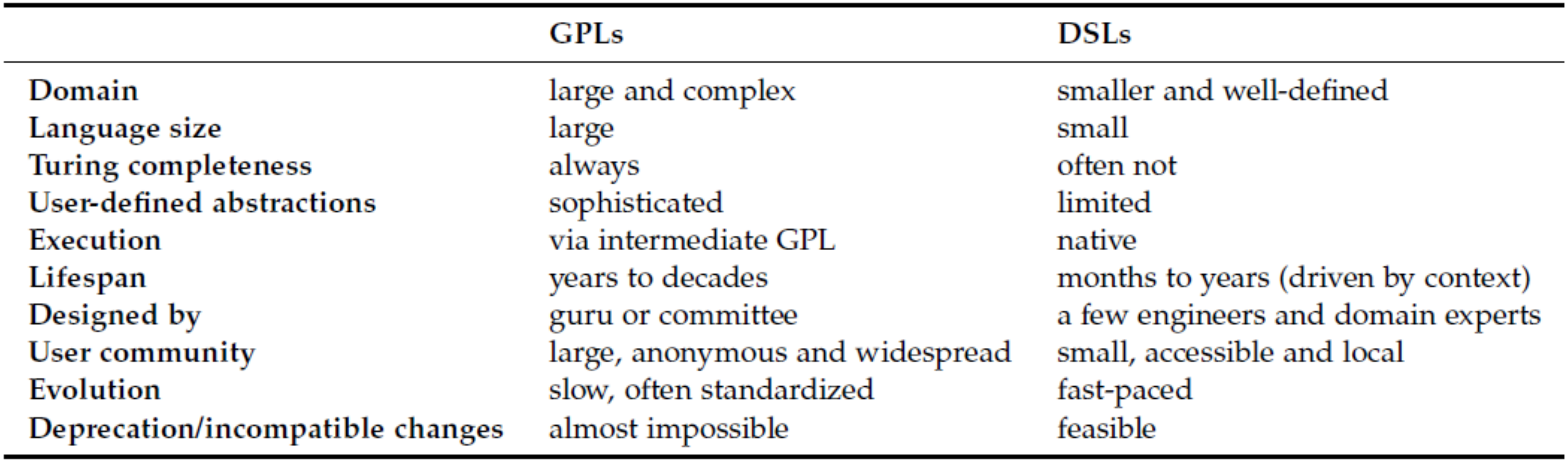
Domain Expert

No Overhead

Platform Independent

GeneralPL vs DomainSL

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



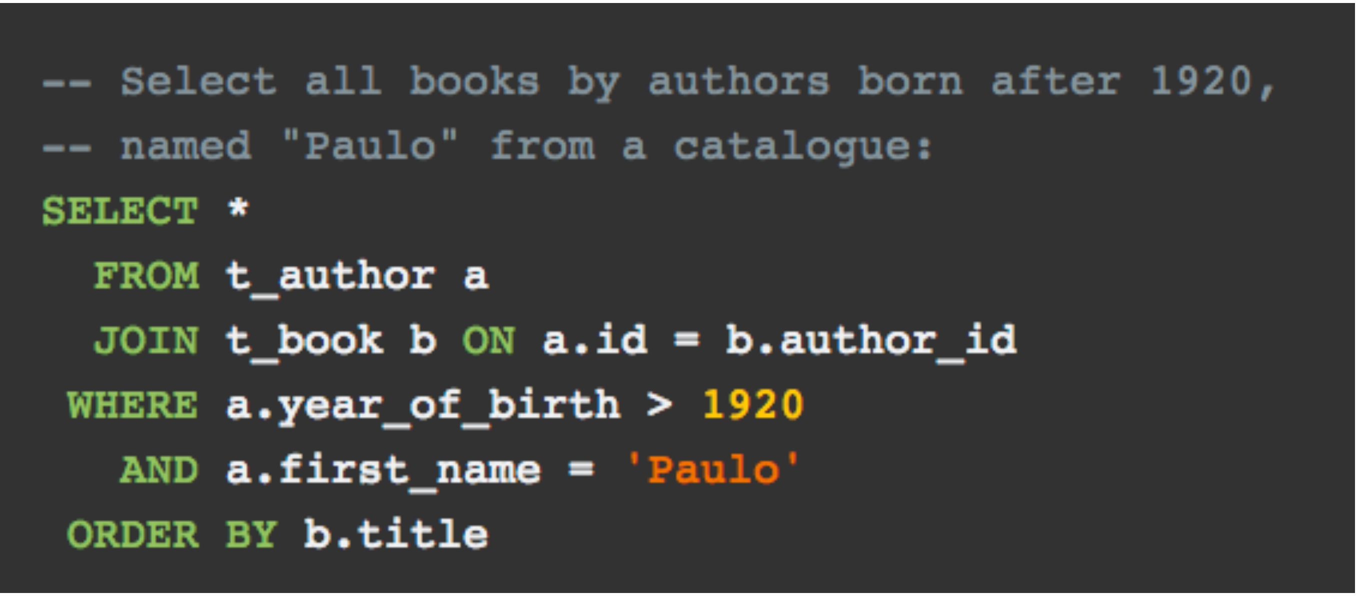
	GPLs	DSLs
Domain	large and complex	smaller and well-defined
Language size	large	small
Turing completeness	always	often not
User-defined abstractions	sophisticated	limited
Execution	via intermediate GPL	native
Lifespan	years to decades	months to years (driven by context)
Designed by	guru or committee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

External DSLs vs Internal DSLs

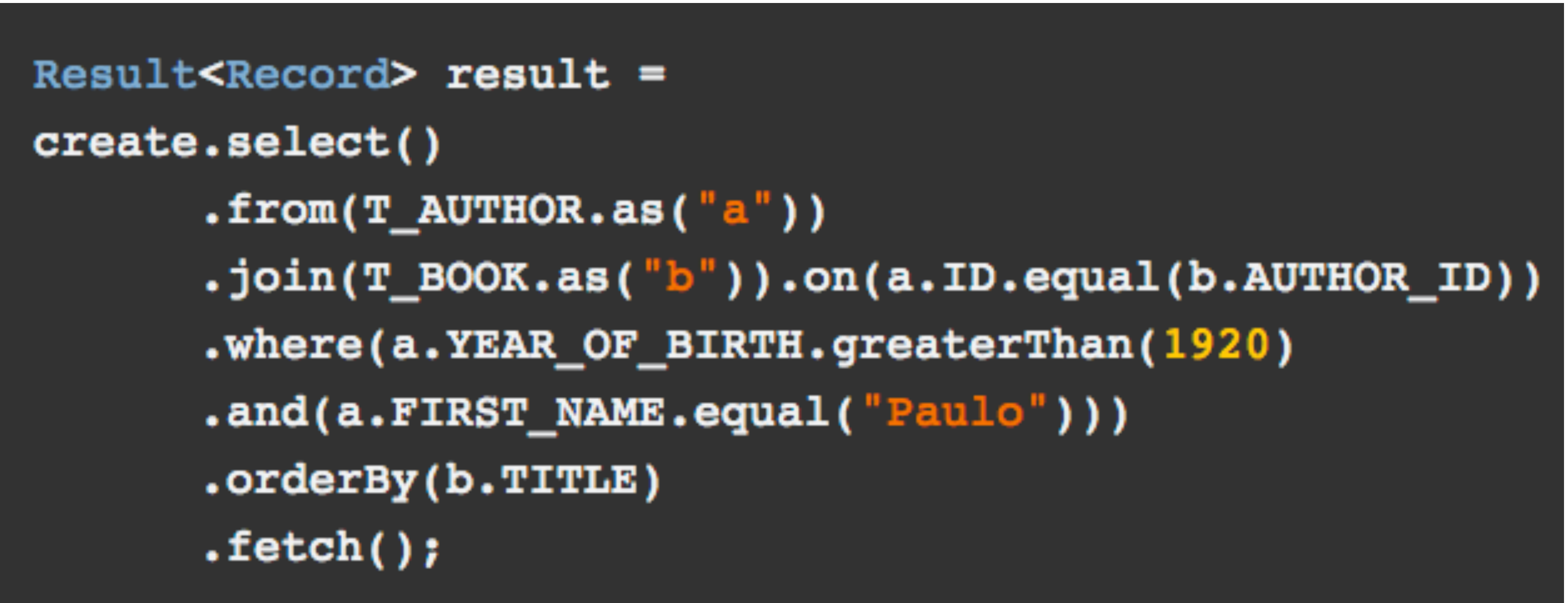
- An **external** DSL is a completely separate language and has its own custom syntax/ tooling support (e.g., editor)

- An internal DSL is more or less a set of APIs written on top of a host language (e.g., Java).
  - Fluent interfaces

External vs Internal DSL (SQL example)

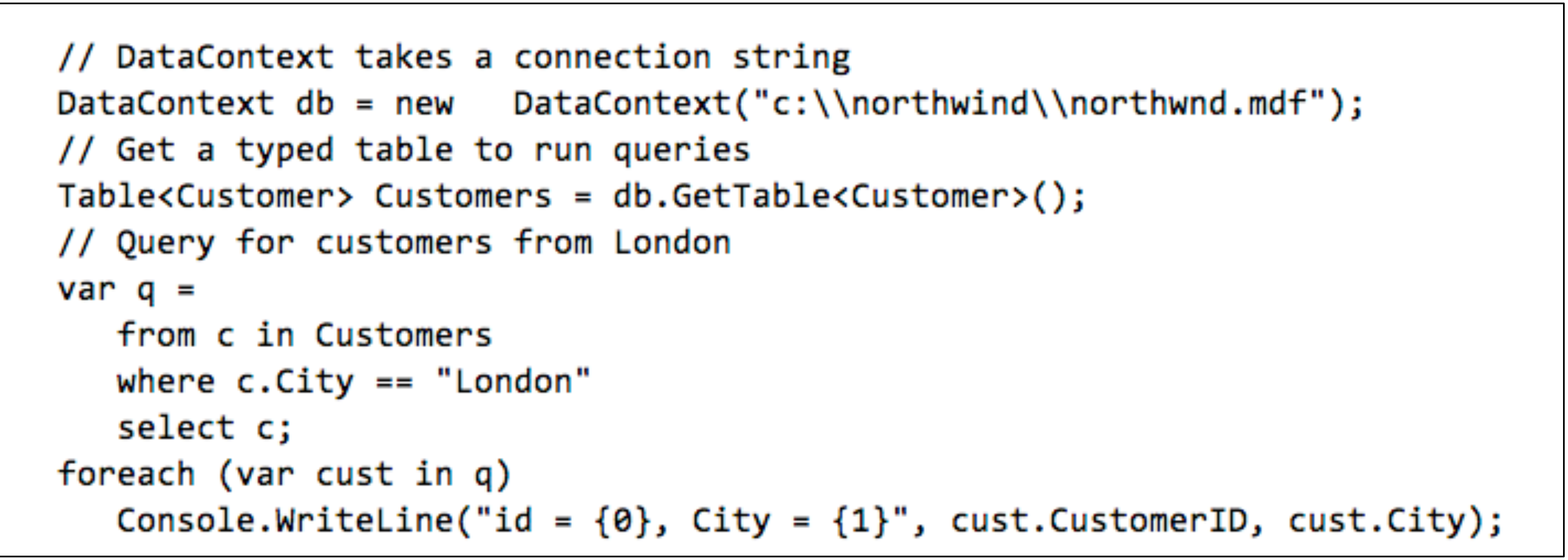


```
-- Select all books by authors born after 1920,
-- named "Paulo" from a catalogue:
SELECT *
  FROM t_author a
  JOIN t_book b ON a.id = b.author_id
 WHERE a.year_of_birth > 1920
   AND a.first_name = 'Paulo'
 ORDER BY b.title
```

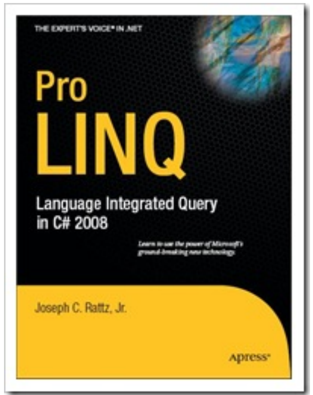


```
Result<Record> result =
create.select()
      .from(T_AUTHOR.as("a"))
      .join(T_BOOK.as("b")).on(a.ID.equal(b.AUTHOR_ID))
      .where(a.YEAR_OF_BIRTH.greaterThan(1920)
      .and(a.FIRST_NAME.equal("Paulo")))
      .orderBy(b.TITLE)
      .fetch();
```

Internal DSL (LINQ/C# example)



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



THE EXPERT'S VOICE® IN .NET

Pro

LINQ

Language Integrated Query
in C# 2008

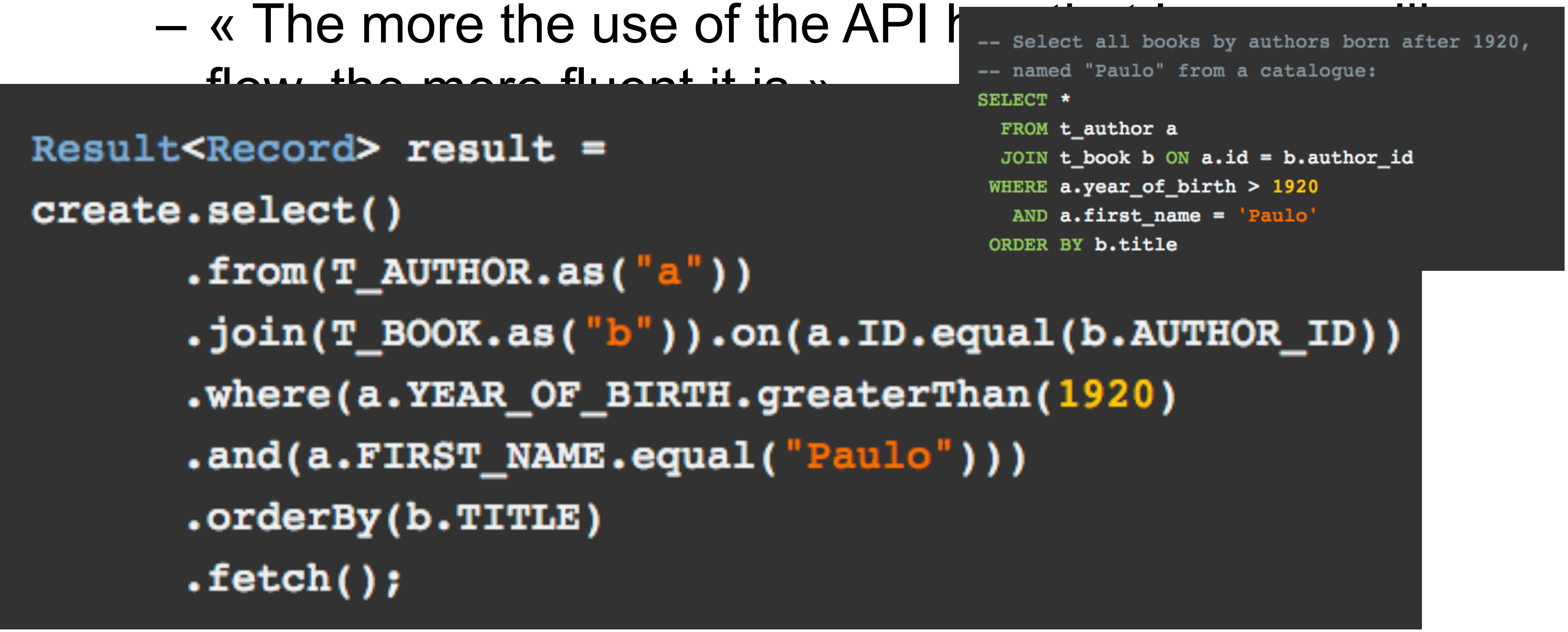
Learn to use the power of Microsoft's ground-breaking new technology.

Joseph C. Rattz, Jr.

Apress®

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 h
    flow, the more fluent it is »

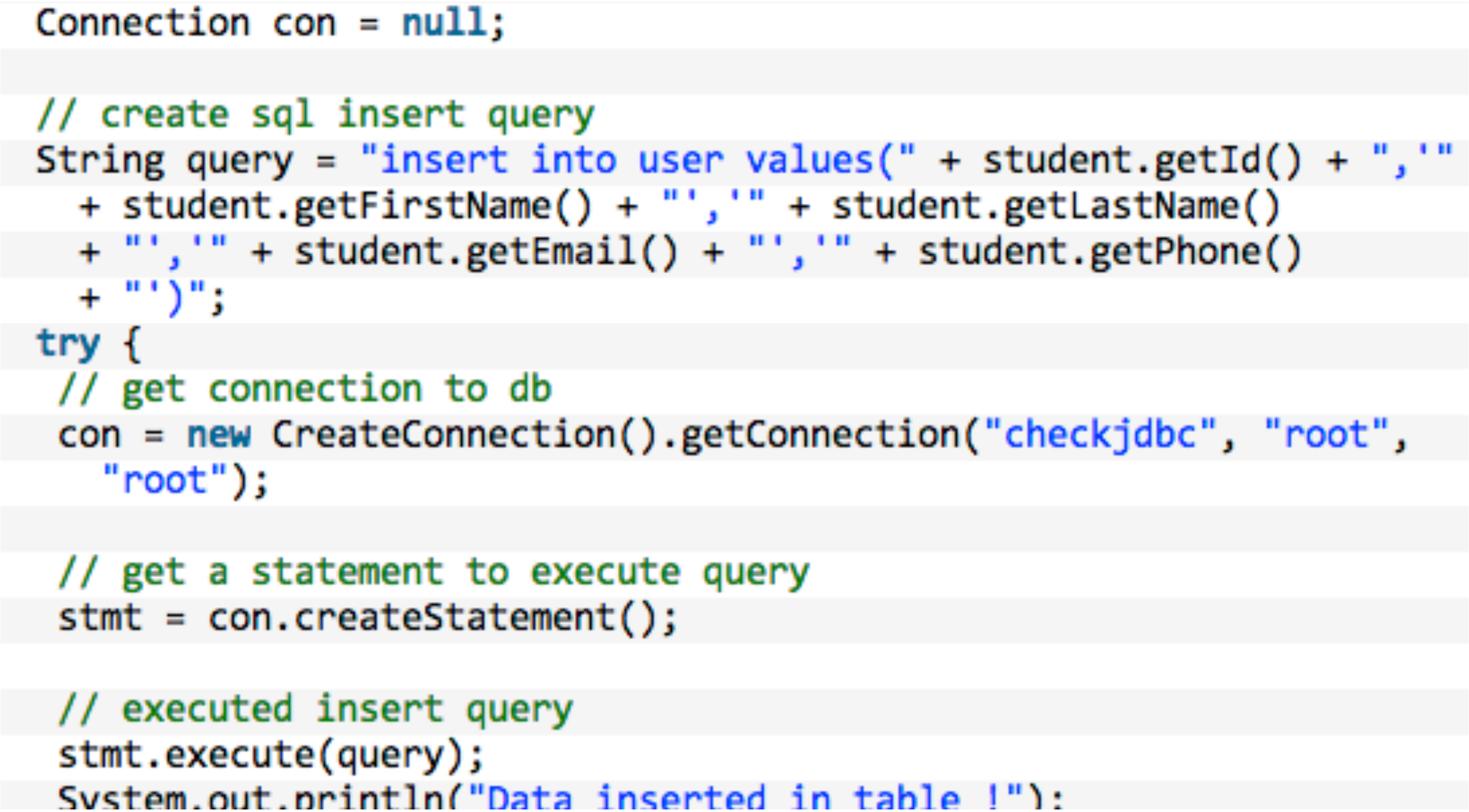


```
-- Select all books by authors born after 1920,
-- named "Paulo" from a catalogue:
SELECT *
  FROM t_author a
  JOIN t_book b ON a.id = b.author_id
 WHERE a.year_of_birth > 1920
   AND a.first_name = 'Paulo'
 ORDER BY b.title
```

```
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();
```

SQL in… Java

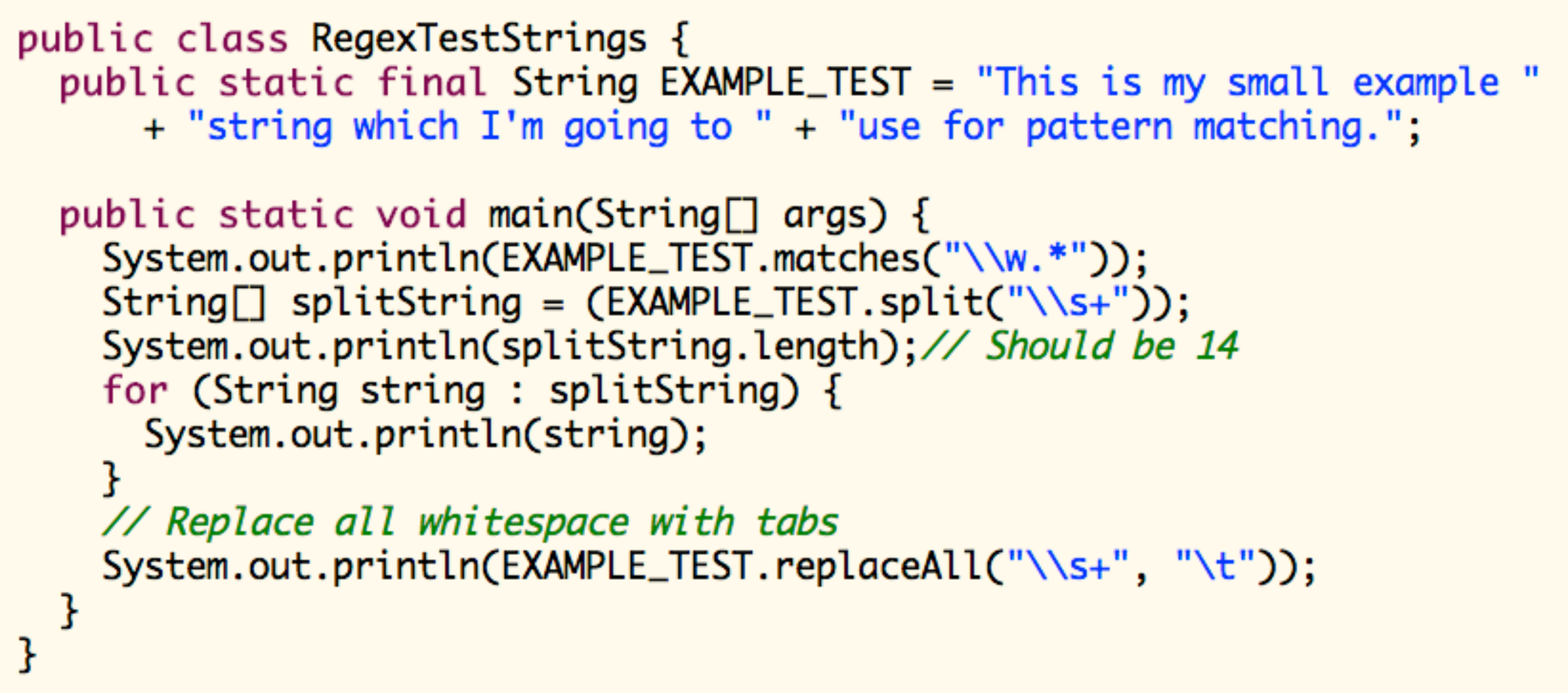
DSL in GPL



```
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)

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

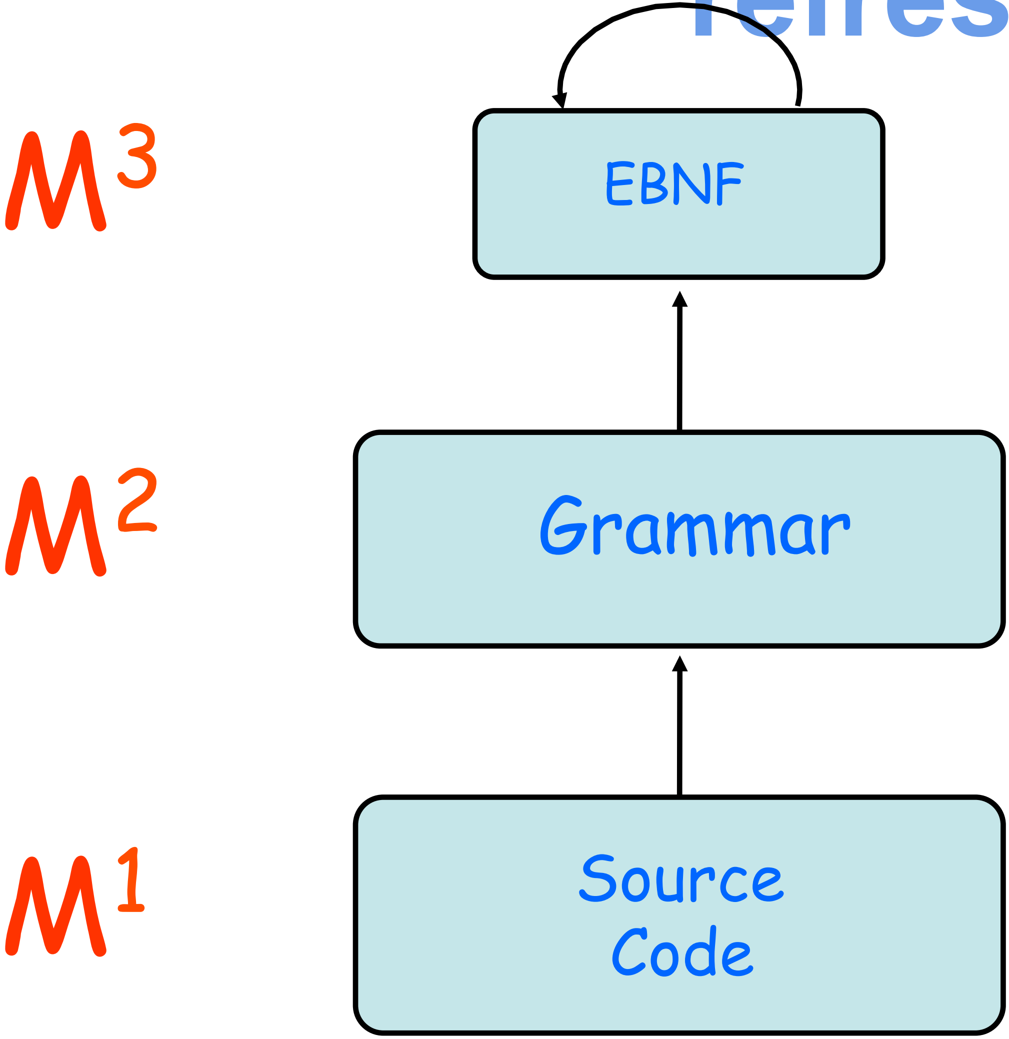
- Foundations and practice of Xtext
  - State-of-the-art language workbench (Most Innovative Eclipse Project in 2010, mature and used in a variety of industries)

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

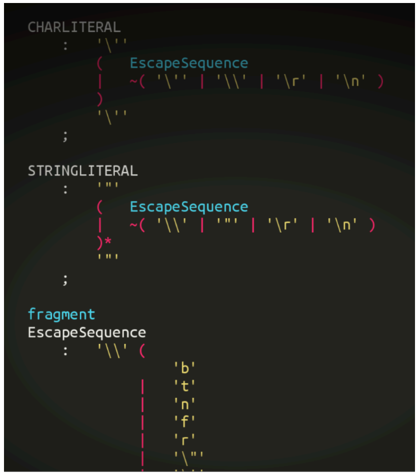
Xtext, a popular, easy-to-use model-based tool for developping DSLs

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

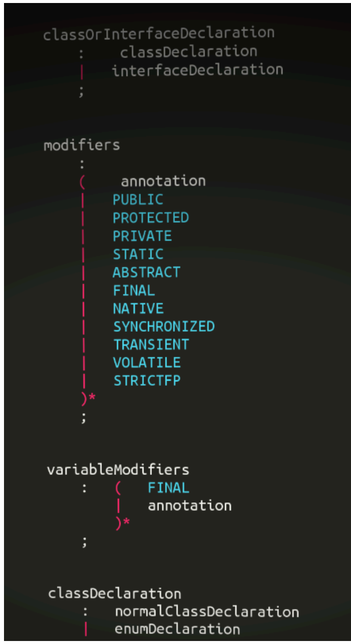
Foundations (or some course refresh)



Java Grammar

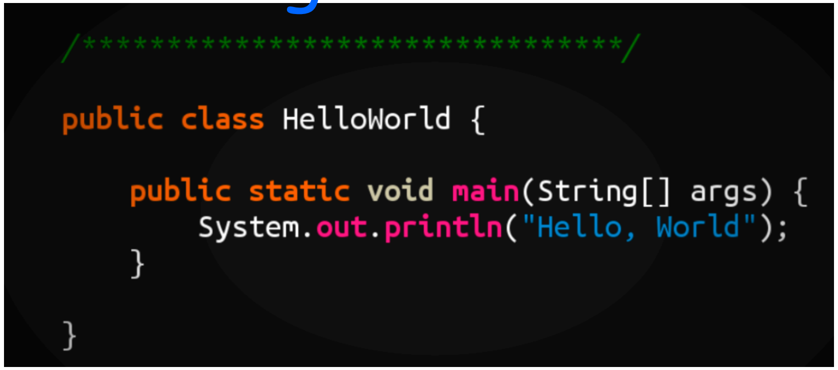


```
CHARLITERAL
    :   '\''
        (   EscapeSequence
        |   ~( '\'' | '\\' | '\r' | '\n' )
        )
        '\''
    ;

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

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



```
classOrInterfaceDeclaration
    :    classDeclaration
    |   interfaceDeclaration
    ;


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


variableModifiers
    :   (   FINAL
        |   annotation
        )*
    ;


classDeclaration
    :   normalClassDeclaration
    |   enumDeclaration
```

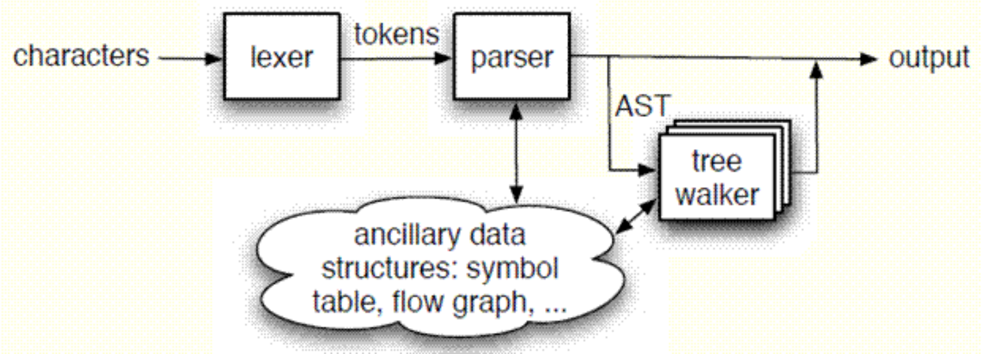
Java Program

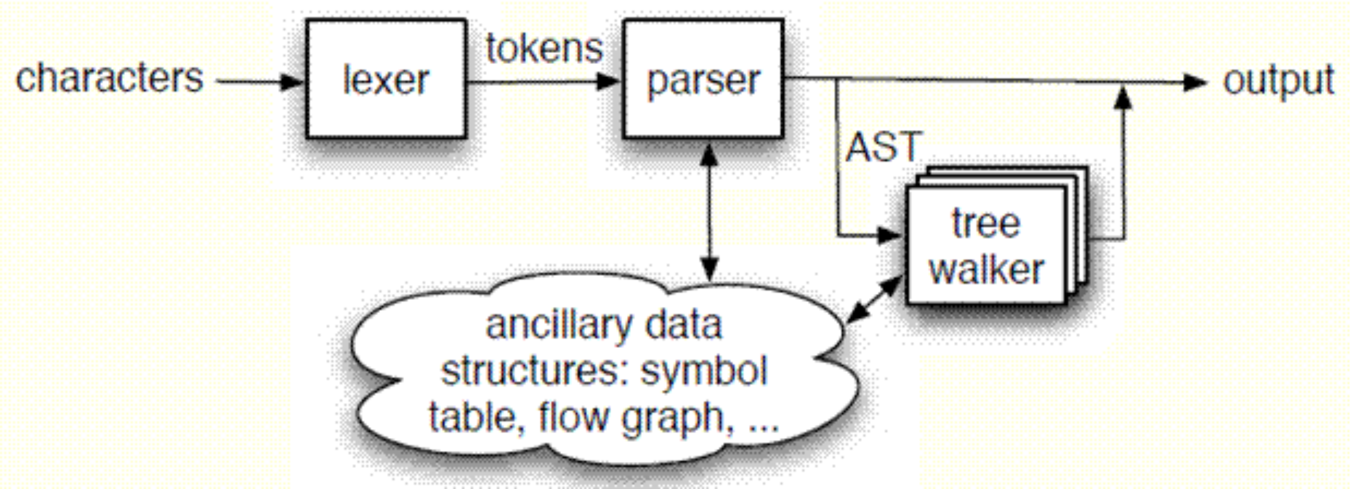


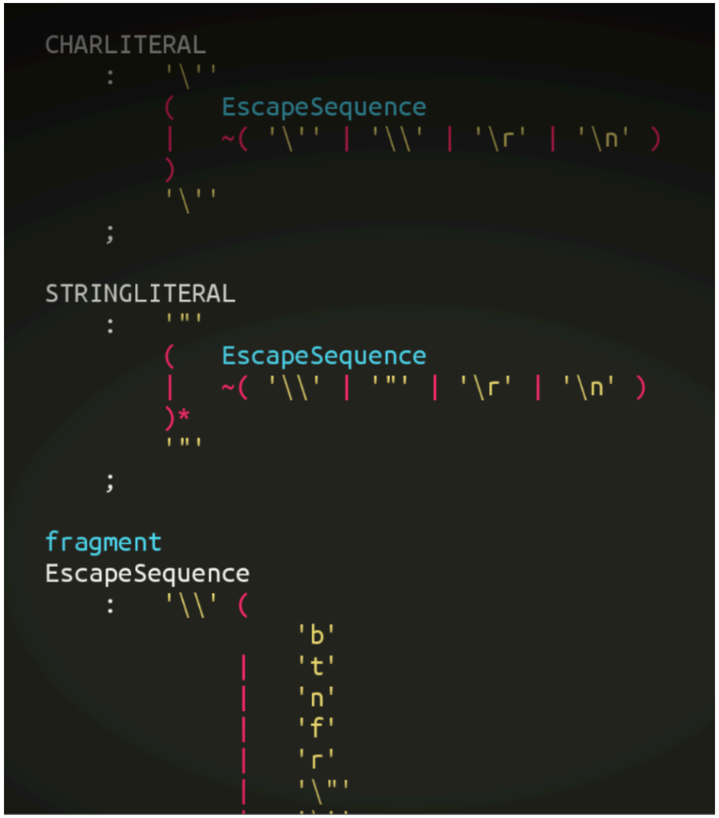
```
/********************************/

public class HelloWorld {

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

}
```

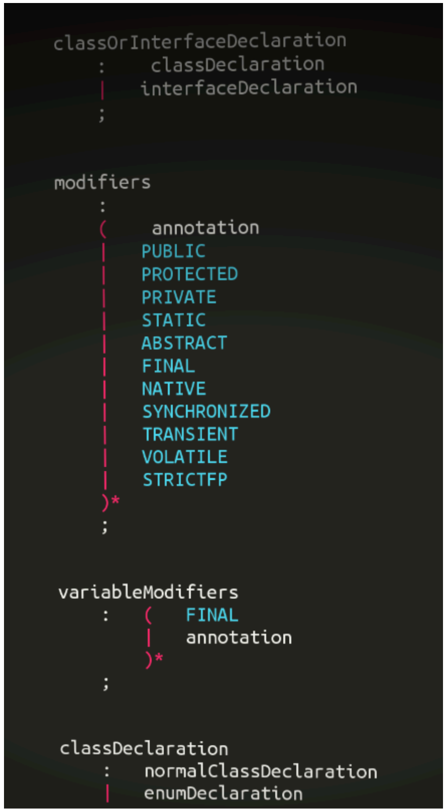
Compilation Process

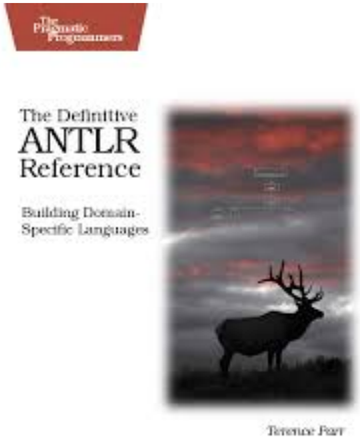
- Source code
  - – Concrete syntax used for specifying a program
  - – Conformant to a grammar
- Lexical analysis
  - – Conveting a sequence of characters into a sequence of **tokens**
- Parsing (Syntactical analysis)
  - – Abtsract Syntax Tree (AST)

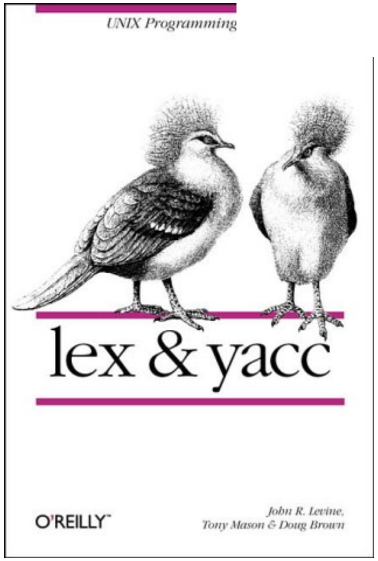





```
CHARLITERAL
    :   '\''
        (   EscapeSequence
        |   ~( '\'' | '\\' | '\r' | '\n' )
        )
        '\''
    ;

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

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


```
classOrInterfaceDeclaration
    :    classDeclaration
    |   interfaceDeclaration
    ;


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


variableModifiers
    :   (   FINAL
        |   annotation
        )*
    ;


classDeclaration
    :   normalClassDeclaration
    |   enumDeclaration
```

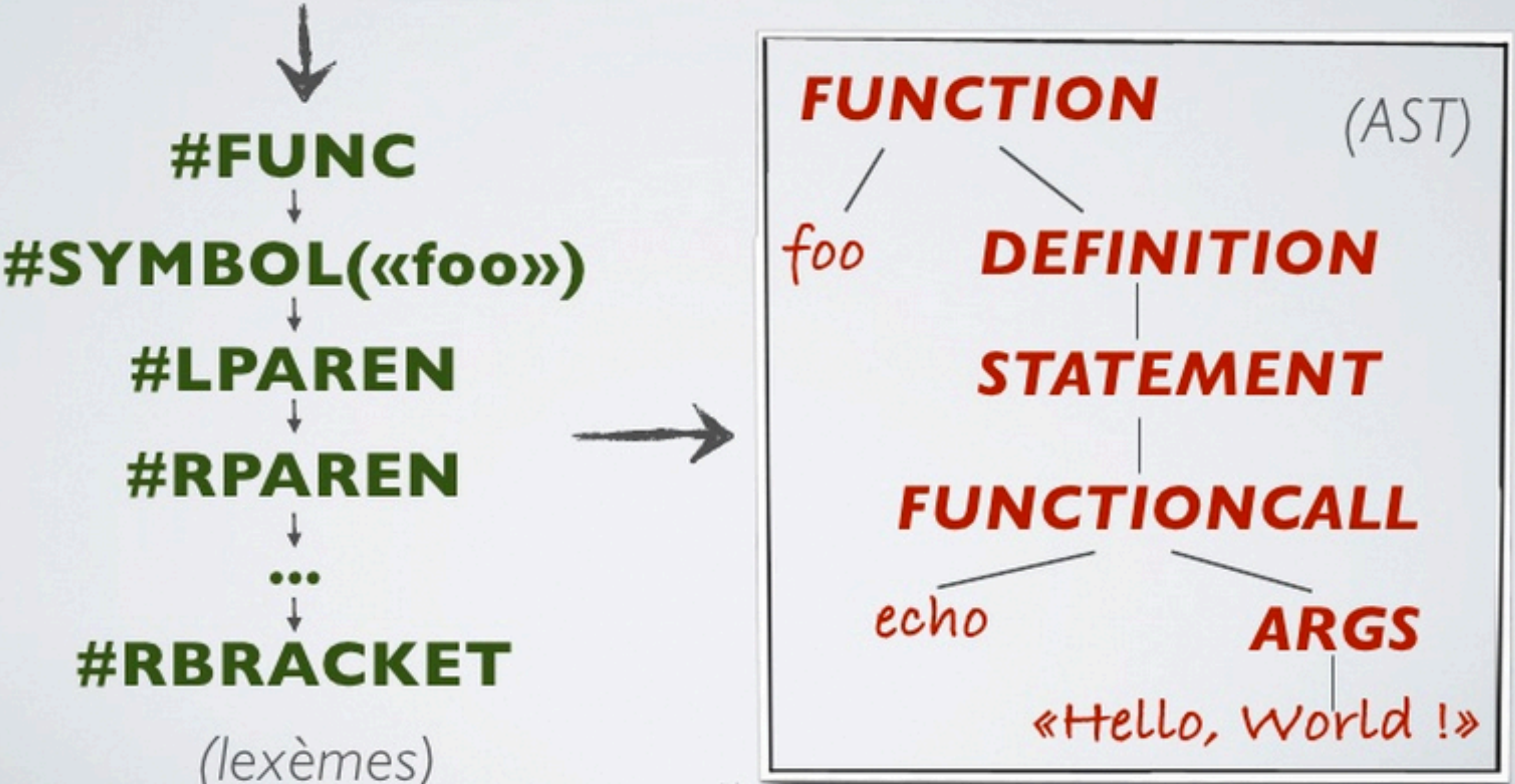




EXEMPLE

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

(Syntaxe concrète)



#FUNC

#SYMBOL(«foo»)

#LPAREN

#RPAREN

...

#RBRACKET

(lexèmes)

FUNCTION

(AST)

foo

DEFINITION

STATEMENT

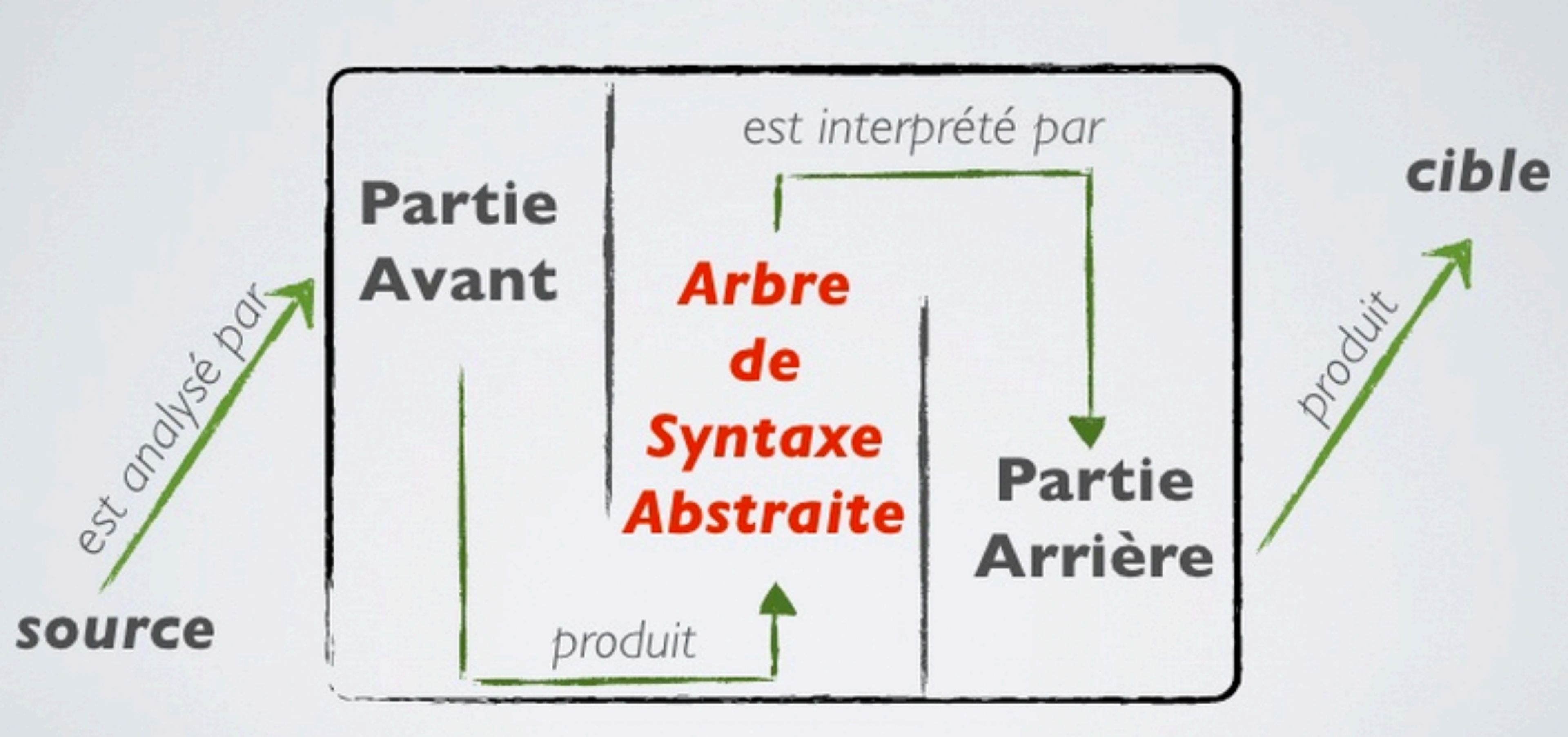
FUNCTIONCALL

echo

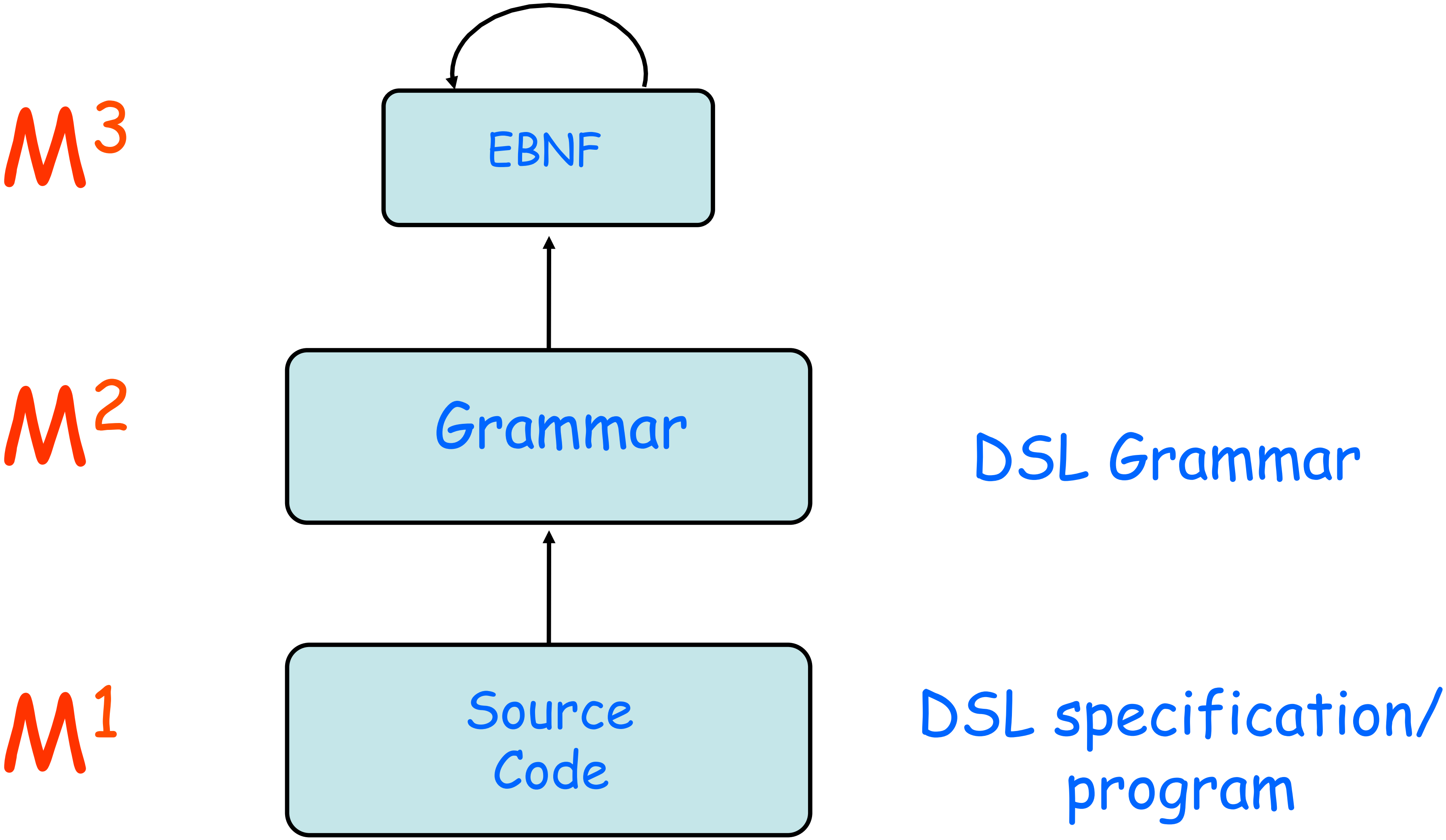
ARGS

«Hello, World !»

Compilation (en français)



DSL? The same!



M3

EBNF

M2

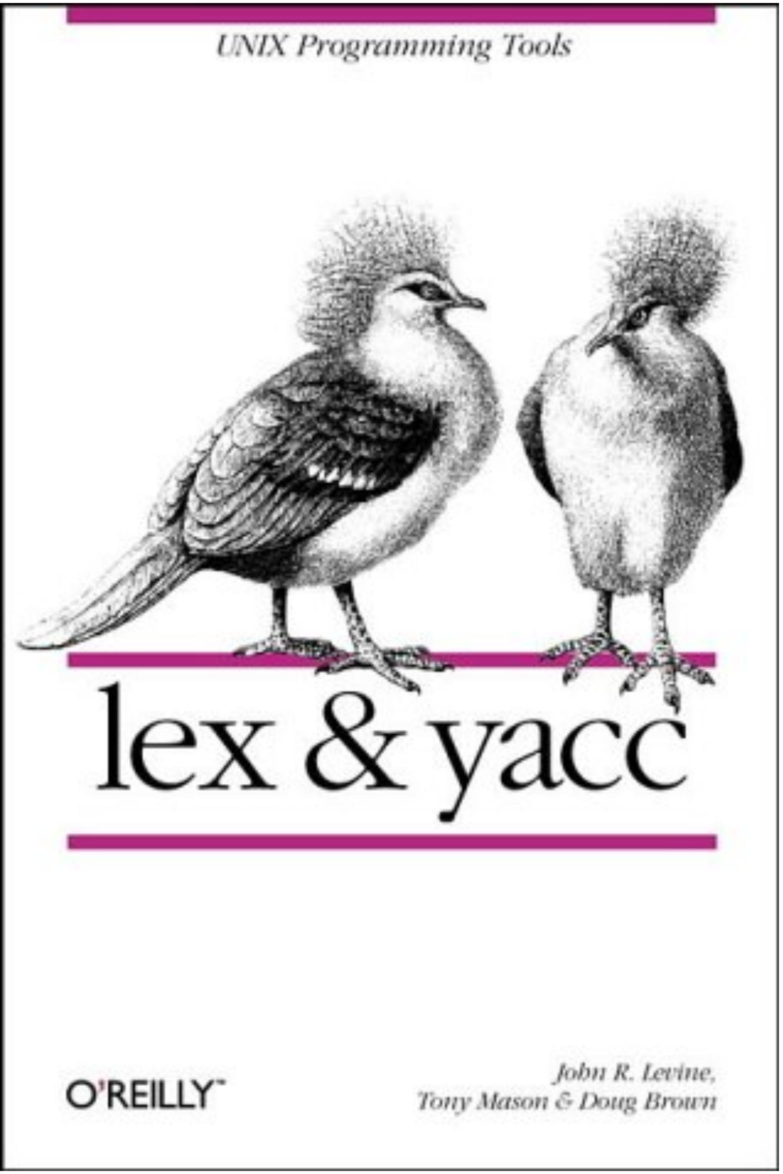
Grammar

DSL Grammar

M1

Source Code

DSL specification/ program

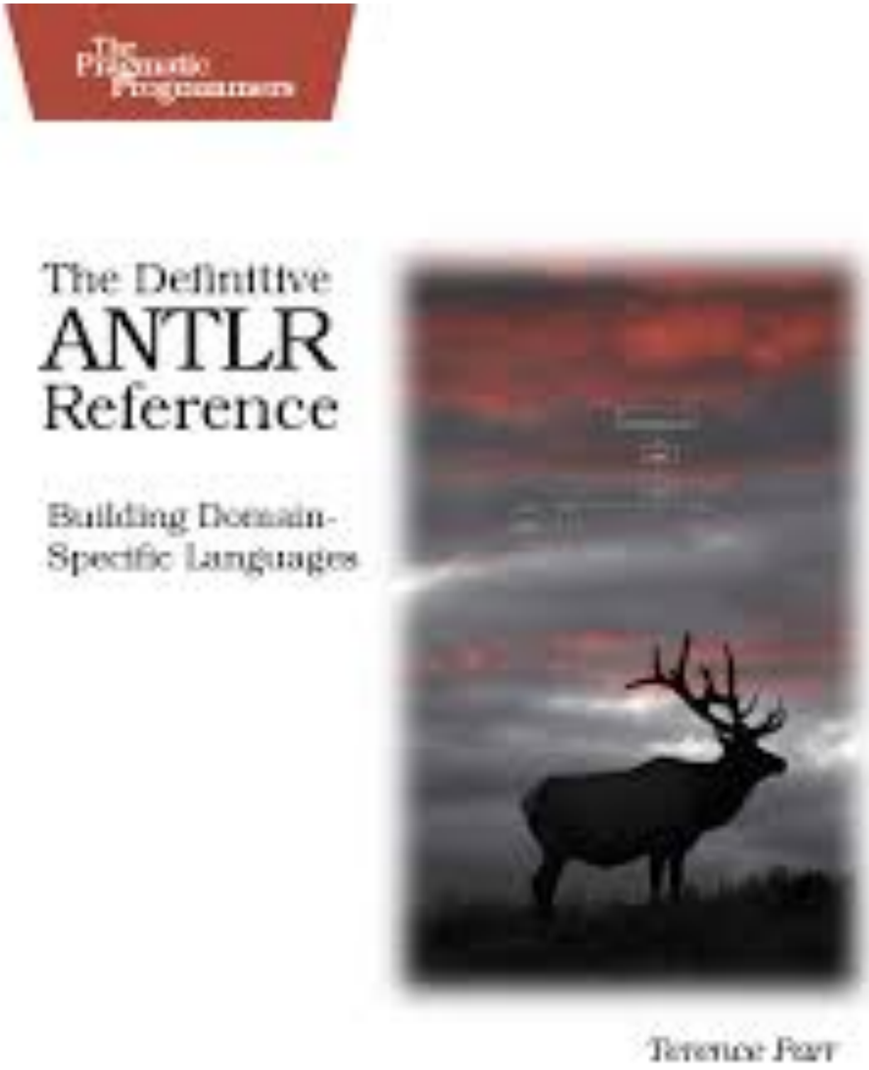


UNIX Programming Tools

lex & yacc

O'REILLY™

John R. Levine,
Tony Mason & Doug Brown

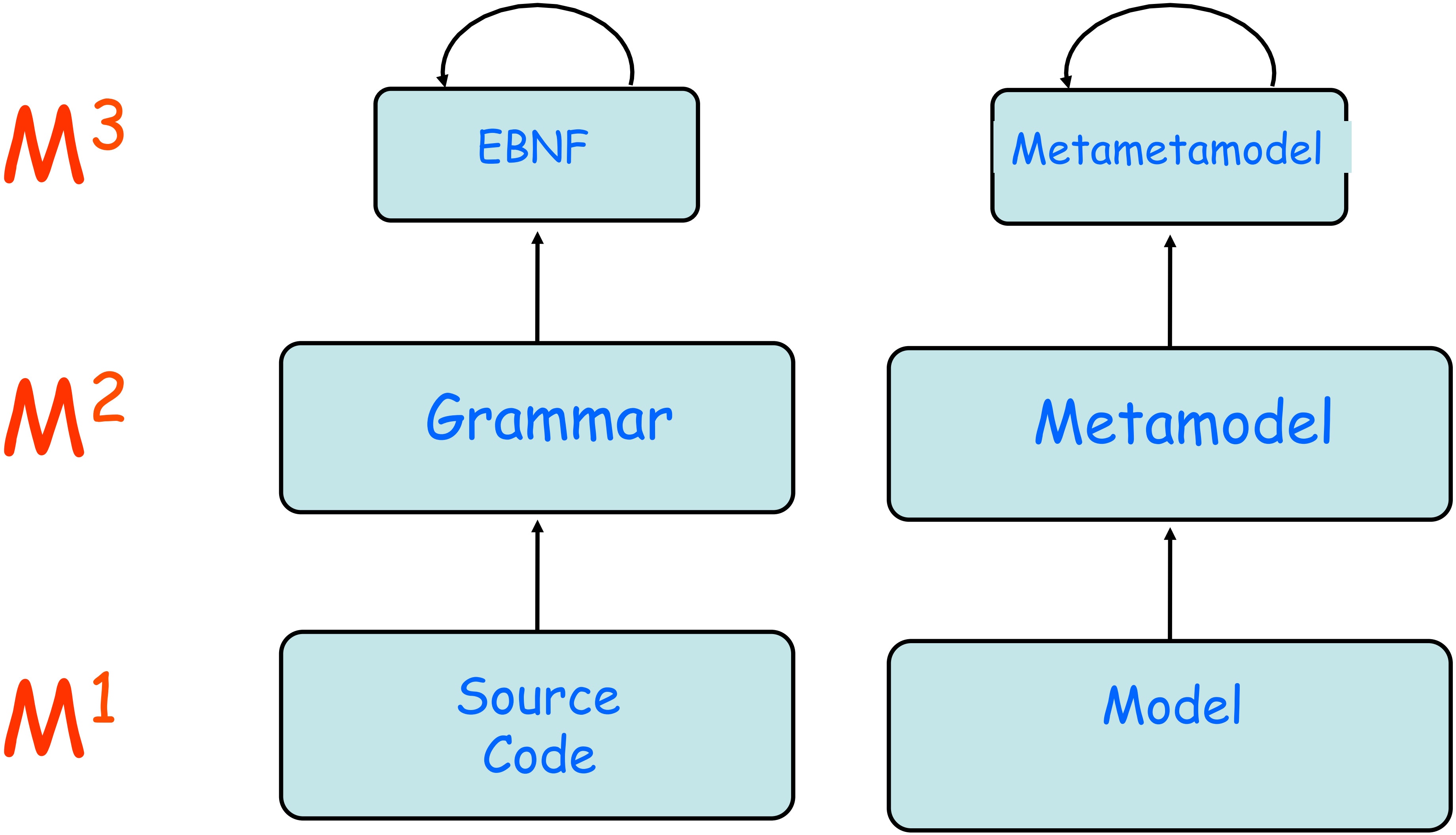


The Pragmatic Programmers

The Definitive ANTLR Reference

Building Domain-Specific Languages

Terence Parr

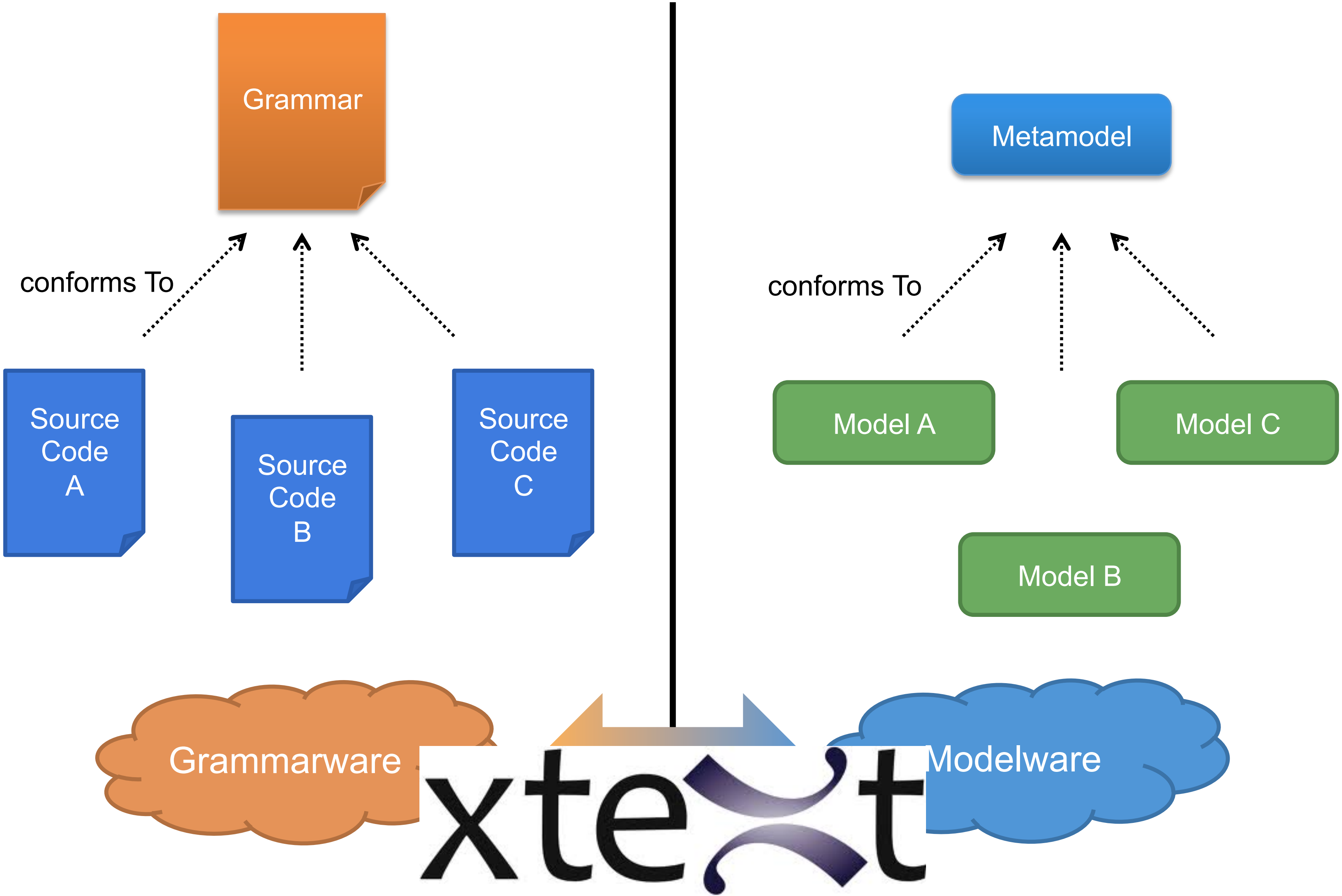


M^3 EBNF Metametamodel

M^2 Grammar Metamodel

M^1 Source Code Model

Language and MDE



xtext

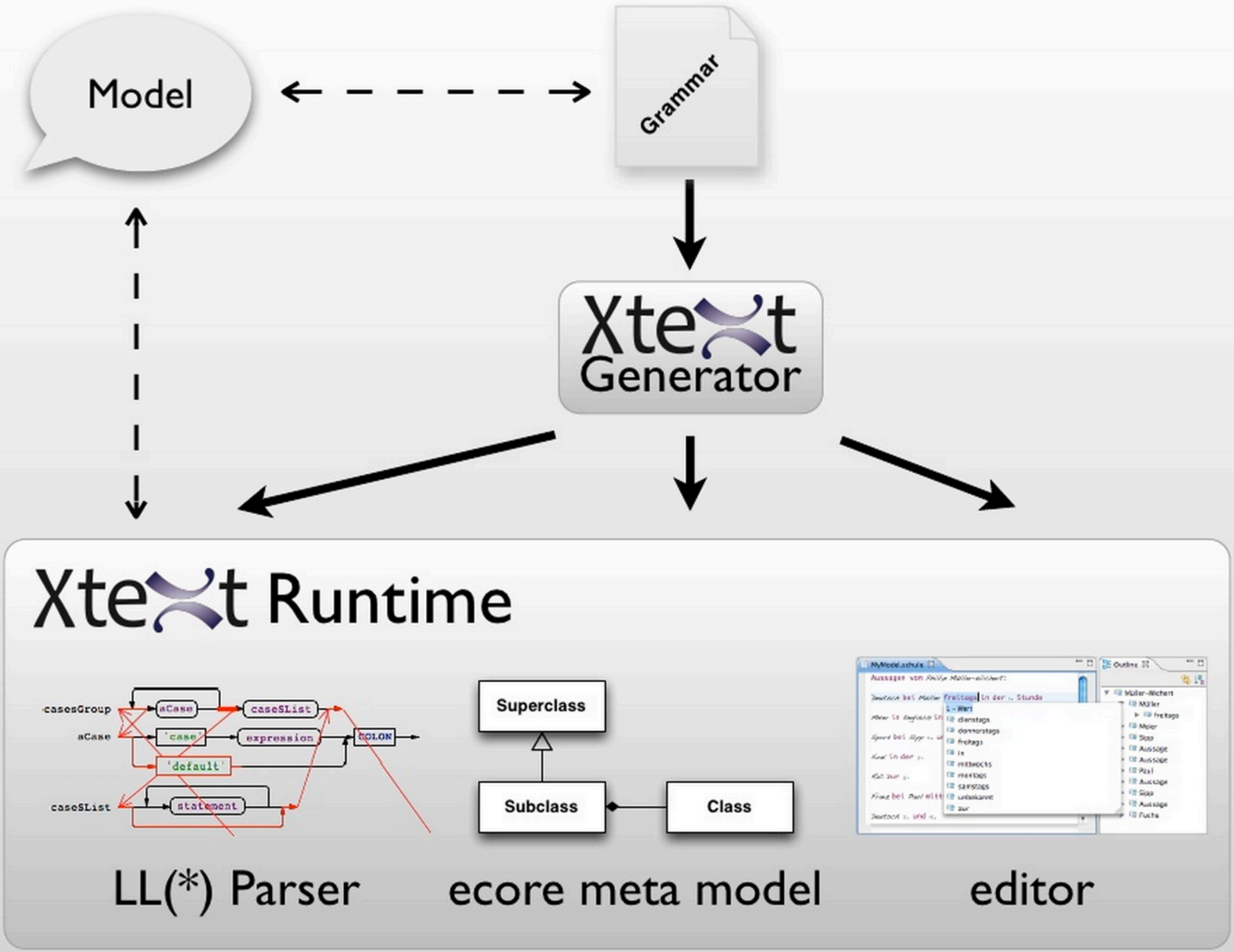
Give me a **grammar**,

I’ll give you (for free)

 * a comprehensive editor (auto-completion, syntax highlitening, etc.) in Eclipse

 * an Ecore metamodel and facilities to load/serialize/visit conformant models (Java ecosystem)

 * extension to override/extend « default » facilities (e.g., checker)



Model

Grammar

Xtext Generator

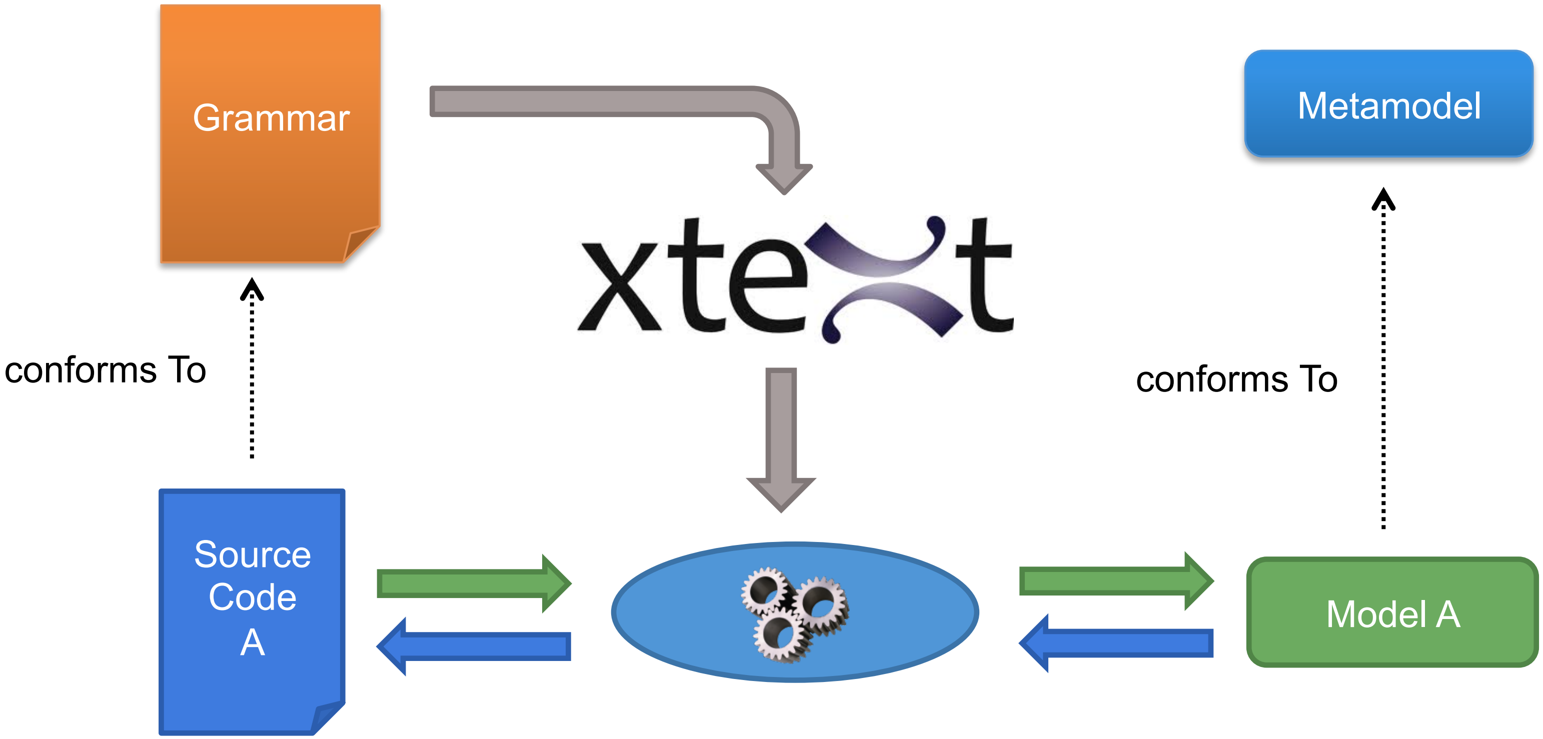
Xtext Runtime

LL(*) Parser

ecore meta model

editor

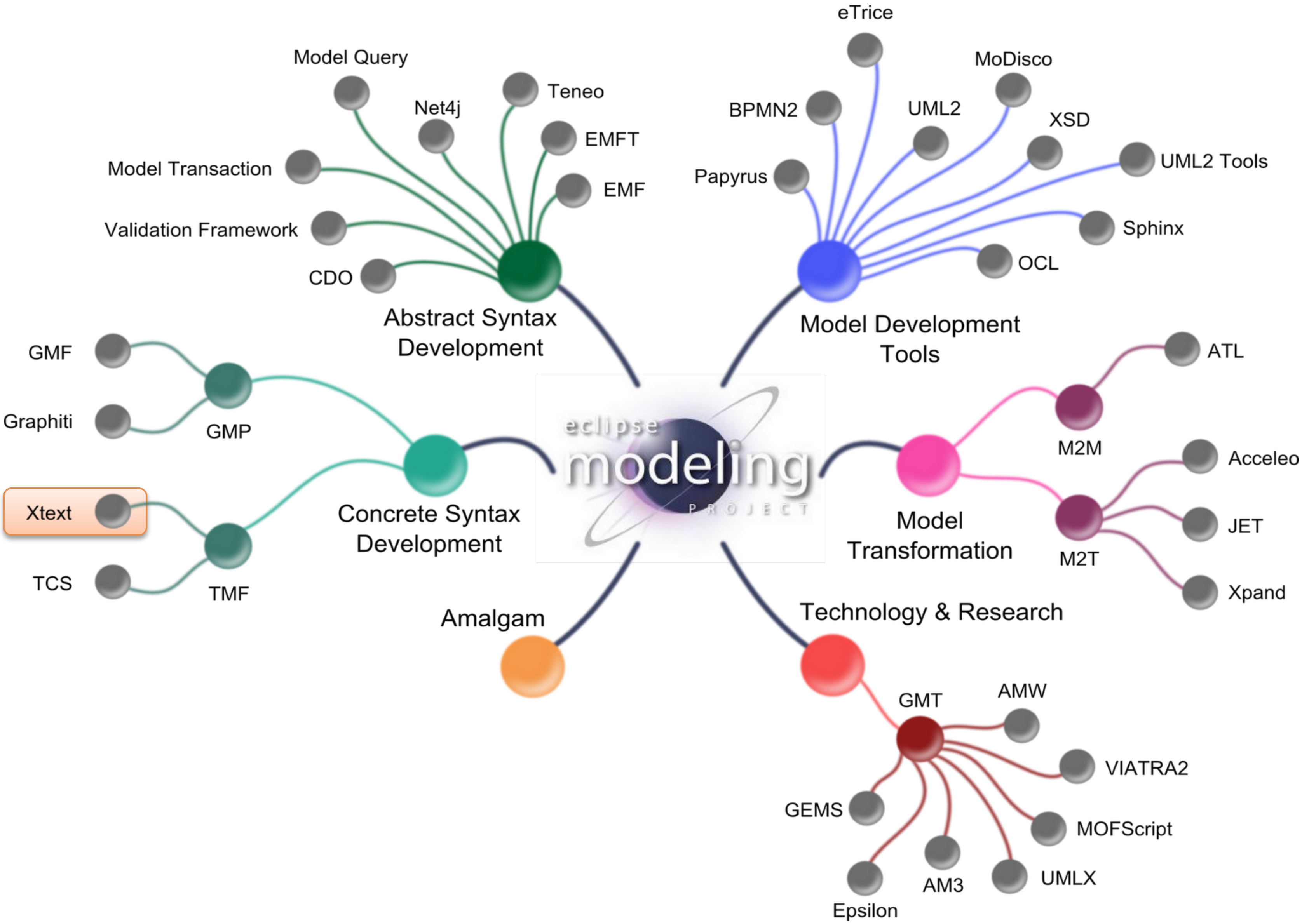
Xtext, Grammar, Metamodel



Xtext Project

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

Eclipse Modeling Project



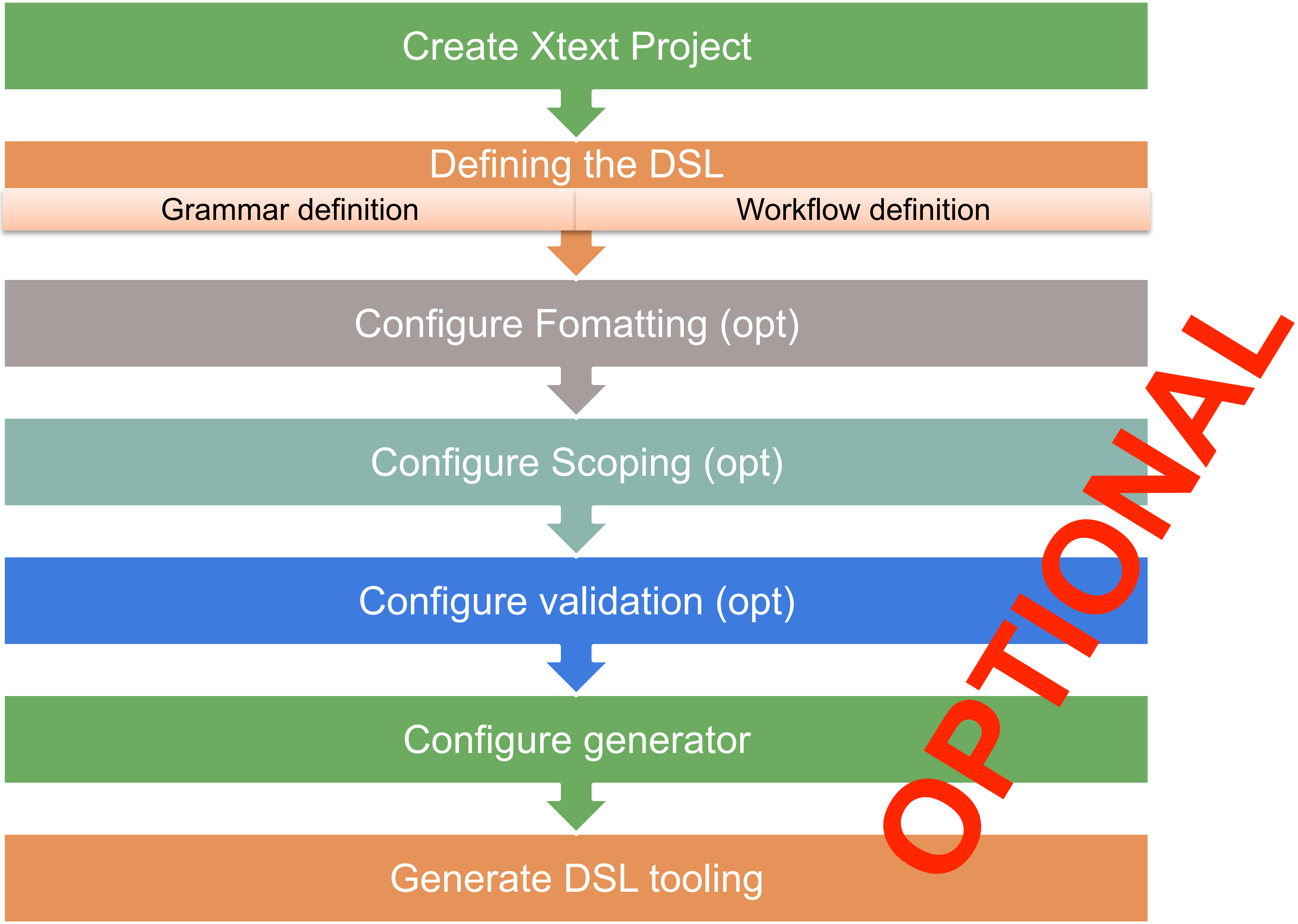
The Grammar Language of Xtext

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

Main Advantages

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

Development Process



Create Xtext Project

Defining the DSL

Grammar definition

Workflow definition

Configure Fomatting (opt)

Configure Scoping (opt)

Configure validation (opt)

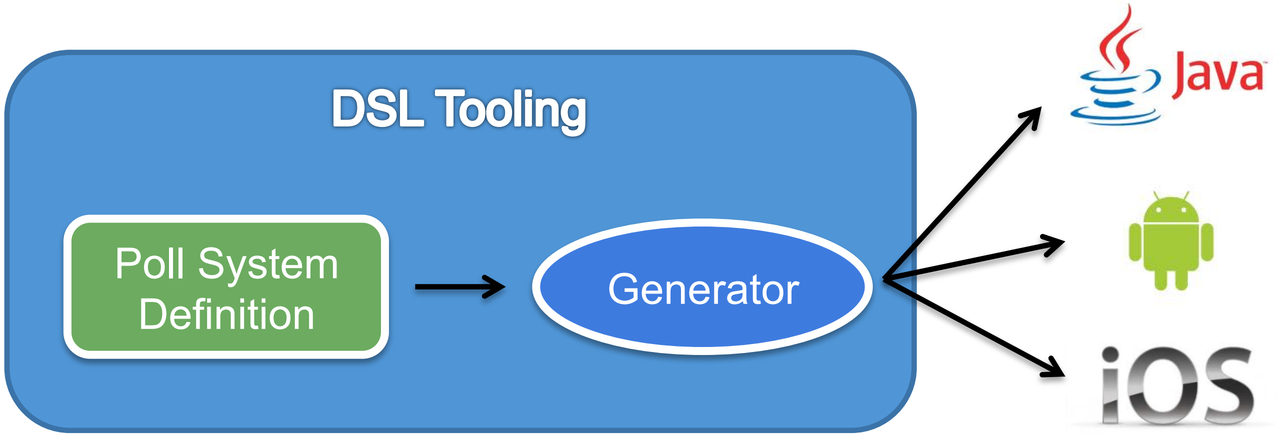
Configure generator

Generate DSL tooling

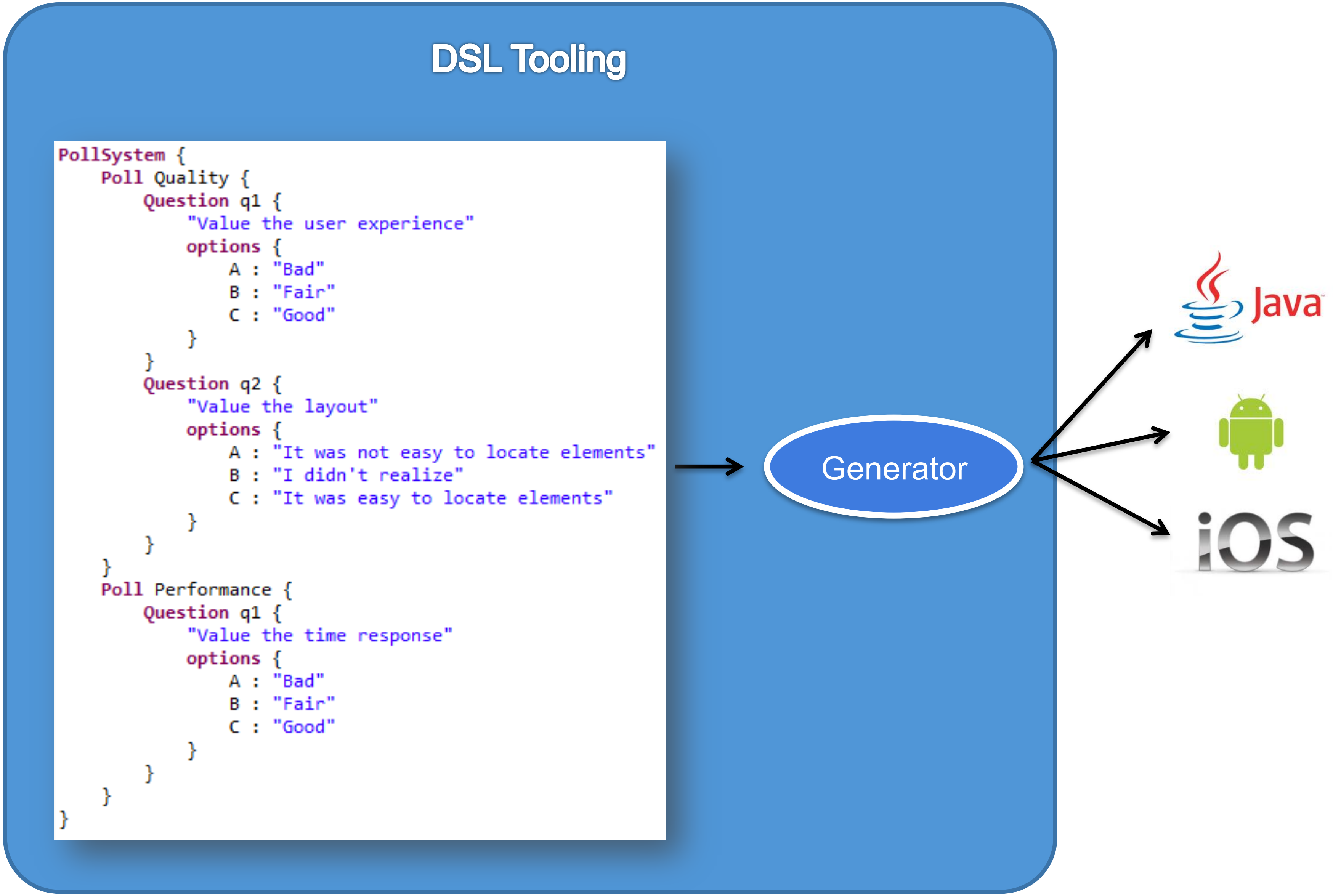
OPTIONAL

Motivating Scenario

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



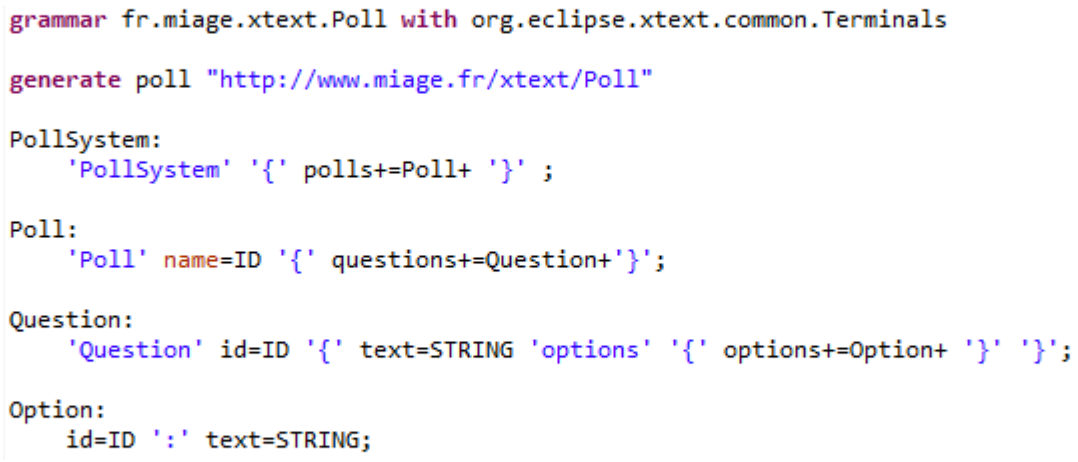
Motivating Scenario (2)



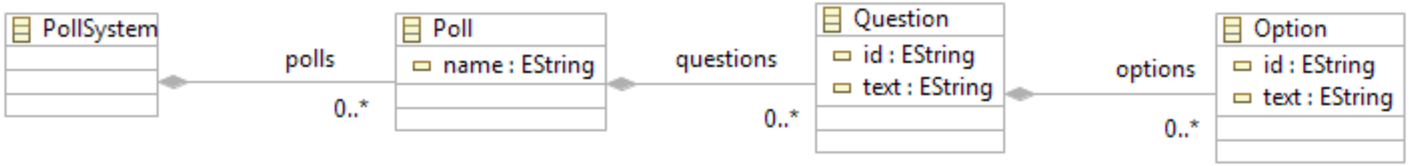
```
PollSystem {
    Poll Quality {
        Question q1 {
            "Value the user experience"
            options {
                A : "Bad"
                B : "Fair"
                C : "Good"
            }
        }
        Question q2 {
            "Value the layout"
            options {
 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

Grammar Definition

Grammar definition →

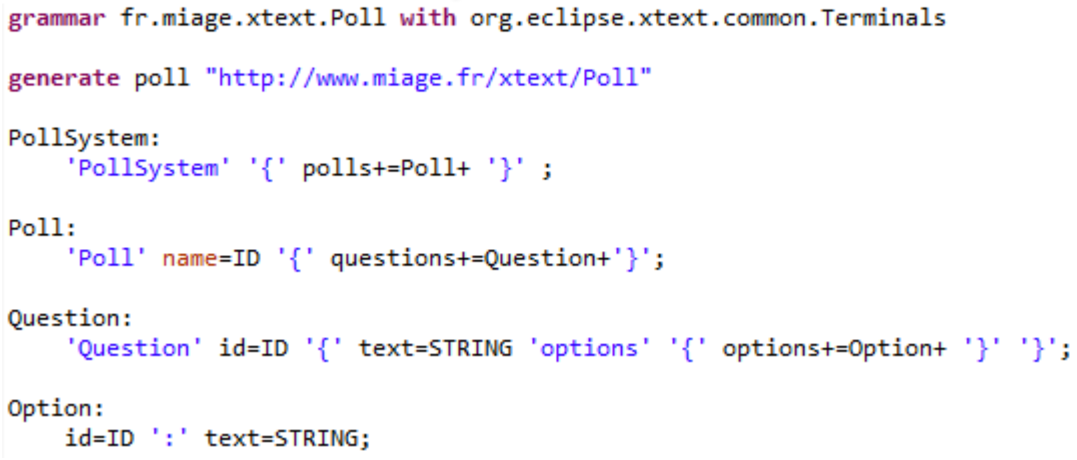


```
grammar fr.miage.xtext.Poll with org.eclipse.xtext.common.Terminals

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

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

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

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

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

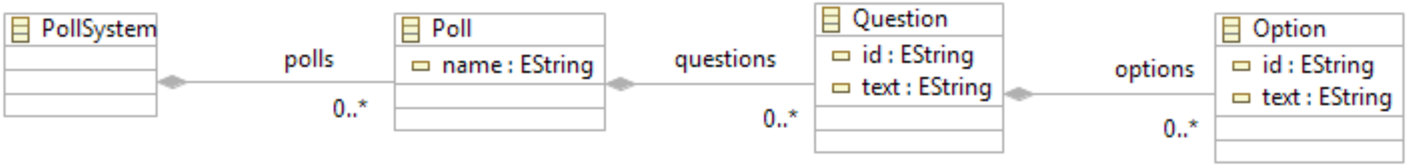


Grammar Definition

Grammar reuse

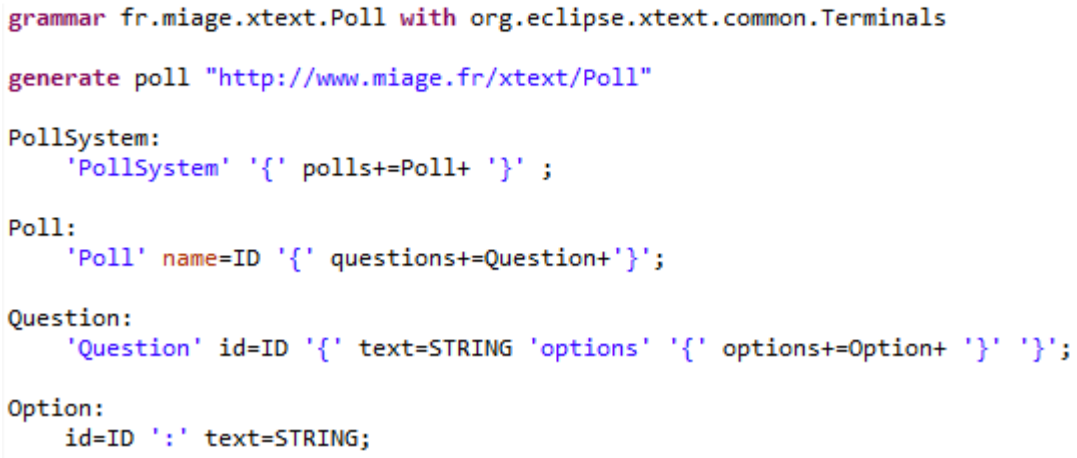


```
grammar fr.miage.xtext.Poll with org.eclipse.xtext.common.Terminals

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

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

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

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

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

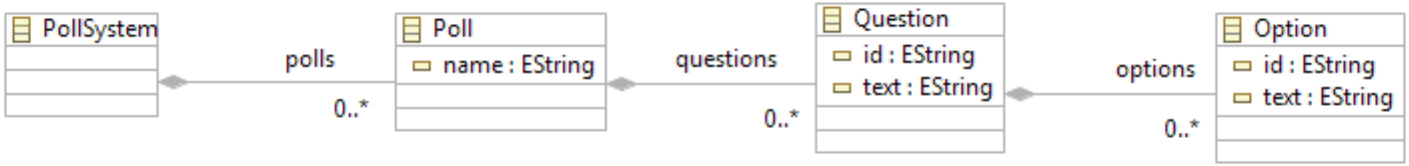


Grammar Definition

Derived metamodel →

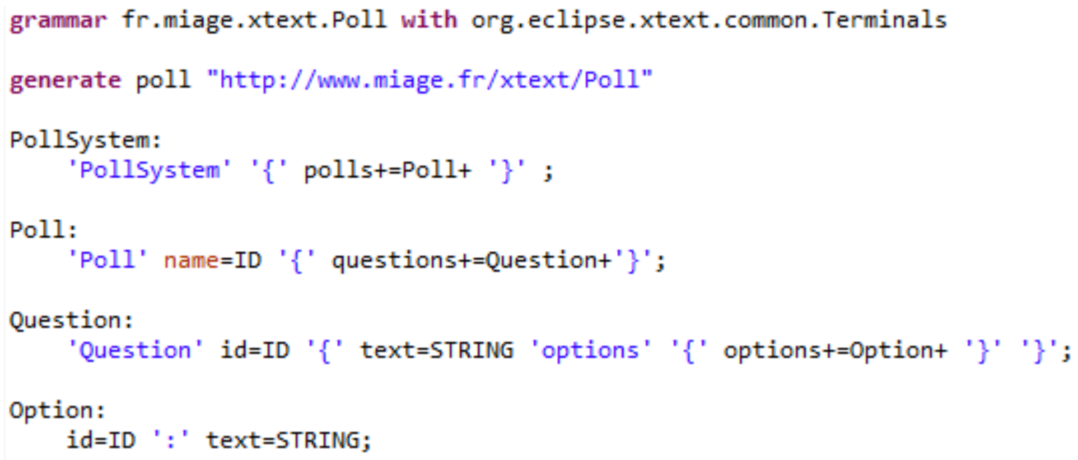


```
grammar fr.miage.xtext.Poll with org.eclipse.xtext.common.Terminals

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

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

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

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

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

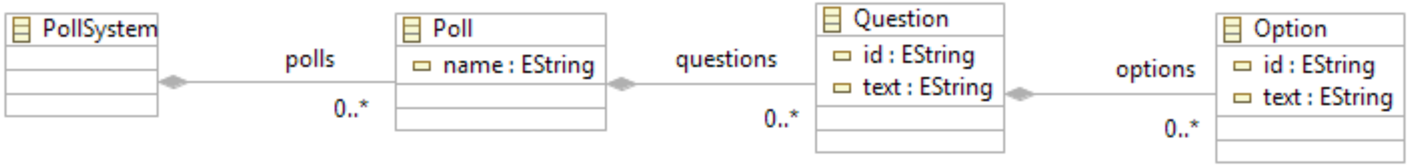


Grammar Definition

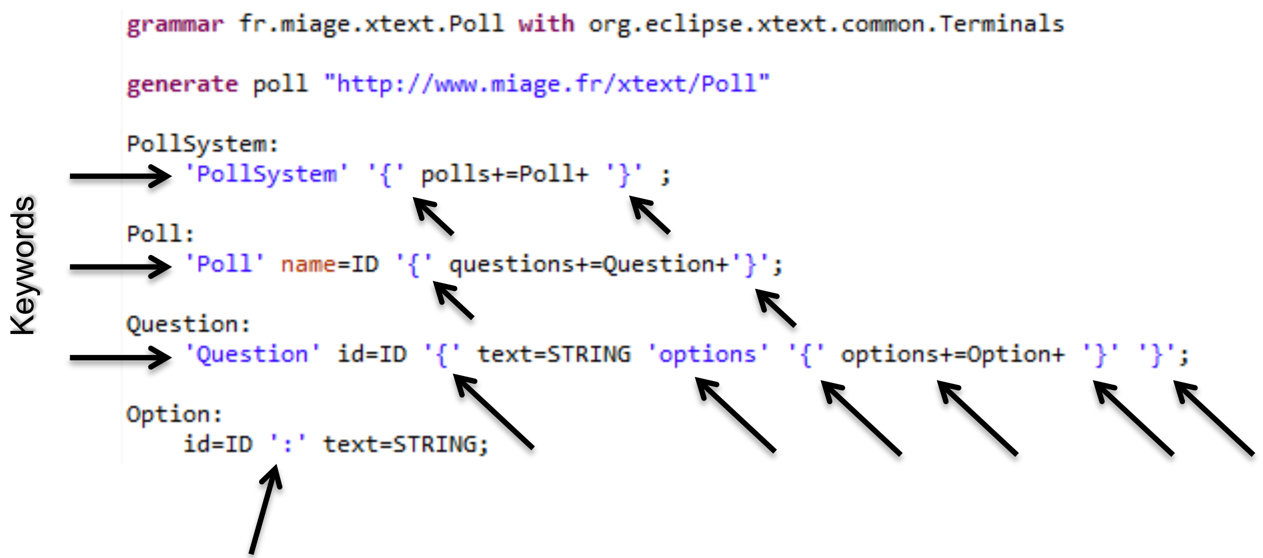
Parser Rules



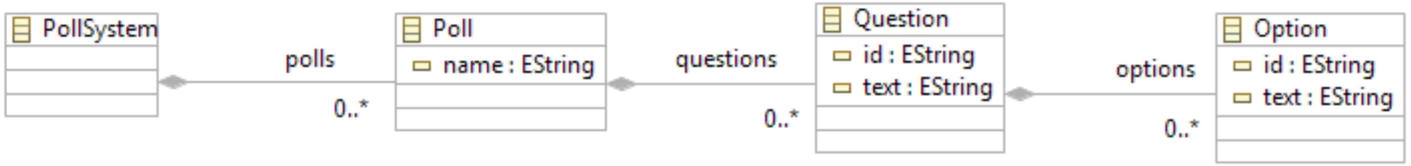
```
grammar fr.miage.xtext.Poll with org.eclipse.xtext.common.Terminals

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;
```



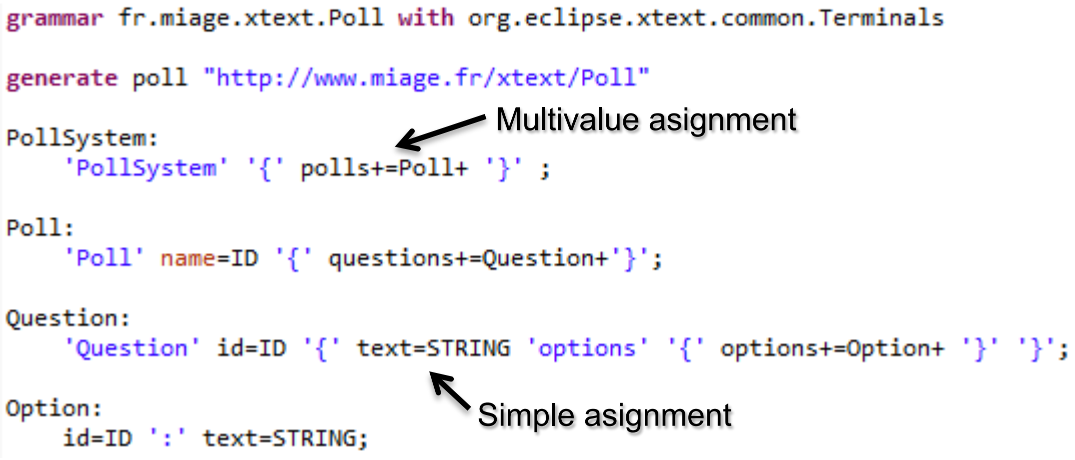
Grammar Definition



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

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

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

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

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

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



Grammar Definition

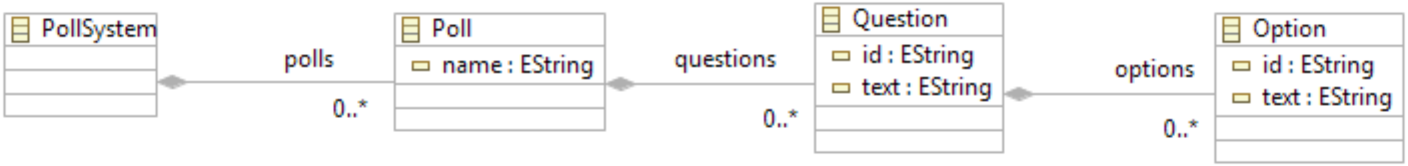


```
grammar fr.miage.xtext.Poll with org.eclipse.xtext.common.Terminals

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

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

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

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

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

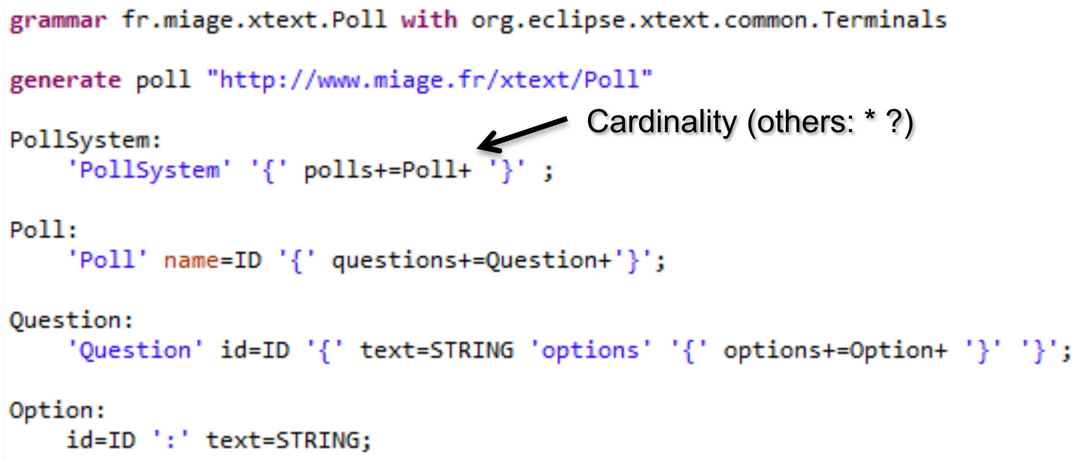
Multivalue asignment

Simple asignment

?= Boolean asignment

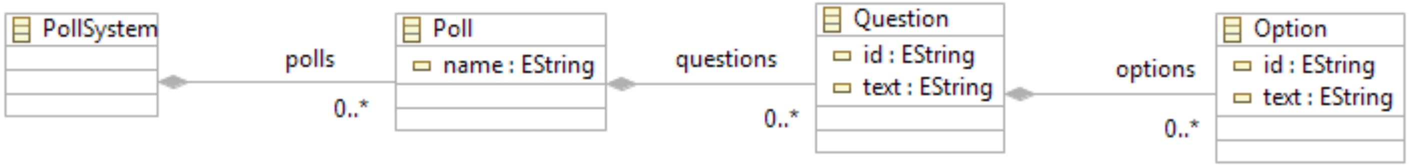


Grammar Definition

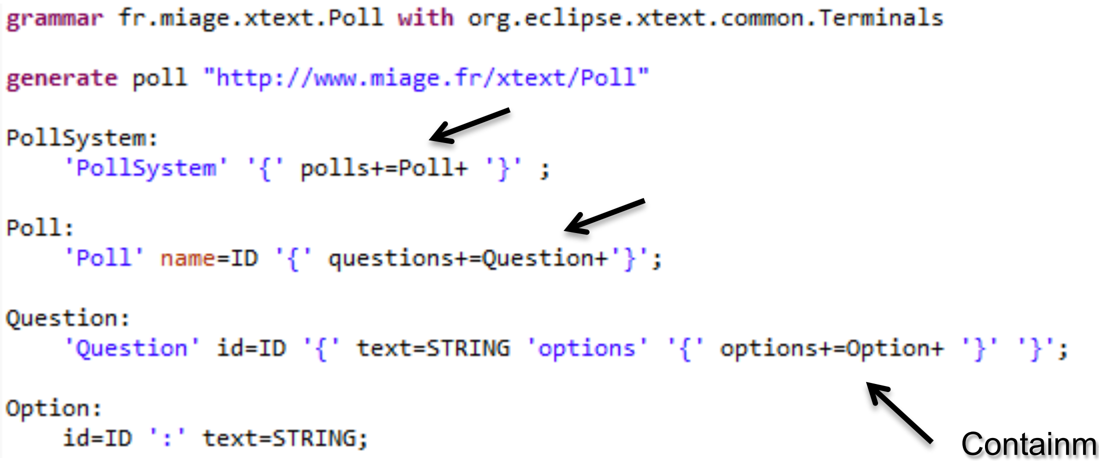


```
grammar fr.miage.xtext.Poll with org.eclipse.xtext.common.Terminals

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;
```

Cardinality (others: * ?)

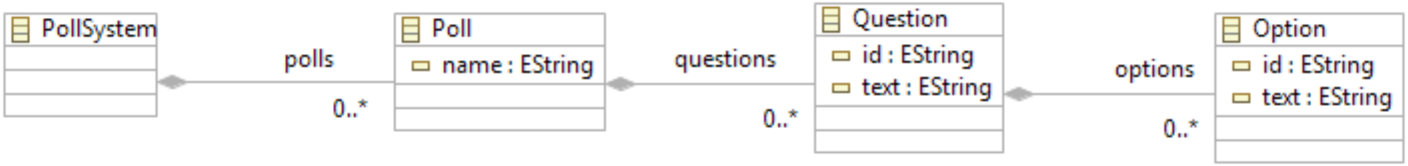


Grammar Definition

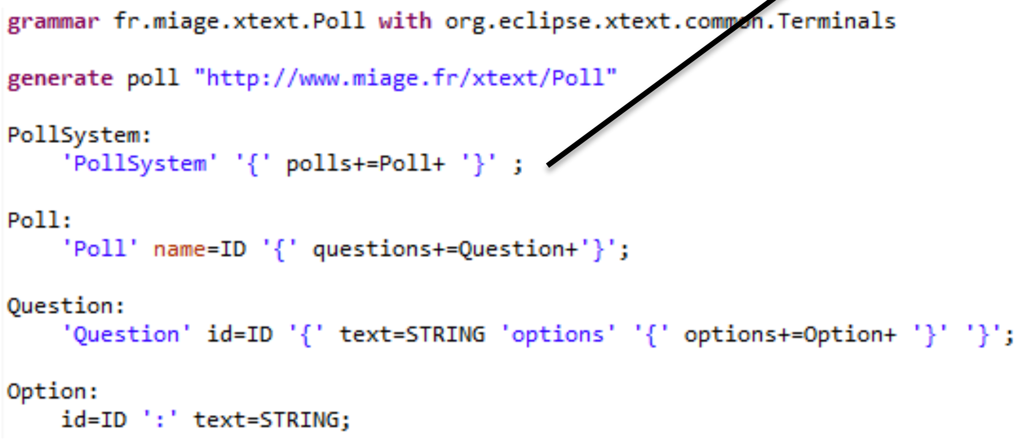


```
grammar fr.miage.xtext.Poll with org.eclipse.xtext.common.Terminals

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

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

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

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

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

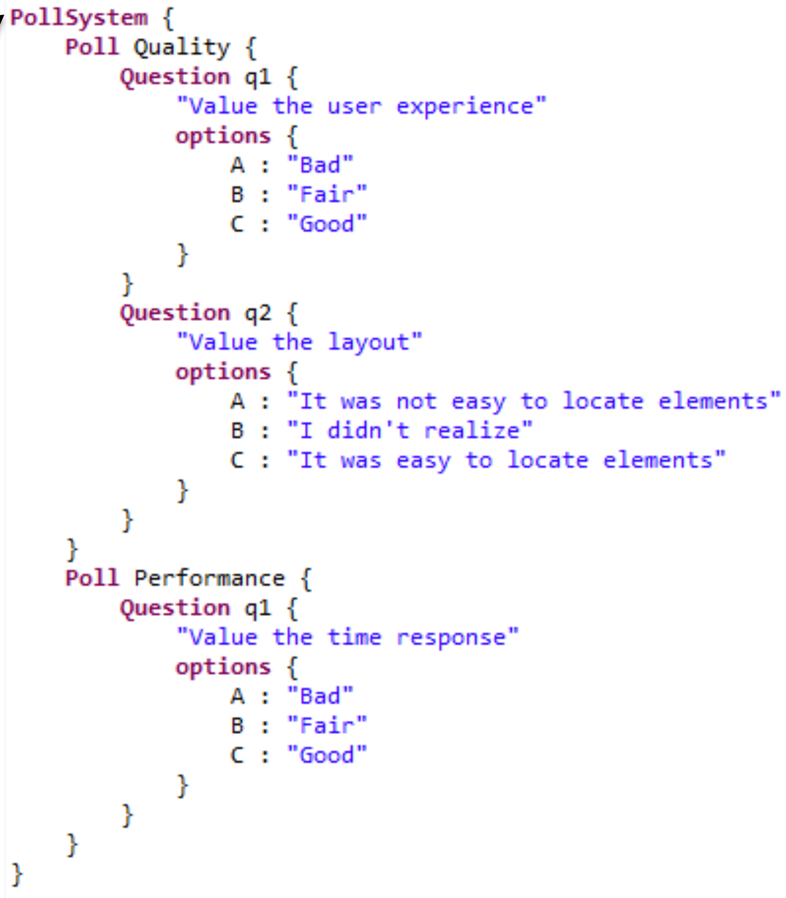
Containment



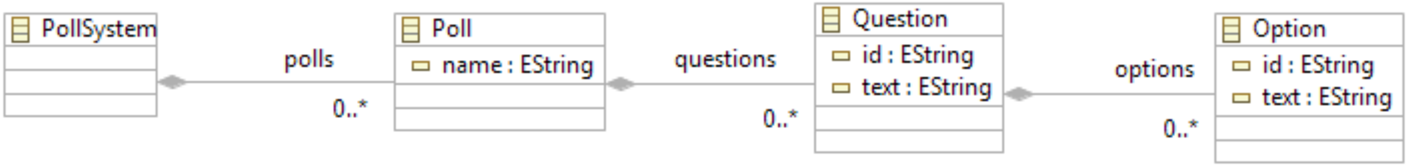
Grammar Definition



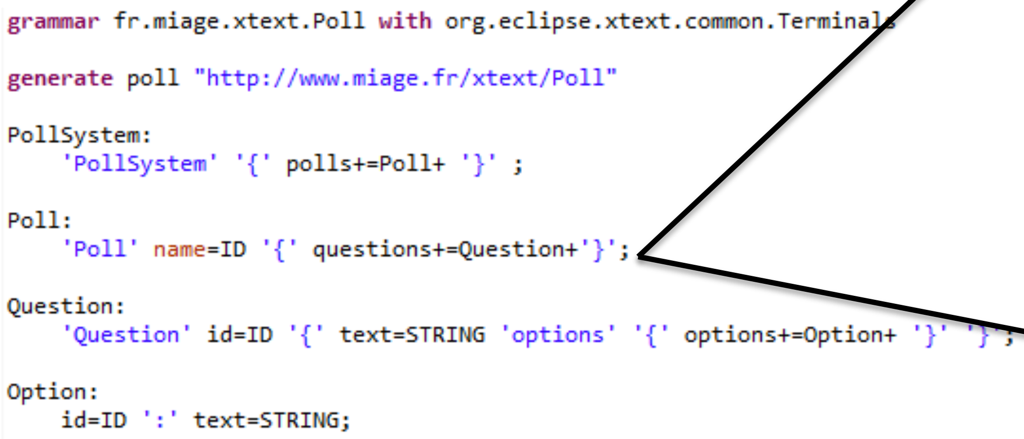
```
grammar fr.miage.xtext.Poll with org.eclipse.xtext.common.Terminals

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

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

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

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

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



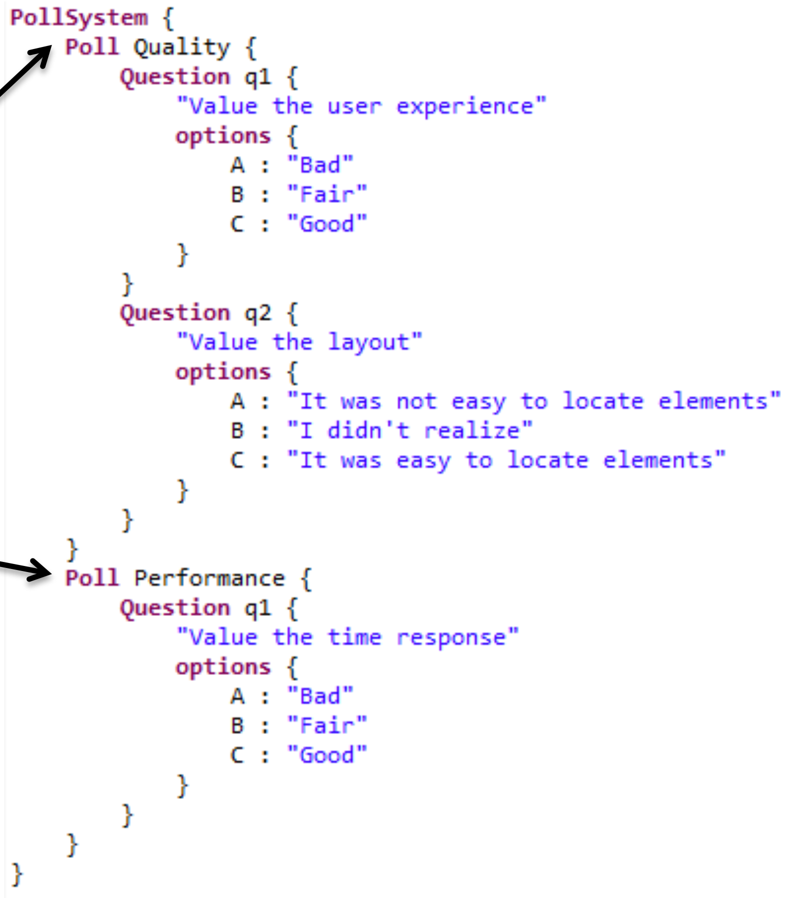
```
PollSystem {
    Poll Quality {
        Question q1 {
            "Value the user experience"
            options {
                A : "Bad"
                B : "Fair"
                C : "Good"
            }
        }
        Question q2 {
            "Value the layout"
            options {
 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



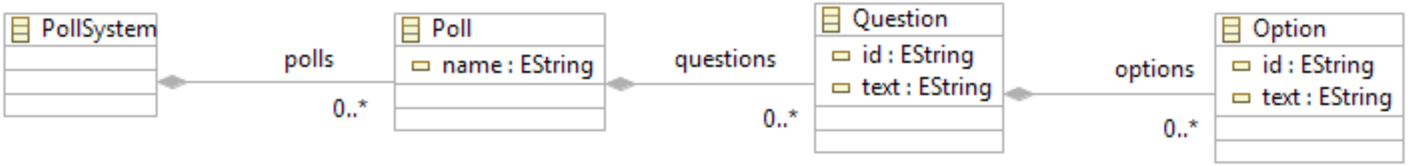
Grammar Definition



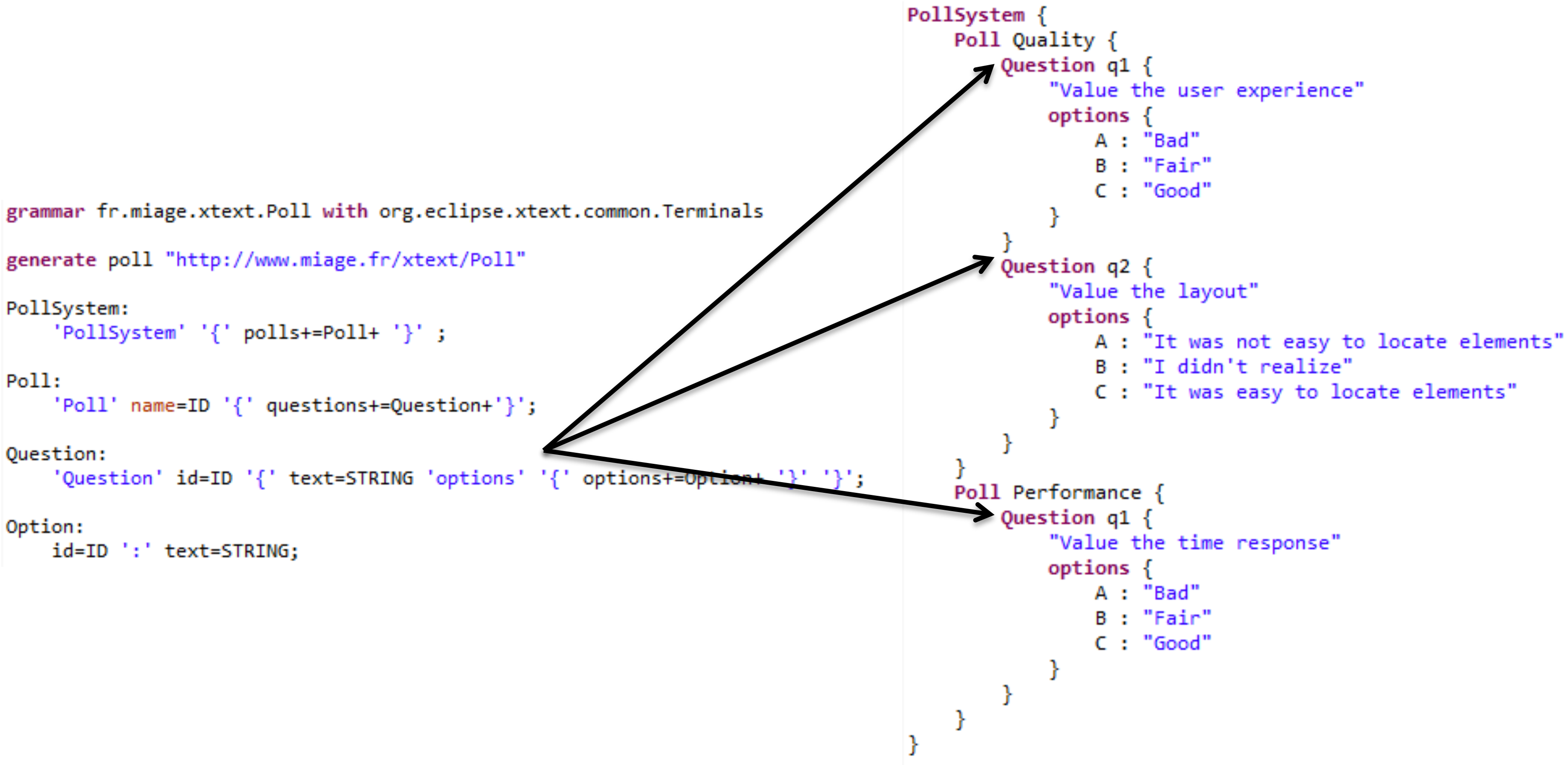
```
grammar fr.miage.xtext.Poll with org.eclipse.xtext.common.Terminals

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

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

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

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

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



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

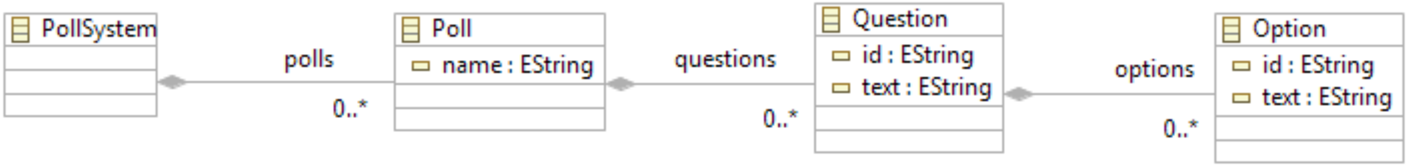


Grammar Definition

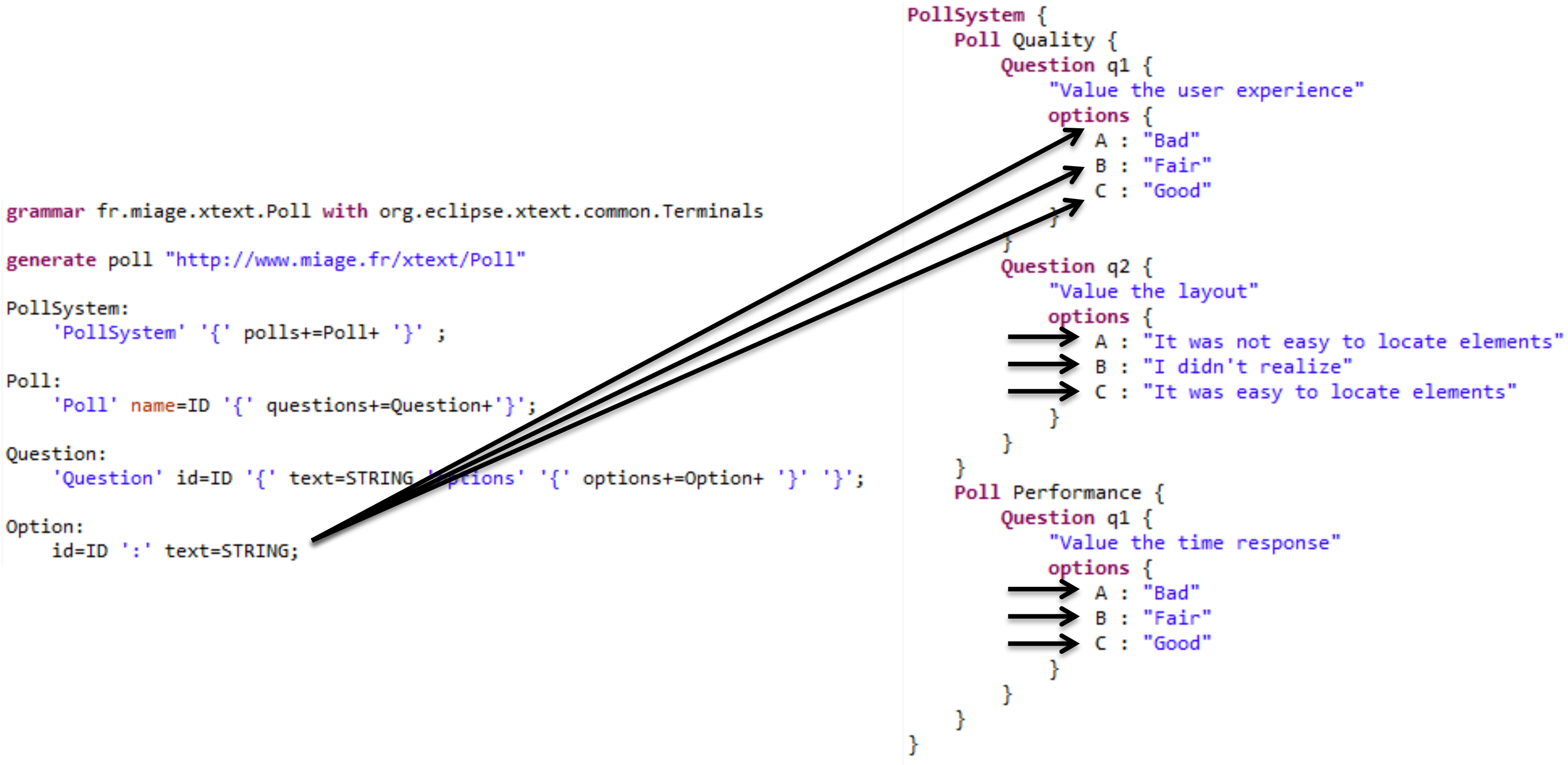


```
grammar fr.miage.xtext.Poll with org.eclipse.xtext.common.Terminals

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

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

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

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

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

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

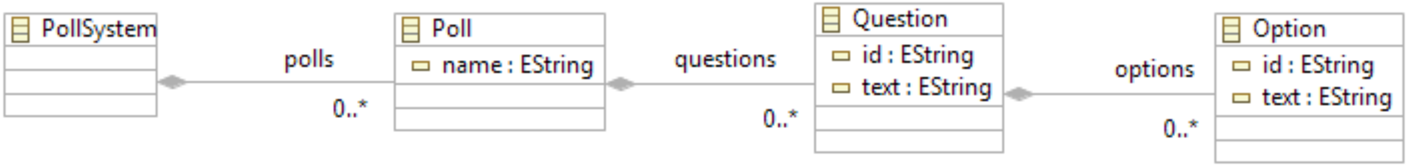


Grammar Definition



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

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

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

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

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

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

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



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

Live Demonstration

Another example:

Chess

"Queen to c7. Check."

"Rd2-c2, rook at d2 moves to c2."

Moves in Chess:

Rook at a1 moves to a5.

Piece Square Action Destination

Bishop at c8 captures knight at h3.

Piece Square Action Destination

N b1 x c3

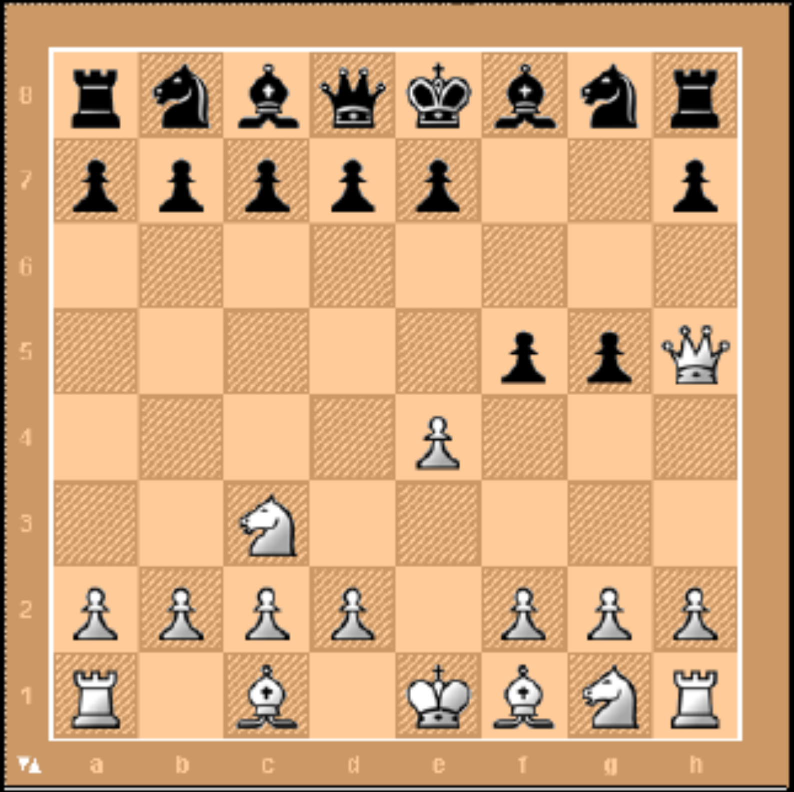
Piece Square Action Destination

g2 - g4

Square Action Destination

Bishop at c8 captures knight at h3

B c8 x h3



P e2 – e4

p g7 – g5

Knight at b2 moves to c3

pawn at f7 moves to f5

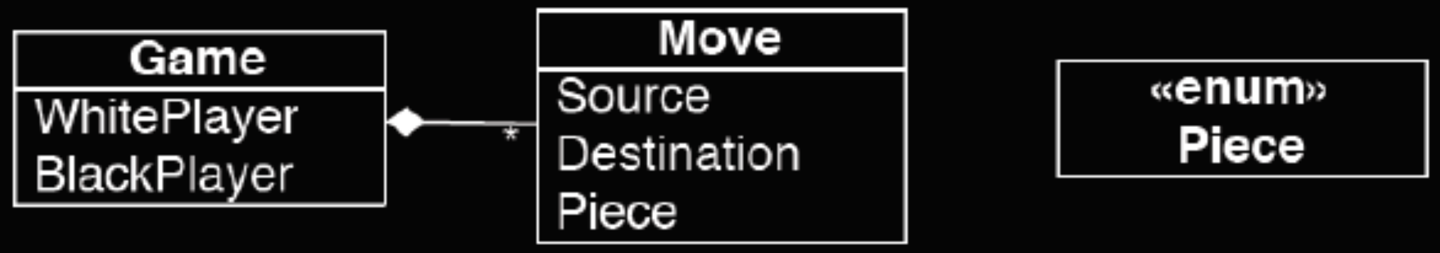
Q d1 – h5

1-0

Concrete Syntax

Constraints !!!

Abstract Syntax



Game

WhitePlayer

BlackPlayer

*

Move

Source

Destination

Piece

«enum»

Piece

Chess Example - Grammar

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

Move:
 AlgebraicMove | SpokenMove;
AlgebraicMove:
 (piece=Piece)? source=Square (captures?='x'|'-') dest=Square;

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

terminal Square:
 ('a'..'h')('1'..'8');

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

Chess Example - Model

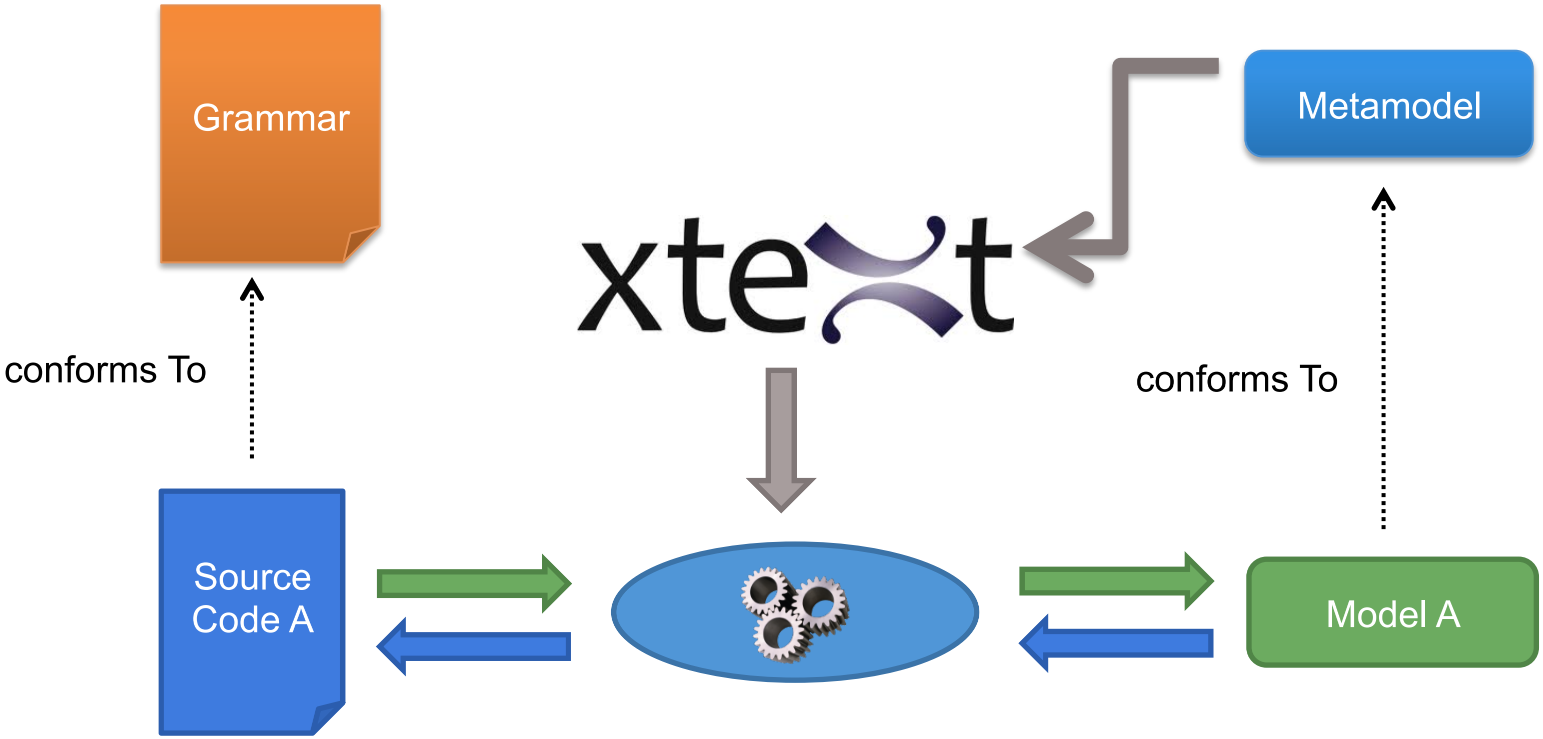
```
White: "Mayfield"
Black: "Trinks“

pawn at e2 moves to e4
pawn at f7 moves to g5

K b1 - c3
f7 - f5

queen at d1 moves to h5
// 1-0
```

From Metamodel

To

Grammar (other side)

From Metamodel to Grammar



xtext

Give me a **metamodel**,

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

xtext

Give me a **metamodel**,

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

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

[We will experiment in the lab sessions]

Live

Demonstration

Graphical DSL

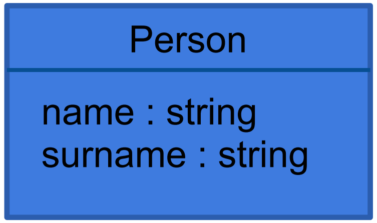
(vs Textual DSL)

Graphical vs Textual DSLs

- Success depends on how the notation fits the domain

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

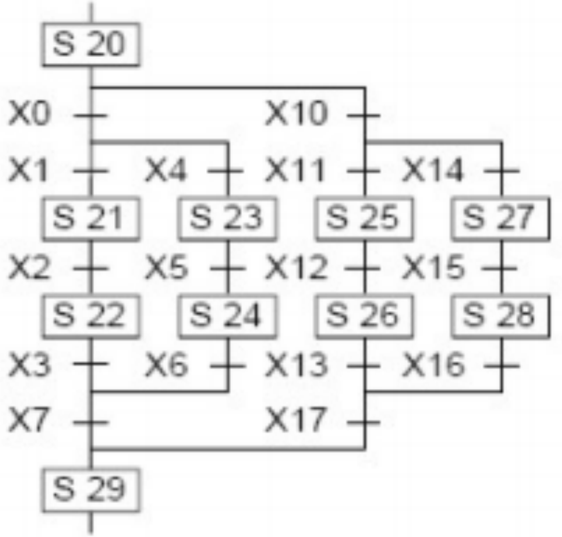
```
Person has (name, surname)
```

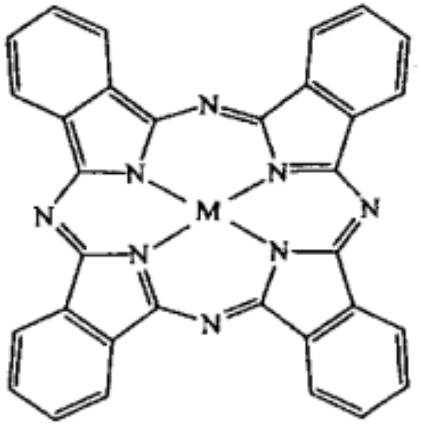


Person

name : string
surname : string

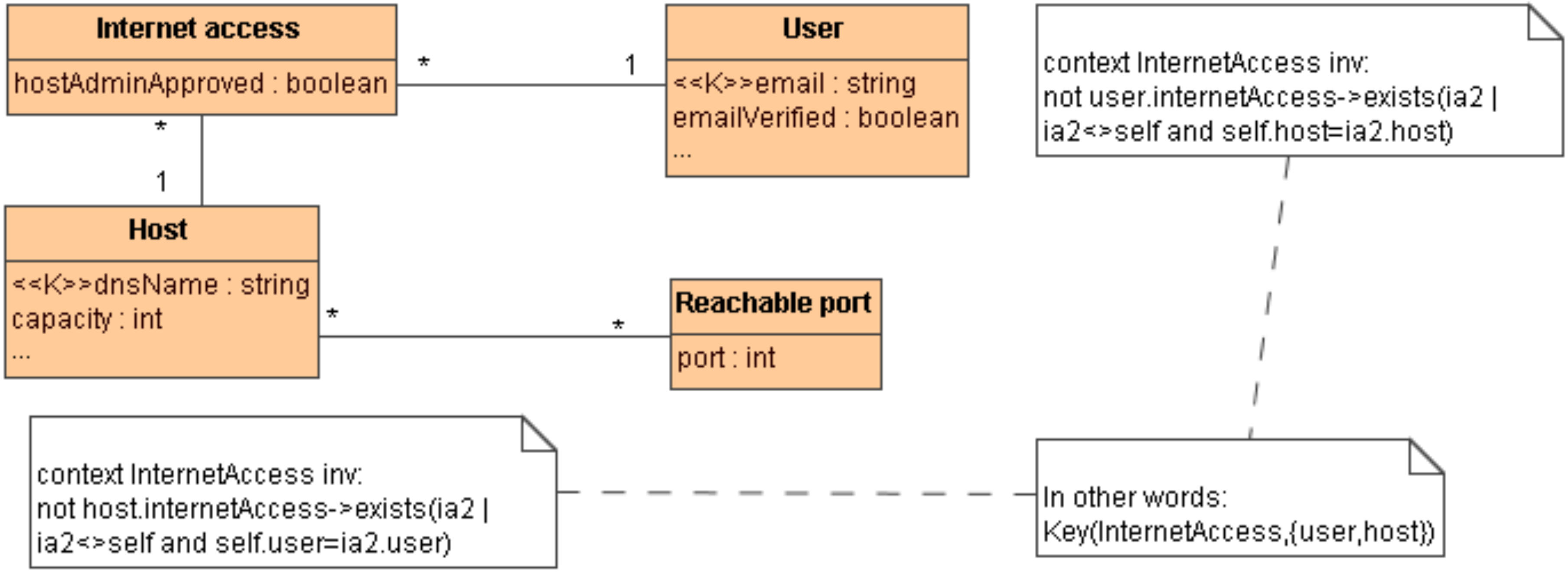
- Graphical DSLs are not always easier to understand





phthalocyanine

A language can be graphical and textual



Internet access

hostAdminApproved : boolean

User

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

Host

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

Reachable port

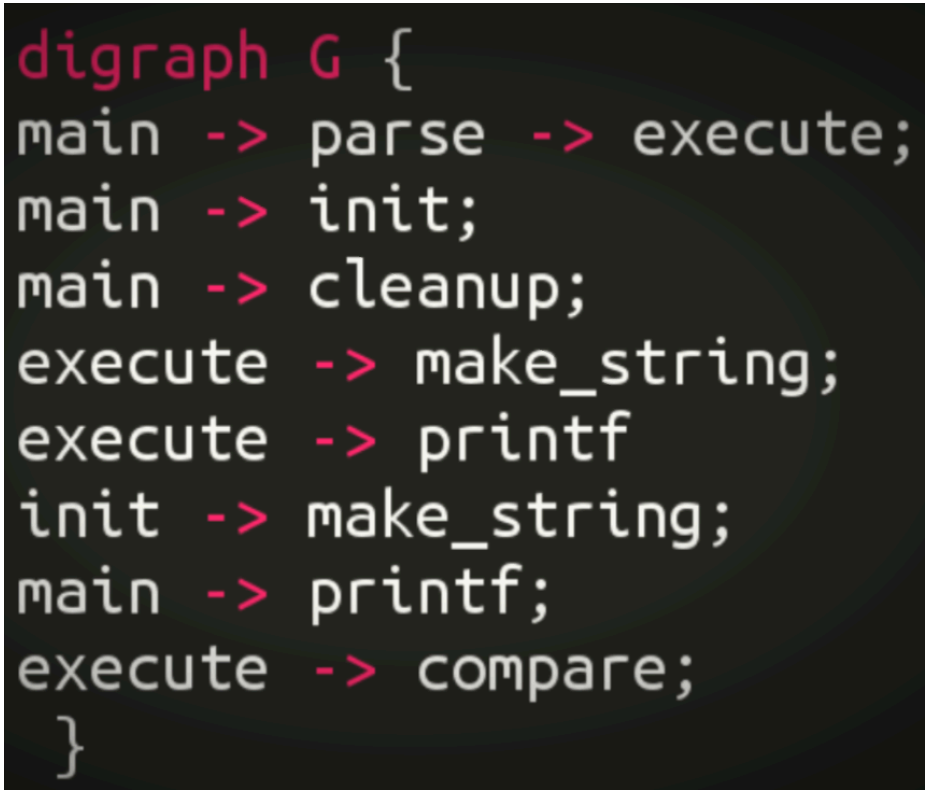
port : int

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

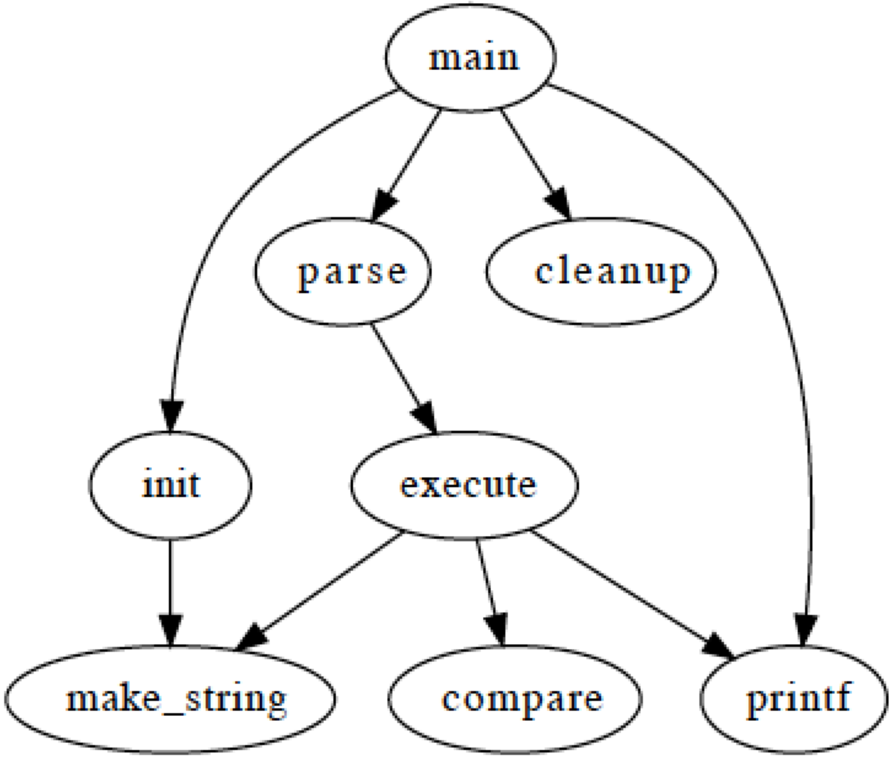
context InternetAccess inv:
not host.internetAccess->exists(ia2 |
ia2<>self and self.user=ia2.user)

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

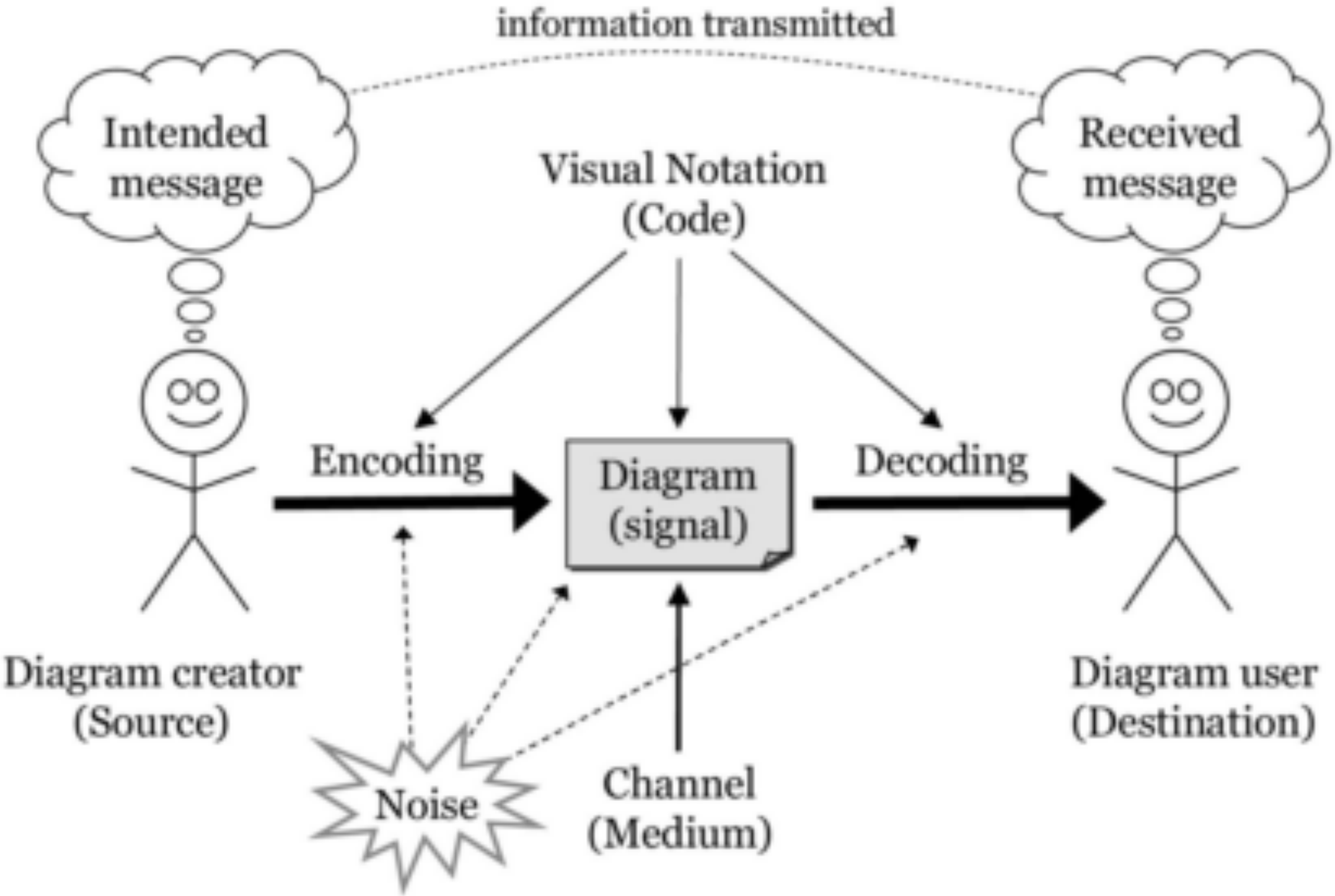
Alternative representation



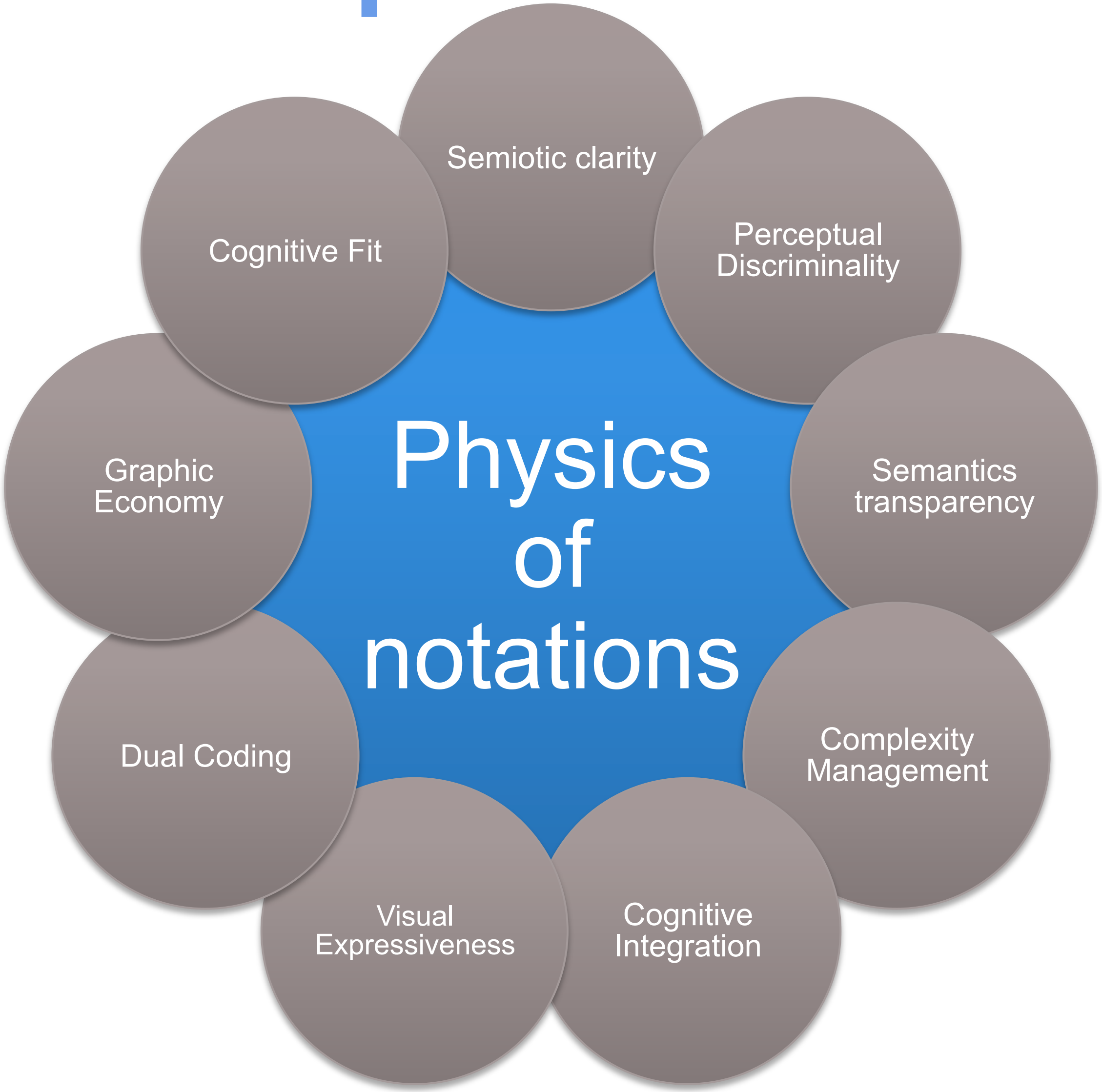
```
digraph G {
main -> parse -> execute;
main -> init;
main -> cleanup;
execute -> make_string;
execute -> printf
init -> make_string;
main -> printf;
execute -> compare;
 }
```



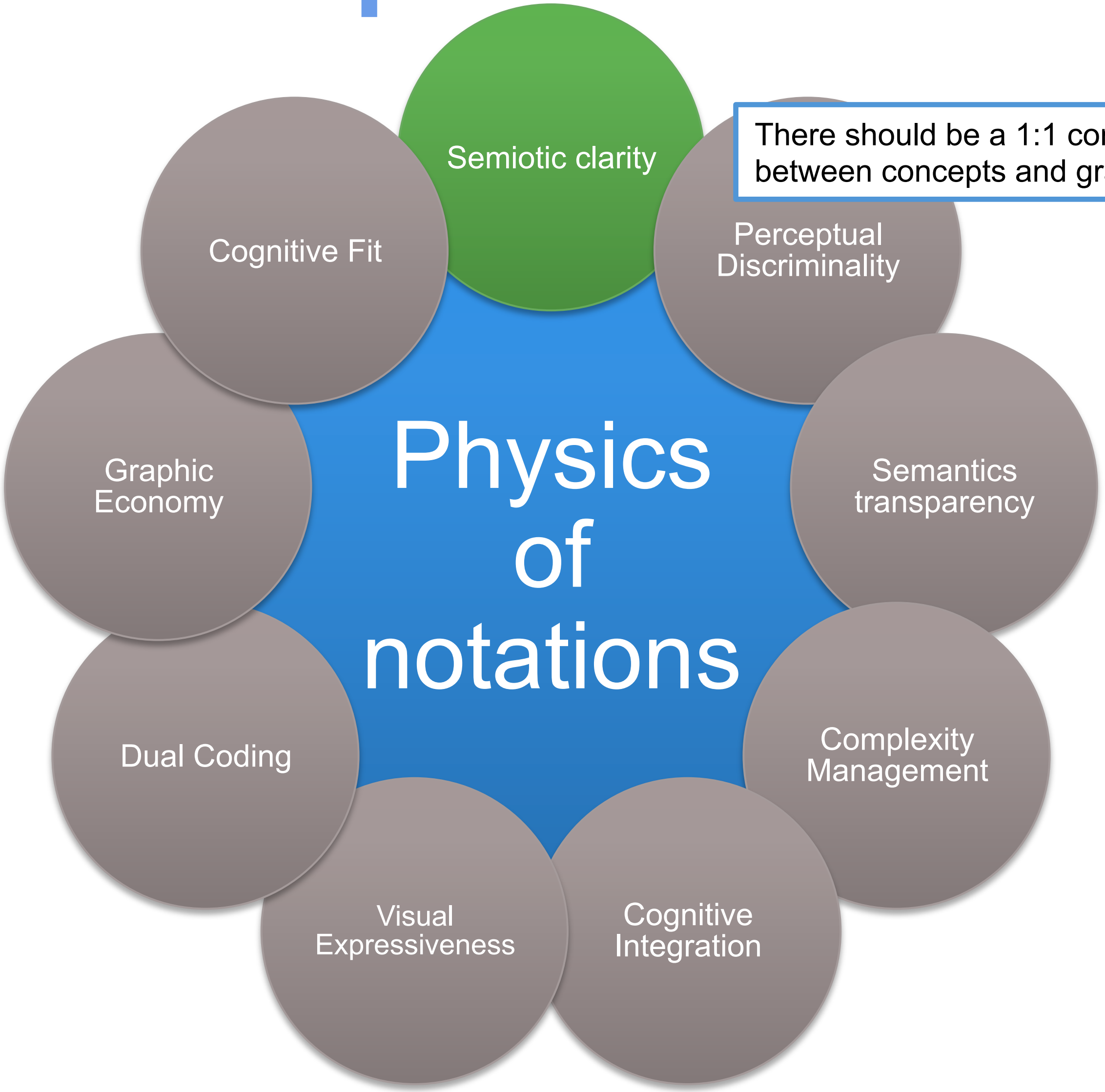
Recommendations for Graphical DSLs



Recommendations for Graphical DSLs

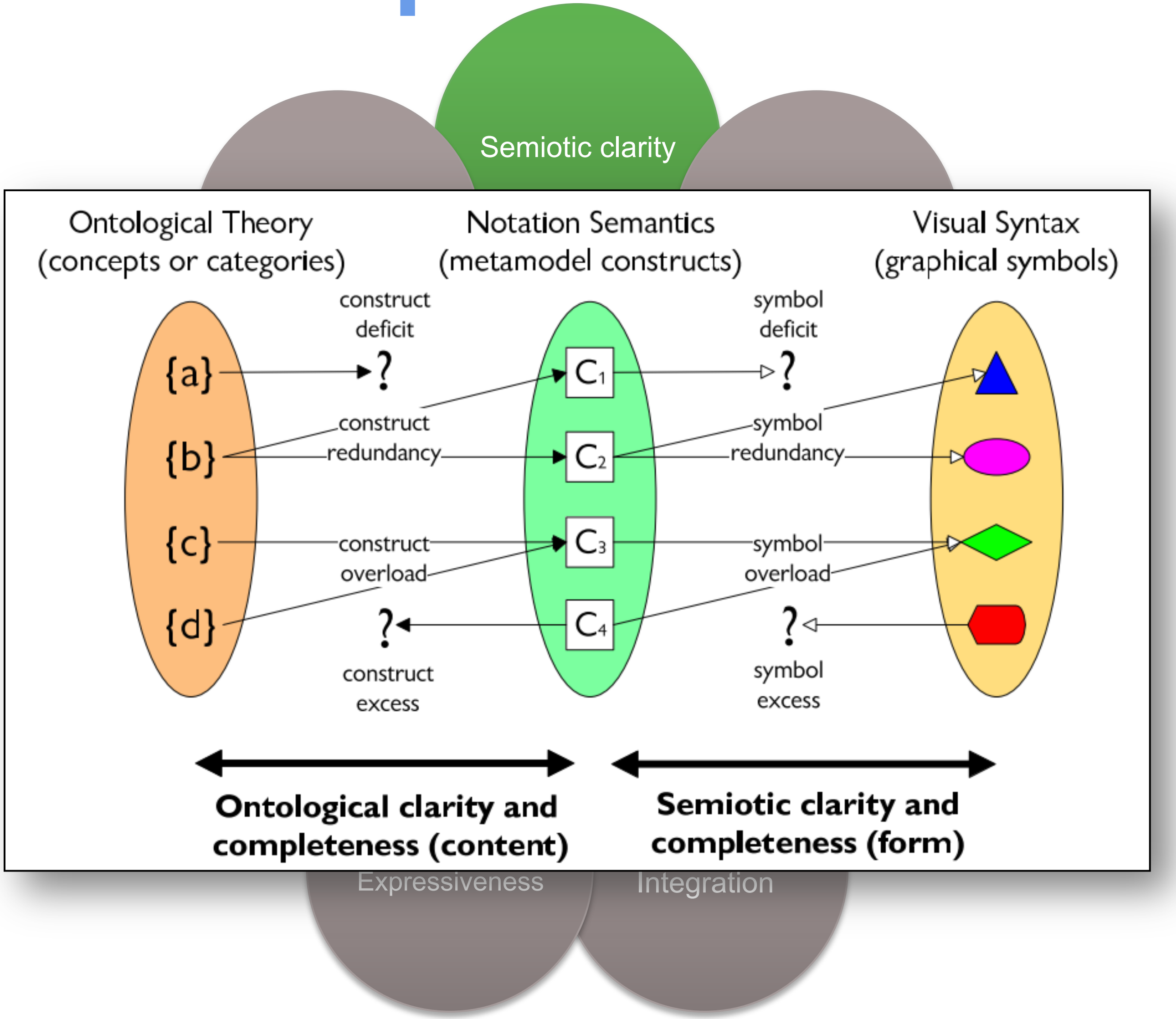


Recommendations for Graphical DSLs

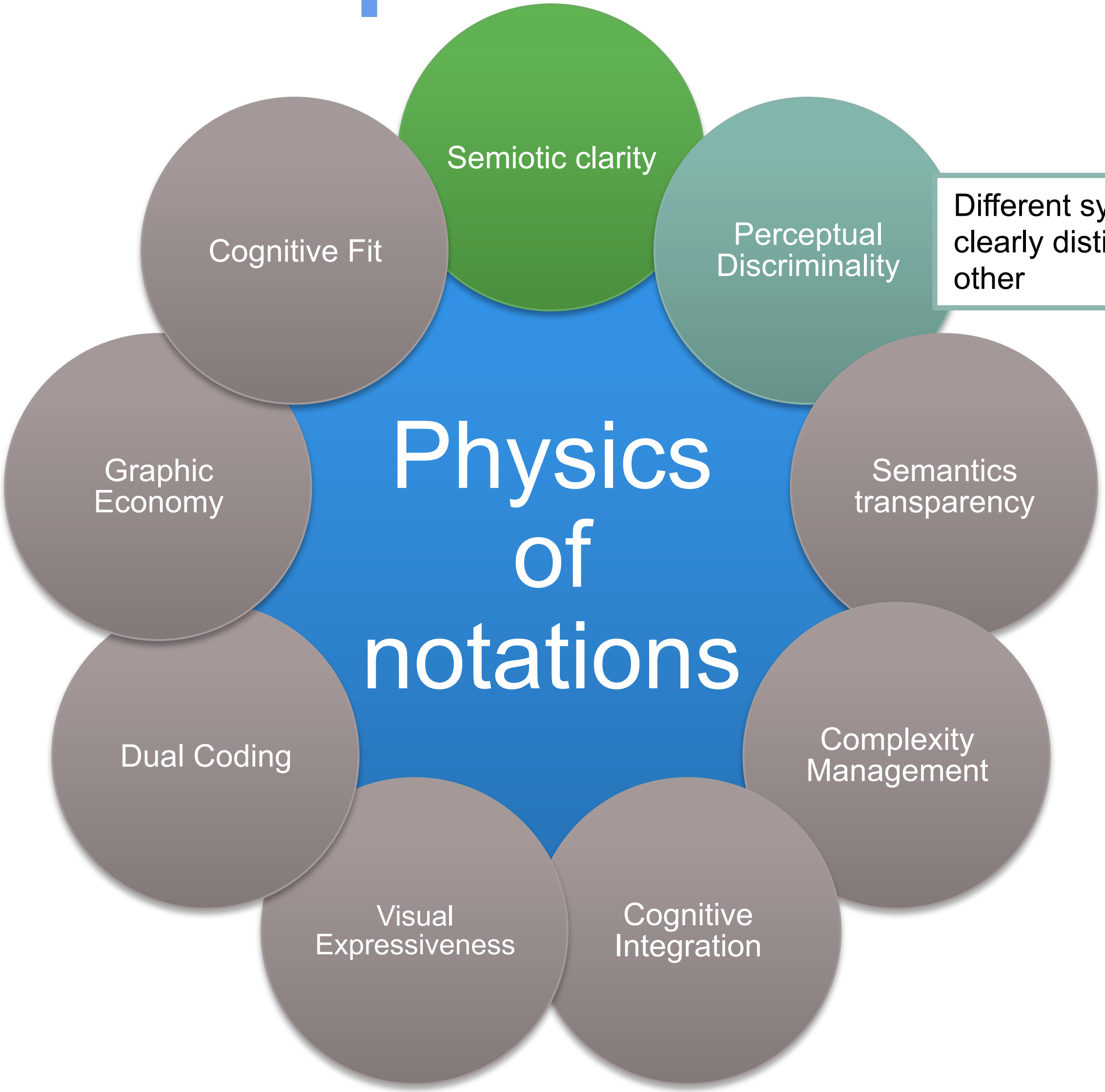


There should be a 1:1 correspondence between concepts and graphical symbols

Recommendations for Graphical DSLs

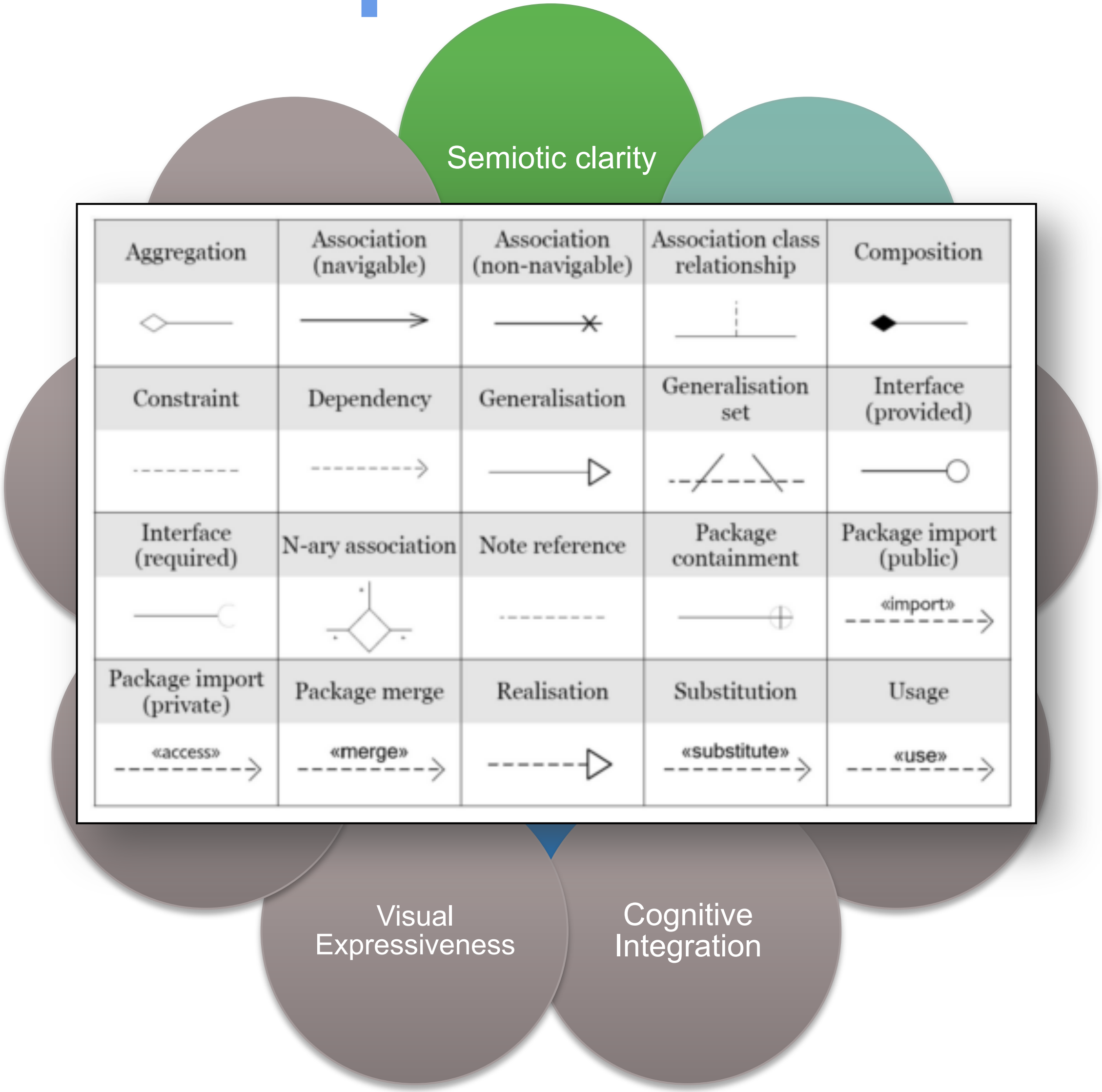


Recommendations for Graphical DSLs



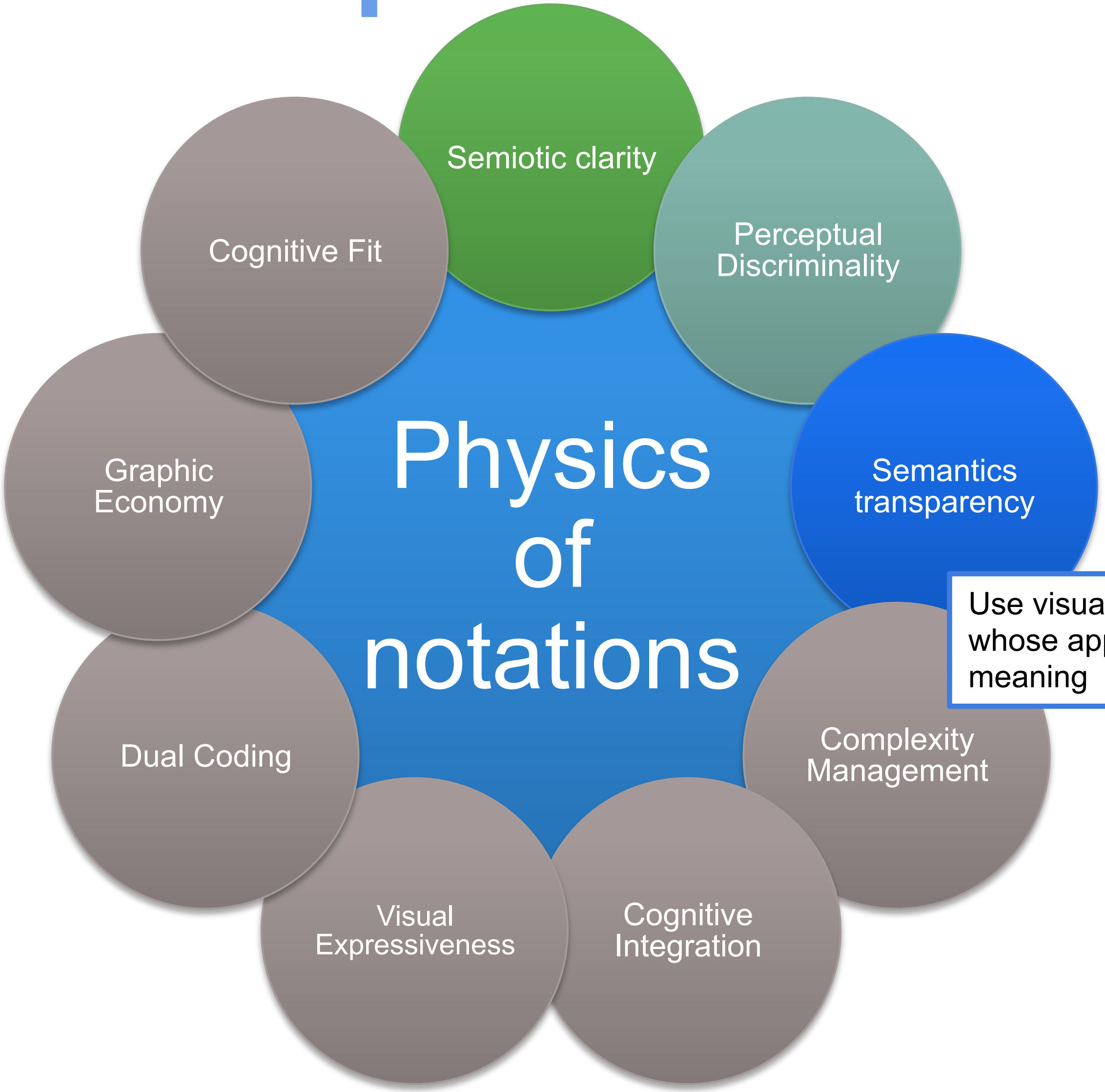
Different symbols should be clearly distinguishable from each other

Recommendations for Graphical DSLs



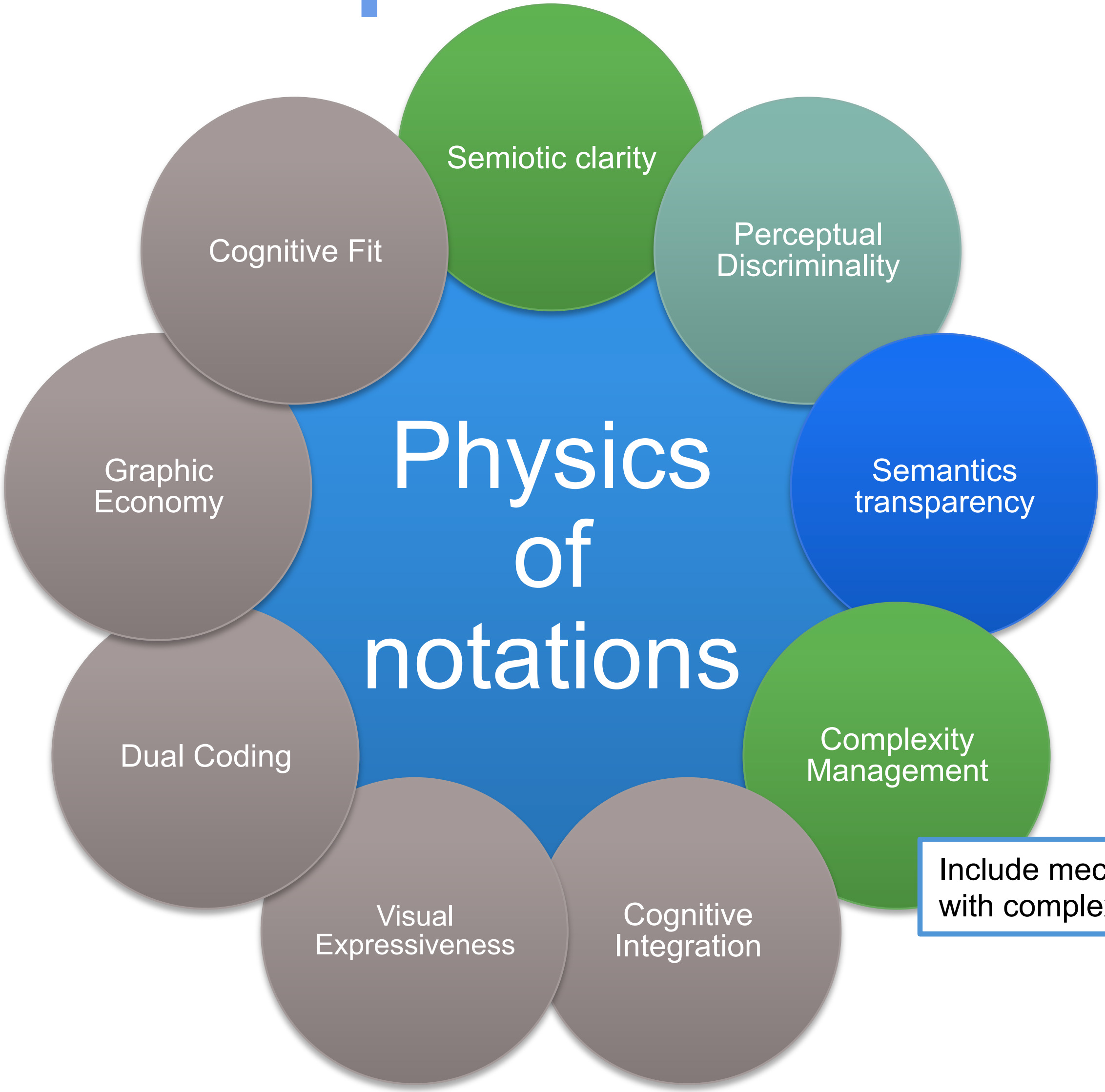
Aggregation	Association (navigable)	Association (non-navigable)	Association class relationship	Composition
Constraint	Dependency	Generalisation	Generalisation set	Interface (provided)
Interface (required)	N-ary association	Note reference	Package containment	Package import (public)
				«import»
Package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

Recommendations for Graphical DSLs



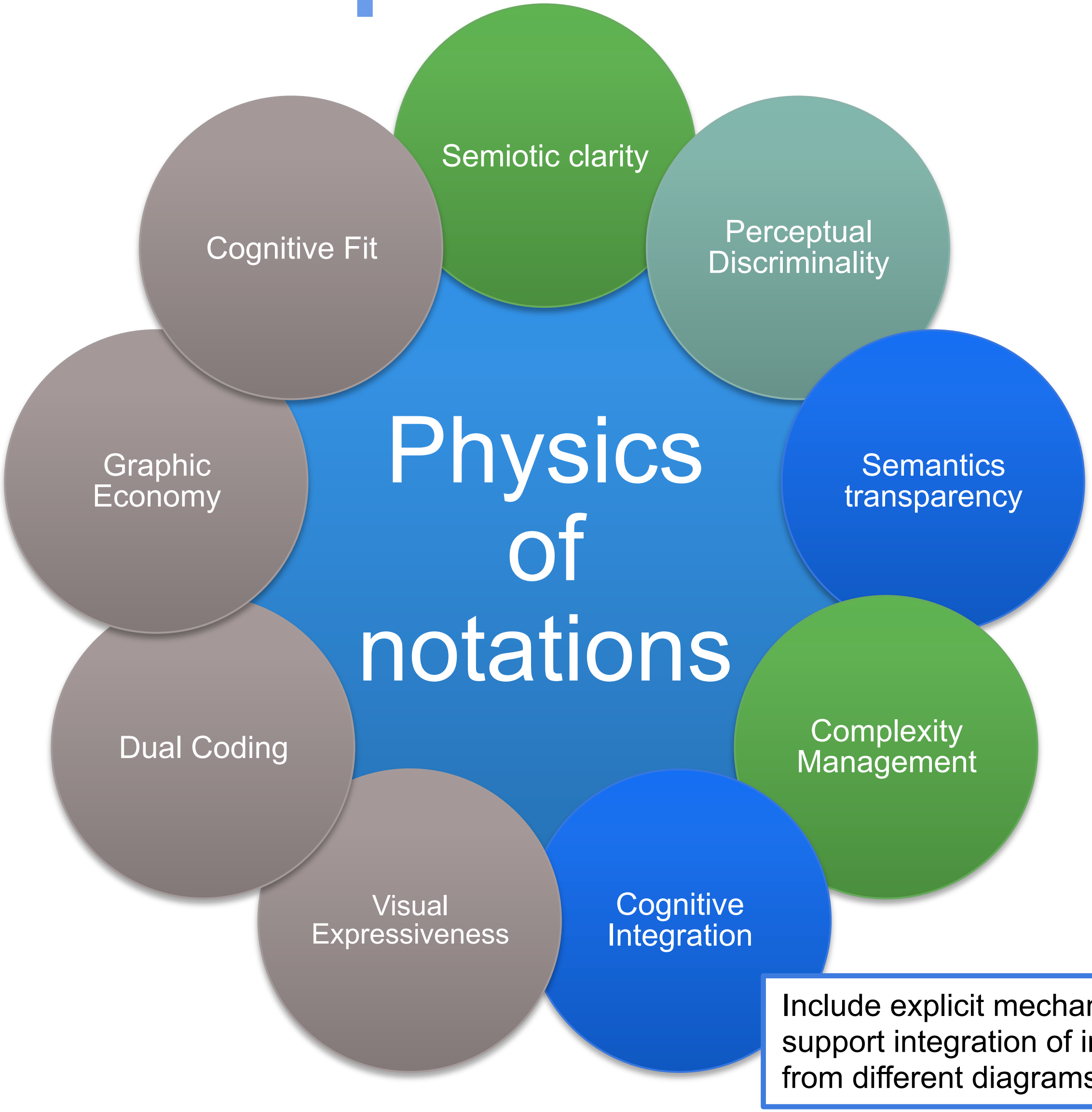
Use visual representations whose apprarance suggests their meaning

Recommendations for Graphical DSLs



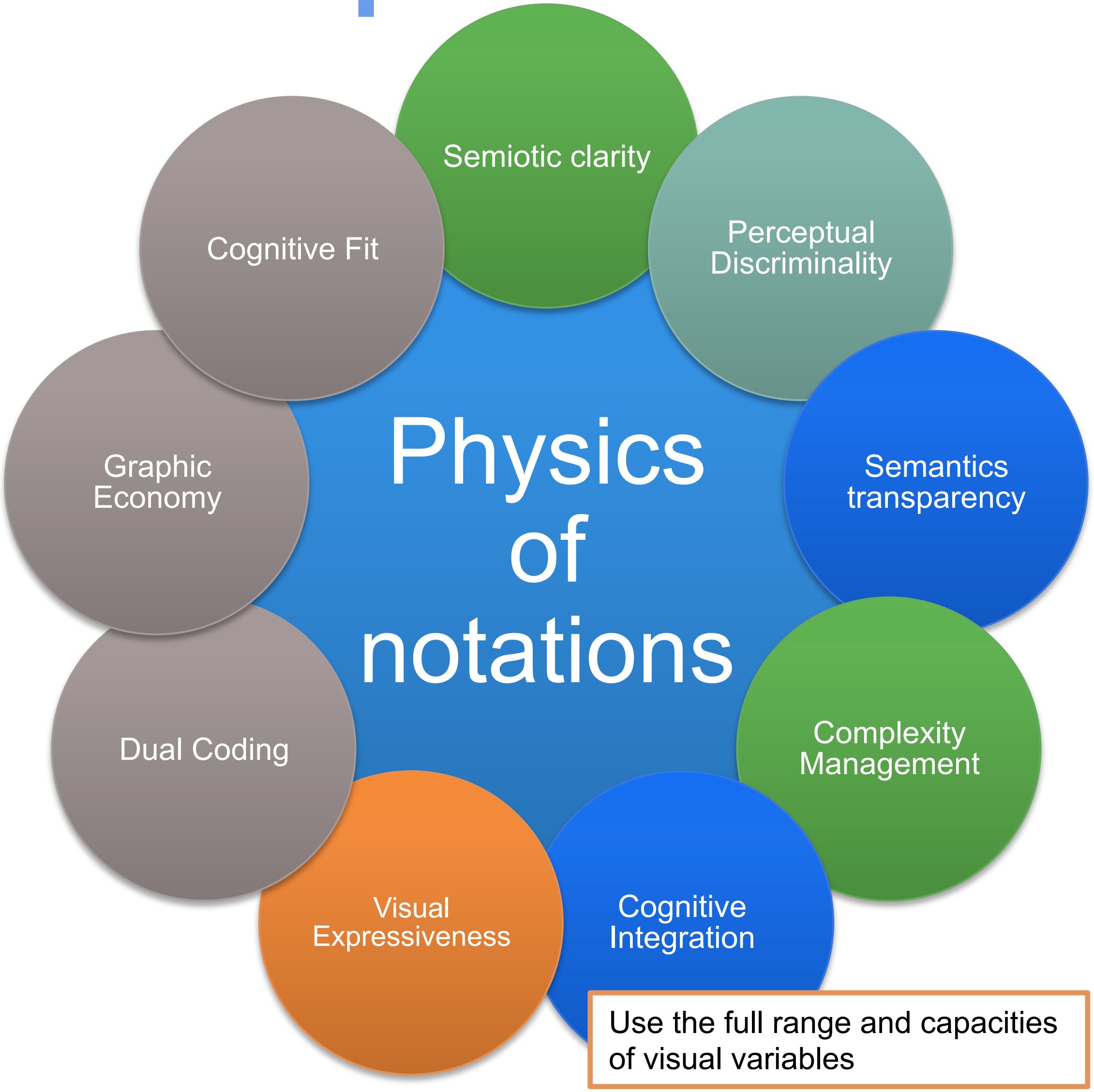
Include mechanisms for dealing with complexity

Recommendations for Graphical DSLs



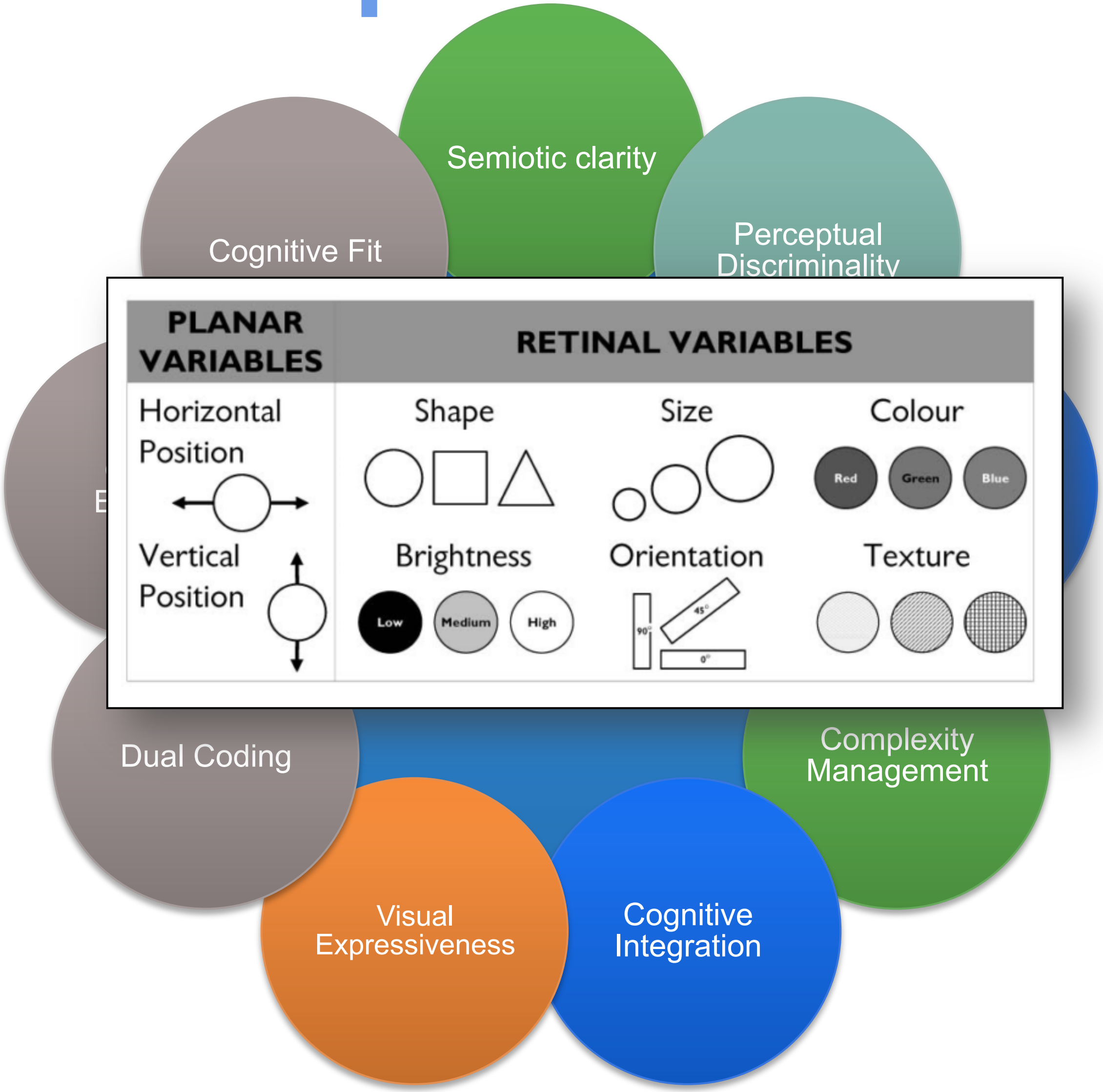
Include explicit mechanisms to support integration of information from different diagrams

Recommendations for Graphical DSLs

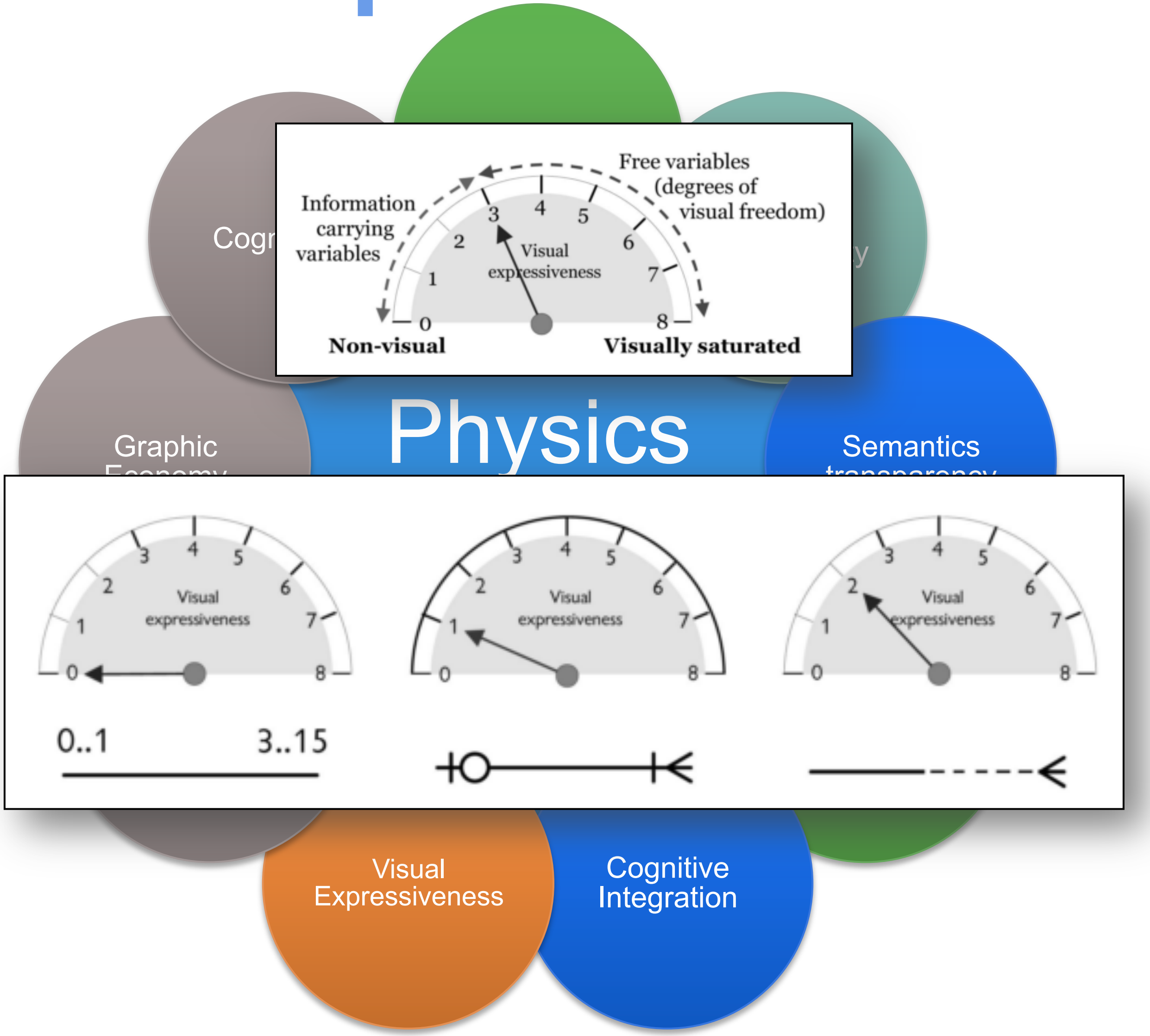


Use the full range and capacities of visual variables

Recommendations for Graphical DSLs



Recommendations for Graphical DSLs



Cogr

y

Physics

Graphic Economy

Semantics transparency

Visual Expressiveness

Cognitive Integration

Information carrying variables
Free variables (degrees of visual freedom)
0
1
2
3
4
5
6
7
8
Visual expressiveness
Non-visual
Visually saturated
0..1
3..15

Recommendations for Graphical DSLs

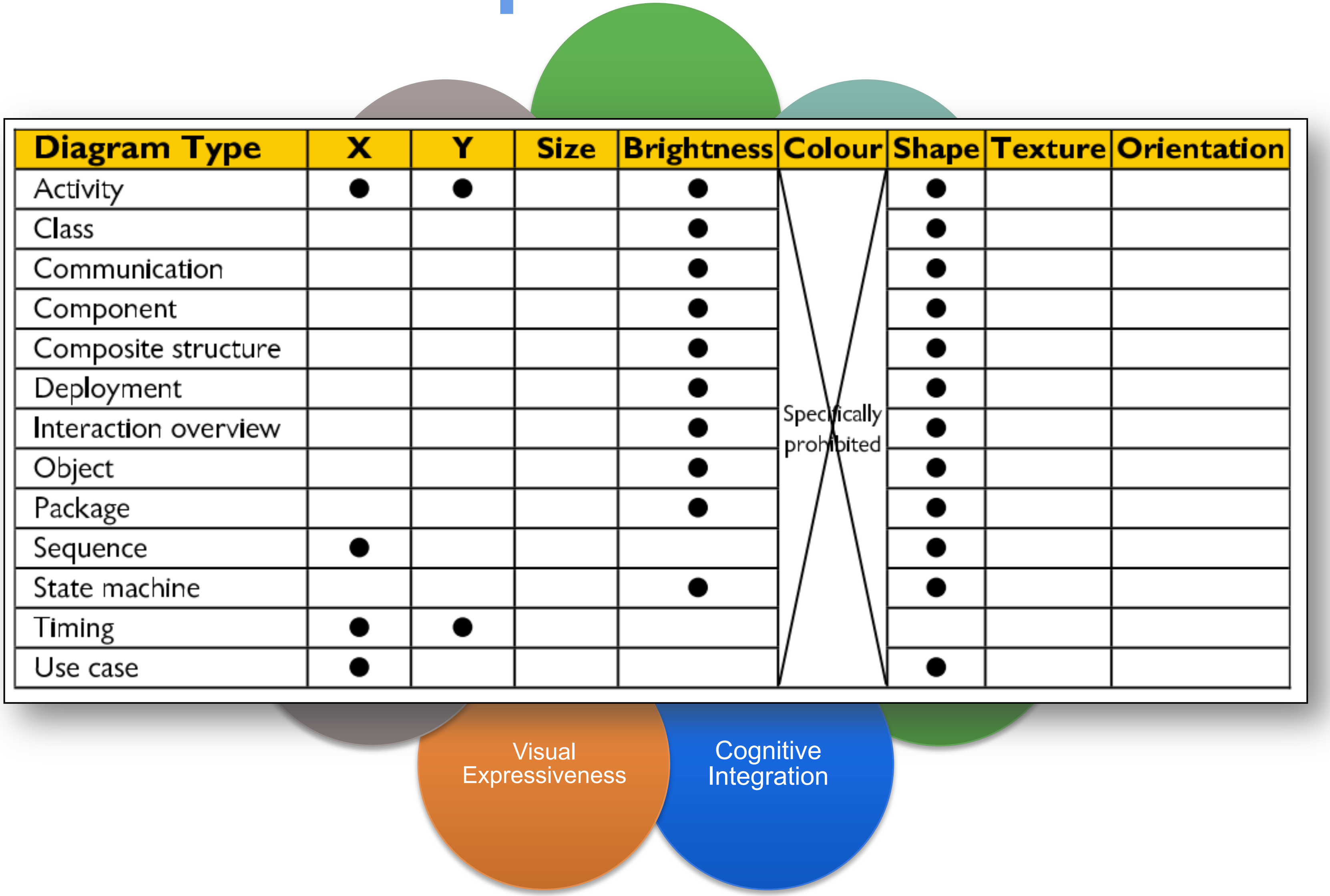
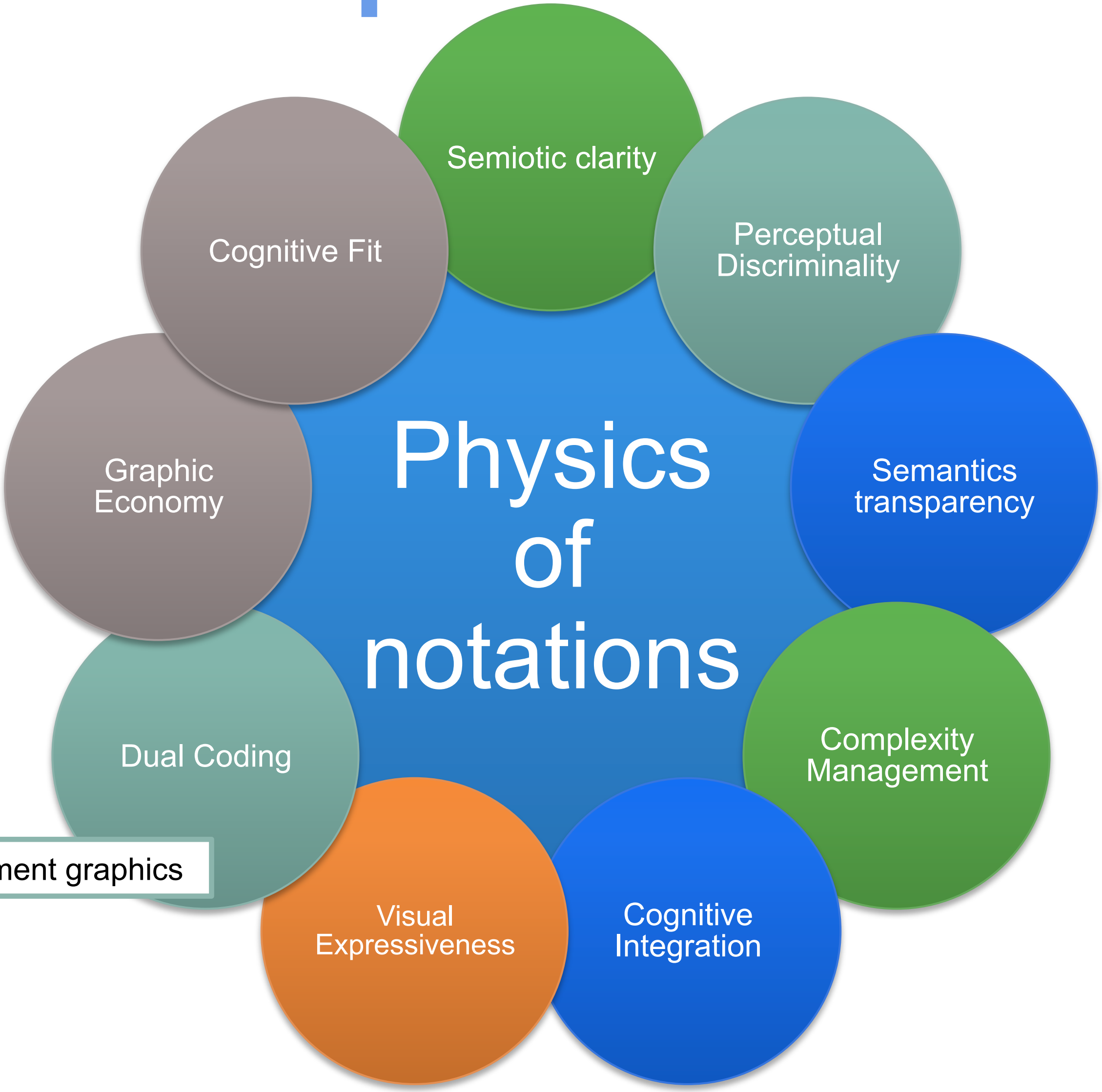


Diagram Type	X	Y	Size	Brightness	Colour	Shape	Texture	Orientation
Activity	●	●		●	Specifically prohibited	●		
Class				●		●		
Communication				●		●		
Component				●		●		
Composite structure				●		●		
Deployment				●		●		
Interaction overview				●		●		
Object				●		●		
Package				●		●		
Sequence	●			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

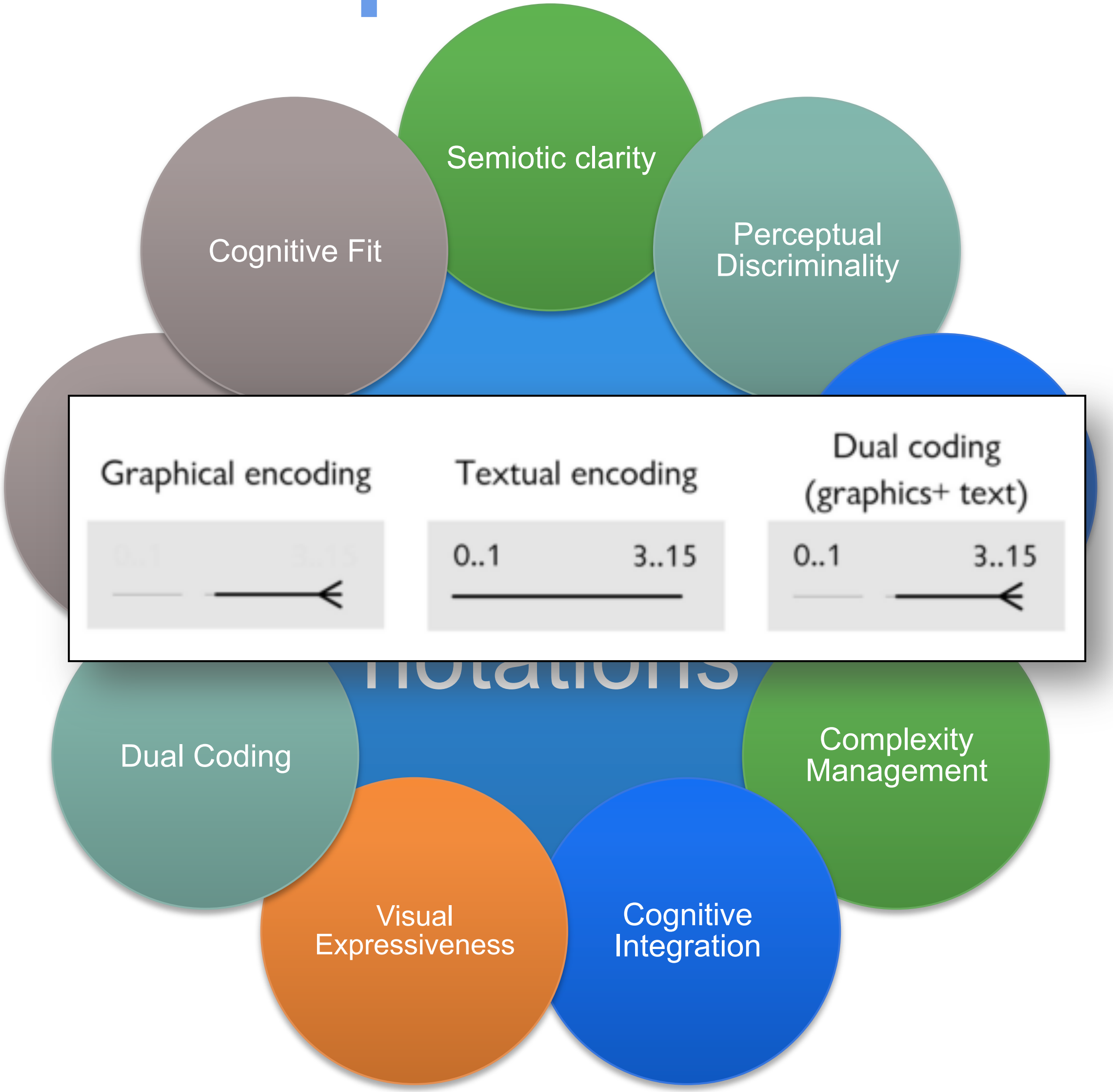
Visual Expressiveness
Cognitive Integration

Recommendations for Graphical DSLs

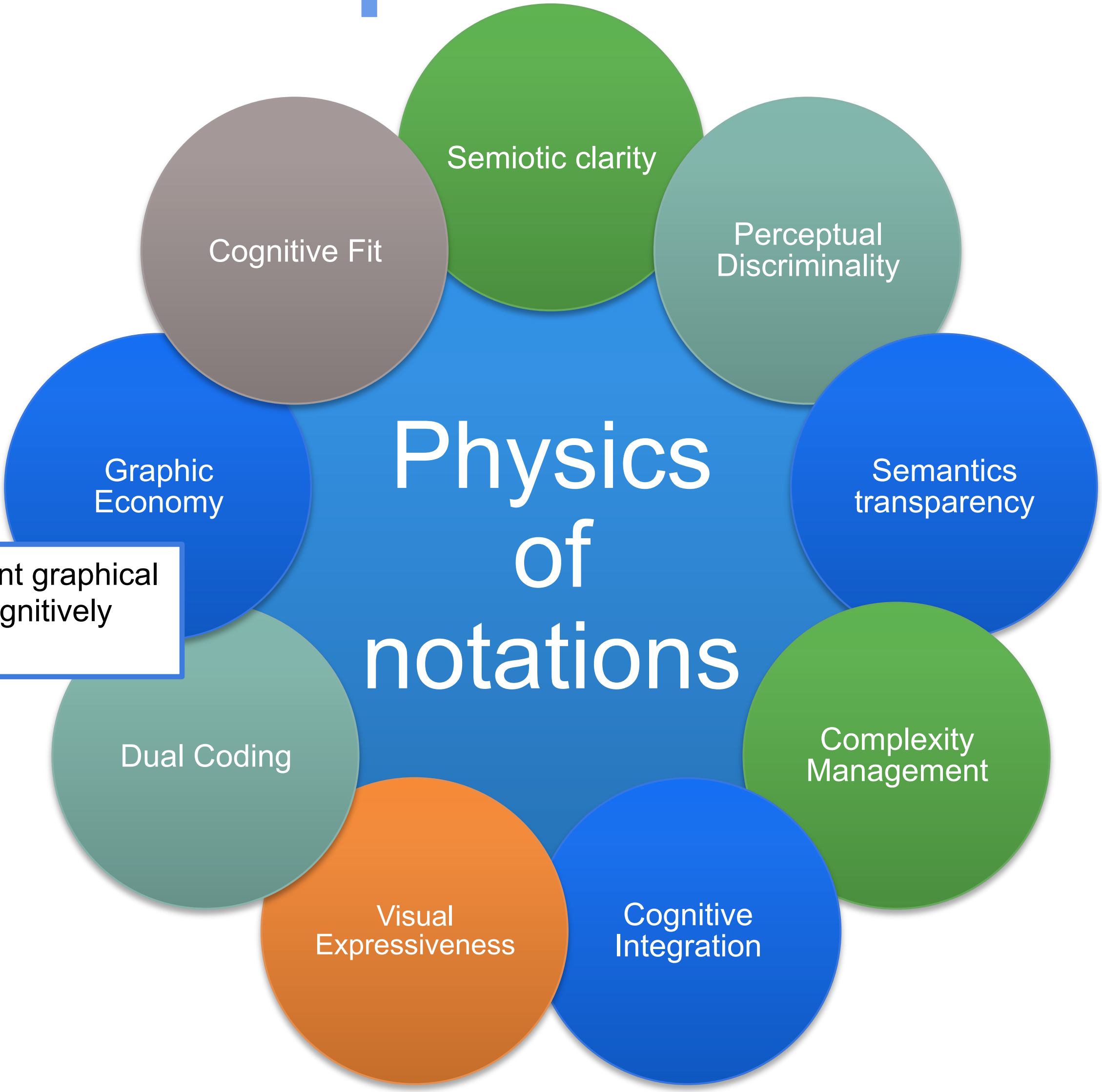


Use text to complement graphics

Recommendations for Graphical DSLs



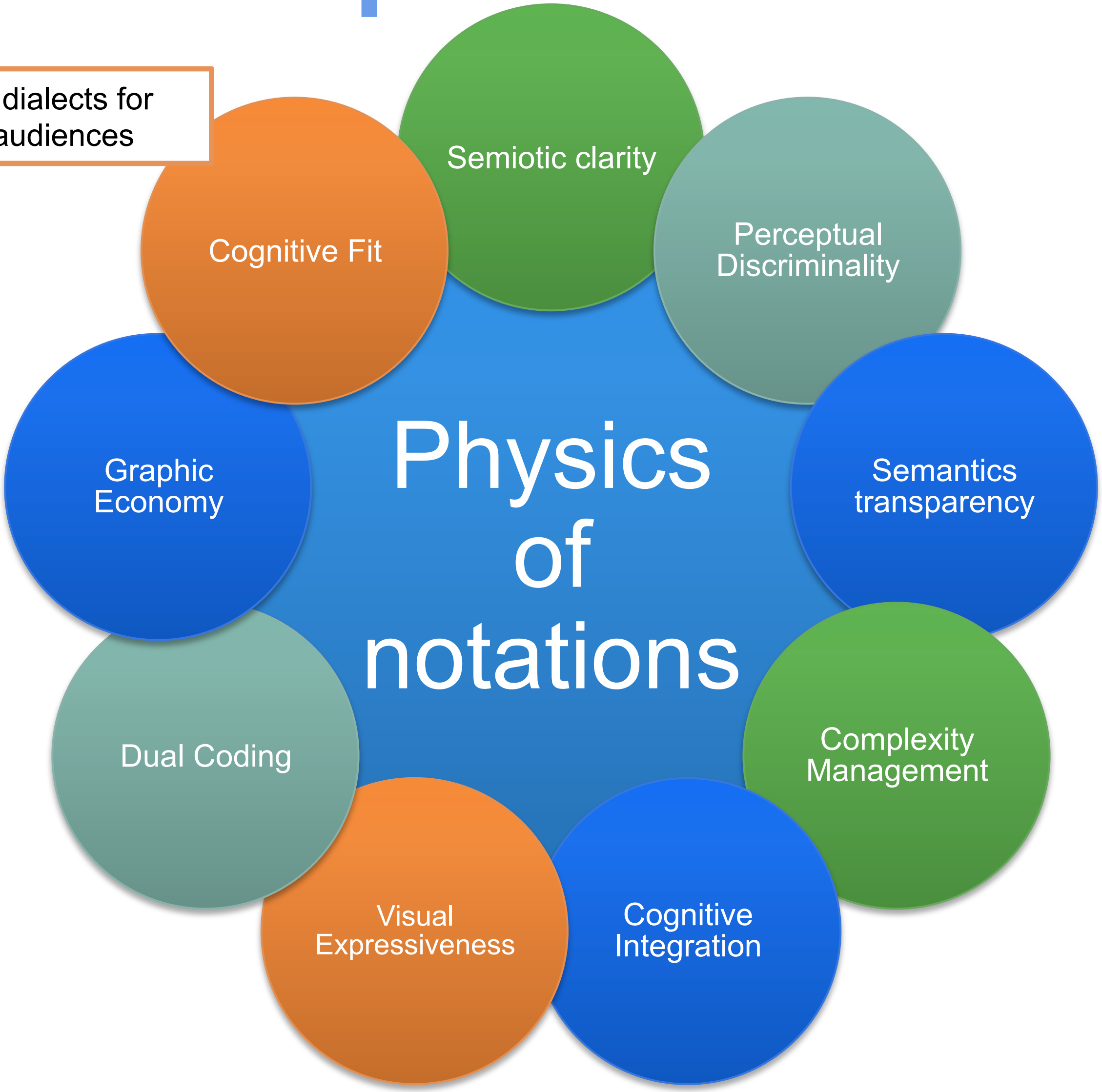
Recommendations for Graphical DSLs



The number of different graphical symbols should be cognitively manageable

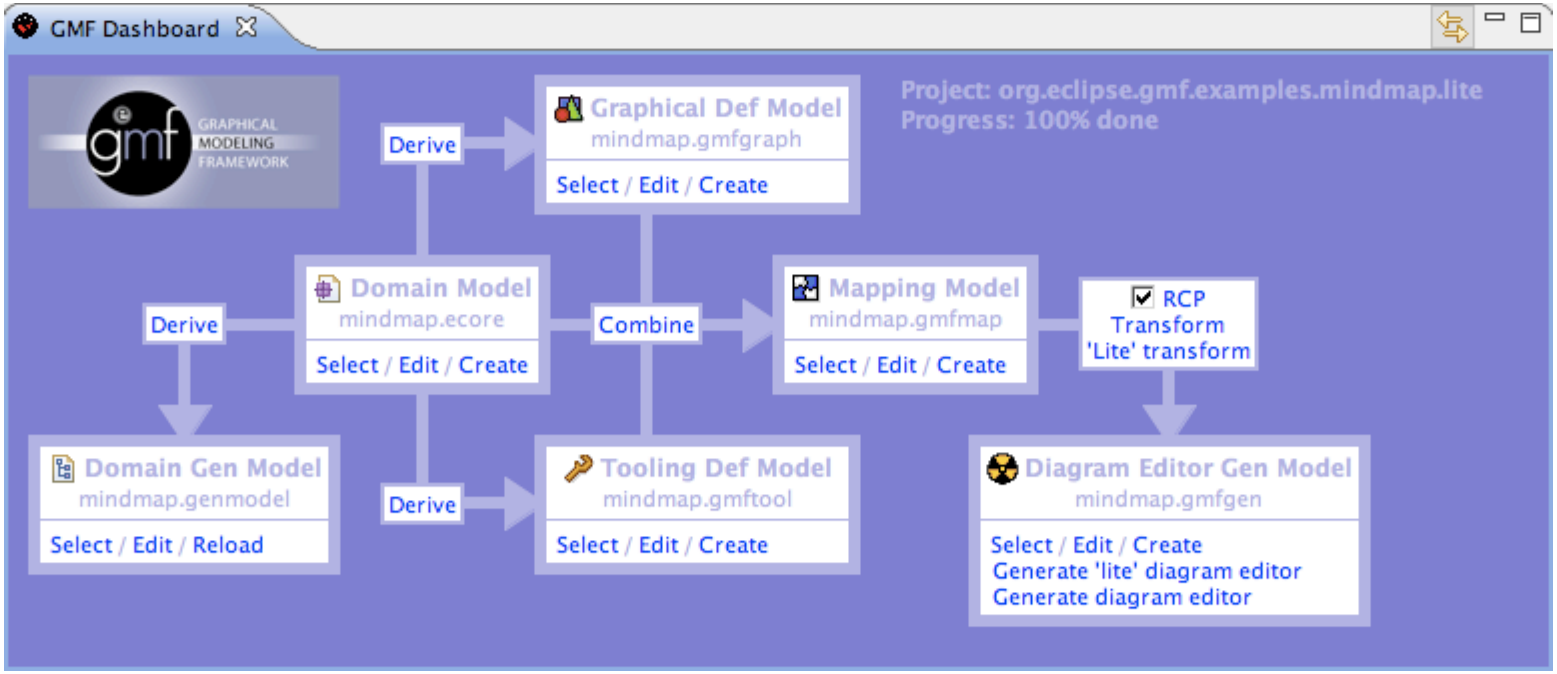
Recommendations for Graphical DSLs

Use different visual dialects for different tasks and audiences



Graphical Modeling Framework (GMF)

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



GMF Dashboard

Project: org.eclipse.gmf.examples.mindmap.lite
Progress: 100% done

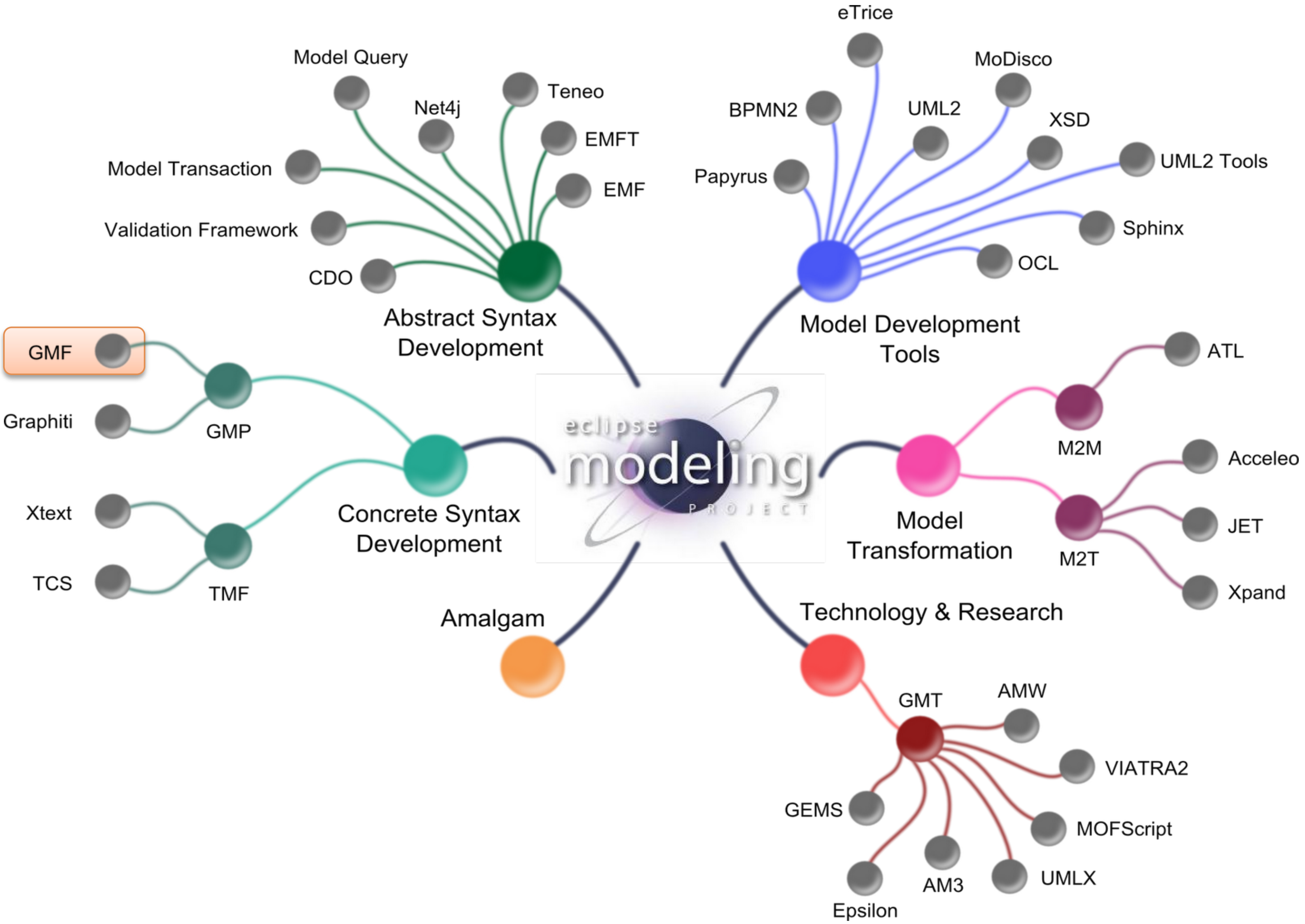
GMF

- Eclipse project
  - – Eclipse Modelling components
  - – Uses
    - EMF (Eclipse Modeling Framework)
    - GEF (Graphical Editing Framework)

- Model-driven framework for Graphical DSLs
  - – Everything is a model

- DSL definition easy, tweaking hard

Eclipse Modeling Project



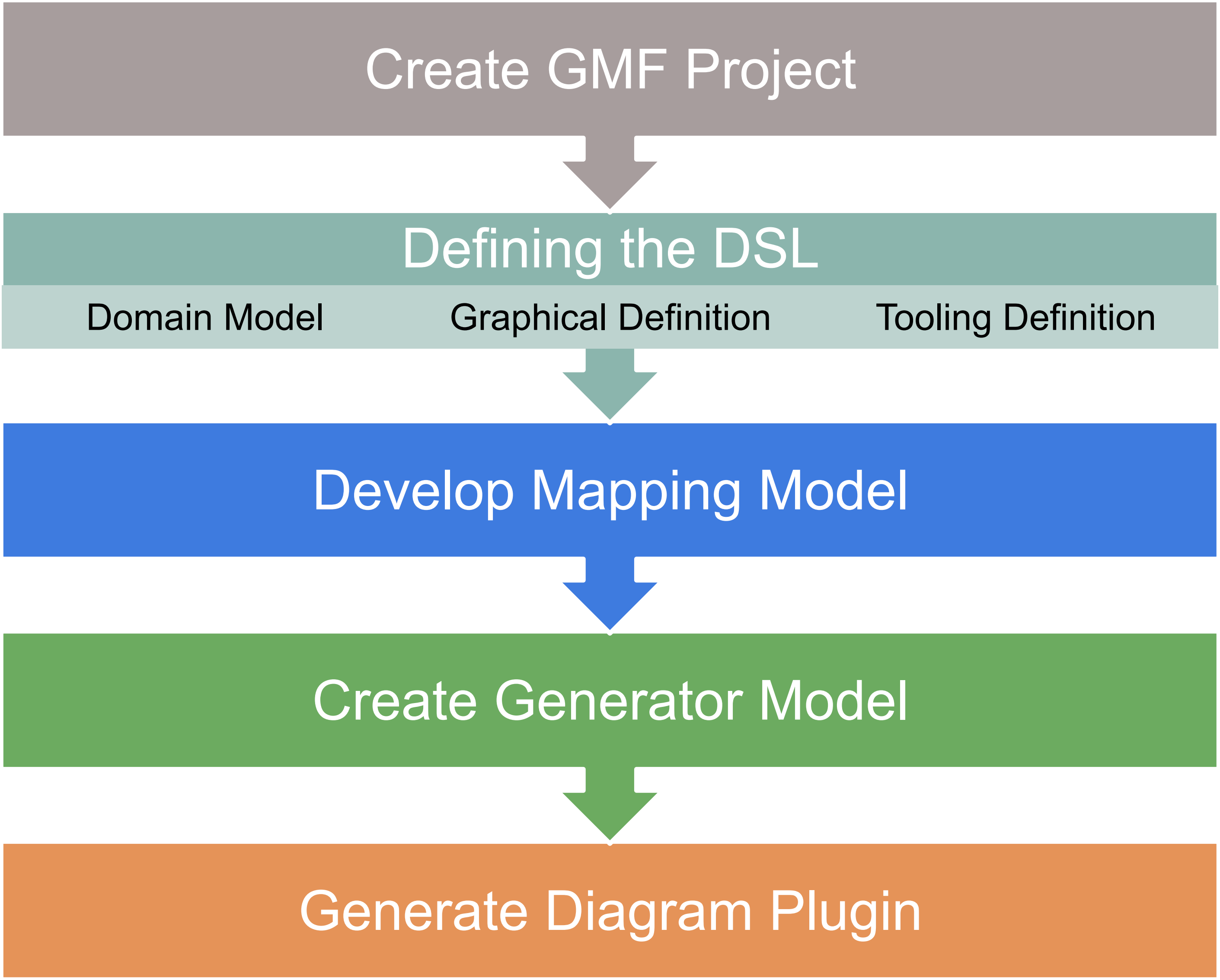
GMF features

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

Main Advantages

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

Development Process



Create GMF Project

Defining the DSL

Domain Model

Graphical Definition

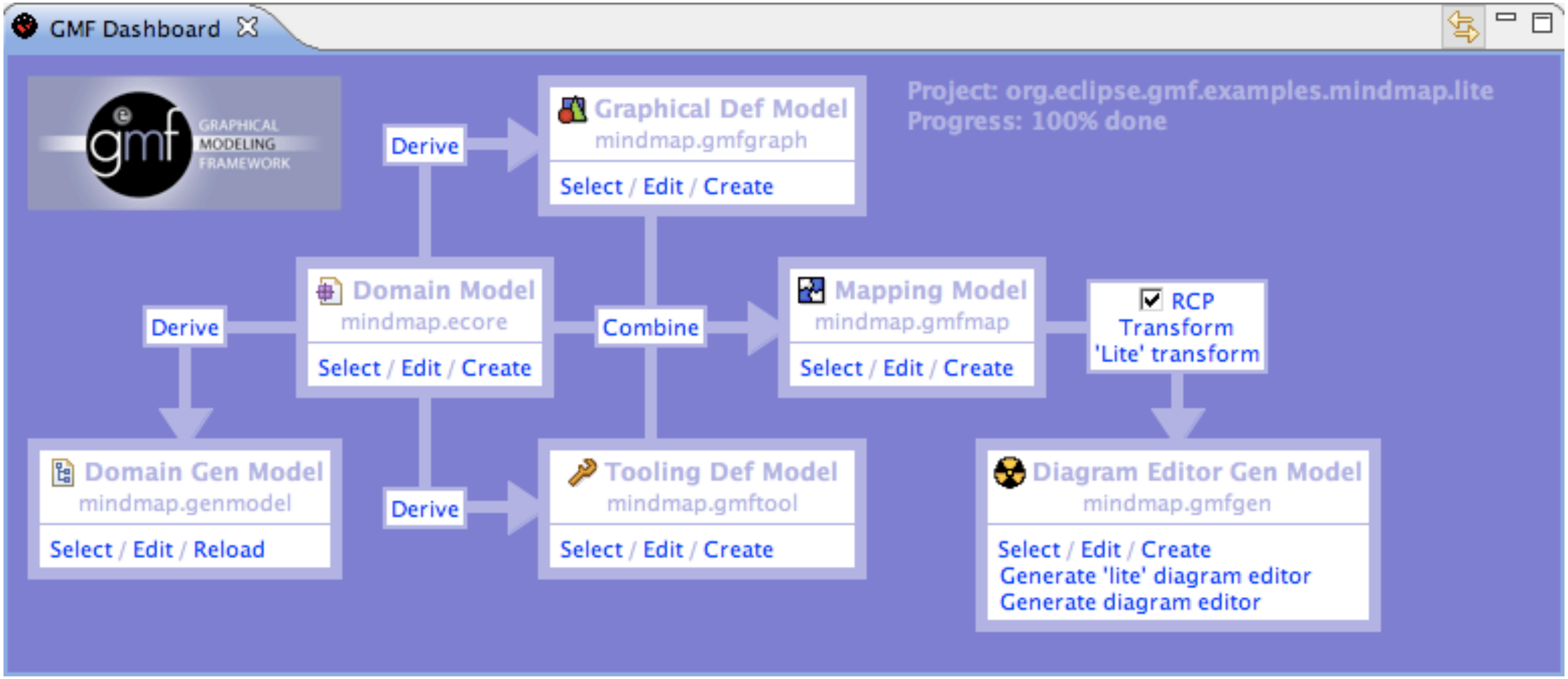
Tooling Definition

Develop Mapping Model

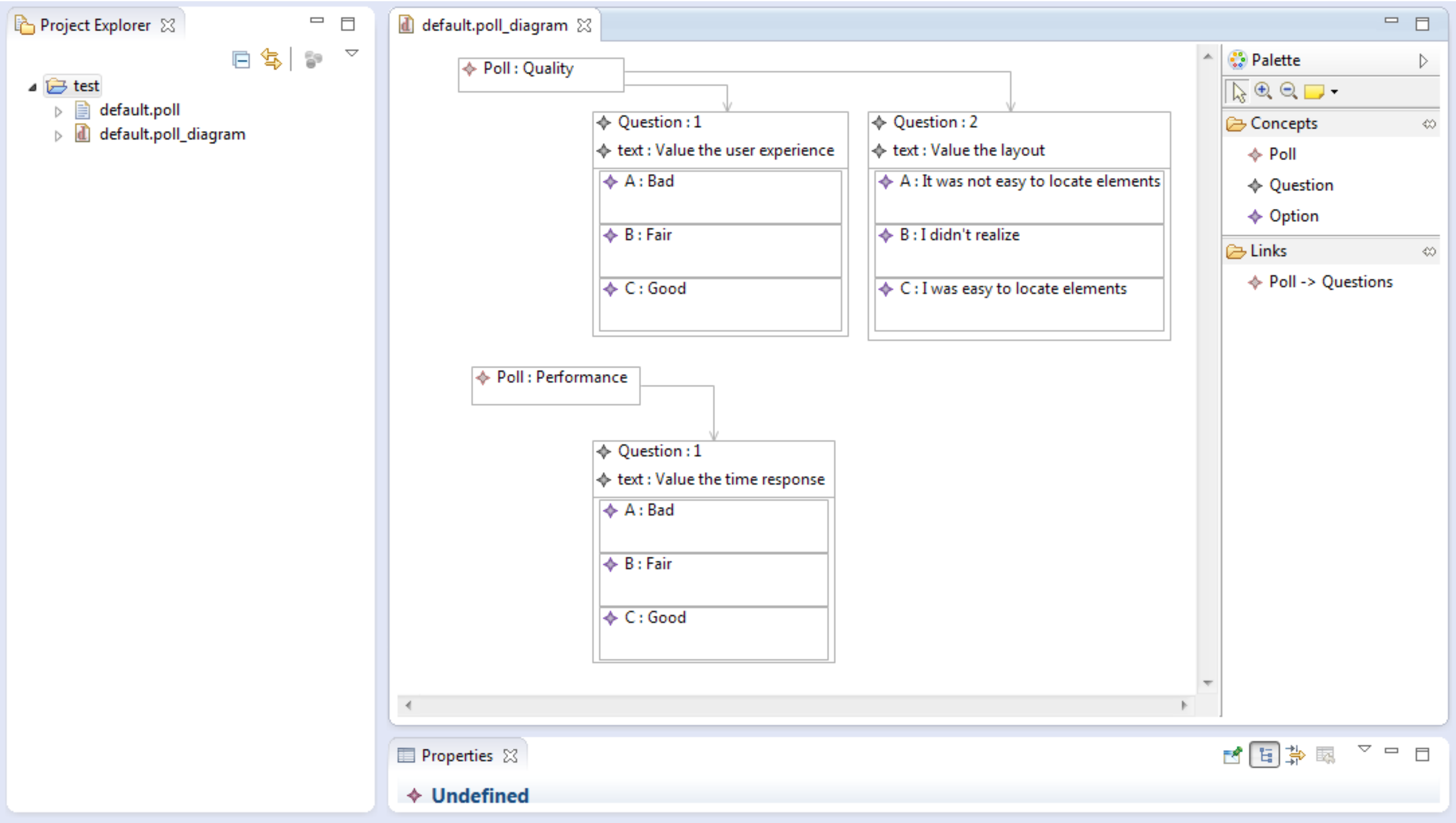
Create Generator Model

Generate Diagram Plugin

Development Process

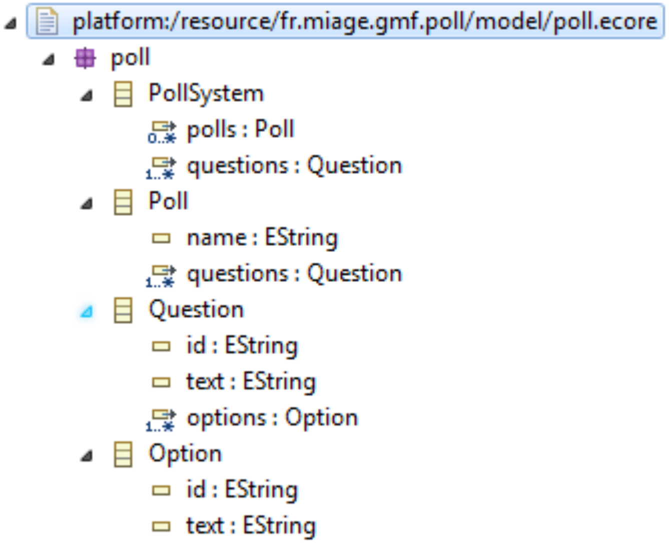


Example (Graphical Notation)



Poll System Metamodel

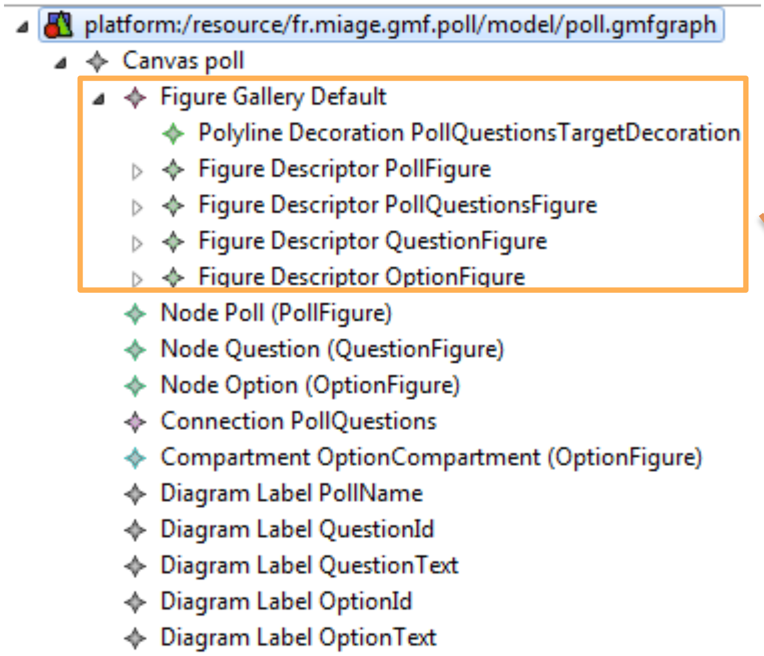
- Concepts
  - PollSystem
  - Poll
  - Question
  - Option
- Attributes
  - A Poll has a name
  - A Question has an identifier and a descriptive text
  - An Option has an identifier and a descriptive text
- Relationships
  - PollSystem is composed of polls and questions
  - Question has a set of options



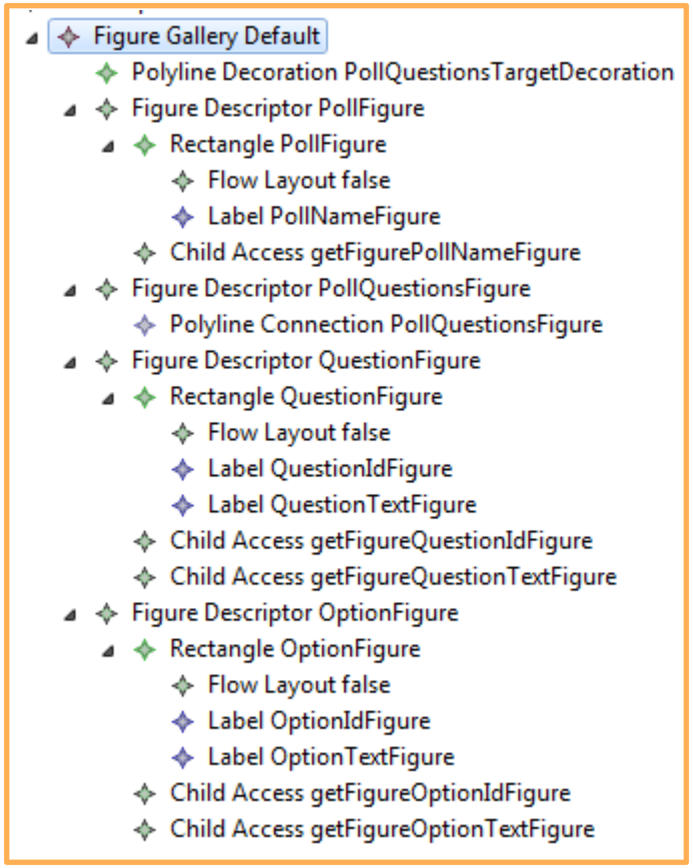
```
platform:/resource/fr.miage.gmf.poll/model/poll.ecore
  poll
    PollSystem
      polls : Poll
      questions : Question
    Poll
      name : EString
      questions : Question
    Question
      id : EString
      text : EString
      options : Option
    Option
      id : EString
      text :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

Graphical Definition

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



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



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

Plan

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

Contract

- Better understanding/source of inspiration of software languages and DSLs
  – Revisit of history and existing languages

- Foundations and practice of Xtext
  – State-of-the-art language workbench (Most Innovative Eclipse Project in 2010, mature and used in a variety of industries)

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

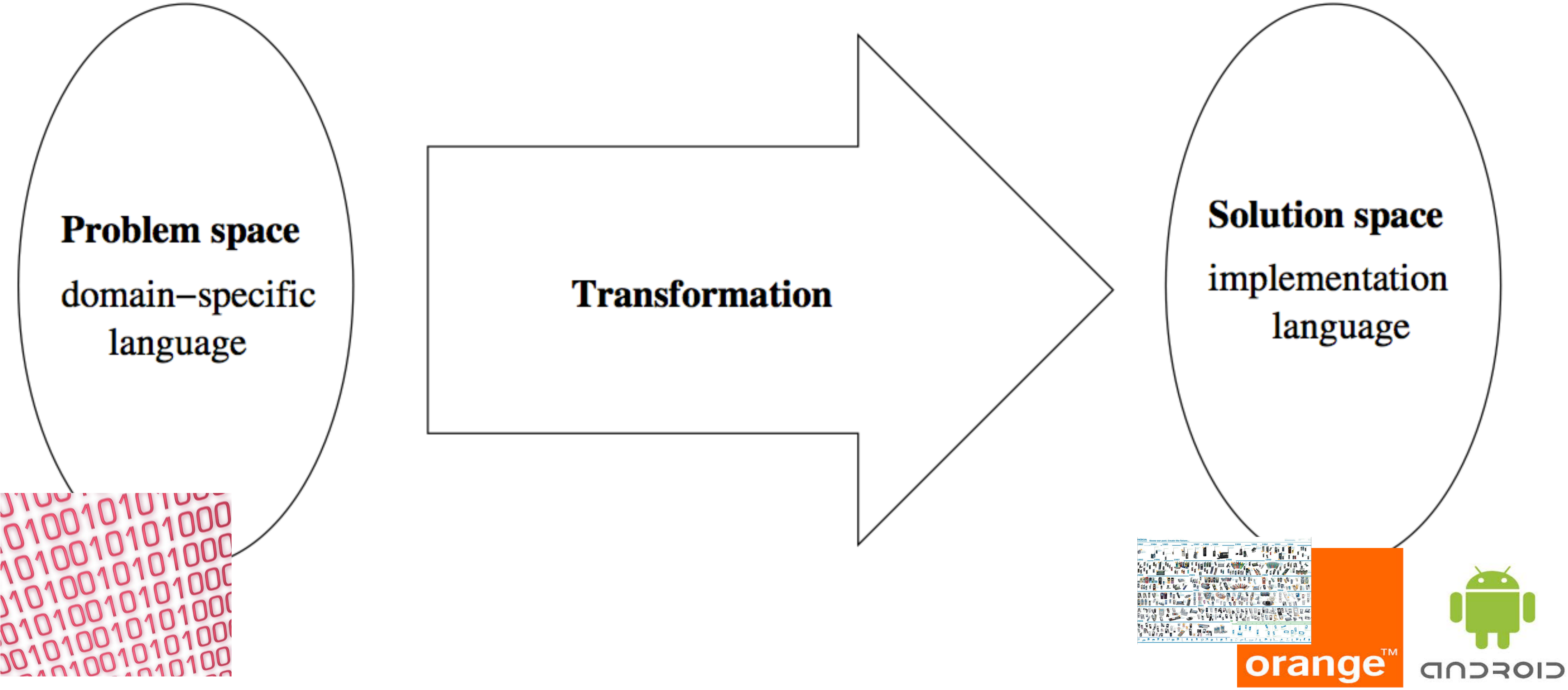
DSL,

Model,

Metamodel,

Summary

Abstraction Gap



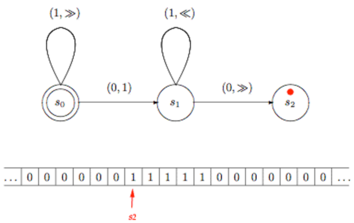
Problem space

domain–specific language

Transformation

Solution space

implementation language



Models/MDE

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

Generative approach

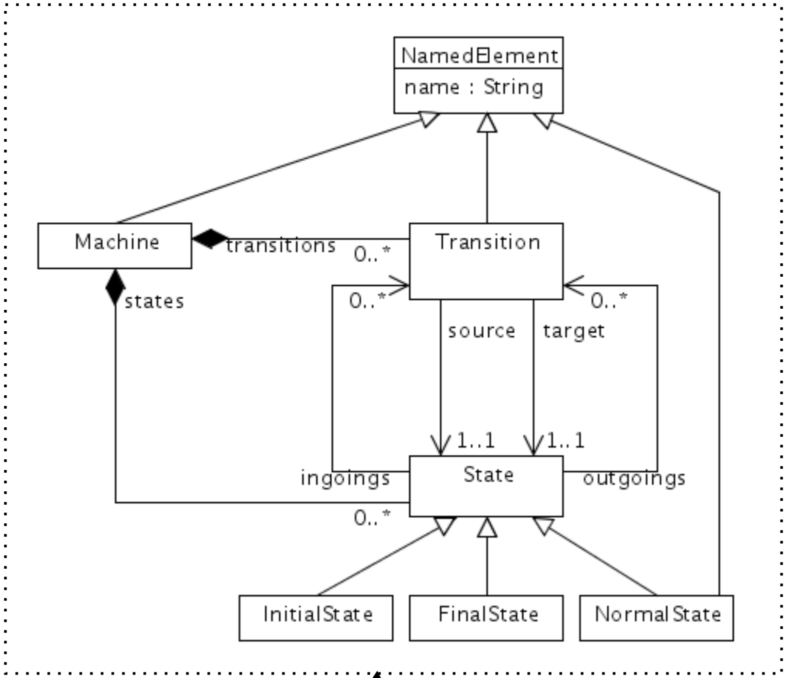
- Programming the generation of programs
  - Very old practice
  - Metaprogramming: generative language and target language are the same
    - Reflection capabilities

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

Grammar

```
machineDefinition:
   MACHINE OPEN_SEP stateList
   transitionList CLOSE_SEP;

stateList:
   state (COMMA state)*;

state:
   ID_STATE;

transitionList:
   transition (COMMA transition)*;

transition:
   ID_TRANSITION OPEN_SEP
   state state CLOSE_SEP;

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

MetaModel



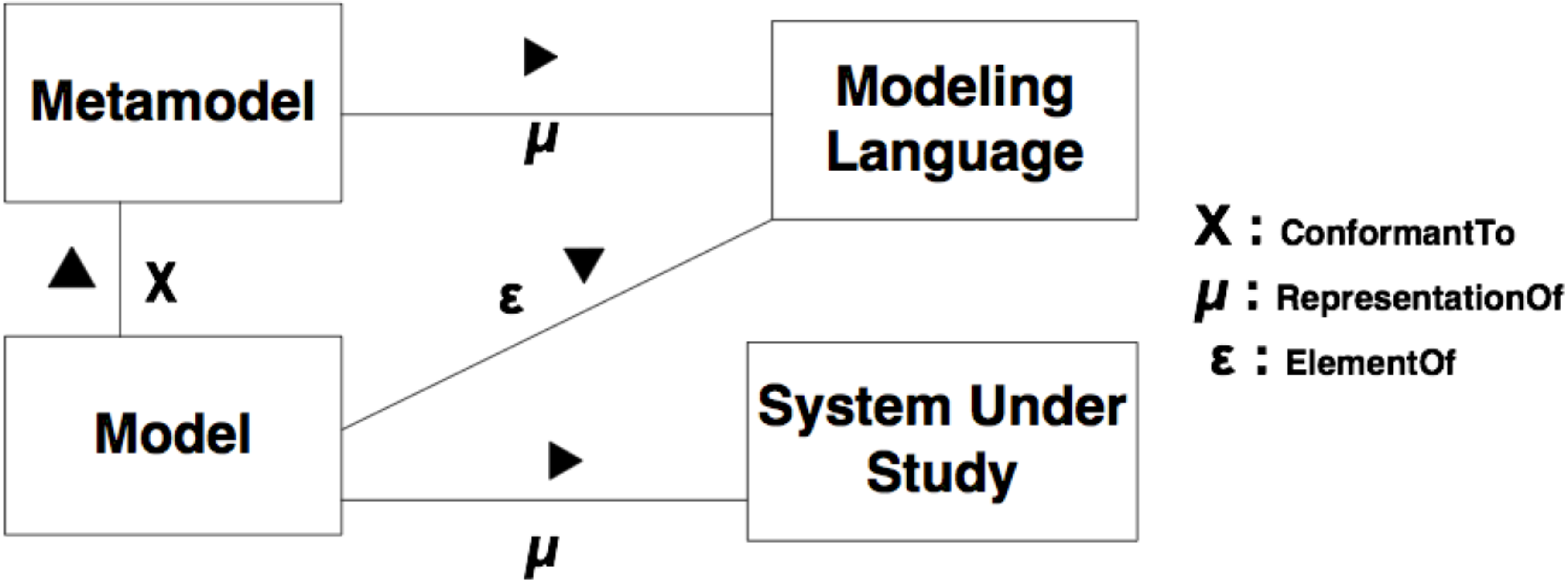
conforms To

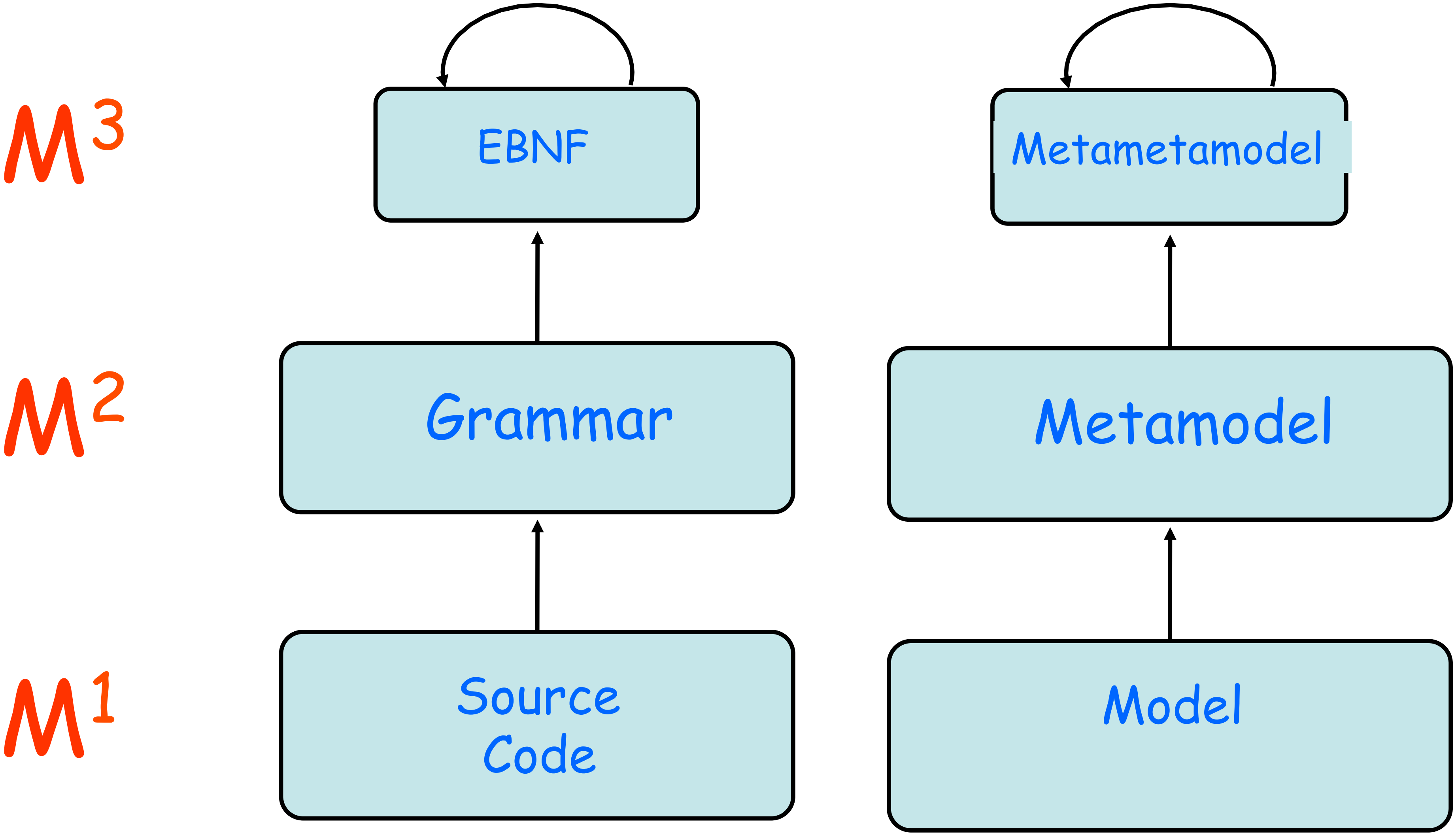
conforms To

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

Source Code/Mode

Model, Metamodel, Metametamodel, DSML





M^3

EBNF

Metametamodel

M^2

Grammar

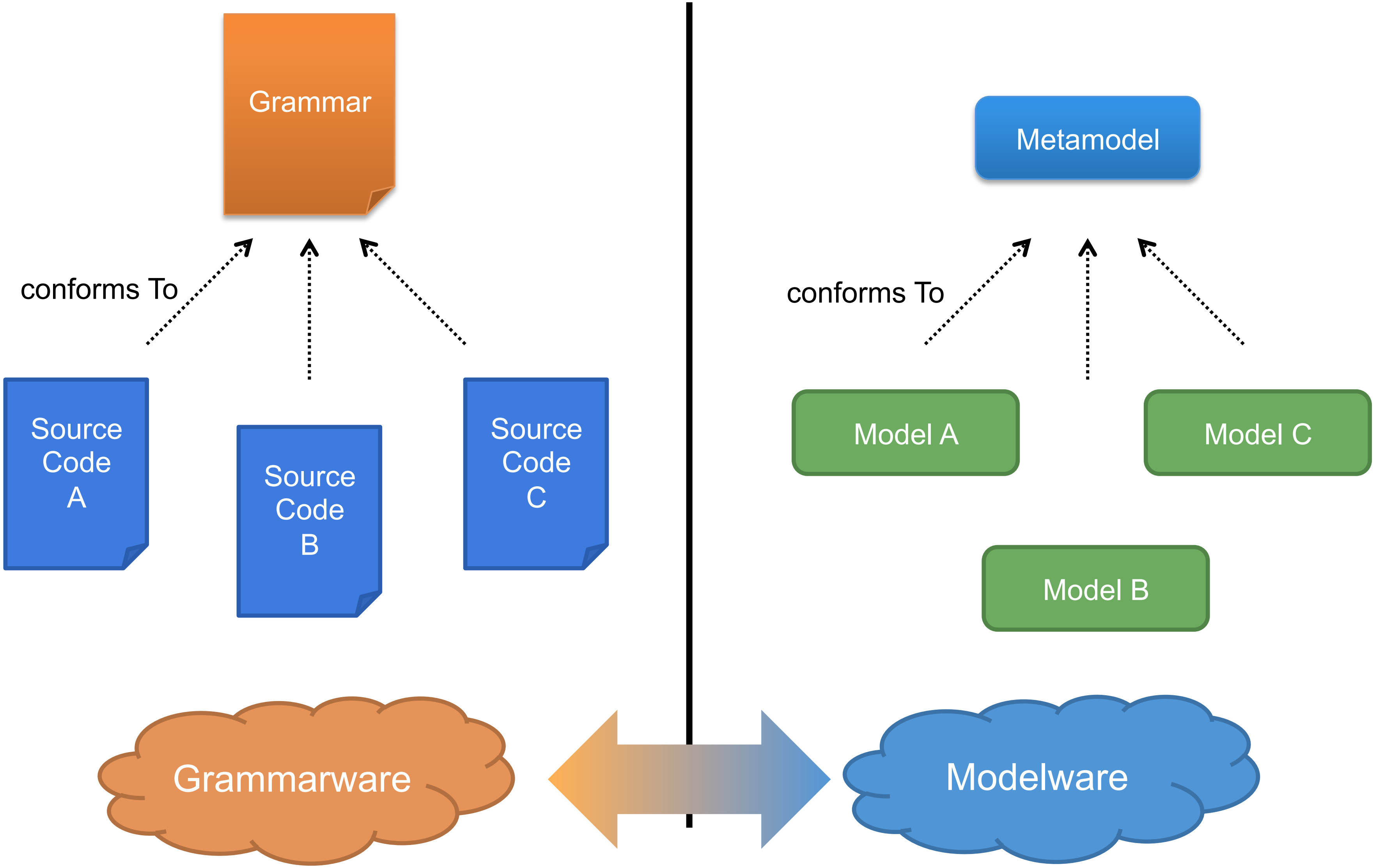
Metamodel

M^1

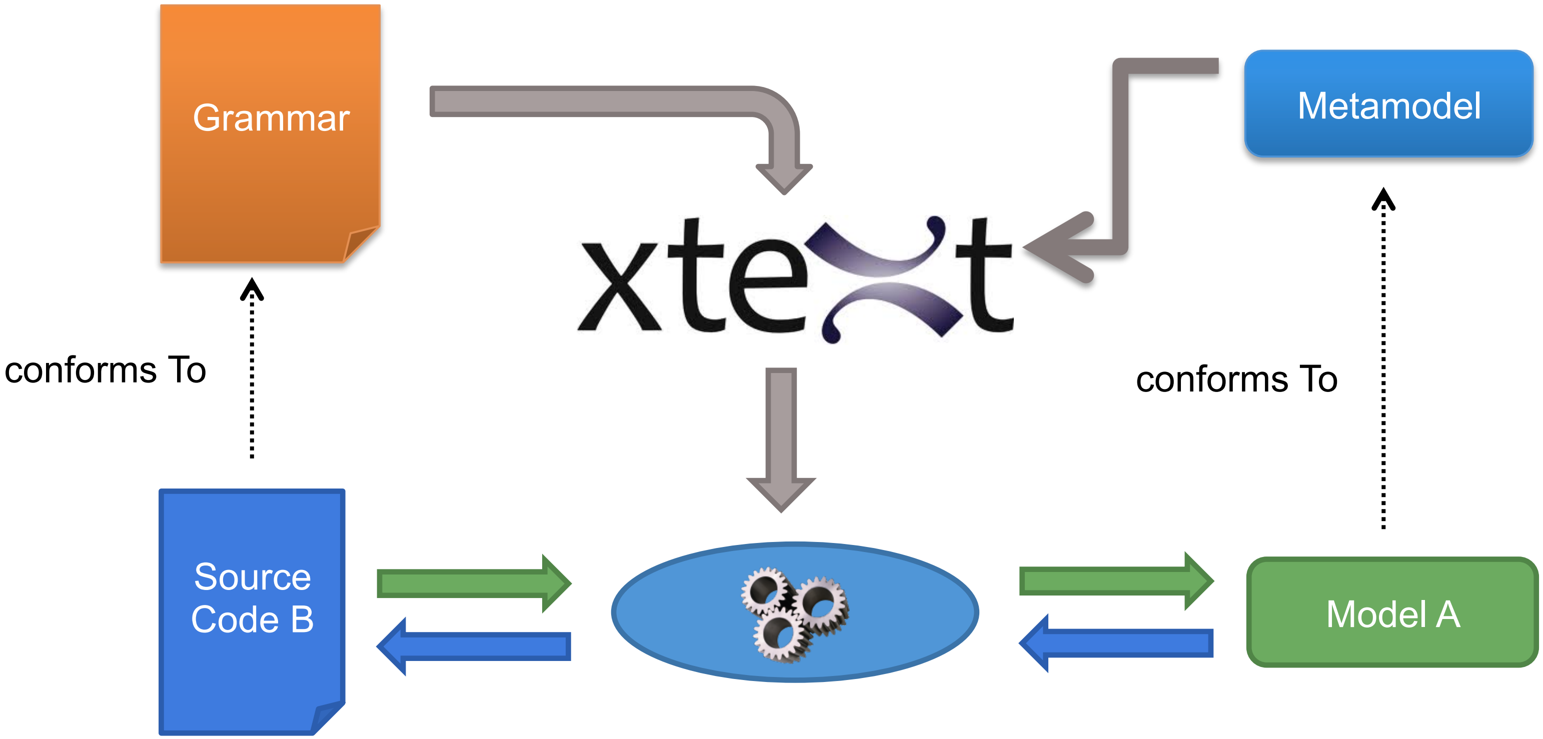
Source Code

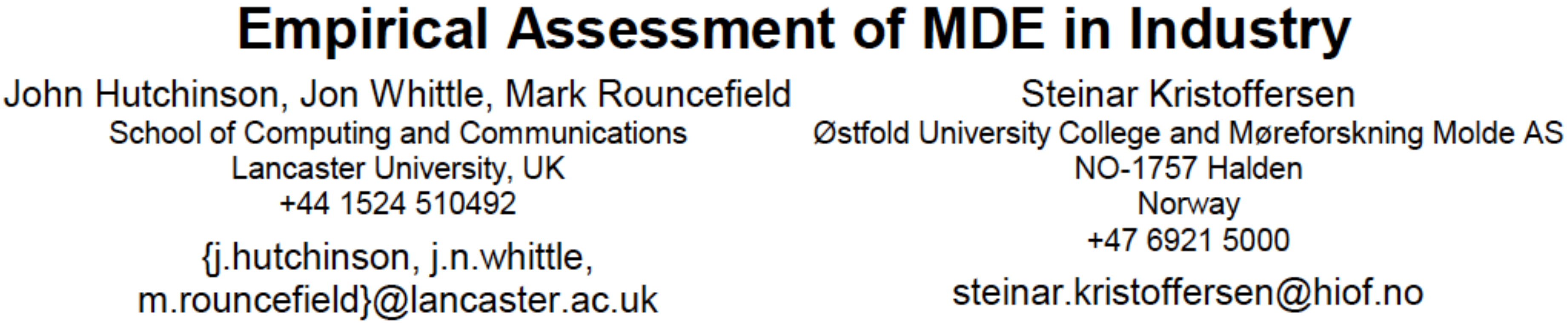
Model

Language and MDE



MDE, Grammar: there and back again

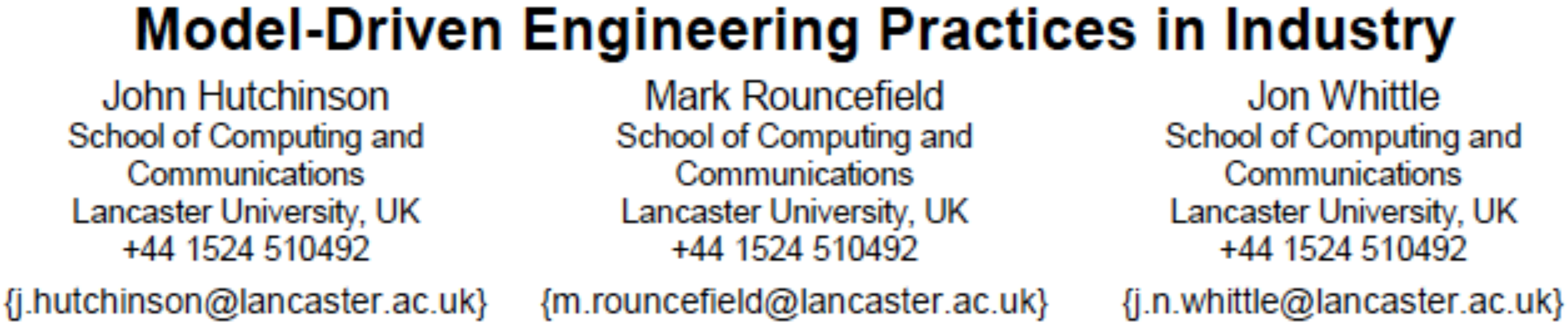




Empirical Assessment of MDE in Industry

John Hutchinson, Jon Whittle, Mark Rouncefield
School of Computing and Communications
Lancaster University, UK
+44 1524 510492
{j.hutchinson, j.n.whittle,
m.rouncefield}@lancaster.ac.uk

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



Model-Driven Engineering Practices in Industry

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

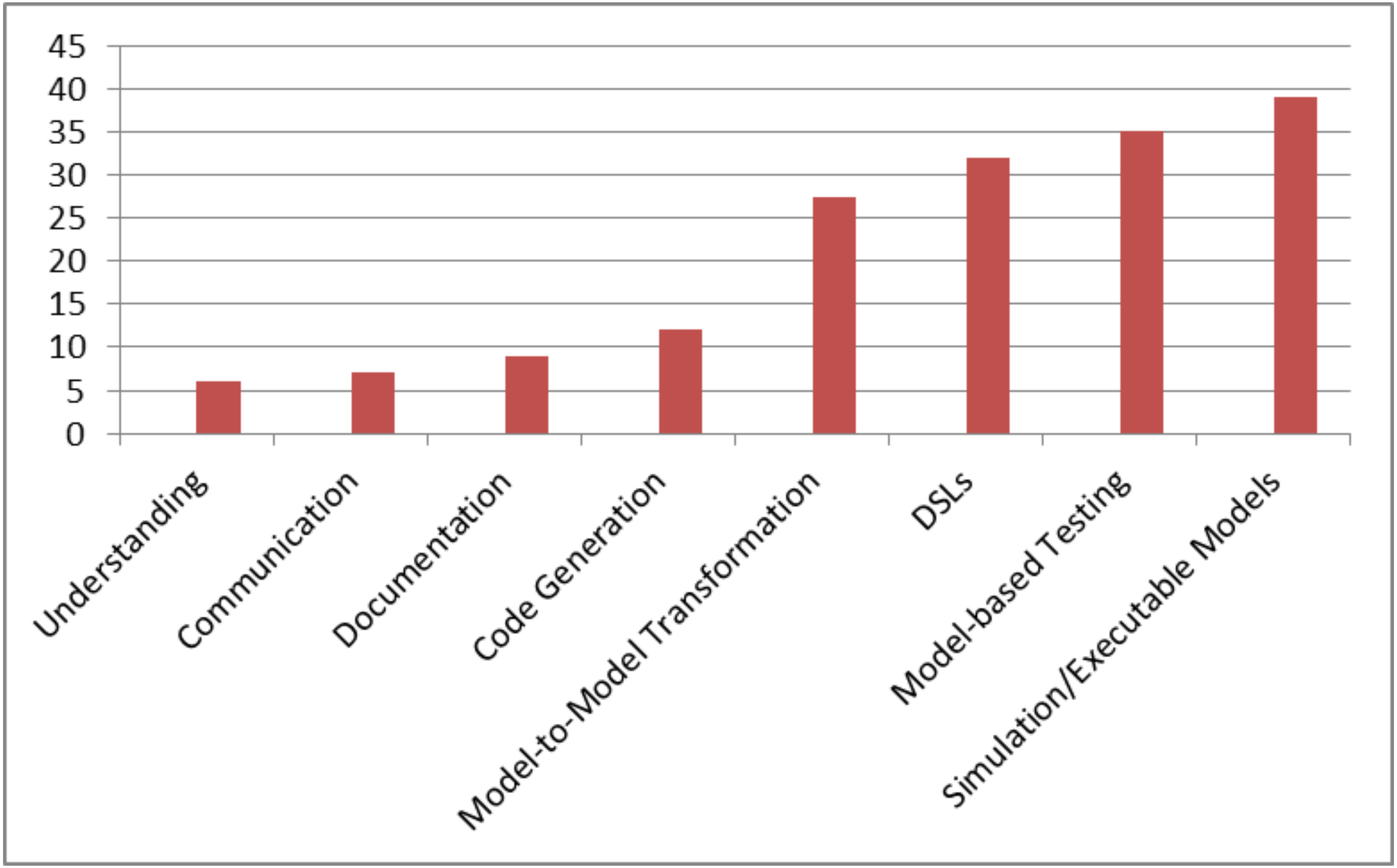
Mark Rouncefield
School of Computing and Communications
Lancaster University, UK
+44 1524 510492
{m.rouncefield@lancaster.ac.uk}

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

2011

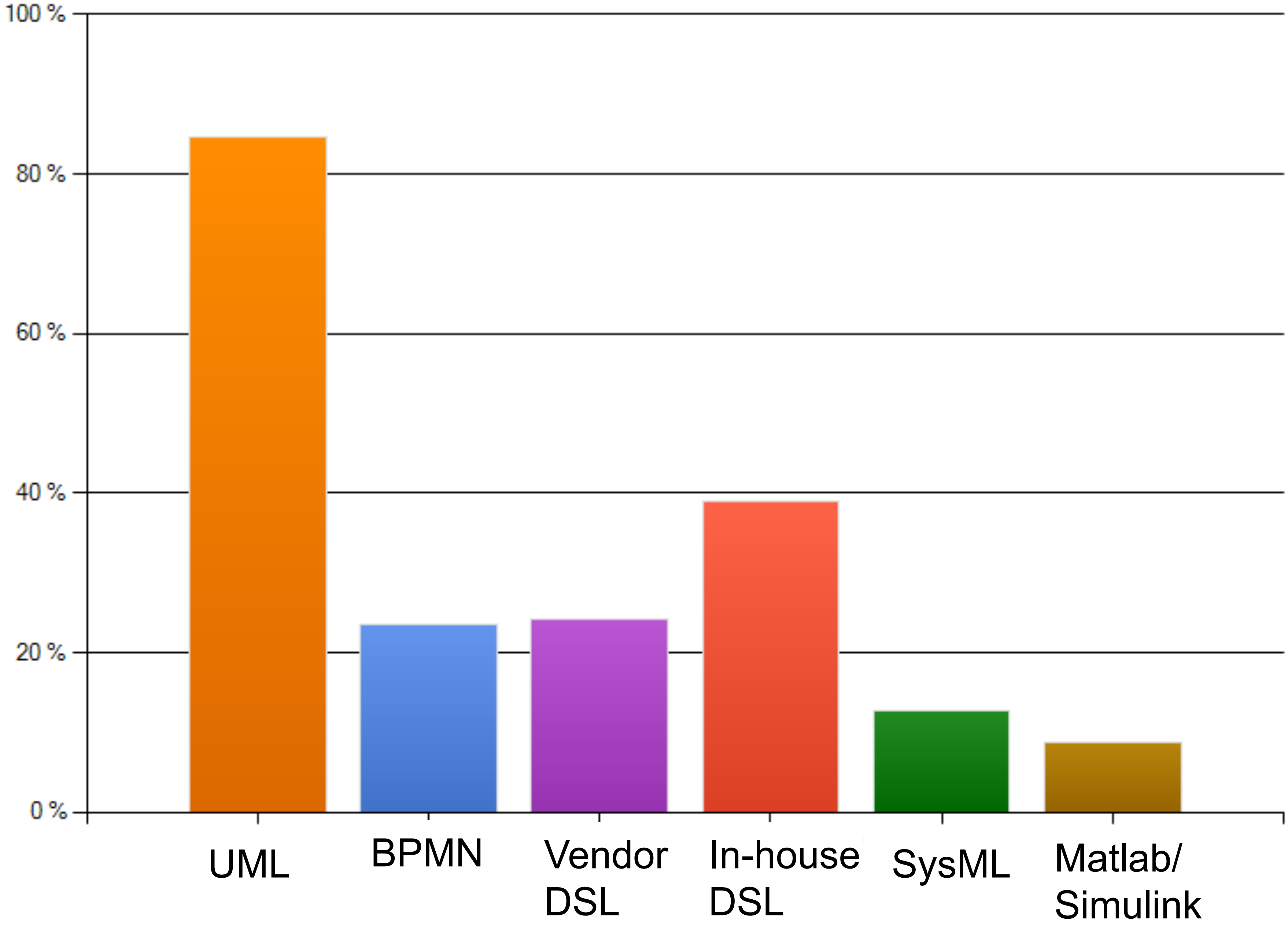
« Domain-specific languages are far more prevalent than anticipated »

What are models used for?

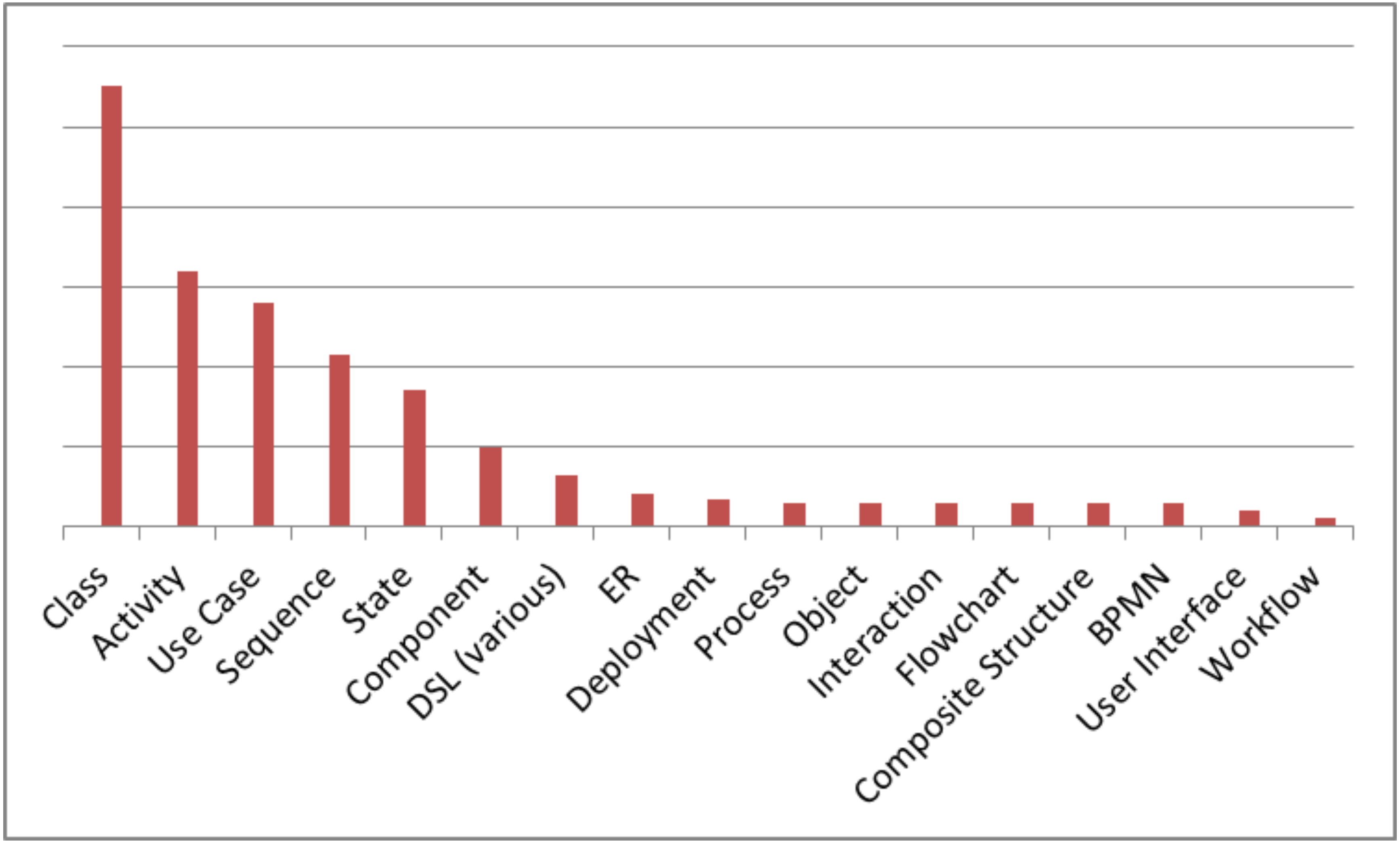


“Do not use” percentages for MDE activities

Which modeling languages do you use?



Which diagrams are used?

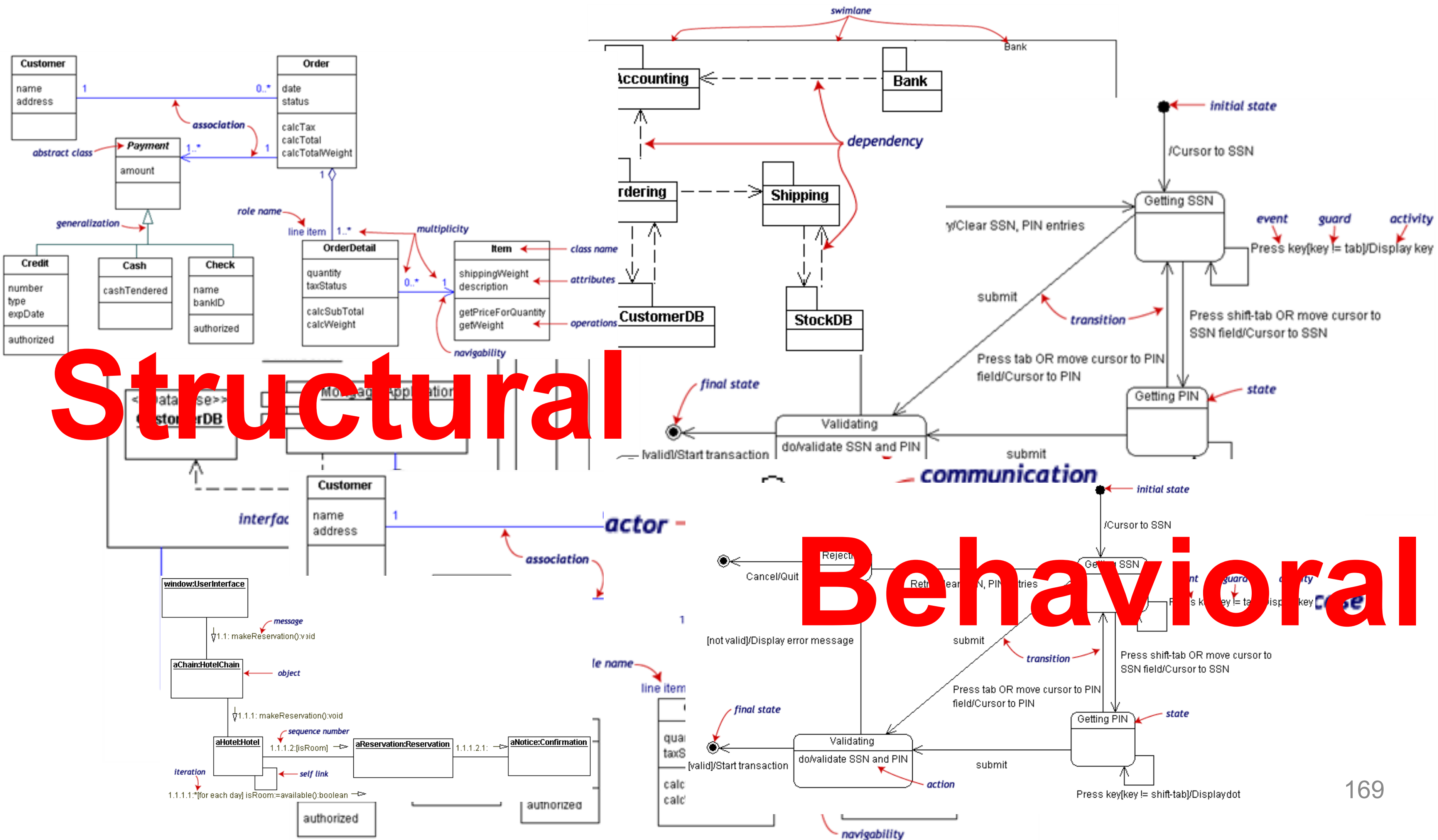


19 different diagram types are used regularly

Use of multiple languages (DSLs)

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

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



Structural

Behavioral

Xtext is built using MDE technologies

Xtext (and alternatives) democratize DSL development

My 3 take away messages

#1 DSLs are important (as intuited for a long time – it will become more and more apparent)

#2 DSL technology is here (no excuse)

#3 MDE meets language engineering

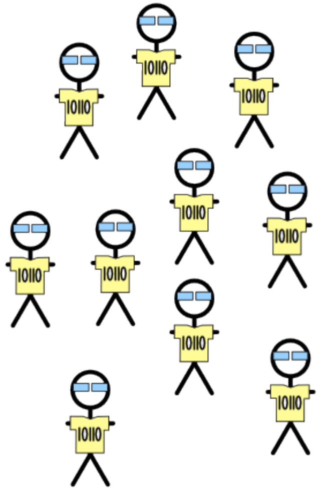
But my take away message is NOT

That DSLs should be used systematically, in every situations

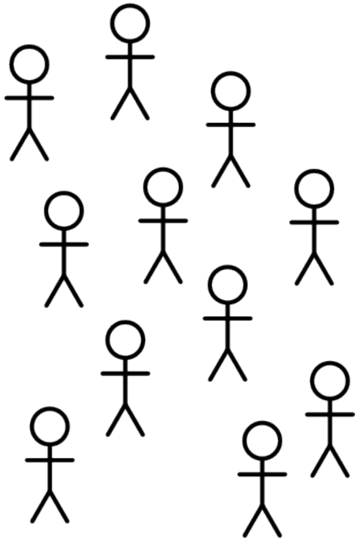
When Developing DSLs?

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

Actors

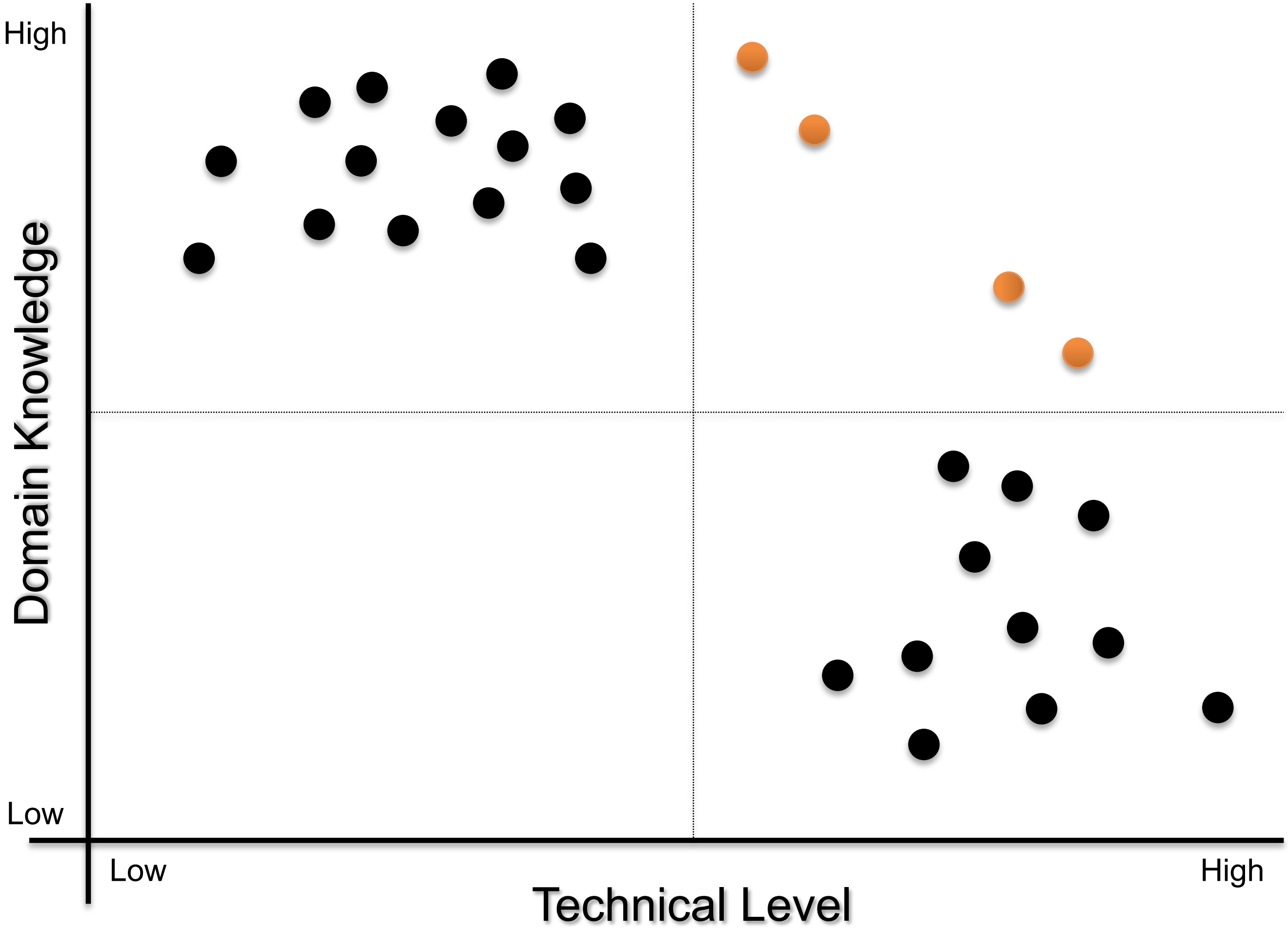


Developers



End-Users

Actors



Best Practices

Limit Expressiveness

Viewpoints

Evolution

Learn from GPLs

Support

Tooling

Worst Practices

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

Worst Practices

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

Worst Practices

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

Worst Practices

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

Worst Practices

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

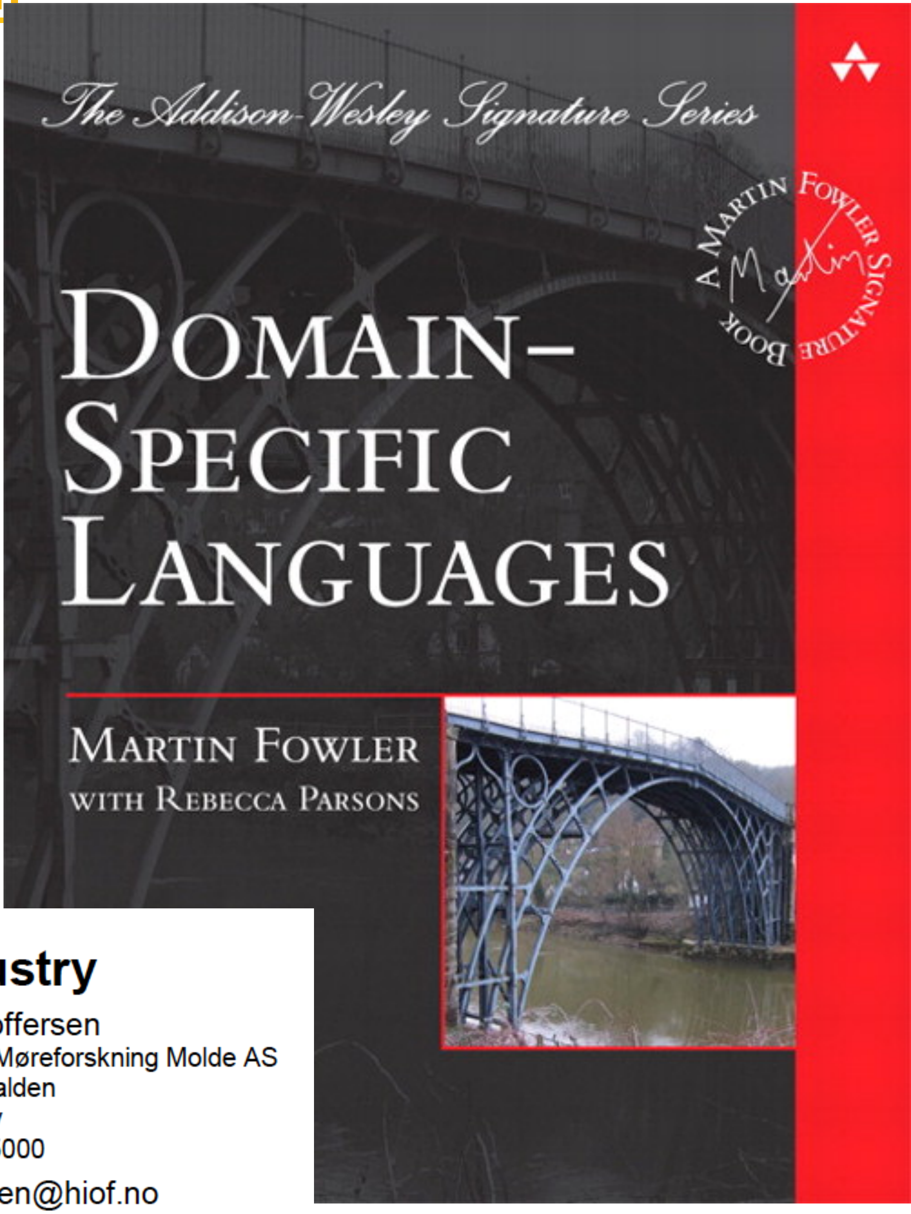
Questions ?

(see also resources and lab sessions)

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

xtext

gmf GRAPHICAL MODELING FRAMEWORK


The Addison-Wesley Signature Series

DOMAIN-SPECIFIC LANGUAGES

MARTIN FOWLER
WITH REBECCA PARSONS

A MARTIN FOWLER SIGNATURE BOOK

Empirical Assessment of MDE in Industry

nn 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