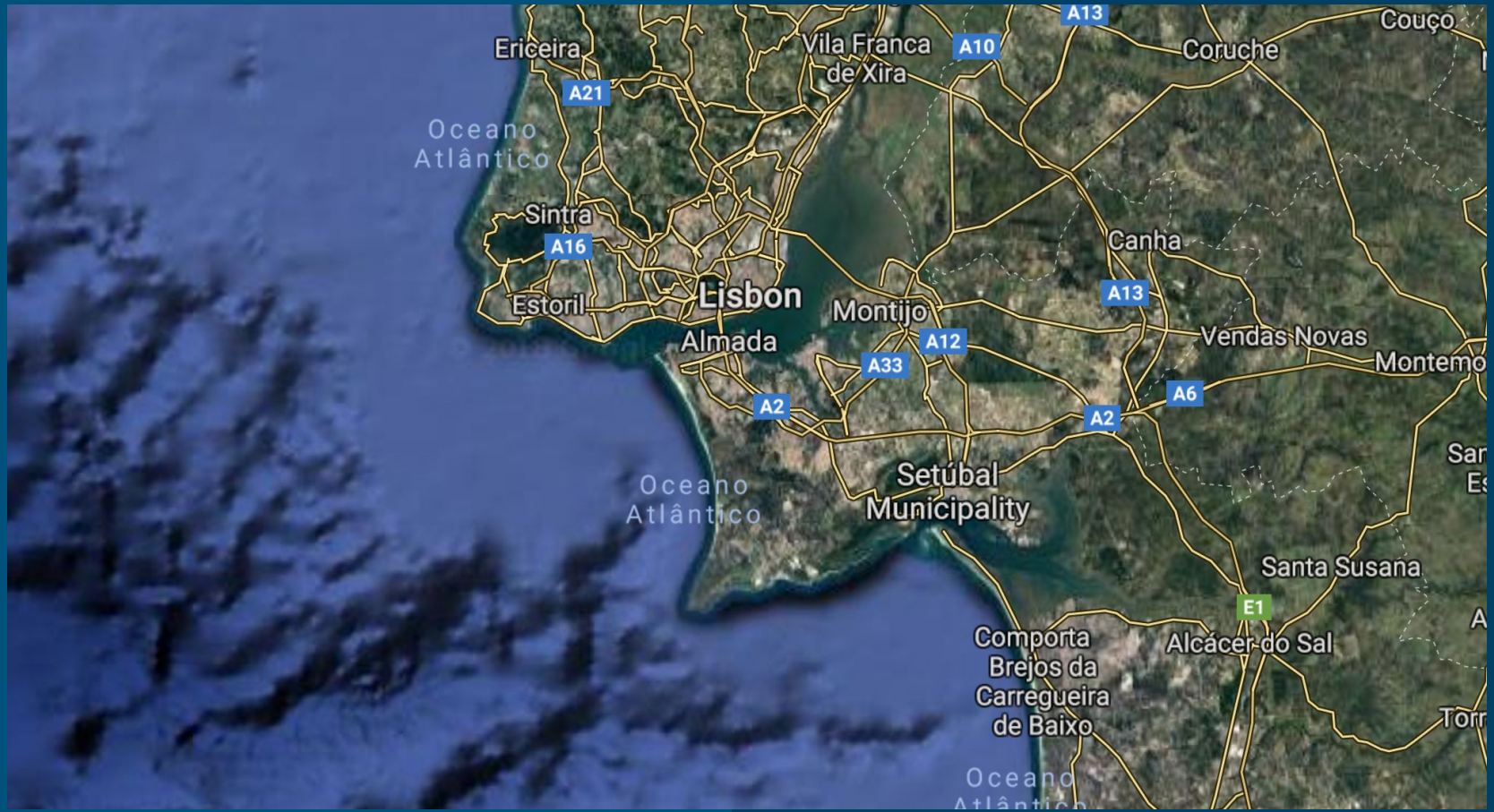
Software Languages Engineering - Being Systematic

by Prof. Vasco Amaral
DI FCT/UNL
ASE group NOVA-LINCS
Lecture at the University of Novi Sad, 21/3/2019

A little bit about myself


North Sea
United Kingdom
Ireland
Denmark
Lithuania
Belarus
Poland
Netherlands
Germany
Ukraine
Belgium
Czechia
Slovakia
Moldova
Austria
Hungary
Romania
France
Black Sea
Croatia
Serbia
Bulgaria
Italy
Turkey
Portugal
Greece
Tyrrhenian Sea
Syria
Spain
Lebanon
Israel
Jordan
Mediterranean Sea


Gijón
A Coruña
Oviedo
Bilbao
San Sebastián
Andorra
Santiago de Compostela
León
Vitoria-Gasteiz
Pontevedra
Ourense
Zaragoza
Vigo
Valladolid
Tarragona
Porto
Salamanca
Madrid
Castellón de la Plana
Coimbra
Toledo
Valencia
Portugal
Spain
Benidorm
Lisbon
Badajoz
Alicante
Murcia
Córdoba
Seville
Granada
Huelva
Almería
Albufeira
Málaga
Faro


Ericeira
Vila Franca de Xira
A13
A10
Couço
Coruche
A21
Oceano Atlântico
Sintra
A16
Canha
Lisbon
Estoril
Montijo
A13
Almada
A12
Vendas Novas
Montemo
A33
A6
A2
A2
Setúbal Municipality
Oceano Atlântico
Santa Susana
E1
Comporta
Brejos da Carregueira de Baixo
Alcácer do Sal
Oceano

THE UNIVERSITY

9 Schools

9 Libraries

3 Halls of Residence

DEGREE PROGRAMMES

25 Bachelor

12 Integrated Masters

103 Masters

79 Doctorate

RESEARCH

40 Research Units

75% evaluated by the Foundation for Science and Technology (FC&T) with "Exceptional", "Excellent" and "Very Good"

4151 Publications

1489 Publications indexed to the Web of Science

17 European Research Council Grants

ACADEMIC AND RESEARCH STAFF

1716 Teachers and Researchers

1172 Teachers and Researchers FTE

99,50% Tenure track

NOVA IN INTERNATIONAL RANKINGS

QS BY SUBJECT

Top 10 among young European universities in 5 areas evaluated by QS:

- Arts & Humanities (4th);
- Engineering & Technology: (10th);
- Life Sciences & Medicine (6th);
- Natural Sciences (7th);
- Social Sciences & Management (7th).

INTERNACIONAL

2119 International students enrolled

103 Nationalities

491 International graduate students

111 International Teachers and Researchers

82 Degree Programmes taught in English

+400 Mobility partnership

63 Countries

903 Erasmus Incoming students

590 Erasmus Outgoing students

INTERNACIONAL

2119 International students enrolled

103 Nationalities

491 International graduate students

111 International Teachers and Researchers

82 Degree Programmes taught in English

+400 Mobility partnership

63 Countries

903 Erasmus Incoming students

590 Erasmus Outgoing students

STUDENTS

19867 Students

5930 Bachelor

7023 Integrated Master

4489 Master

2048 Doctorate

377 non-degree programmes

SCHOOLS

- Faculdade de Ciências e Tecnologia
- Faculdade de Ciências Sociais e Humanas
- NOVA School of Business and Economics
- NOVA Medical School / Faculdade de Ciências Médicas
- Faculdade de Direito
  Instituto de Higiene e Medicina Tropical
- NOVA Information Management School
- Instituto de Tecnologia Química e Biológica António Xavier
- Escola Nacional de Saúde Pública

UNIVERSIDADE NOVA DE LISBOA

- Faculdade de Ciências e Tecnologia
- Faculdade de Ciências Sociais e Humanas
- NOVA School of Business and Economics
- NOVA Medical School / Faculdade de Ciências Médicas
- Faculdade de Direito
  Instituto de Higiene e Medicina Tropical
- NOVA Information Management School
- Instituto de Tecnologia Química e Biológica António Xavier
- Escola Nacional de Saúde Pública

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Department of computer science FCT

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Faculty of Sciences and Technology (FCT NOVA)

16 research centres

1600 PhD

7800 Master’s students

550 academic staff (90% holding a Ph.D)

180 non-academic staff

14 departments

NOVA LINCS

Mission:

Principled Engineering for Global Software Services

User and Communities Empowering Tools

Research threads:

COMPUTER SYSTEMS
KNOWLEDGE-BASED SYSTEMS
MULTIMODAL SYSTEMS
SOFTWARE SYSTEMS

NOVA LINCS

Automated software Engineering team

Research threads:

Model-Driven Software Development

Requirements Engineering

Domain-Specific (Modelling) Languages

Experimental Software Engineering

NOVA LINCS

Automated software Engineering team






Ana Moreira
Miguel Goulão
João Araújo
Miguel Monteiro
Ankica Barišić

TÉCNICO LISBOA

CERN

DESY.

LIP

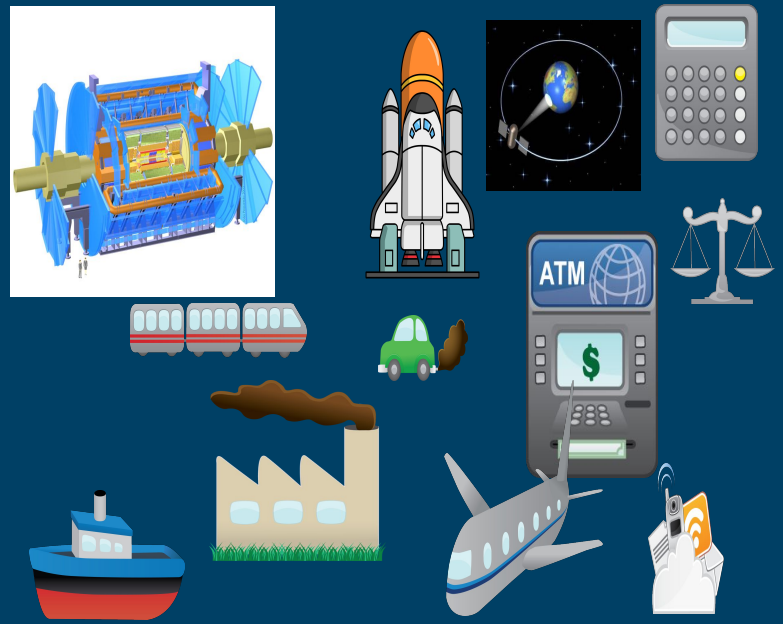
UNIVERSITY OF MANNHEIM

- Model-Driven Software Development
- Domain-Specific (Modelling) Languages
- Software Languages Engineering

- Multi-Paradigm Modelling for Cyber-Physical Systems

Software (Modelling) Languages Engineering

Increasing complexity of (Software) Systems



Software Crisis

The term "**software crisis**" was coined by F. L. Bauer at the first NATO Software Engineering Conference in 1968

The major cause of the software crisis is that the machines have become several orders of magnitude more powerful!

To put it quite bluntly: as long as there were no machines, programming was no problem at all;

when we had a few weak computers, programming became a mild problem, and now we have gigantic computers, programming has become an equally gigantic problem.

– Edsger Dijkstra, The Humble Programmer (EWD340), Communications of the ACM

“No Silver Bullet – Essence and Accidents of Software Engineering”, Fred Brooks, 1986

"there is no single development (...) which by itself promises even one order of magnitude [tenfold] improvement within a decade in productivity, in reliability, in simplicity."

"we cannot expect ever to see two-fold gains every two years"

“No Silver Bullet – Essence and Accidents of Software Engineering”, Fred Brooks, 1986

Accidental Complexity

non essential to the problem solved.

Essential complexity

is inherent and unavoidable

Need for Industrial Revolution in SE

Interchangeable parts, introduced by John Hall

Assembly lines, introduced by Ransom Olds

Automated Assembly lines, (first industrial robot installed in 1961 by GM)

Craftsmen workshops

Factories

1826

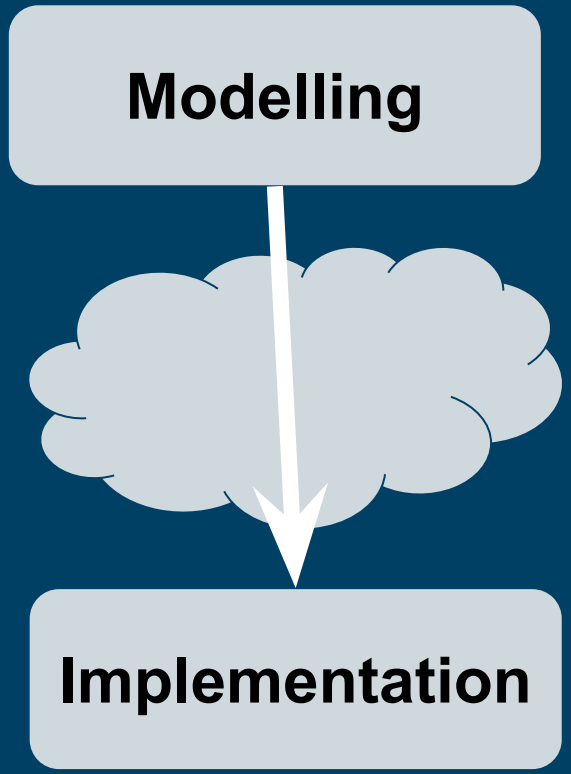
1901

1980

Software Factories, Jack Greenfield et.al, Wiley,2004
Generative Programming, Czarnecki. et al, Wesley 2000

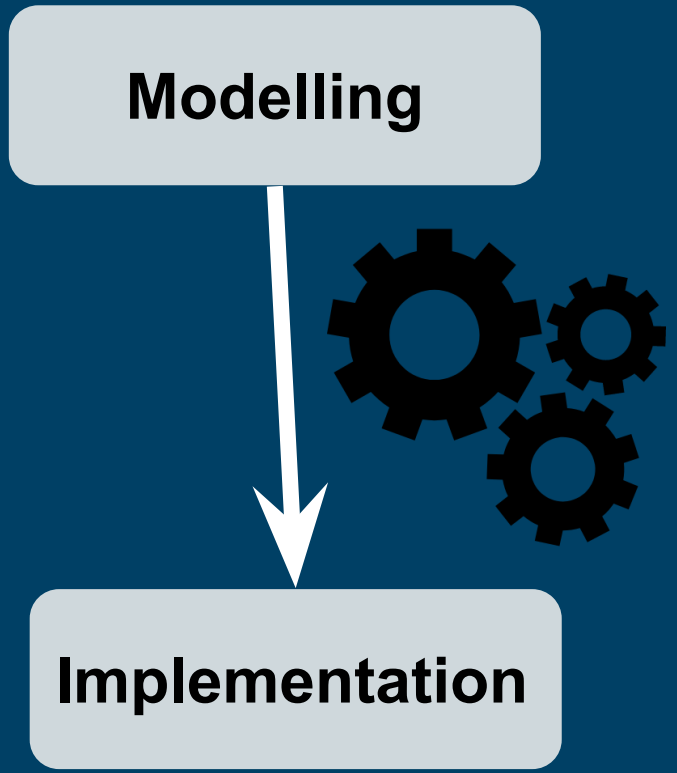
The Modelling Gap

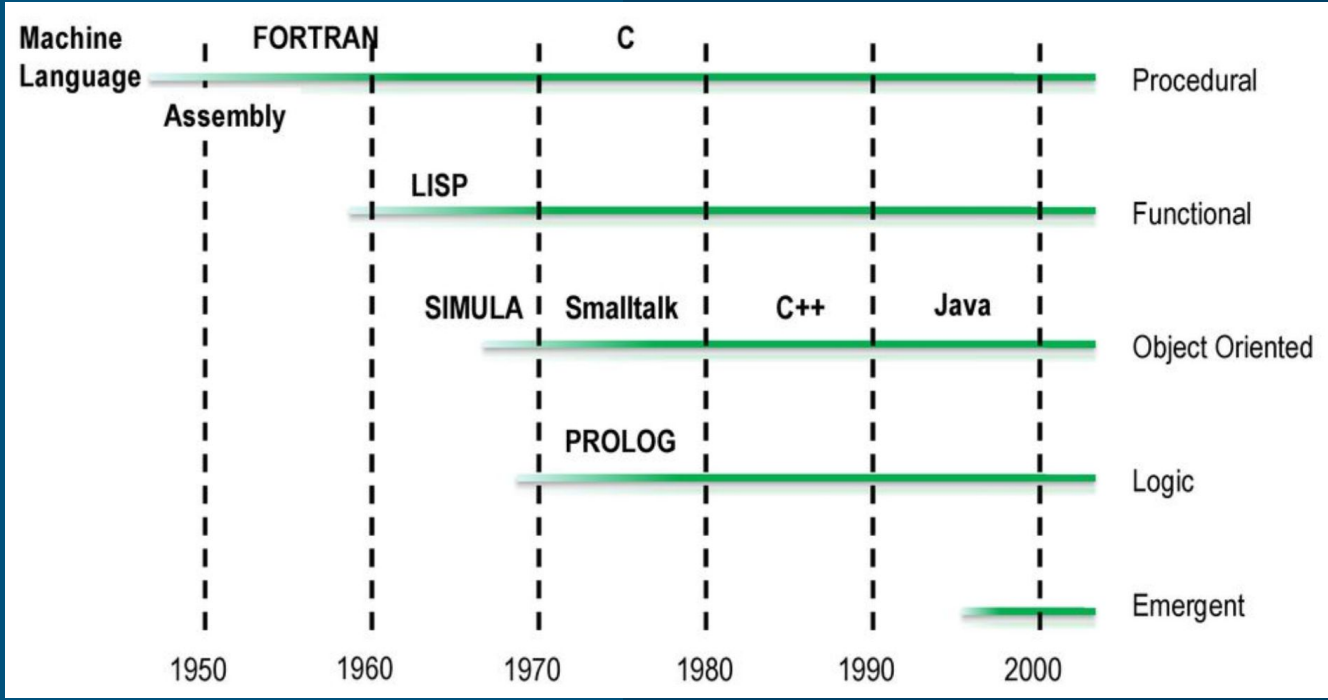
The tradition...



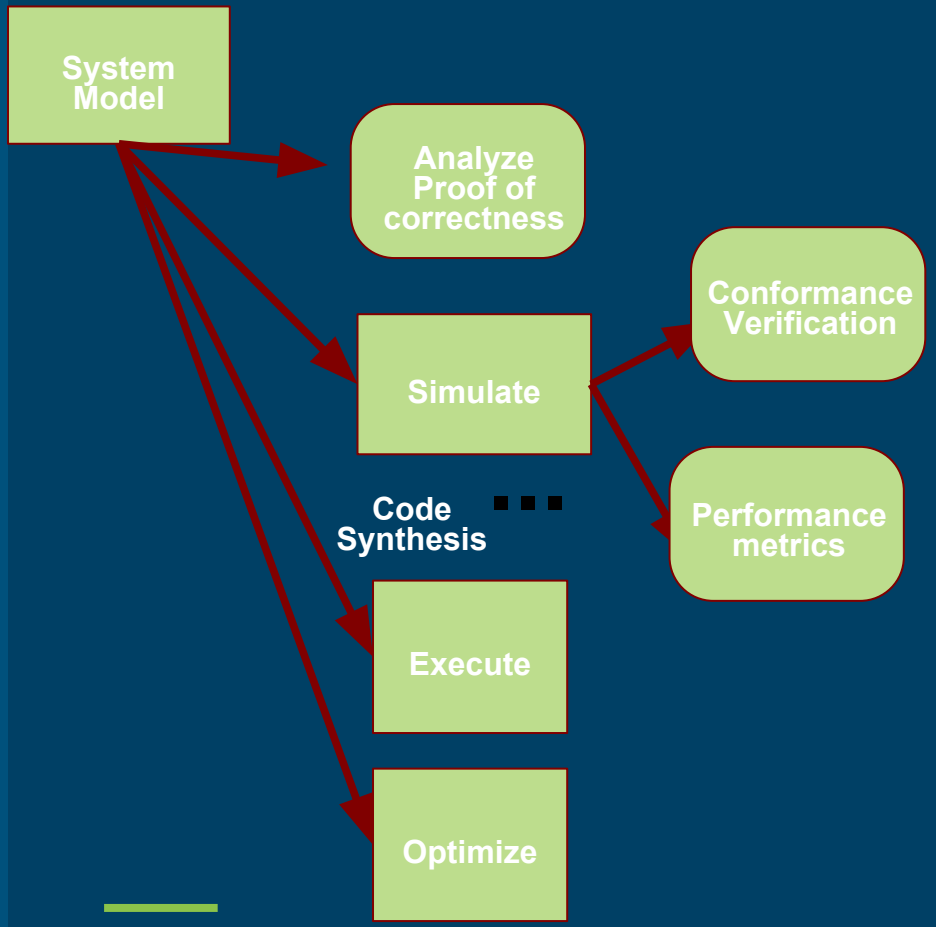
Modelling Gap

We need automation




Machine Language
FORTRAN
C
Assembly
LISP
SIMULA
Smalltalk
C++
Java
PROLOG
Procedural
Functional
Object Oriented
Logic
Emergent
1950
1960
1970
1980
1990
2000

Model-Driven Roadmap



What is a Model?

What is a Model?

"A model is an abstraction of a (real or language-based) system allowing predictions or inferences to be made."

Thomas Khune, Matters of (Meta-) Modeling, SoSYM, 2006

Models+Transformations=Programs

Stachowiak

Properties of a Model

mapping feature
based on an original.

reduction feature
selection of an original’s properties.

pragmatic feature
A model needs to be usable in place of an original with respect to some purpose.

“no abstraction” → “no model”

Sylvie and Bruno Concluded, by Lewis Carroll, 1893

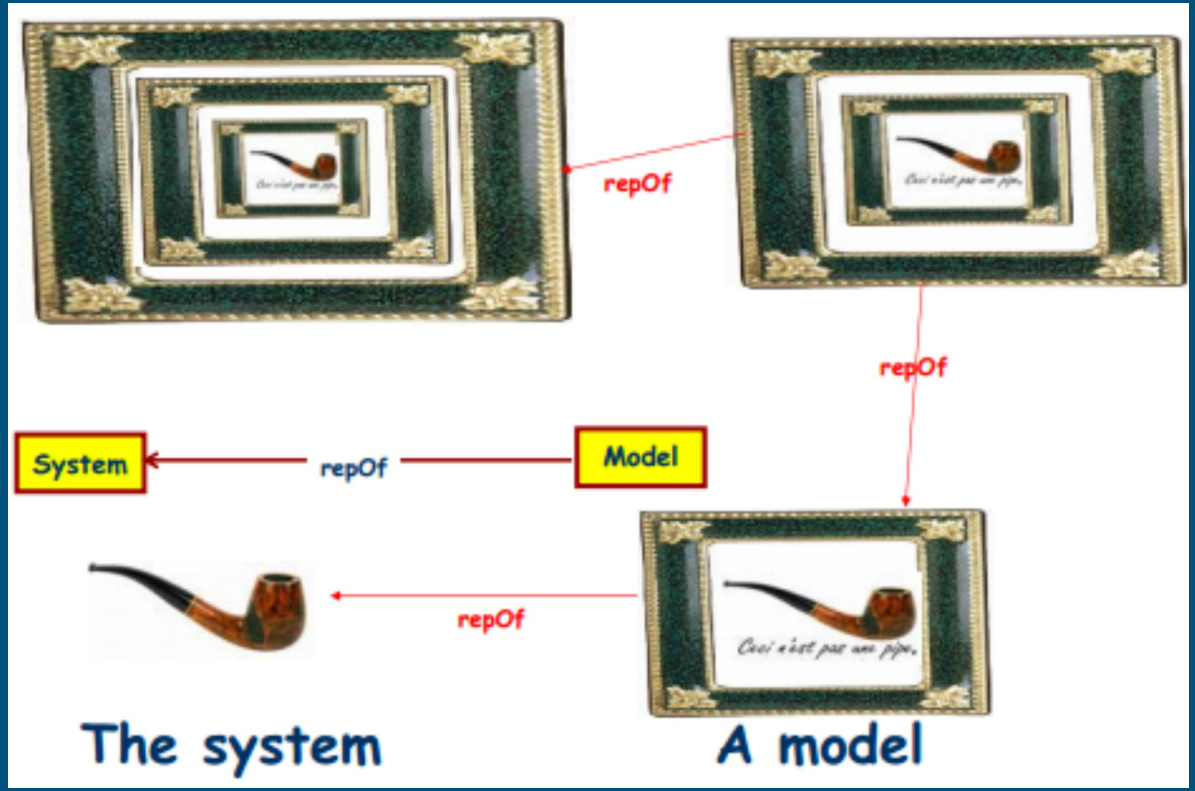
“no abstraction” → “no model”

- A copy is not a model
- Any representation of a real world subject automatically implies reduction and thus can be granted model status.

René Magritte

1928–29

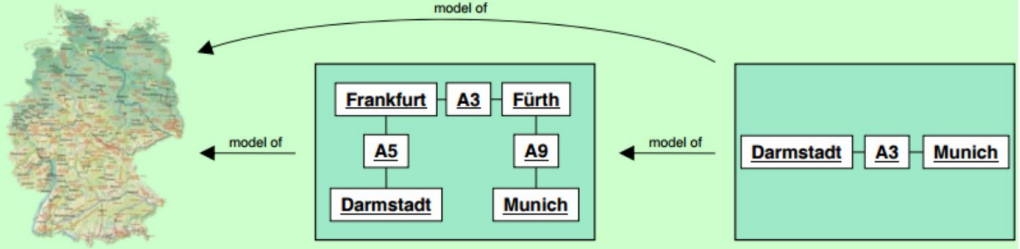
Difference between a model and System



Jean-Marie Favre:
Megamodelling and Etymology. Transformation Techniques in Software Engineering 2005

Kinds of Model Roles -Token models

Elements of a token model capture singular (as opposed to universal) aspects of the original's elements, i.e., they model individual properties of the elements in the system.

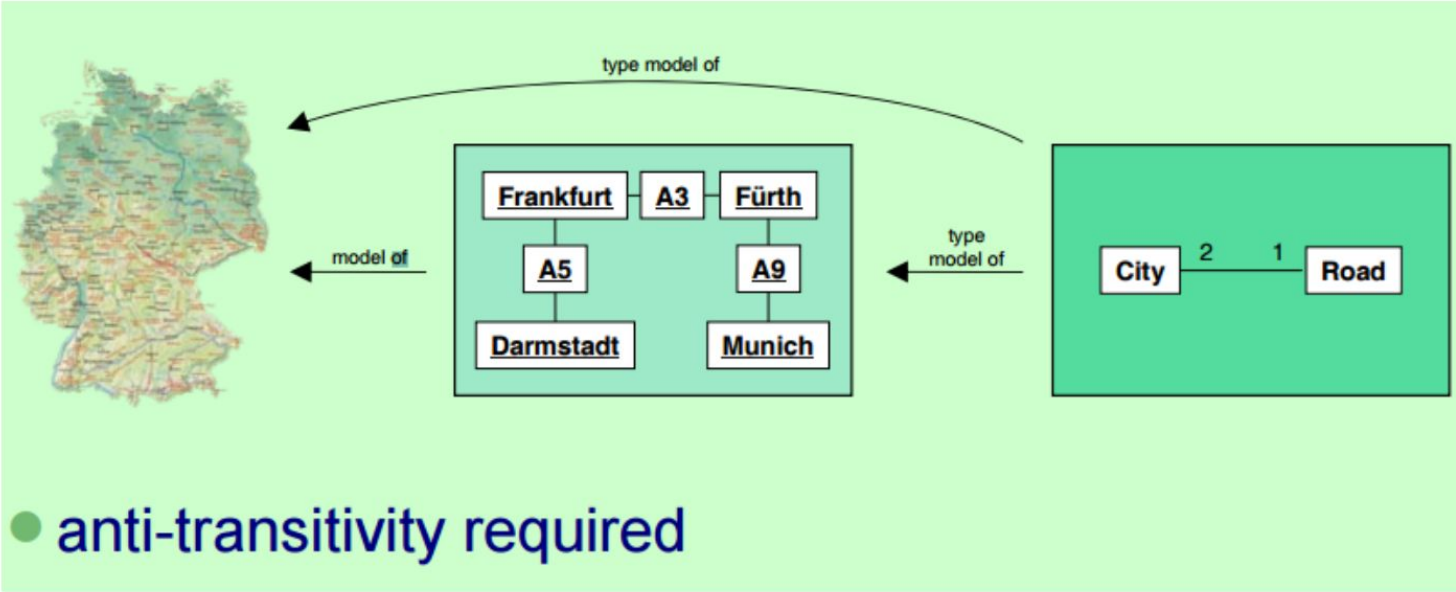


- anti-transitivity required

one-to-one representation of elements in the (relevant part of the) system.

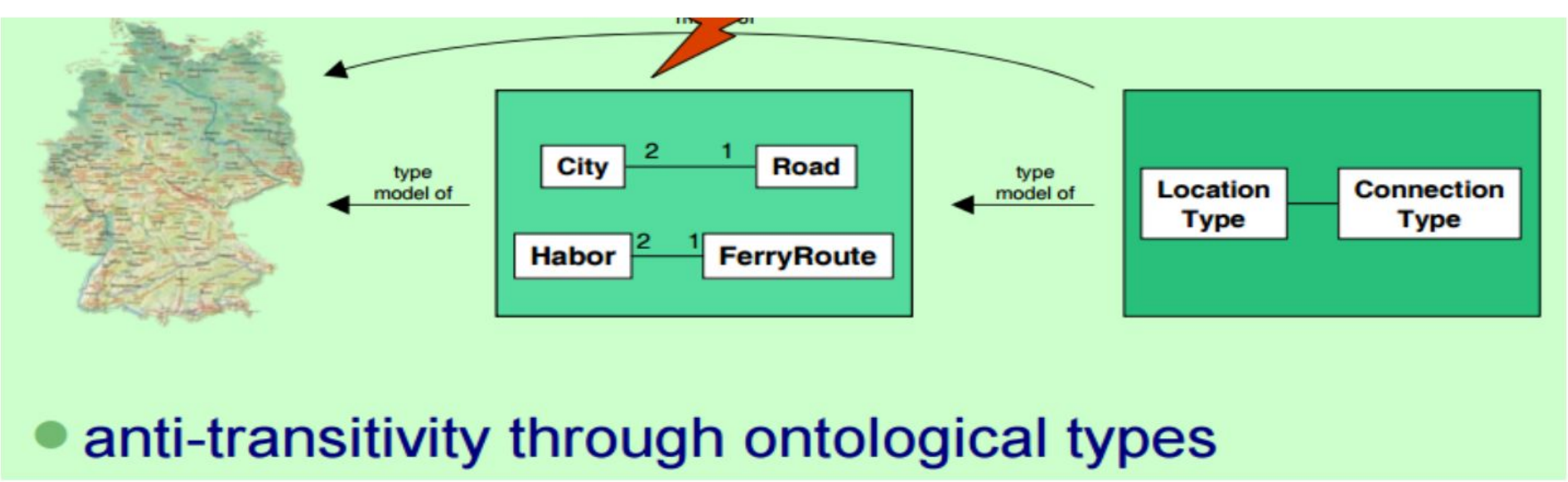
Thomas Kühne:
Matters of (Meta-)Modeling. Software and System Modeling 5(4): 369-385 (2006)

Kinds of Model Roles -Type models



Thomas Kühne:
Matters of (Meta-)Modeling. Software and System Modeling 5(4): 369-385 (2006)

Kinds of Model Roles -Type models



- anti-transitivity through ontological types

Thomas Kühne:
Matters of (Meta-)Modeling. Software and System Modeling 5(4): 369-385 (2006)

Kinds of Model Roles -Type models

type model of

City 2 1 Road

type model of

UML Class — UML Association

- anti-transitivity through linguistic types

Thomas Kühne:
Matters of (Meta-)Modeling. Software and System Modeling 5(4): 369-385 (2006)

A metamodel is...

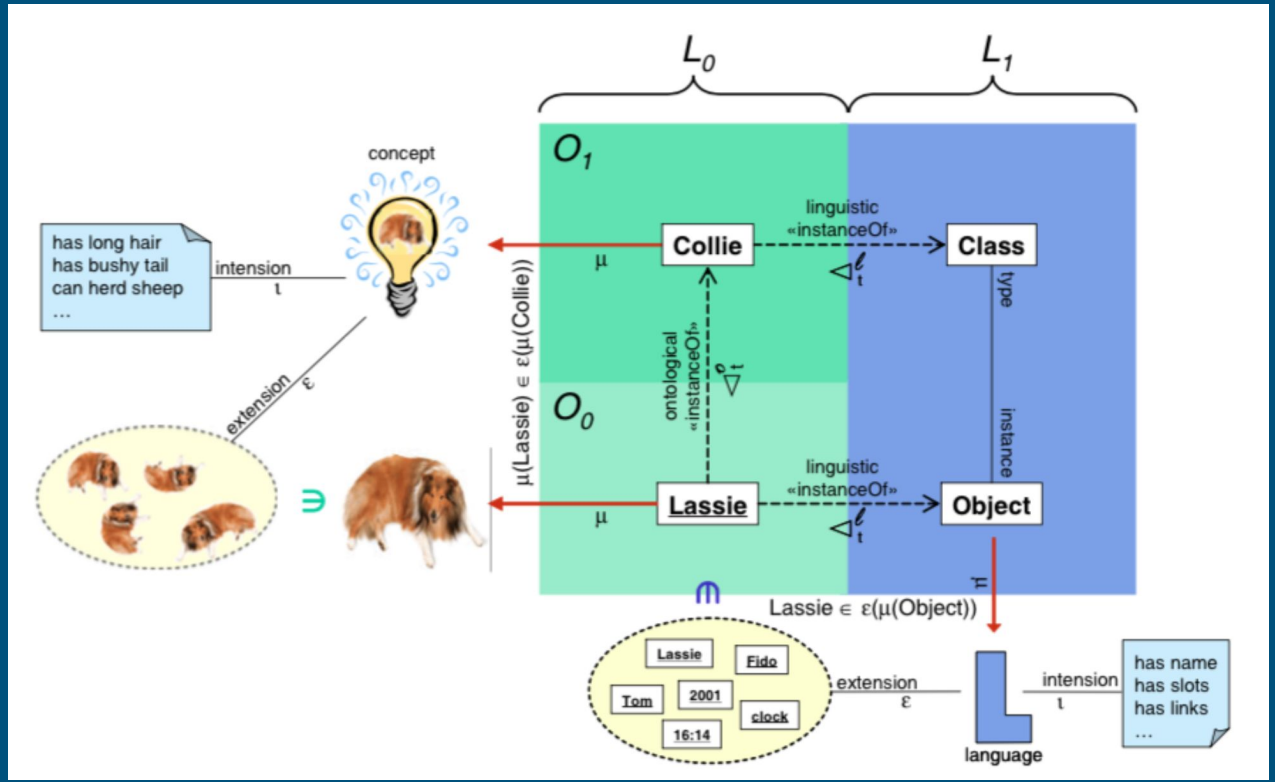
A metamodel is a model of models

- **A model is an instance of a metamodel**
- **implies that a metamodel is a model of another model.**

Note that:

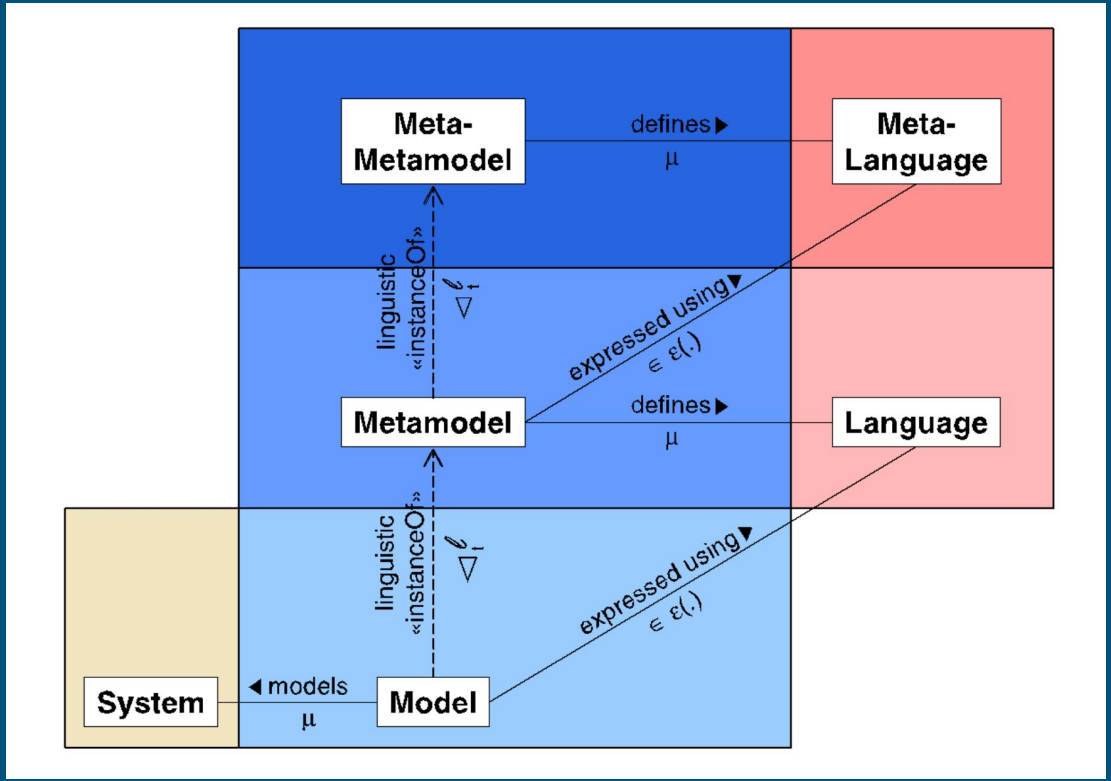
- **a token model of a token model is not a metamodel.**
- **in order to create a metamodel we need the same non-transitive relationship.**

Ontological vs. Language



Thomas Kühne:
Matters of (Meta-)Modeling. Software and System Modeling 5(4): 369-385 (2006)

Language mechanism stack



Thomas Kühne:
Matters of (Meta-)Modeling. Software and System Modeling 5(4): 369-385 (2006)

Models

Can be represented as diagrams or as text!

Software Language Engineering

SLE is the application of a systematic, disciplined and quantifiable approach to the development, usage, and maintenance of software languages.

Software Language Engineer

One task of a language engineer is to develop languages that make the job of creating software easier.

Another task is to create a language that will support the language end-user (also known as domain expert) efficiently and effectively.

General purpose

General Purpose

Specific Purpose

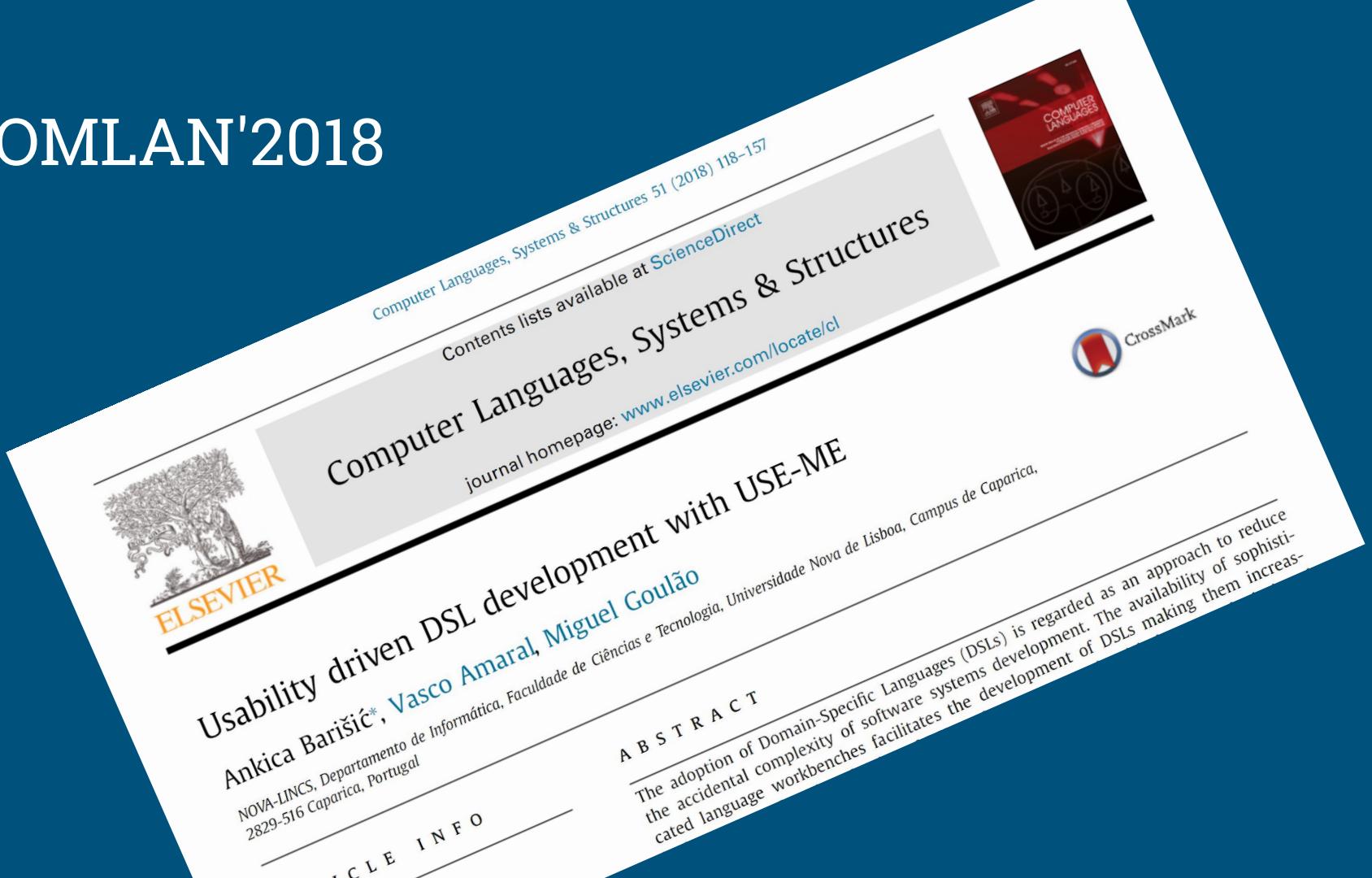
- **match the user's mental model** of the problem domain
- **maximally constrain** the user (to the problem at hand)
  $\Rightarrow$ easier to learn
  $\Rightarrow$ avoid errors
- **separate** domain-expert's work
  from analysis/transformation expert's work

Anecdotal evidence of 5 to 10 times speedup

Steven Kelly and Juha-Pekka Tolvanen. Domain-Specific Modeling: Enabling Full Code Generation. Wiley, 2008.

Laurent Safa. The practice of deploying DSM, report from a Japanese appliance maker trenches. In Proceedings of the 6th OOPSLA Workshop on Domain-Specific Modeling (DSM'06), pp. 185-196, 2006.

COMLAN'2018



Computer Languages, Systems & Structures 51 (2018) 118–157

Contents lists available at ScienceDirect

Computer Languages, Systems & Structures

journal homepage: www.elsevier.com/locate/cl

Usability driven DSL development with USE-ME

Ankica Barišić*, Vasco Amaral, Miguel Goulão

NOVA-LINCS, Departamento de Informática, Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa, Campus de Caparica, 2829-516 Caparica, Portugal

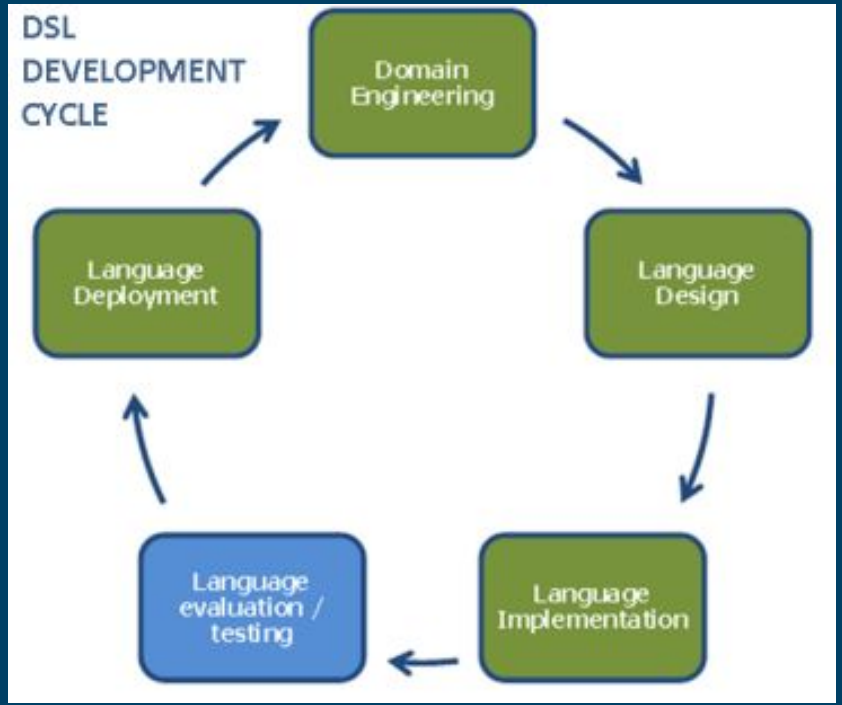
ARTICLE INFO

ABSTRACT

The adoption of Domain-Specific Languages (DSLs) is regarded as an approach to reduce the accidental complexity of software systems development. The availability of sophisticated language workbenches facilitates the development of DSLs making them increas-

SLE Process

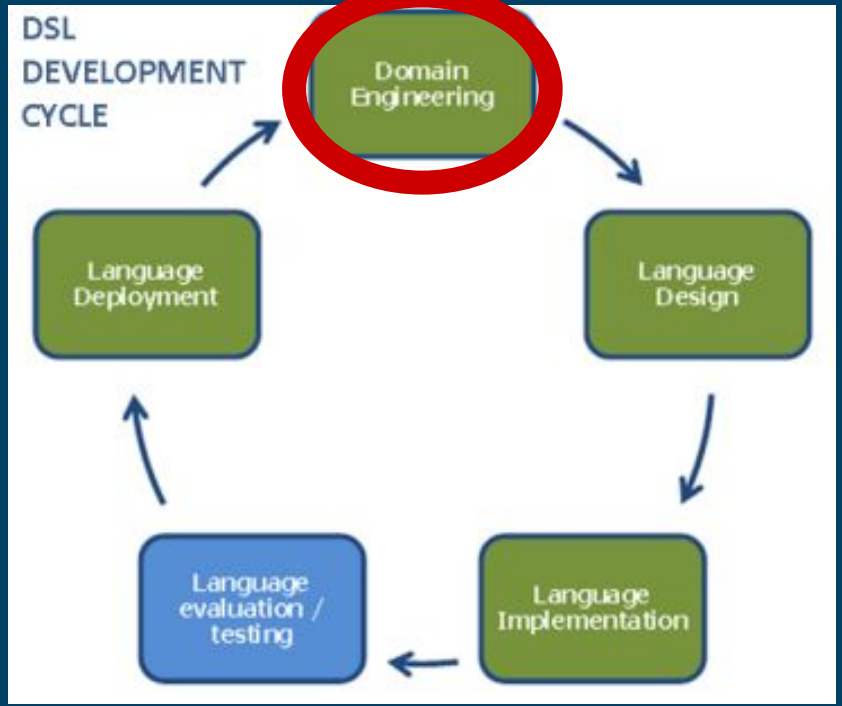
(as systematic)



Ankica Barisic, Vasco Amaral, Miguel Goulão:
Usability driven DSL development with USE-ME. Computer Languages, Systems & Structures 51: 118-157 (2018)

SLE Process

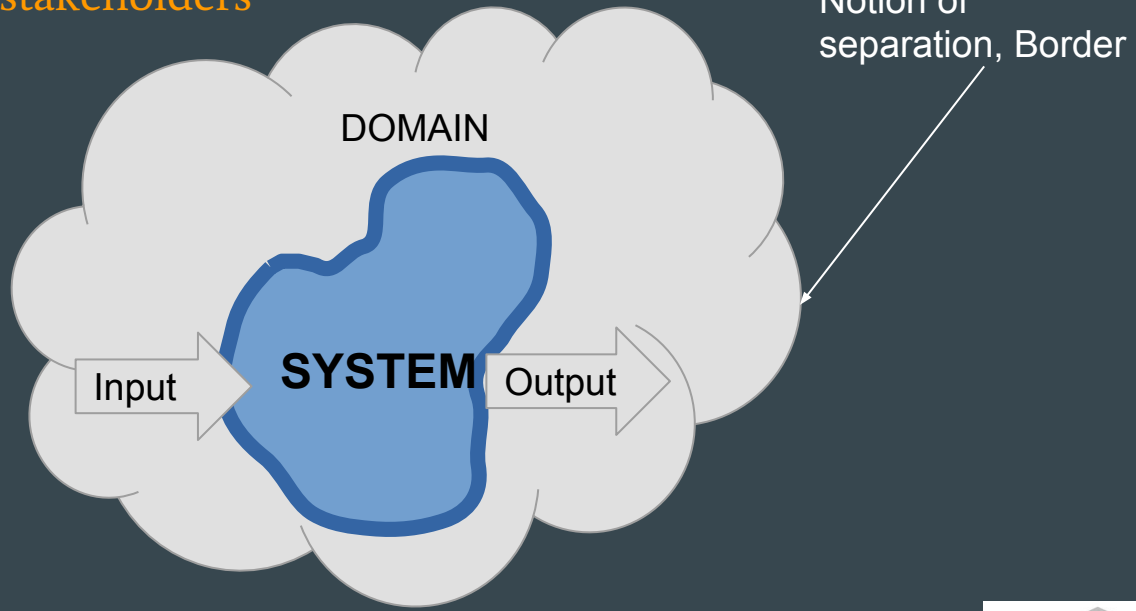
Domain Engineering



Ankica Barisic, Vasco Amaral, Miguel Goulão:
Usability driven DSL development with USE-ME. Computer Languages, Systems & Structures 51: 118-157 (2018)

What is a Domain? In terms of a System

- Area of knowledge or activity
- Context to a set of problems to be solved by a system
- Relevant to some stakeholders'
- Embraces
  - Concepts
  - Processes
  - Terminology
  - Specific rules



Dissecting Domain - Problem Domain vs. Solution Domain

Traffic-Lights Systems

Dissecting a Domain - Problem Domain vs. Solution Domain

Traffic Lights

Change
Traffic
...
Pedestrians
bicycles
Green Light

PROBLEM DOMAIN

Switch-On
Timer
Yellow Light
Priority
Red Light

Concepts

PLC
Java
State-Machine

SOLUTION DOMAIN

TCP/IP
controller
C++

Set of Systems

Dissecting a Domain - Problem Domain vs. Solution Domain

Traffic Lights

Change
Traffic
...
Pedestrians
bicycles
Green Light

PROBLEM DOMAIN

Switch-On
Timer
Yellow Light
Priority
Red Light

What?

PLC
Java
State-Machine

SOLUTION DOMAIN

TCP/IP
controller
C++

How?

What is a Domain model?

Different research areas, different interpretations:

OO - is the UML class model

IS - is the ERD model

AI - is the Domain Ontology

...

Domain Model

"Is an **explicit representation** of

common and the variable properties of the systems in a domain,

counterexamples (i.e., systems outside the domain),

the semantics of the **properties** and domain **concepts**,

and the **dependencies** between the variable properties."

[Czarnecki et. al]

Domain Model

Domain Definition - Scope of the domain, stakeholders involved. Examples and counterexamples of what is and not inside the domain.

Domain Lexicon - lists domain vocabulary

Concept Models - Describes concepts in a domain in some appropriate modeling formalism. Comprehends:

- Entities
- Relationships between entities
- Domain Rules

Feature Models - Define a set of reusable and reconfigurable requirements for specifying a system in a domain.

Processes - Define the business and design processes

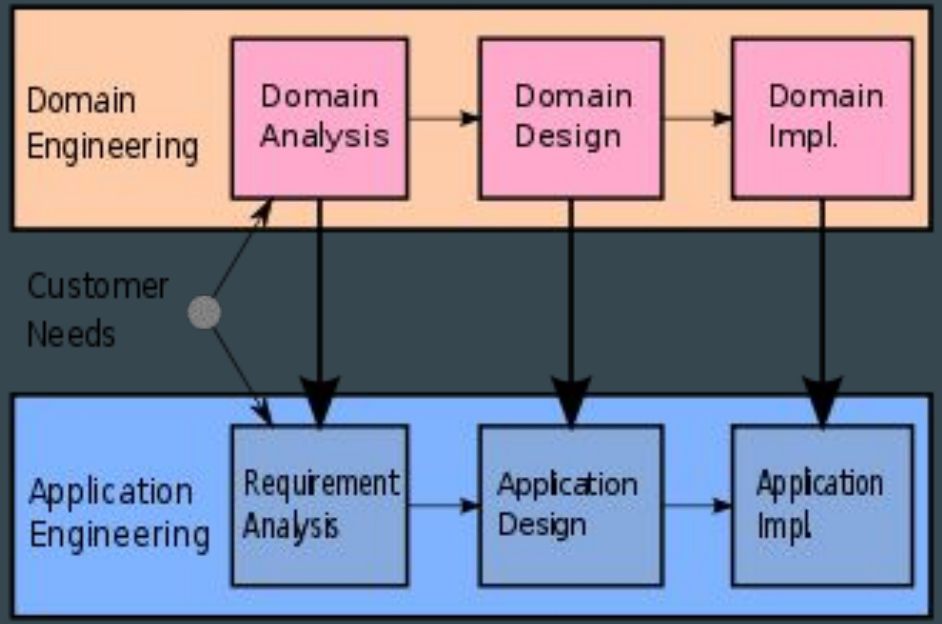
Domain Model - Input information

- Artifacts (such as design documents, requirement documents and user manuals)
- Standards
- Applications
- Stakeholders (users, clients, developers,...)

Domain Engineering

Focus:

- Engineering Reusable Software
- Knowledge Management

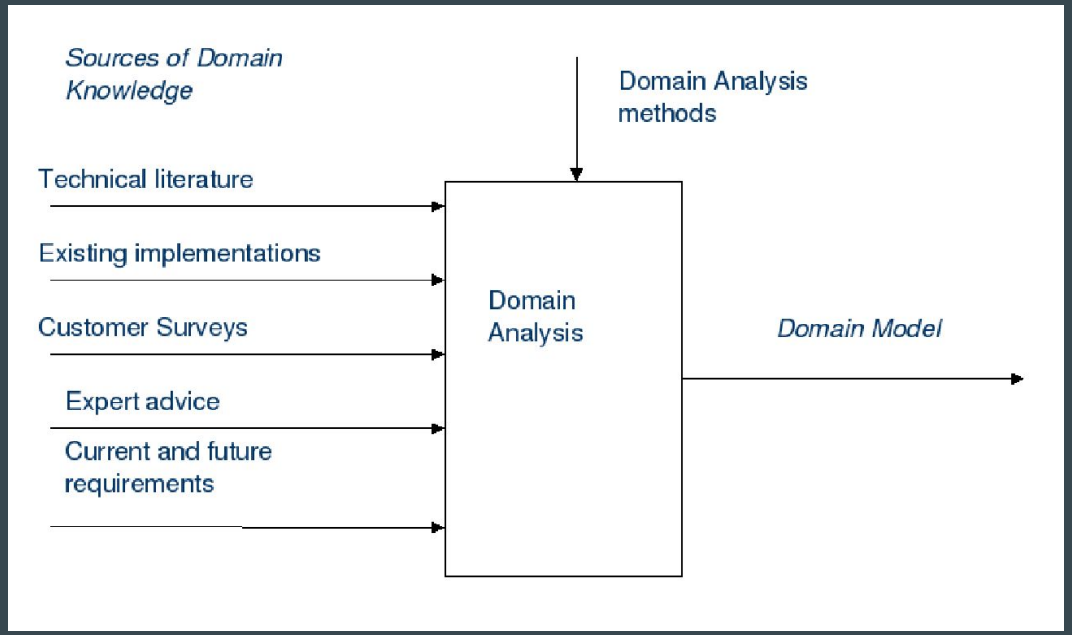


Domain Analysis

Information used in developing software systems is identified, captured and organized for reuse:

Selects and defines Domain focus

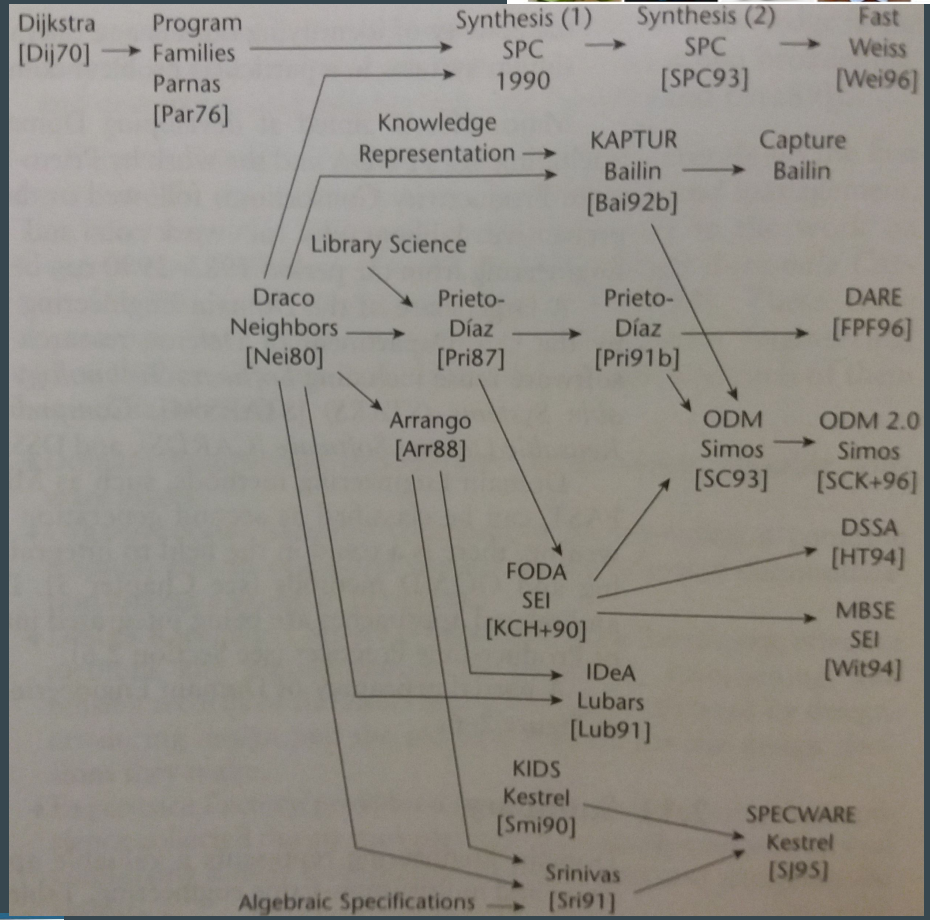
Produces coherent Domain Model

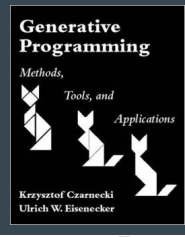


Adapted from, Scott Thibault Phd Thesis, 1998

Domain Engineering Methods

- Feature Oriented Domain Analysis (FODA)
- Organization Domain Analysis (ODM)
- Draco
- Capture
- Domain Analysis and Reuse Environment (DARE)
- Domain-Specific Software Architecture (DSSA) Approach
- Algebraic Approach
- Family-Oriented Abstraction, Specification, and Translation
- PuLSE
- SYNTHESIS
- Defense Information Systems Agency's Domain Analysis and Design Process
- Joint Integrated Avionics Working Group Object Oriented Domain Analysis Method (JODA)
- ...


Dijkstra [Dij70]
Program Families Parnas [Par76]
Synthesis (1) SPC 1990
Synthesis (2) SPC [SPC93]
Fast Weiss [Wei96]
Knowledge Representation
KAPTUR Bailin [Bai92b]
Capture Bailin
Library Science
Draco Neighbors [Nei80]
Prieto-Díaz [Pri87]
Prieto-Díaz [Pri91b]
DARE [FPF96]
Arrango [Arr88]
ODM Simos [SC93]
ODM 2.0 Simos [SCK+96]
DSSA [HT94]
FODA SEI [KCH+90]
MBSE SEI [Wit94]
IDeA Lubars [Lub91]
KIDS Kestrel [Smi90]
SPECWARE Kestrel [SJ95]
Srinivas [Sri91]
Algebraic Specifications


Generative Programming
Methods,
Tools, and
Applications
Krzysztof Czarnecki
Ulrich W. Eisenecker

FODA

Developed by SEI in the 80's

Domain Analysis component of part of Model-Based Software Engineering (MBSE)

Part of the Product Lines Practice (SPL)

FODA - Phase 1 - Domain Analysis

Define the **boundaries of the domain to be analyzed (scoping)**

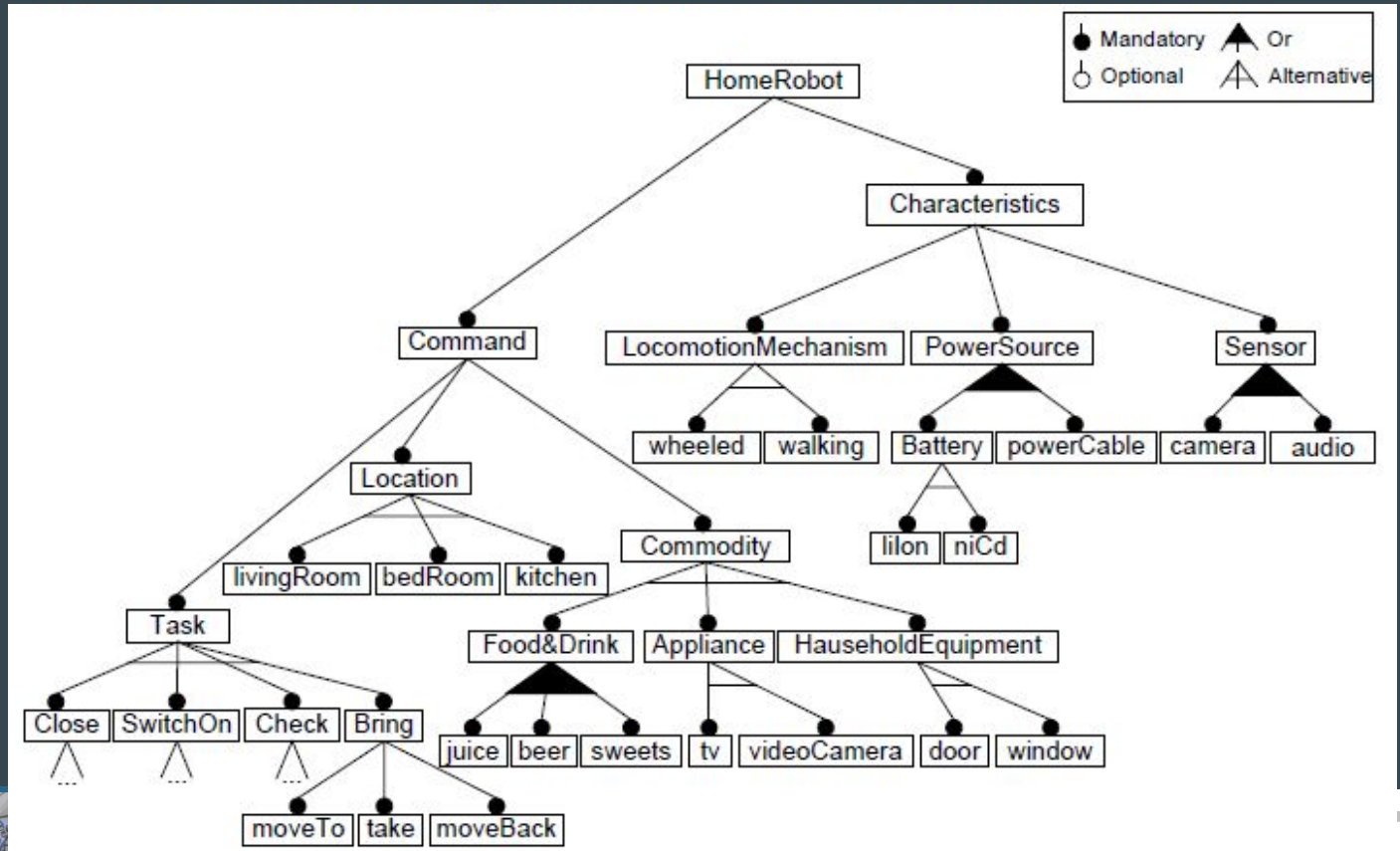
- Scope a domain likely to yield useful domain products
- Availability of domain expertise and project constraints
- Relationship between the domain of focus and other domains or entities

FODA - Phase 2 - Domain Modeling

Produce a domain Model identifying main commonalities and variabilities between the applications in the domain

- Information Analysis
  - Derive the Information Model, with domain entities and relationships. Can be done using OO modeling, ER, or ...
- Feature Analysis
  - Capture a customer's or end-user's understanding of capabilities of applications in a domain. Design the feature model.
- Operational Analysis
  - Identifies the commonalities and differences between control and data flows in the application domain. Captures behavior relating the objects in the Information Model and the features in the feature model.

Feature Model



Part III - How do we formalize the Domain Model?

Textual description

Example Traffic-Light System - from wikipedia

“Traffic lights, also known as traffic signals, traffic lamps, traffic semaphore, signal lights, stop lights, (in South African English) robots and (in technical parlance) traffic control signals,[1] are signalling devices positTraffic lights alternate the right of way accorded to users by displaying lights of a standard color (red, amber (yellow), and green) following a universal color code. In the typical sequence of color phases:

The green light allows traffic to proceed in the direction denoted, if it is safe to do so and there is room on the other side of the intersection.
The amber (yellow) light warns that the signal is about to change to red. In a number of countries – among them the United Kingdom – a phase during which red and yellow are displayed together indicates that the signal is about to change to green.[5] Actions required by drivers on a yellow light vary, with some jurisdictions requiring drivers to stop if it is safe to do so, and others allowing drivers to go through the intersection if safe to do so.ioned at road intersections, pedestrian crossings, and other locations to control flows of traffic.” ...

Domain Model - Textual description - from wikipedia

Traffic lights alternate the right of way accorded to users by displaying lights of a standard color (red, amber (yellow), and green) following a universal color code. In the typical sequence of color phases:

The green light allows traffic to proceed in the direction denoted, if it is safe to do so and there is room on the other side of the intersection.
The amber (yellow) light warns that the signal is about to change to red. In a number of countries – among them the United Kingdom – a phase during which red and yellow are displayed together indicates that the signal is about to change to green. Actions required by drivers on a yellow light vary, with some jurisdictions requiring drivers to stop if it is safe to do so, and others allowing drivers to go through the intersection if safe to do so.
A flashing Amber indication is a warning signal. In the United Kingdom, a flashing amber light is used only at pelican crossings, in place of the combined red–amber signal, and indicates that drivers may pass if no pedestrians are on the crossing.
The red signal prohibits any traffic from proceeding.
A flashing red indication is treated as a stop sign.
In some countries traffic signals will go into a flashing mode if the controller detects a problem, such as a program that tries to display green lights to conflicting traffic. The signal may display flashing yellow to the main road and flashing red to the side road, or flashing red in all directions. Flashing operation can also be used during times of day when traffic is light, such as late at night

Entity-Relationship Diagrams (ERD)
+
Data Flow Diagrams (DFD)

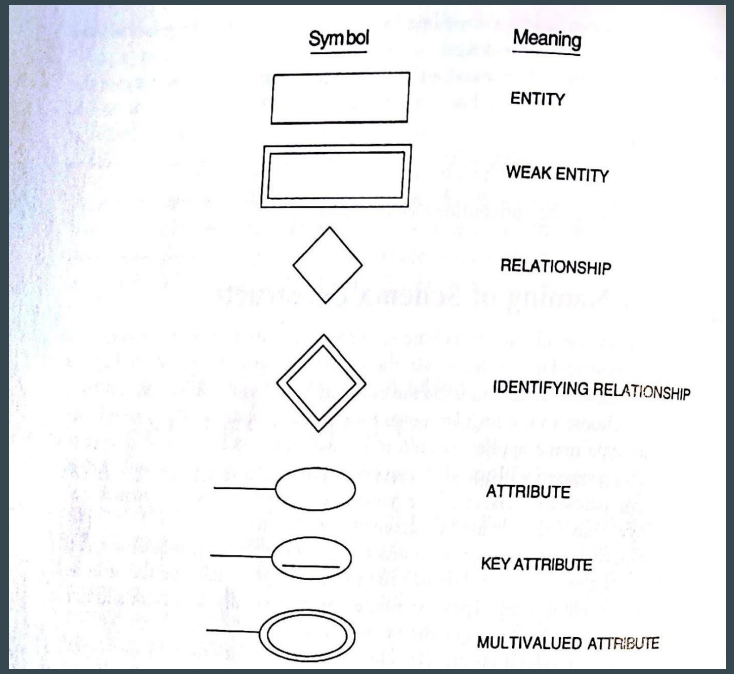
Vasco Amaral - FCT/UNL - NOVA LINCS

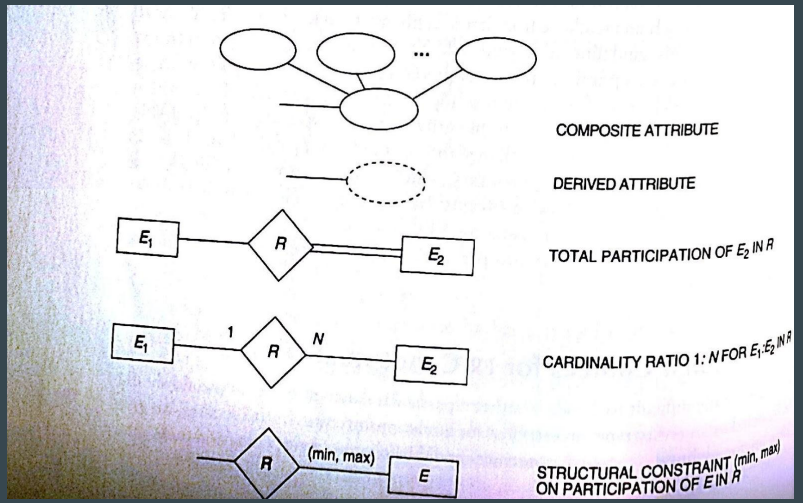
ERD - Developed by Peter Chen in the 70's. Help to describe inter-related things of interest in a specific domain of knowledge.

DFD (aka bubble charts) - Developed by Larry Constantine in the 70's . Help to visualize

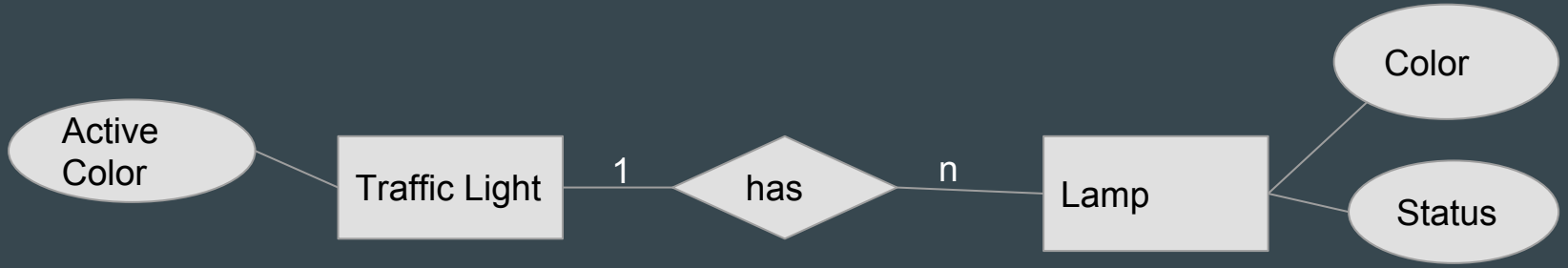
- how the system will operate,
- what the system will accomplish,
- and how the system will be implemented.

ERD

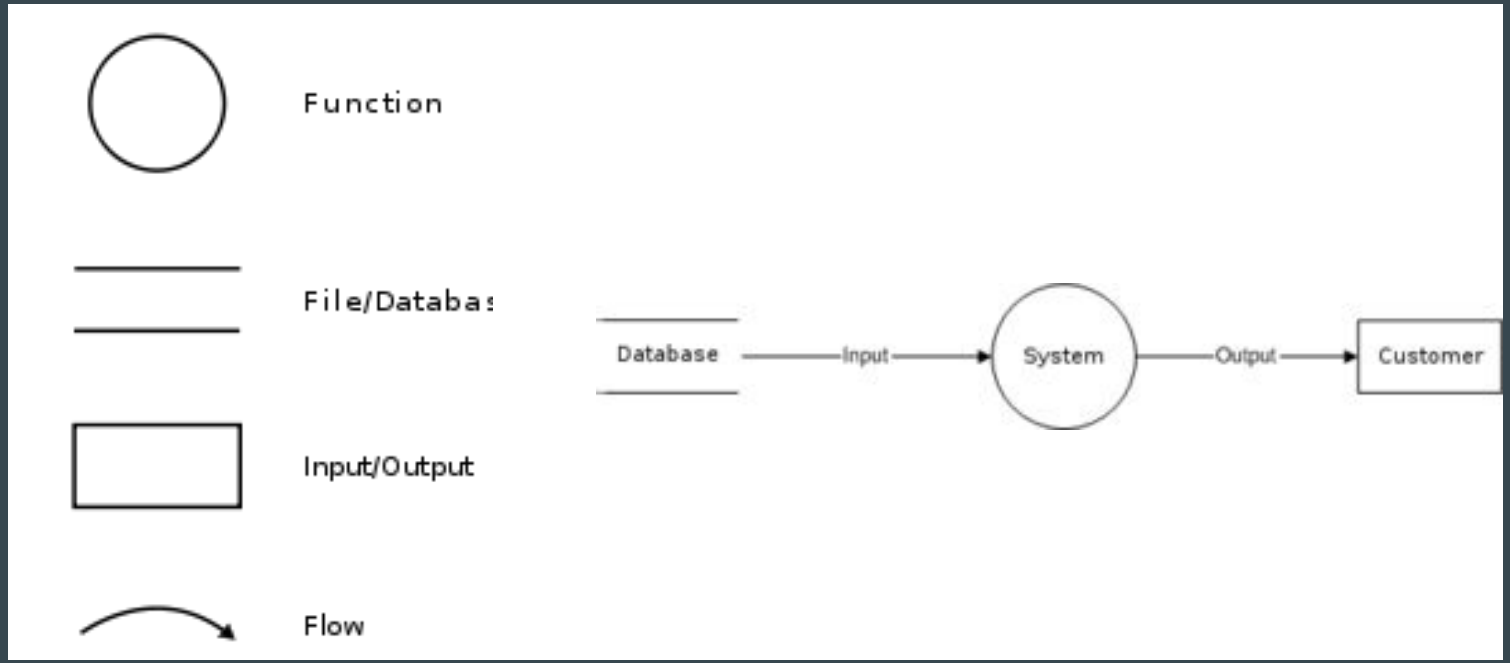

Symbol
Meaning
ENTITY
WEAK ENTITY
RELATIONSHIP
IDENTIFYING RELATIONSHIP
ATTRIBUTE
KEY ATTRIBUTE
MULTIVALUED ATTRIBUTE


COMPOSITE ATTRIBUTE
DERIVED ATTRIBUTE
E1
R
E2
TOTAL PARTICIPATION OF E2 IN R
1
N
CARDINALITY RATIO 1: N FOR E1:E2 IN R
(min, max)
E
STRUCTURAL CONSTRAINT (min, max) ON PARTICIPATION OF E IN R

Traffic Lights - ERD

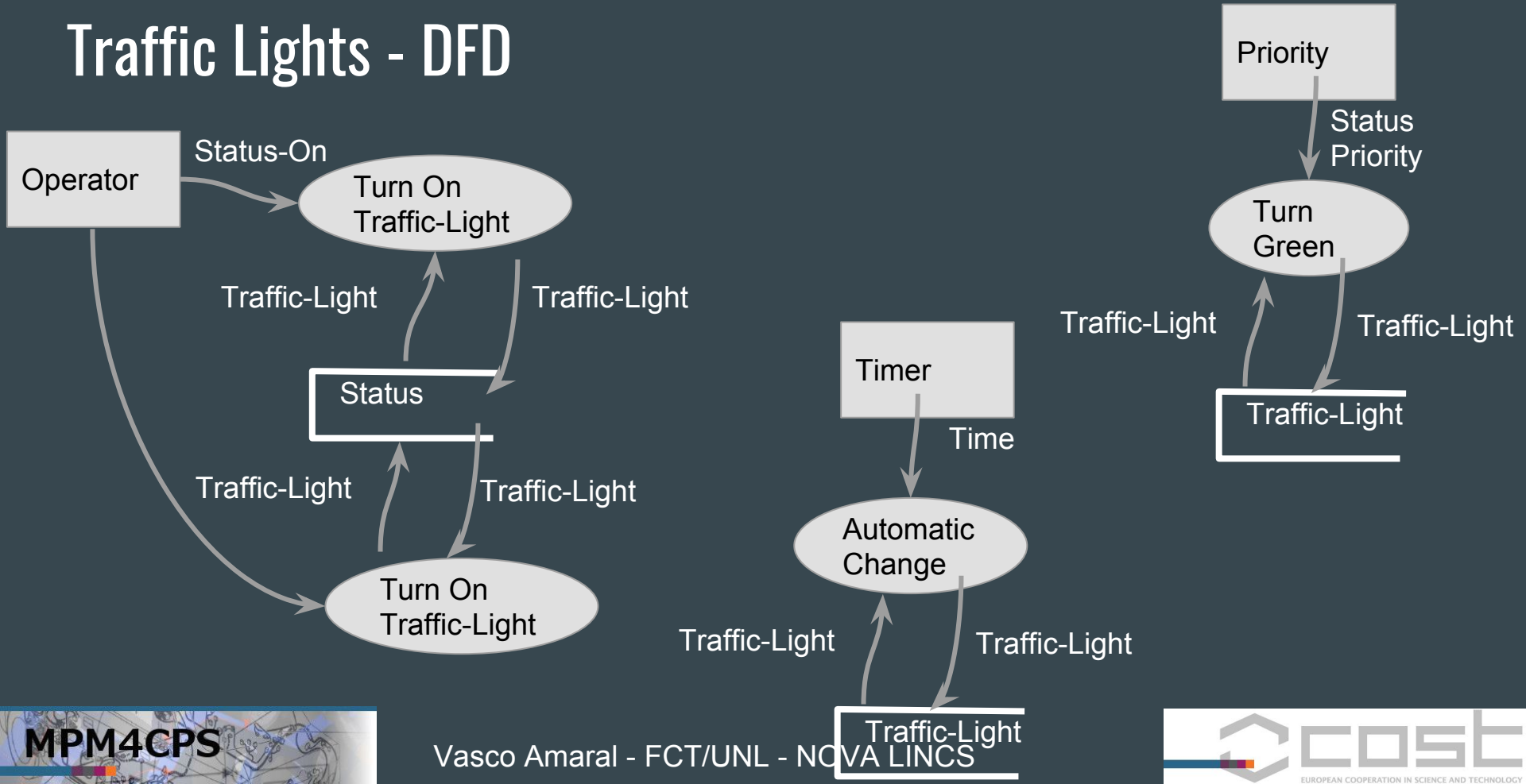


DFD



Taken from wikipedia

Traffic Lights - DFD



UML
Class Diagrams + Statecharts + OCL

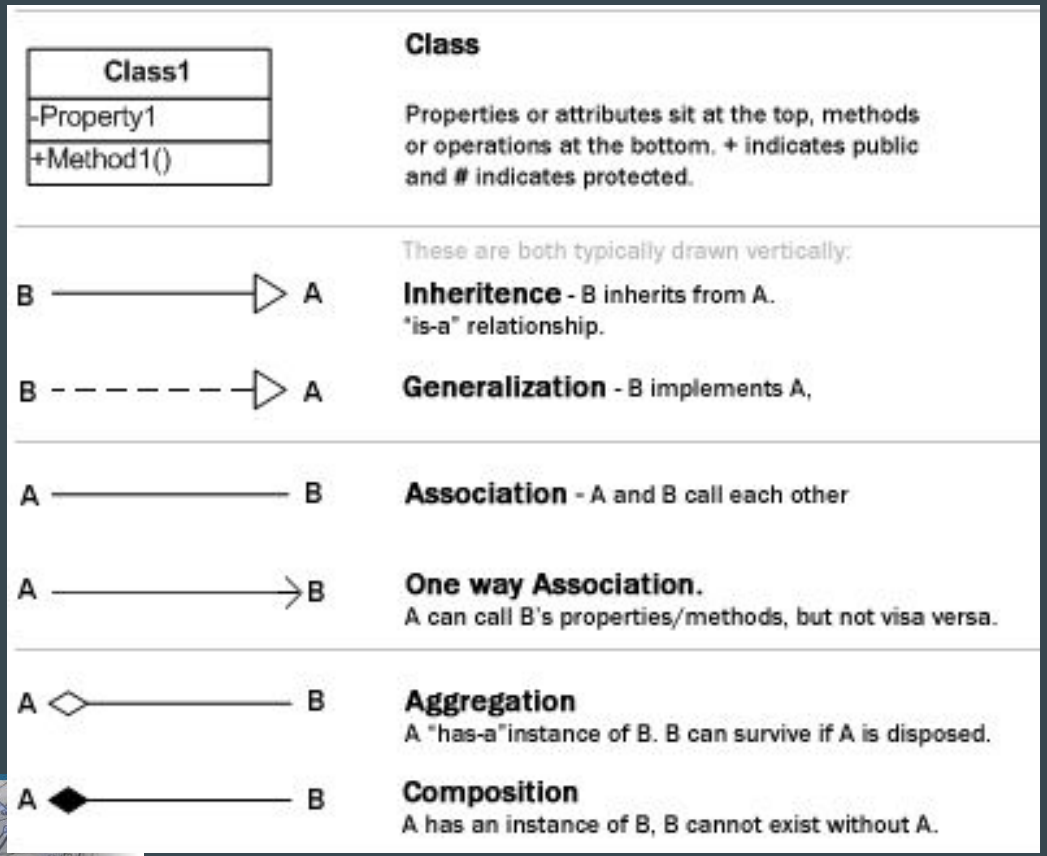
Vasco Amaral - FCT/UNL - NOVA LINCS

Class Diagrams - showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

Statecharts - Proposed by David Harel

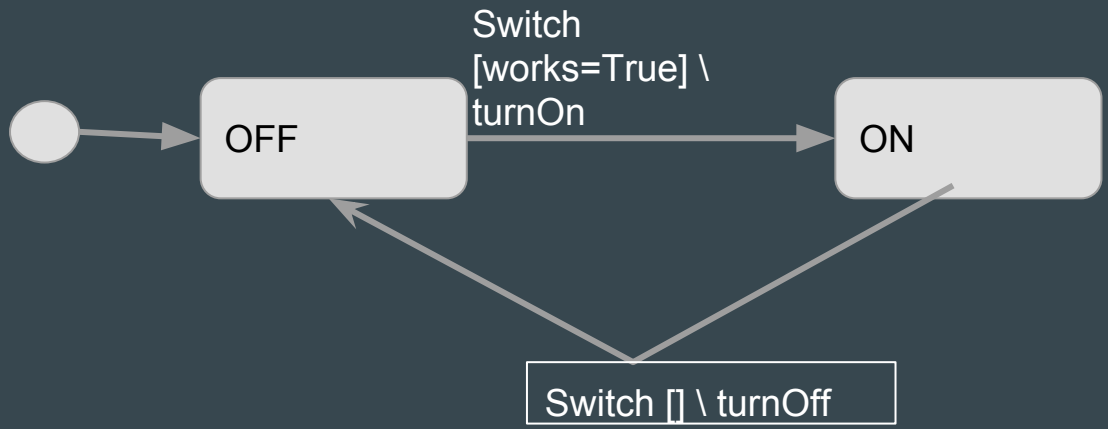
OCL - Proposed by Jos Warmer and Anneke Kleppe

Class diagrams

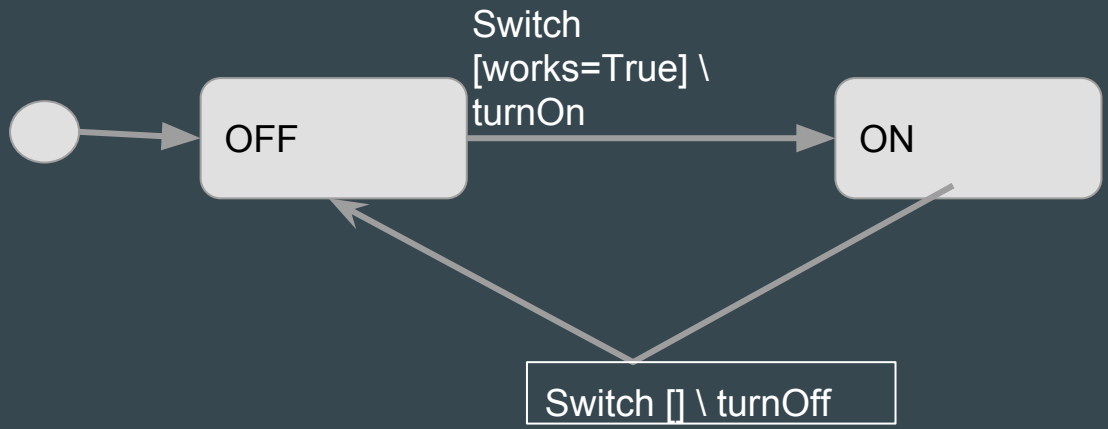


Class Diagrams

Statecharts



Statecharts



OCL Constraints

Context Traffic-Light

INV Only-One Color: self.Light (l| l.status = On)->count ()=1

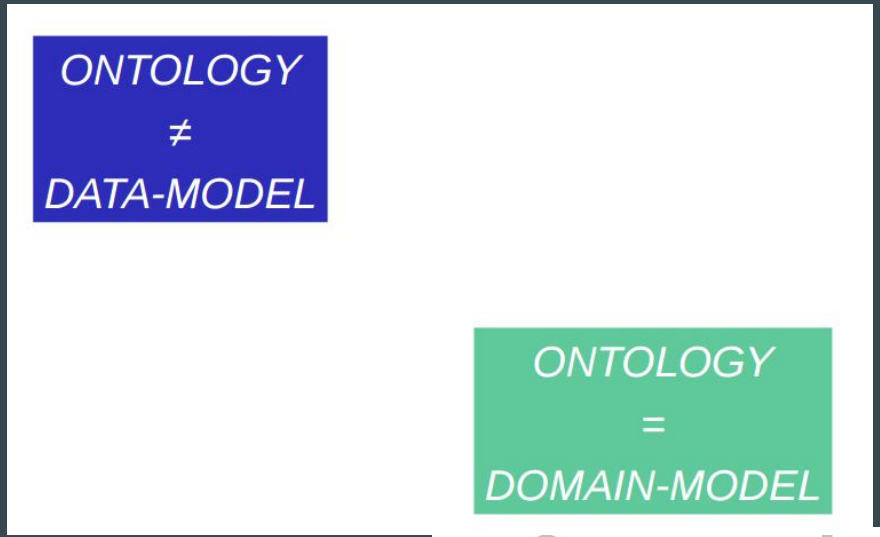
Ontologies

Vasco Amaral - FCT/UNL - NOVA LINCS

What is an Ontology?

It is a specification of a conceptualization in the context of knowledge description.

Ability to reason and make inferences.

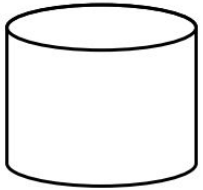


Several Description Language Families...

Syntax		Semantics	$\mathcal{AL}$	$\mathcal{ALU}$	$\mathcal{ALE}$	$\mathcal{ALN}$	$\mathcal{ALC}$
Concepts	$\top$	Δ^I	✓	✓	✓	✓	✓
	$\bot$	$\varnothing$	✓	✓	✓	✓	✓
	$\neg A$	$\Delta^I \setminus A^I$	✓	✓	✓	✓	✓
	$C \sqcap D$	$C^I \cap D^I$	✓	✓	✓	✓	✓
	$\exists R.\top$	$\{a \mid \exists b, (a,b) \in R^I\}$	✓	✓	✓	✓	✓
	$\forall R.C$	$\{a \mid \forall b, (a,b) \in R^I \rightarrow b \in C^I\}$	✓	✓	✓	✓	✓
	$\mathcal{U}: C \sqcup D$	$(C \sqcup D)^I = C^I \cup D^I$		✓			✓
	$\mathcal{E}: \exists R.C$	$(\exists R.C)^I = \{a \mid \exists b, (a,b) \in R^I \text{ and } b \in C^I\}$			✓		✓
	$\mathcal{N}: \geq nR$	$(\geq nR)^I = \{a \mid \sharp\{b \mid (a,b) \in R^I\} \geq n\}$				✓	
	$\leq nR$	$(\leq nR)^I = \{a \mid \sharp\{b \mid (a,b) \in R^I\} \leq n\}$				✓	
	$\mathcal{C}: \neg C$	$(\neg C)^I = \Delta^I \setminus C^I$					✓
TBox	$C \sqsubseteq D$	$C^I \subseteq D^I$	✓	✓	✓	✓	✓
ABox	$a: C$	$a^I \in C^I$	✓	✓	✓	✓	✓
	$(a,b): R$	$(a^I, b^I) \in R^I$	✓	✓	✓	✓	✓

Open World vs. Closed world assumption

Closed World Assumption ←→ Open World Assumption



enhance

Bart speaks French?

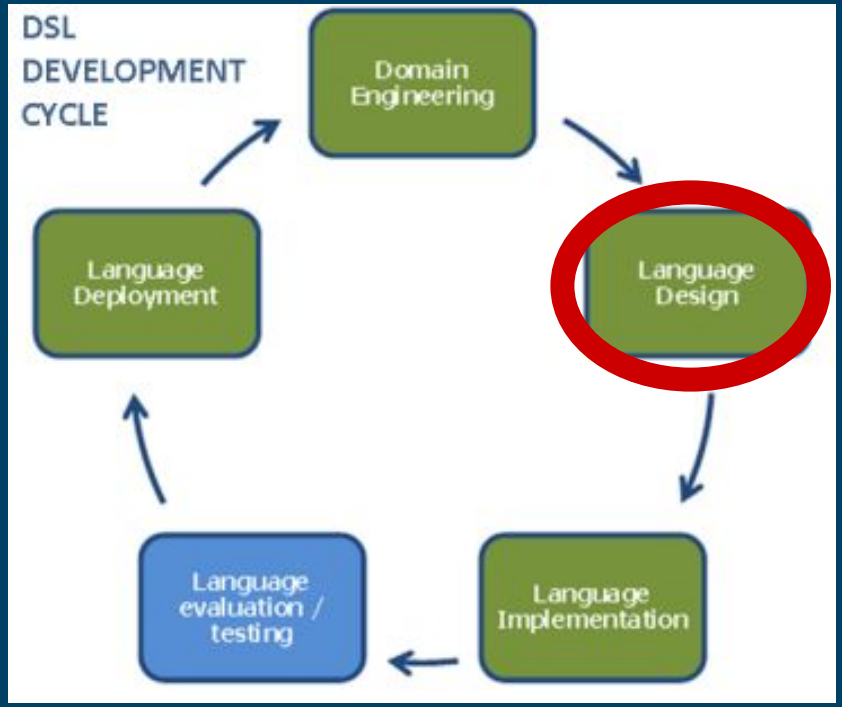
Yes
No

Yes
Maybe
No

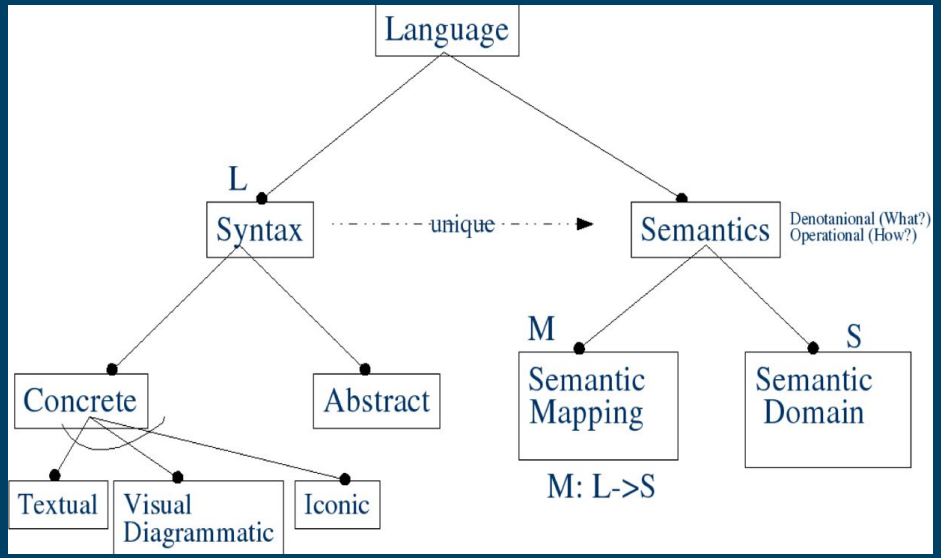
Absence of information must not be validated as negative information

SLE Process

Language Design

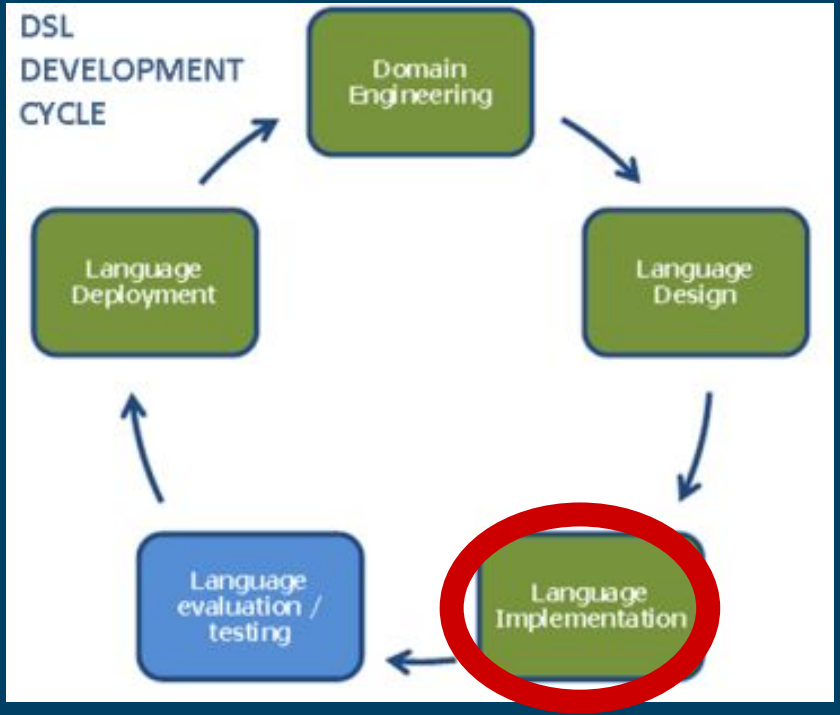


What is a Language?



SLE Process

Implementation



Pick the most adequate tool for your purpose

Visual?

- Modelling workbenches (ideal for prototyping languages):
  - EMF/GMF; GME; DSL Tools; MetaEdit+; AtomPM
- UML Stereotypes?

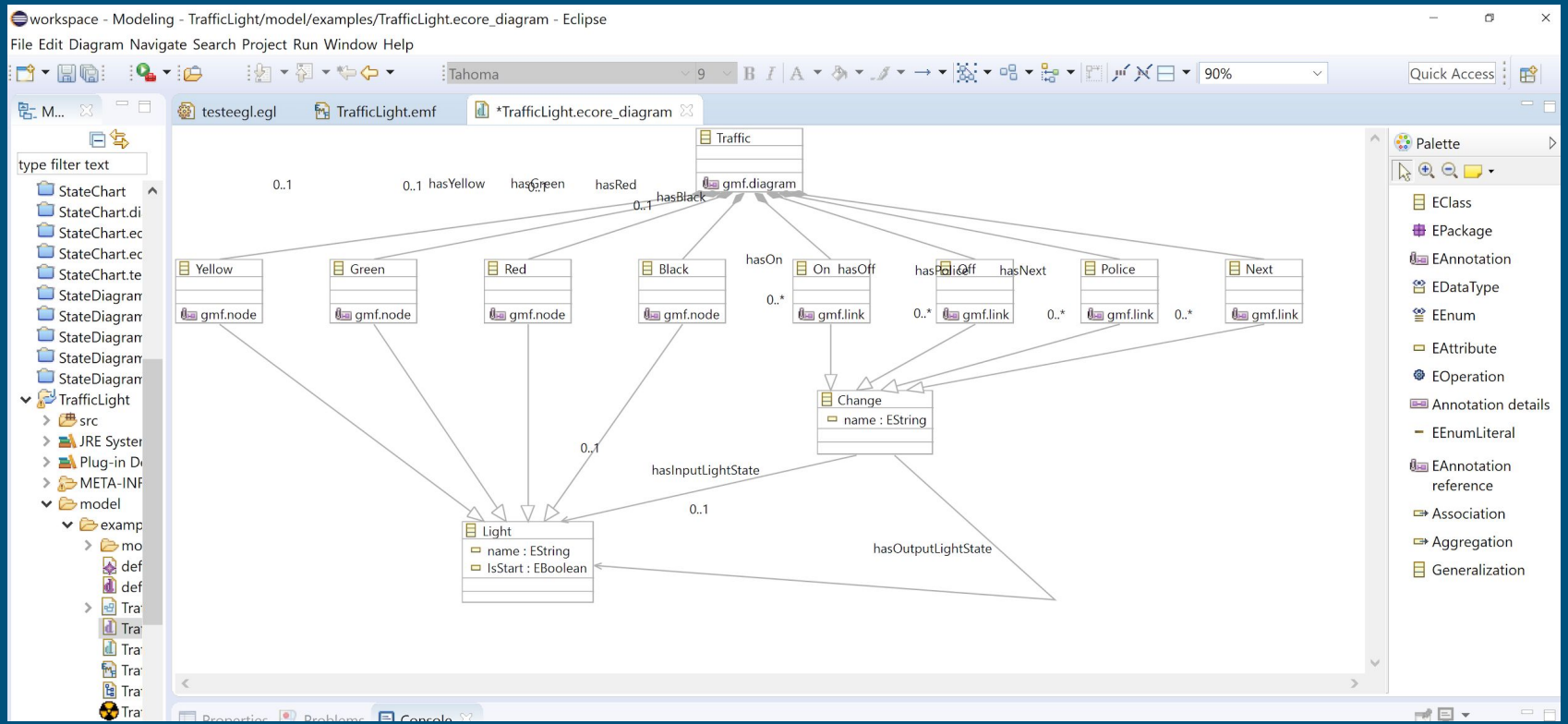
Textual?

- Macro
- Embedded Language? (Ex. Ruby)
- Standalone:
  - Traditional Compiler approach
  - Modelling Workbenches: MPS (JetBrains); Xtext Eclipse;...

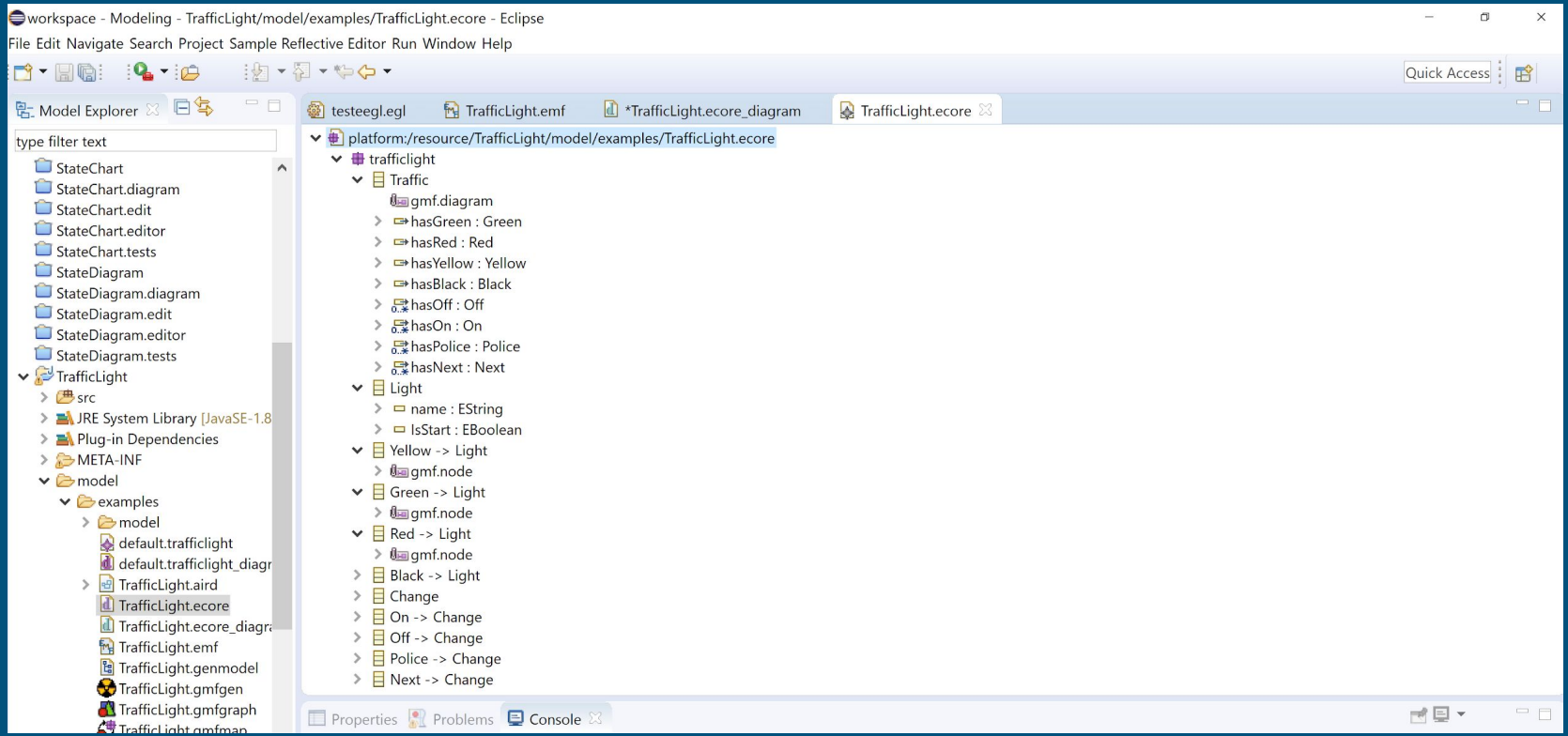
MDD approach

Lets use Eclipse GMF/EMF with the Epsilon toolset

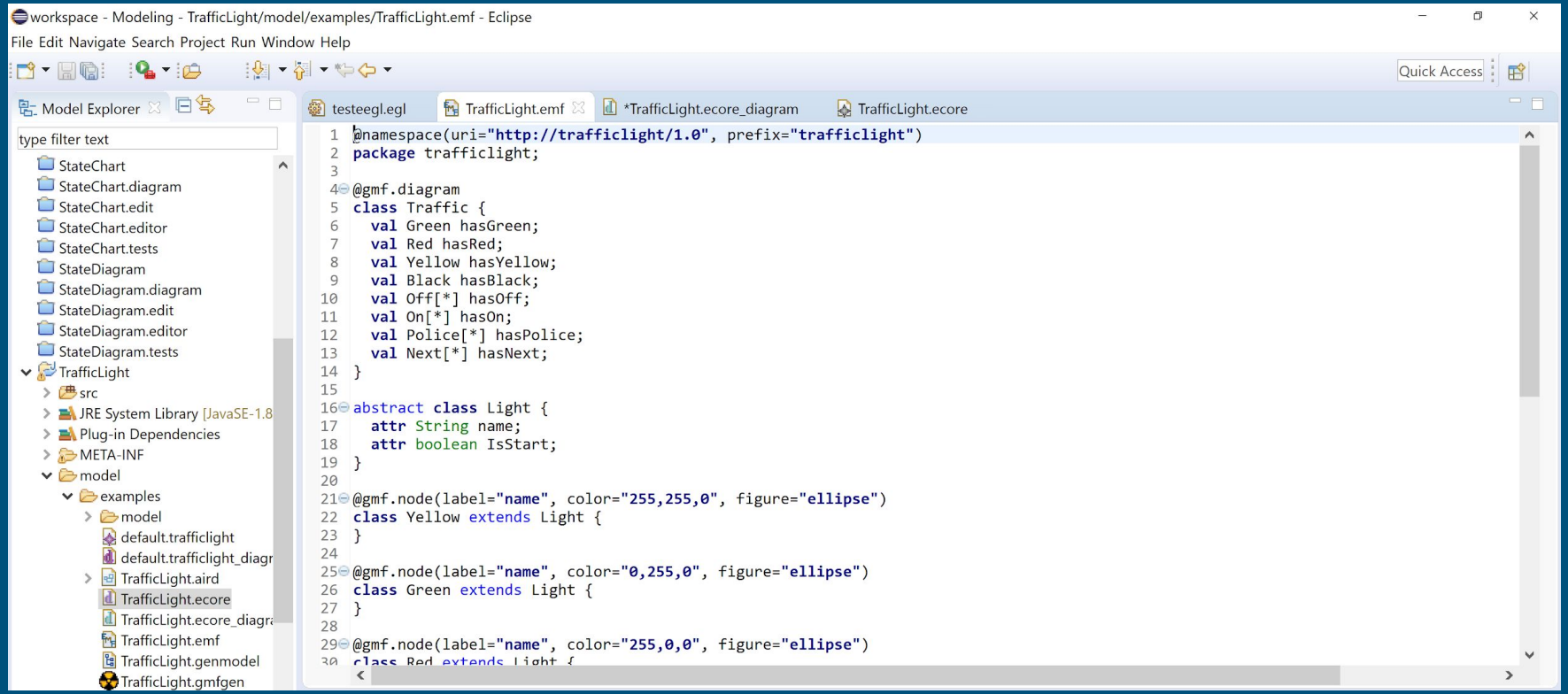
The Abstract Syntax - Ecore Metamodel



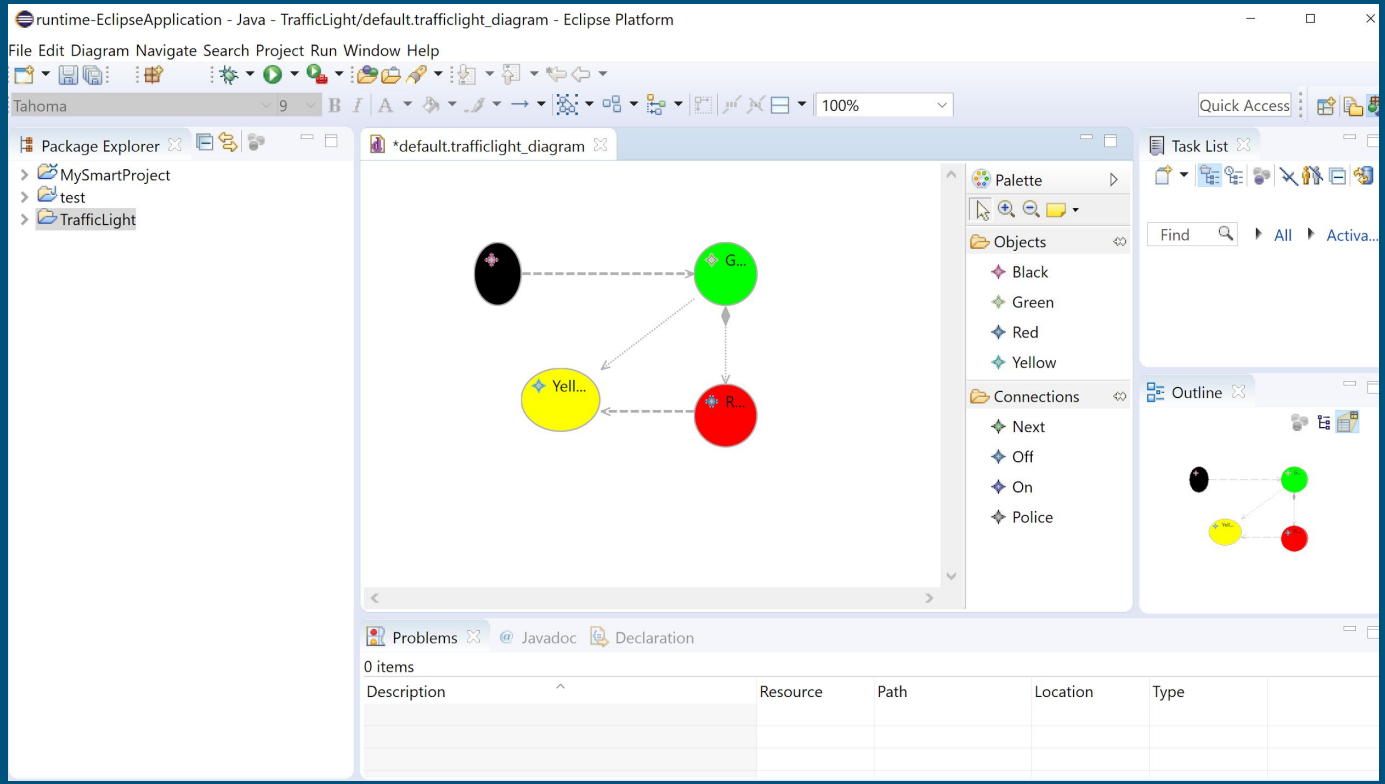
The Abstract Syntax - Ecore Metamodel



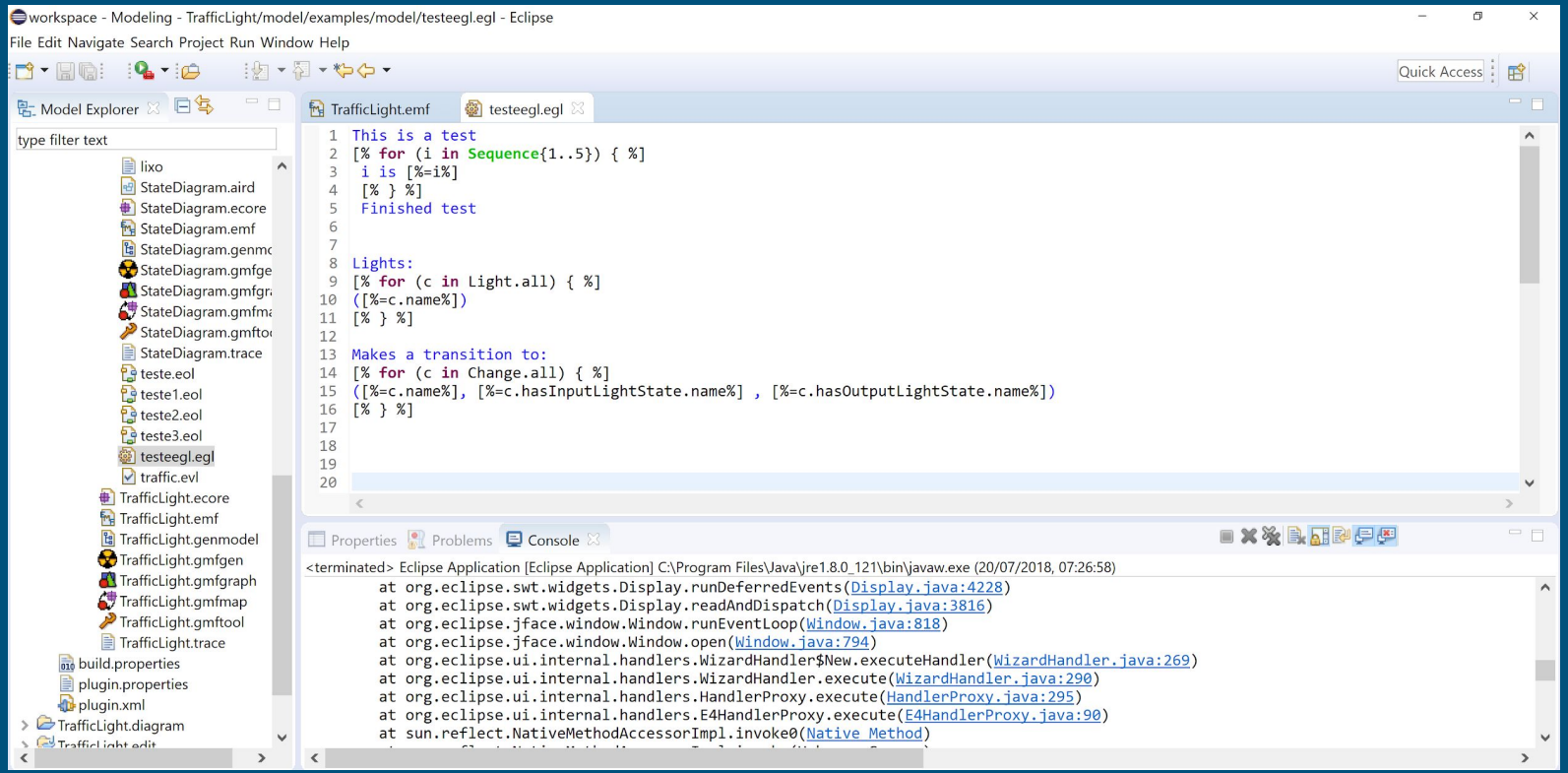
The Concrete Syntax



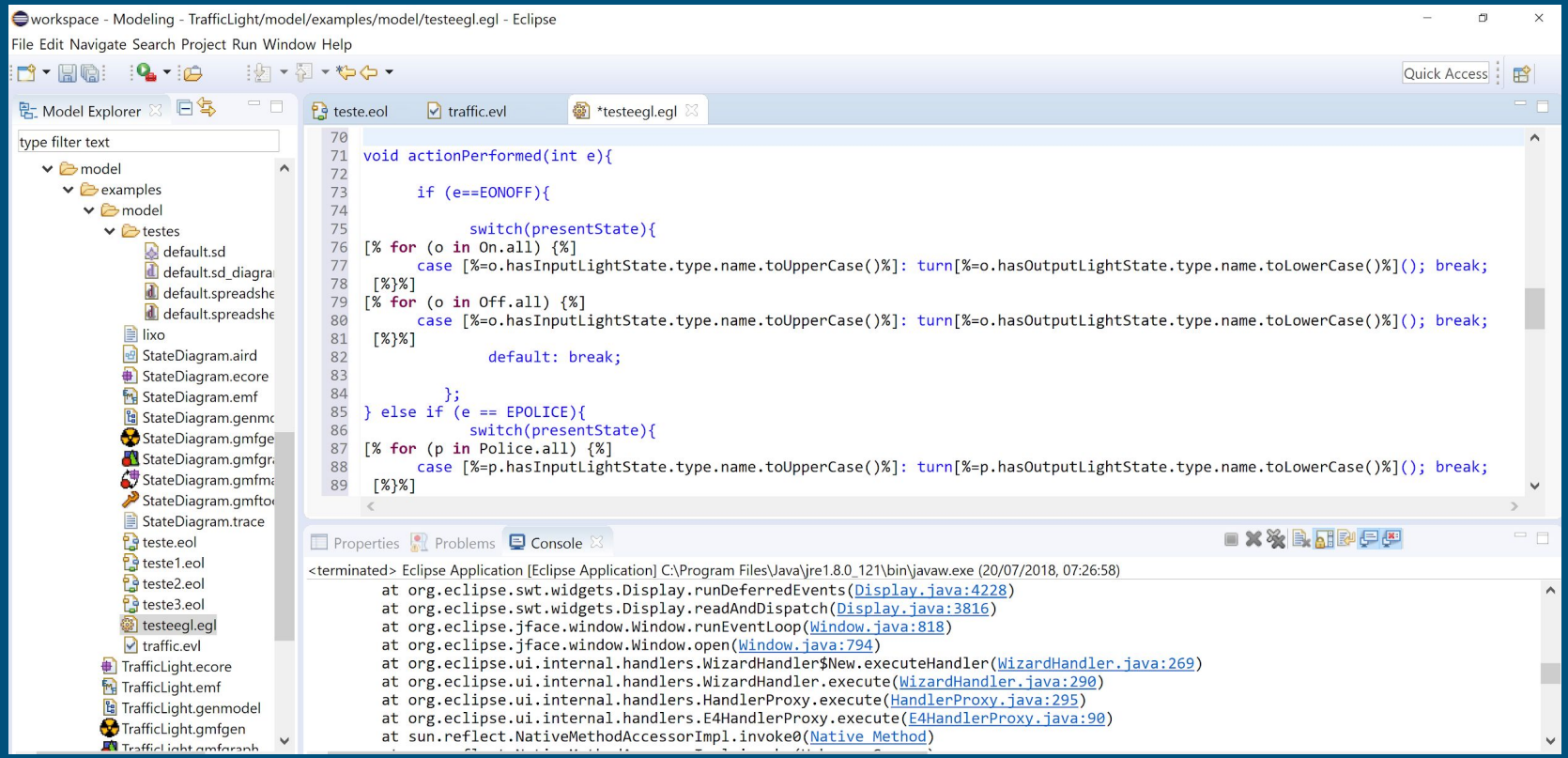
Model to Code - Just push a button



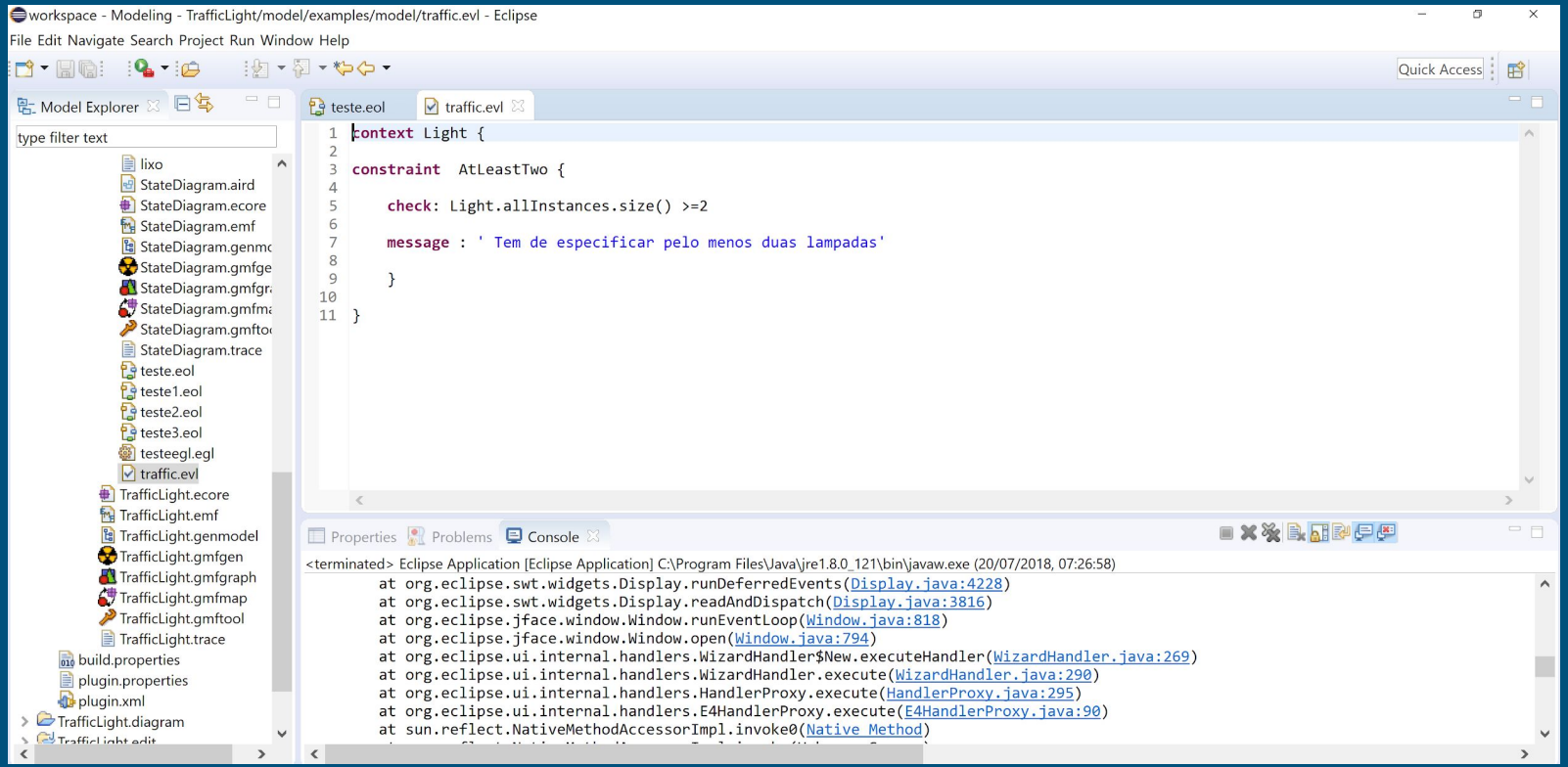
Model to Code - Epsilon Generation Language (Template based)



Model to Code - Epsilon Generation Language (Template based) Arduino

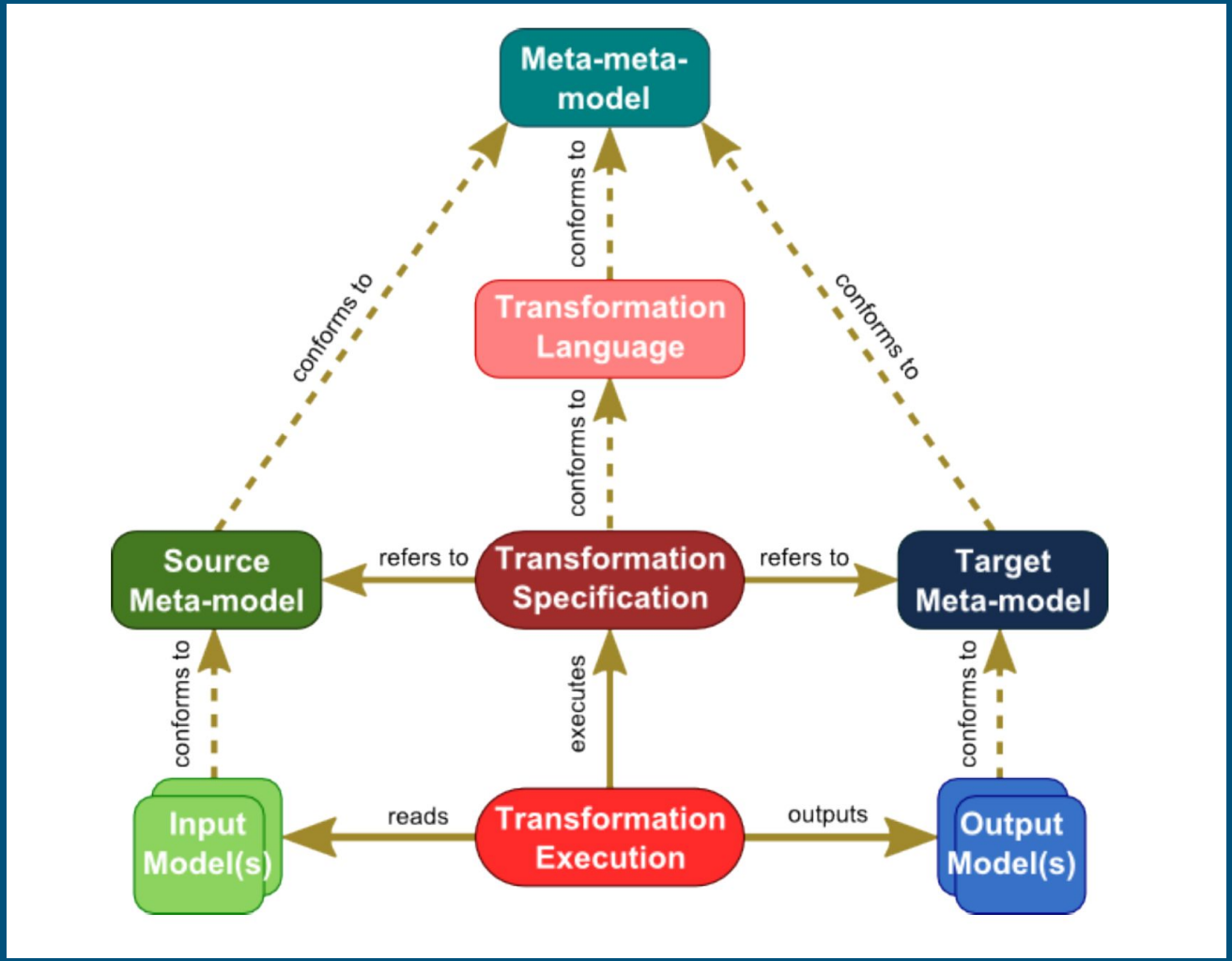


Model to Code - Epsilon Validation Language (Template based)



Model to Code - Epsilon Validation Language (Template based)

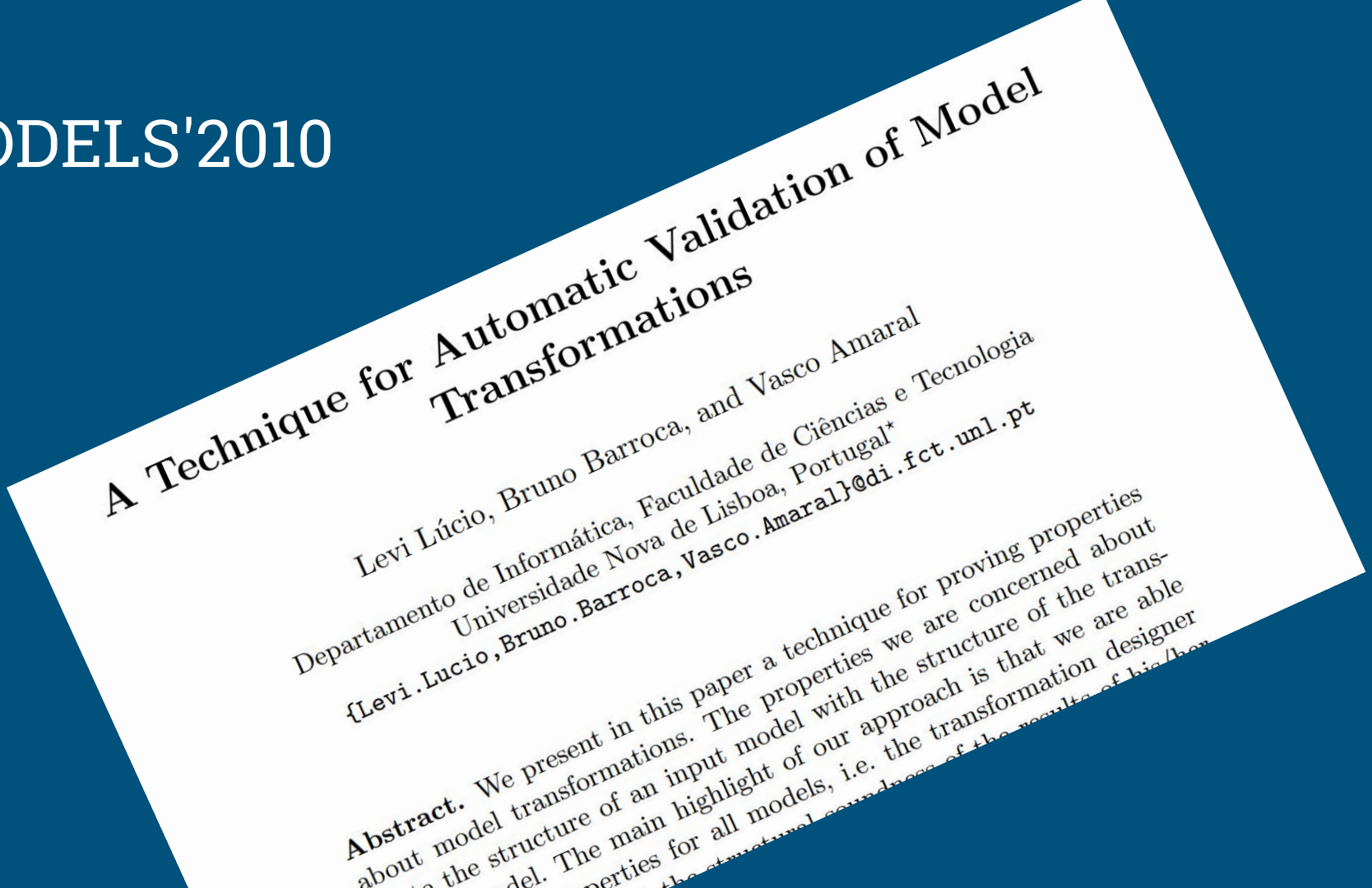
```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
import javax.swing.border.*;

public class TrafficLight extends JFrame implements ActionListener {
    JButton b1, b2, b3;

    Signal green = new Signal(Color.green);
    Signal yellow = new Signal(Color.yellow);
    Signal red = new Signal(Color.red);

    public TrafficLight(){
        super("Traffic Light");
        getContentPane().setLayout(new GridLayout(2, 1));
        b1 = new JButton("Red");
        b2 = new JButton("Yellow");
        b3 = new JButton("Green");
        b1.addActionListener(this);
        b2.addActionListener(this);
        b3.addActionListener(this);

        green.turnOn(false);
        yellow.turnOn(false);
        red.turnOn(true);

        JPanel p1 = new JPanel(new GridLayout(3,1));
        p1.add(red);
        p1.add(yellow);
        p1.add(green);
        JPanel p2 = new JPanel(new FlowLayout());
        p2.add(b1);
```


Meta-meta-model
conforms to
conforms to
conforms to
Transformation Language
conforms to
Source Meta-model
refers to
Transformation Specification
refers to
Target Meta-model
conforms to
executes
conforms to
Input Model(s)
reads
Transformation Execution
outputs
Output Model(s)

Transformation rules (ETL)

```
rule Tree2Node

  transform t : Tree!Tree
  to n : Graph!Node {

  n.name = t.label;

  // If t is not the top tree
  // create an edge connecting n
  // with the Node created from t's parent

  if (t.parent.isDefined()) {
    var e : new Graph!Edge;
    e.source ::= t.parent;
    e.target = n;
  }
}
```

MODELS'2010



A Technique for Automatic Validation of Model Transformations

Levi Lúcio, Bruno Barroca, and Vasco Amaral

Departamento de Informática, Faculdade de Ciências e Tecnologia
Universidade Nova de Lisboa, Portugal*
{Levi.Lucio,Bruno.Barroca,Vasco.Amaral}@di.fct.unl.pt

Abstract. We present in this paper a technique for proving properties about model transformations. The properties we are concerned about
the structure of an input model with the structure of the trans-
del. The main highlight of our approach is that we are able
perties for all models, i.e. the transformation designer

SQJ'2018

Software Qual J (2018) 26:417–453
DOI 10.1007/s11219-016-9352-4

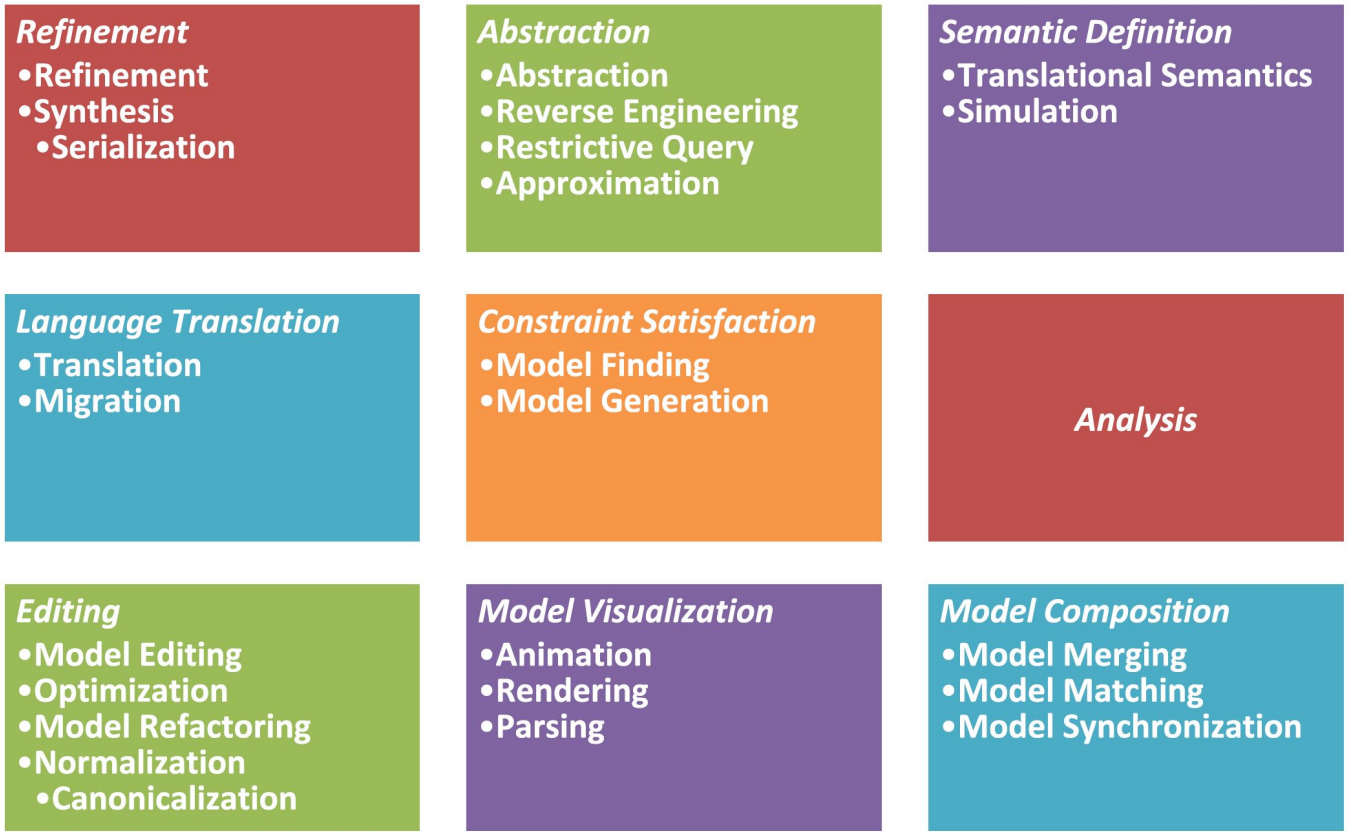
CrossMark

Semantic languages for developing correct language translations

Bruno Barroca[1] · Vasco Amaral[1] · Didier Buchs[2]

uary 2017
s Media New York 2016

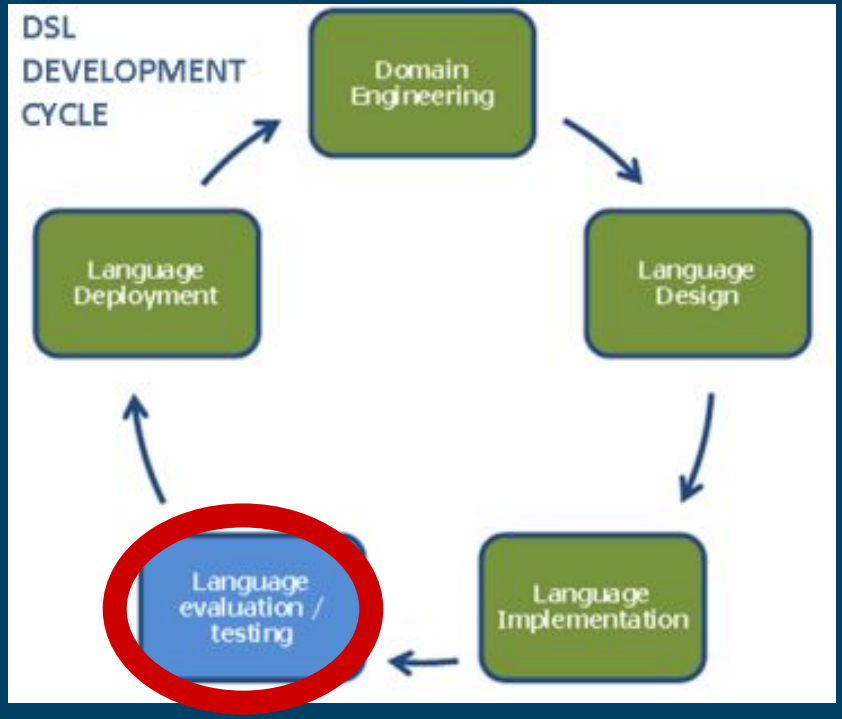
Model transformation intent classification



L. Lúcio, M. Amrani, J. Dingel, L. Lambers, R. Salay, G. Selim, E. Syriani and M. Wimmer. Model transformation intents and their properties. *Journal on Software and Systems Modeling*: 10.1007/s10270-014-0429-x. Springer (2014).

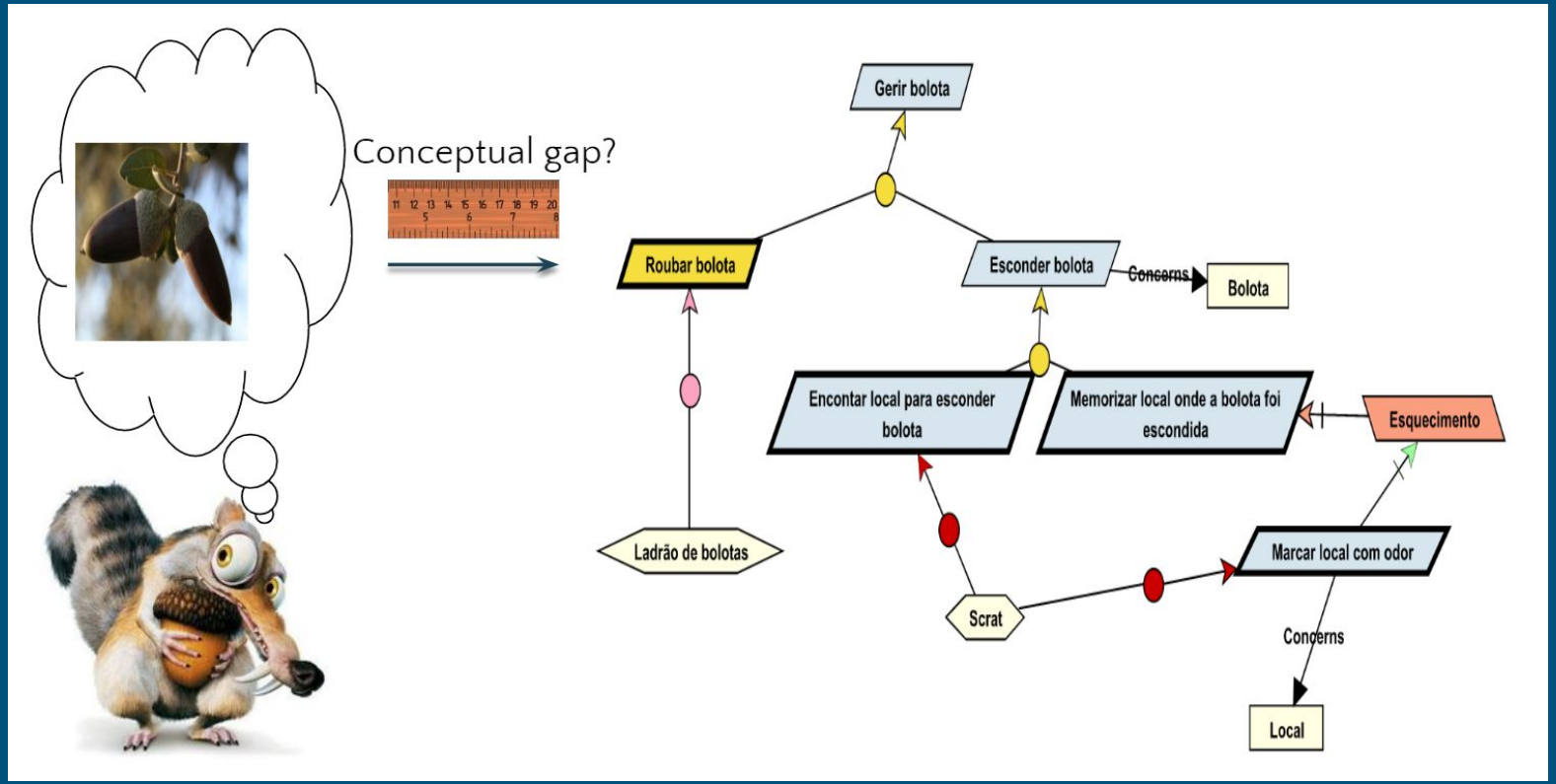
SLE Process

Evaluation



Gabriel, P., M. Goulão, and V. Amaral, "Do Software Languages Engineers Evaluate their Languages?", XIII Congreso Iberoamericano en "Software Engineering" (CIbSE'2010), ISBN: 978-9978-325-10-0, Cuenca, Ecuador, Universidad del Azuay, pp. 149-162, 2010

The language has to empower its user…
or he will end up using something else



CIbSE'2010

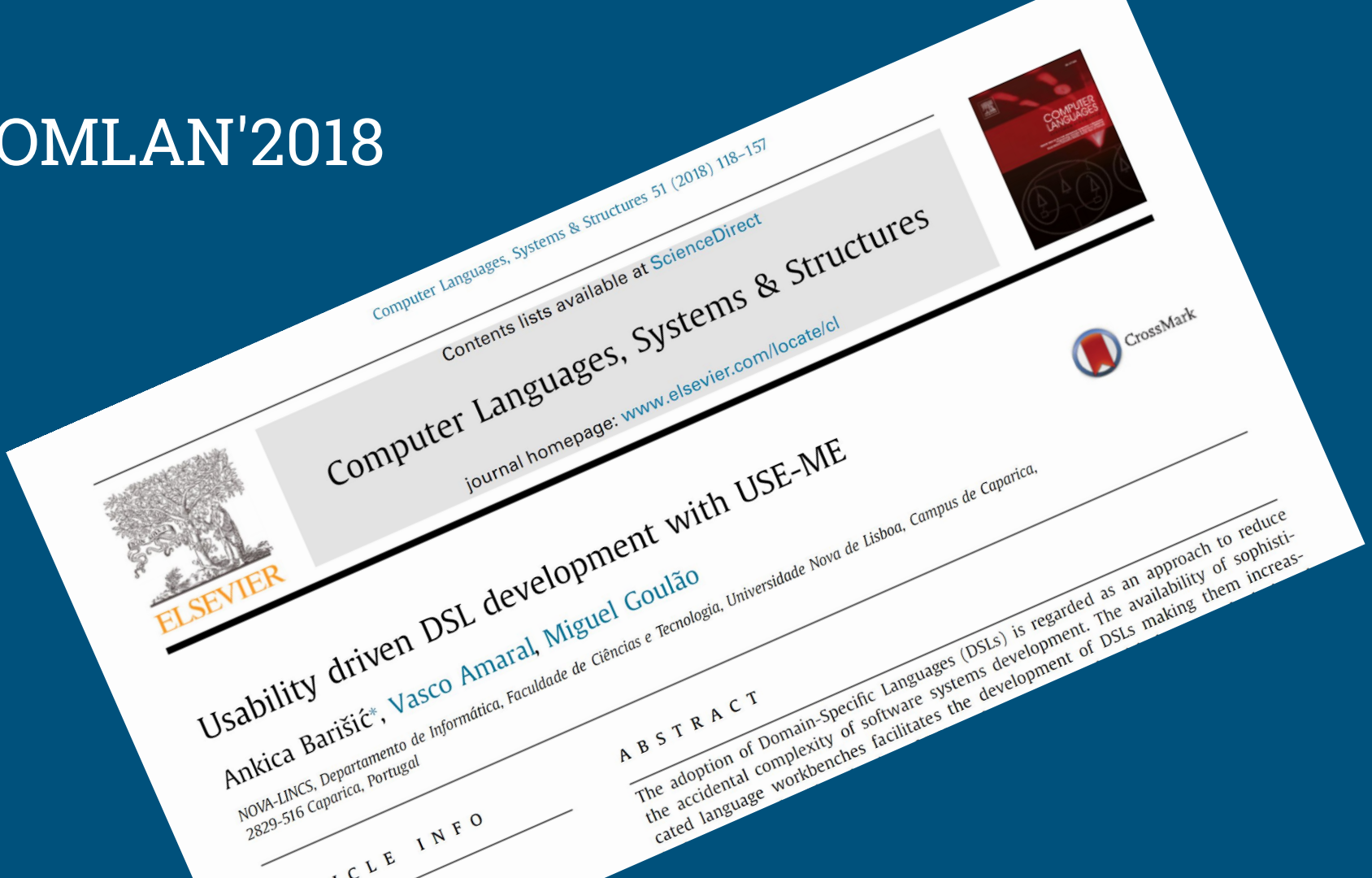
Do Software Languages Engineers Evaluate their Languages?

Pedro Gabriel, Miguel Goulão, and Vasco Amaral

CITI, Departamento de Informática, Faculdade de Ciências e Tecnologia, FCT, Universidade Nova de Lisboa, 2829-516 Caparica, Portugal
pedrohgabriel@gmail.com,{miguel.goulao,vasco.amaralj}@di.fct.unl.pt
http://citi.di.fct.unl.pt/

Abstract. Domain Specific Languages (DSLs) can contribute to increment productivity, while reducing the required maintenance and programming expertise. We hypothesize that Software Languages Engineering (SLE) developers consistently skip, or relax, Language Evaluation. Based on the experience of engineering other types of software products, we assume that this may potentially lead to the deployment of inadequate languages. The fact that the languages already deal with concepts from the problem domain, and not the solution domain, is not enough to validate several issues at stake, such as its expressiveness, usability, effectiveness, maintainability, or even the domain expert's productivity while using them. We present a systematic review on articles published in top ranked venues, from 2001 to 2008, which report DSLs' construction, to characterize the common practice. This work confirms our initial hypothesis and lays the ground for the discussion on how to include a approach to DSL evaluation in the SLE process.

COMLAN'2018



Computer Languages, Systems & Structures 51 (2018) 118–157

Contents lists available at ScienceDirect

Computer Languages, Systems & Structures

journal homepage: www.elsevier.com/locate/cl

Usability driven DSL development with USE-ME

Ankica Barišić*, Vasco Amaral, Miguel Goulão

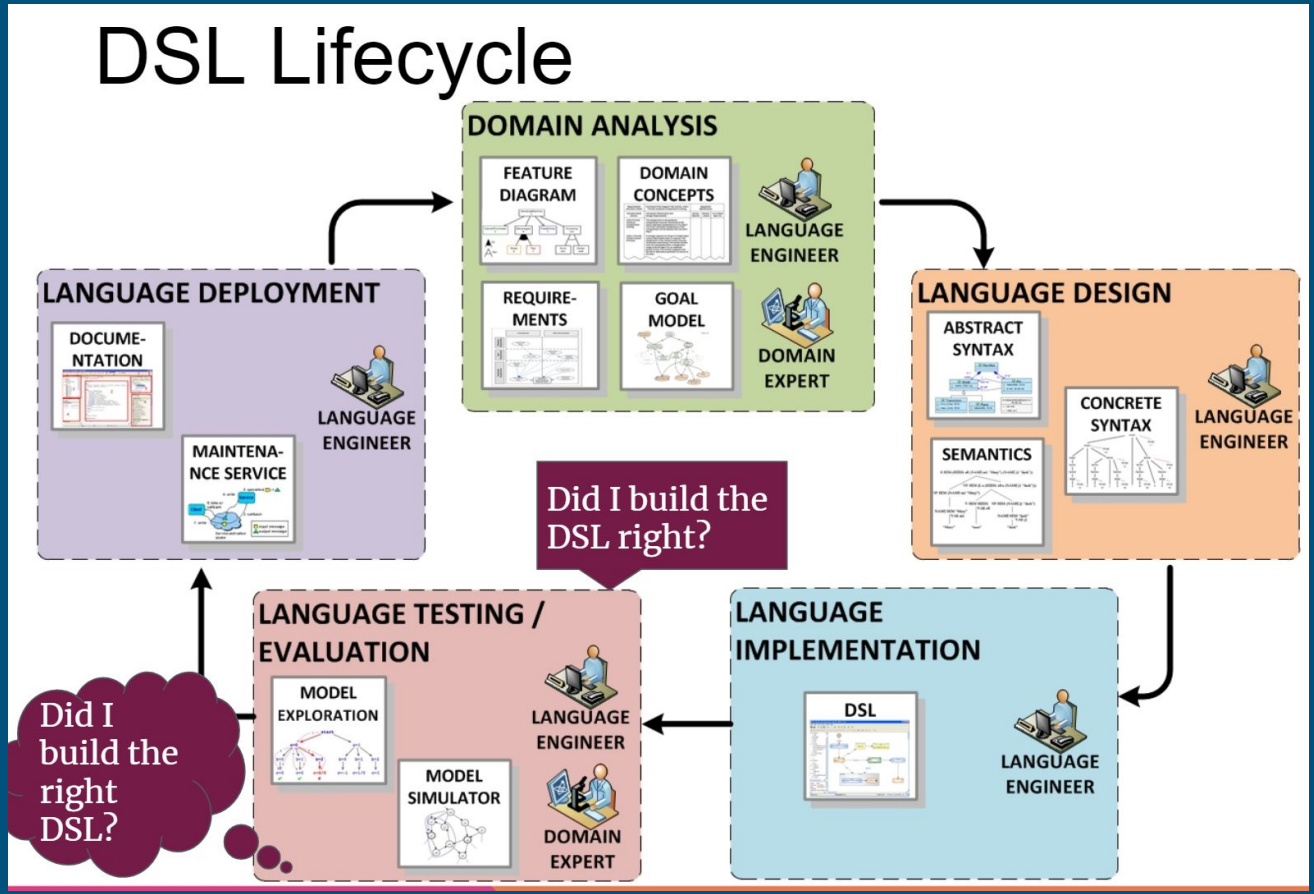
NOVA-LINCS, Departamento de Informática, Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa, Campus de Caparica, 2829-516 Caparica, Portugal

ARTICLE INFO

ABSTRACT

The adoption of Domain-Specific Languages (DSLs) is regarded as an approach to reduce the accidental complexity of software systems development. The availability of sophisticated language workbenches facilitates the development of DSLs making them increas-

DSL Lifecycle



What strategies are available to us?

Constructive approaches:

- Our own expertise and common sense
- Usability heuristics such as the “Physics of notations”

Evaluation-based approaches:

- “Traditional” usability evaluations
- User monitoring while using the DSML

Language usage tasks

writing
reading
interpretation
comprehension
memorization
problem solving

How is SLE doing?

SLE misses:

- More examples of successful industry projects showing SLE
- Maturity in some stages
- Language Composition/Integration
- Language Evolution
- Improved Supporting Tools
- Cost estimation strategies
- ...

Are we there yet?

Not yet. Next Lecture.

Thank you!

Contact: vma@fct.unl.pt