Domain Modelling

Model-Driven Engineering course 2022/2023

Vasco Amaral

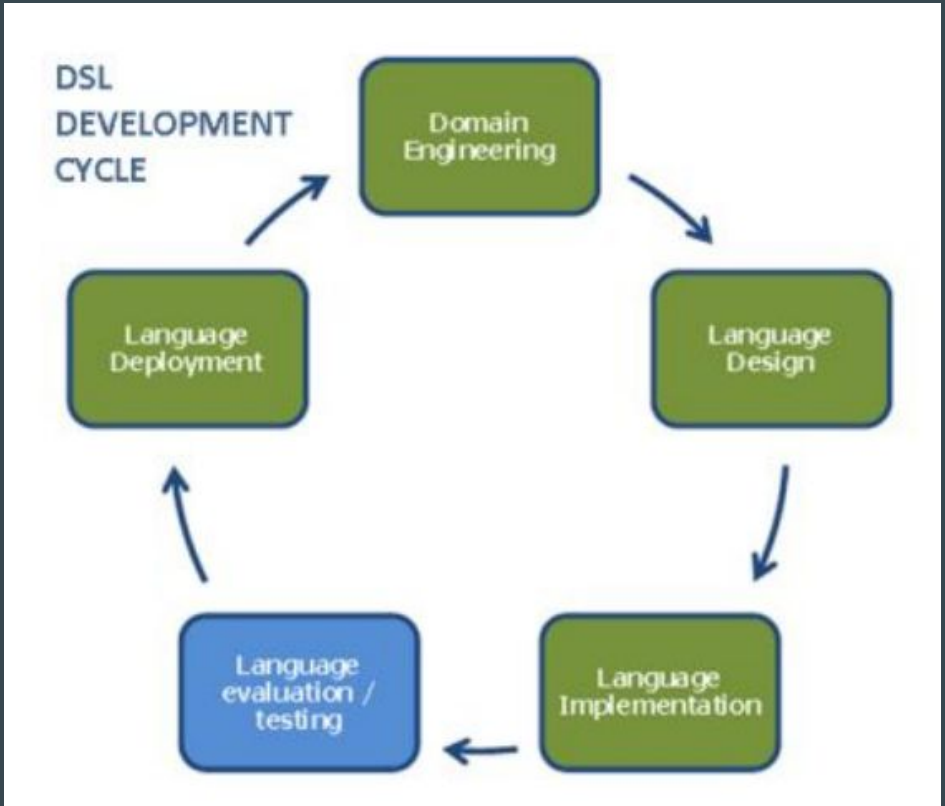
General Purpose

example from MDESystem

example from MDESystem

example from MDESystem

Software Languages Engineering



Questions

What is a Domain?

What is a Domain Model?

What is Domain Modelling?

What is Domain Engineering?

What is Domain Analysis?

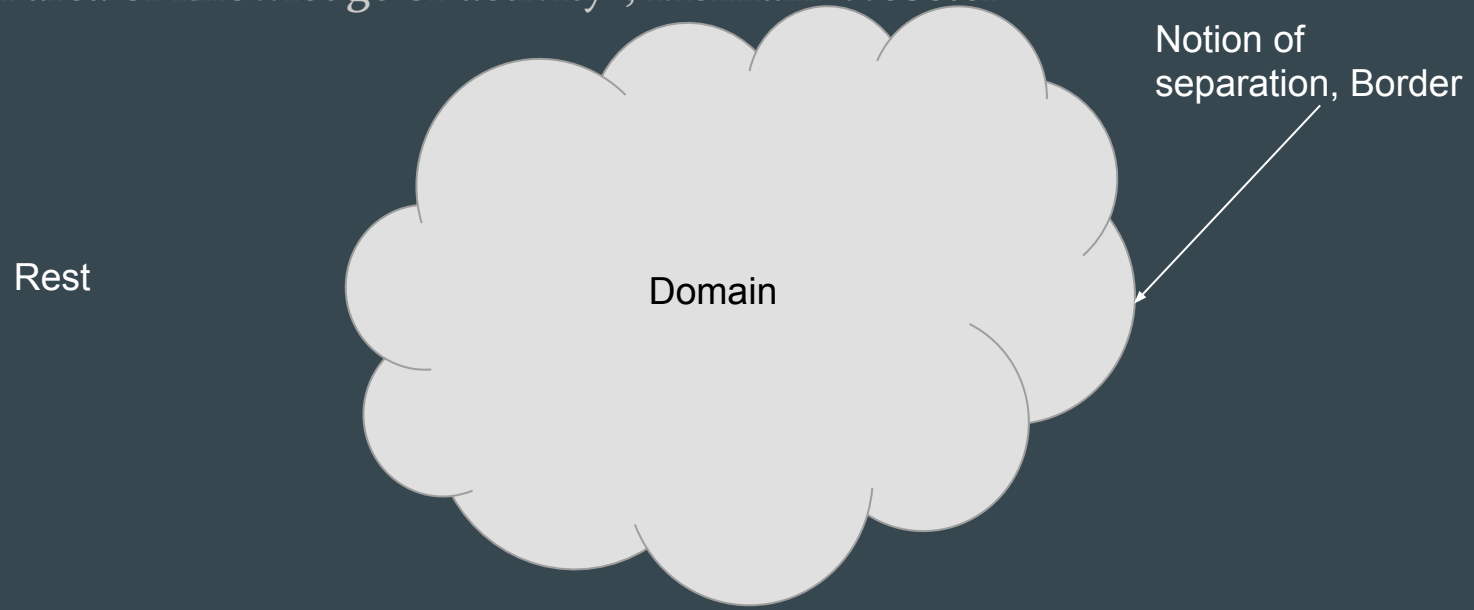
Which formalisms to represent a Domain Model?

How to capture them?

What is a Domain? - From the dictionary

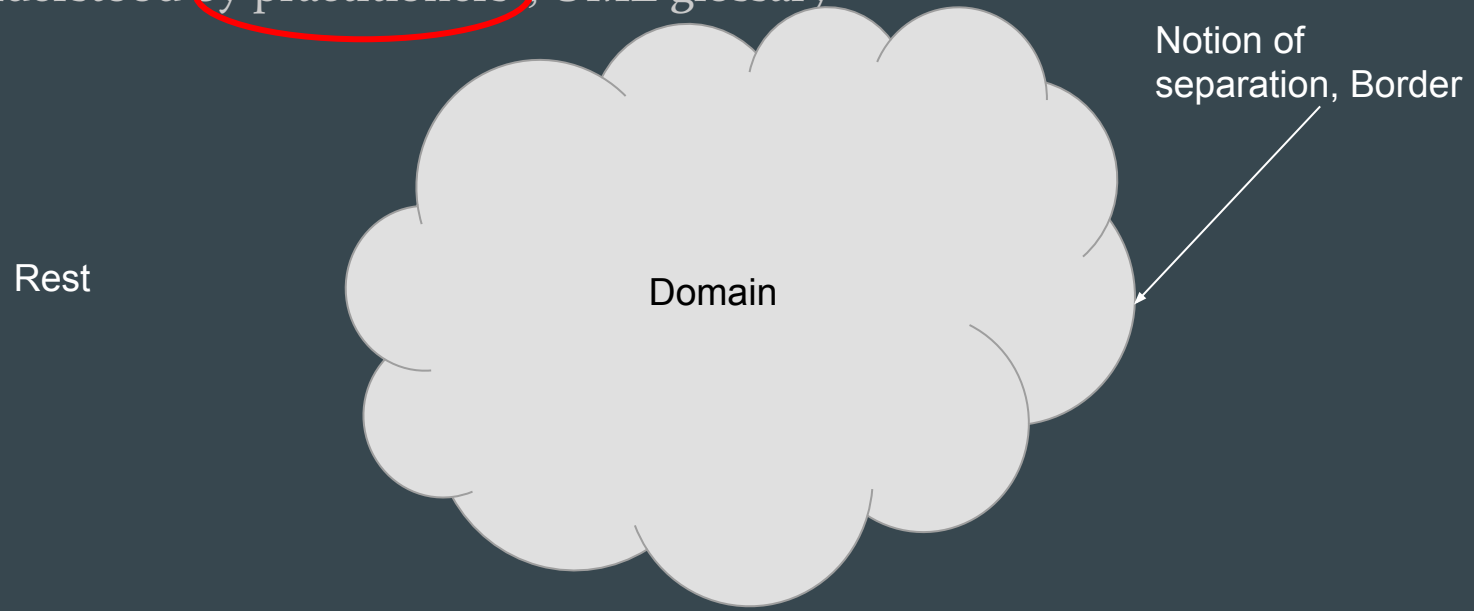
"a realm or range of personal knowledge, responsibility, etc.", Dictionary.com

"an area of knowledge or activity", Merriam-Webster



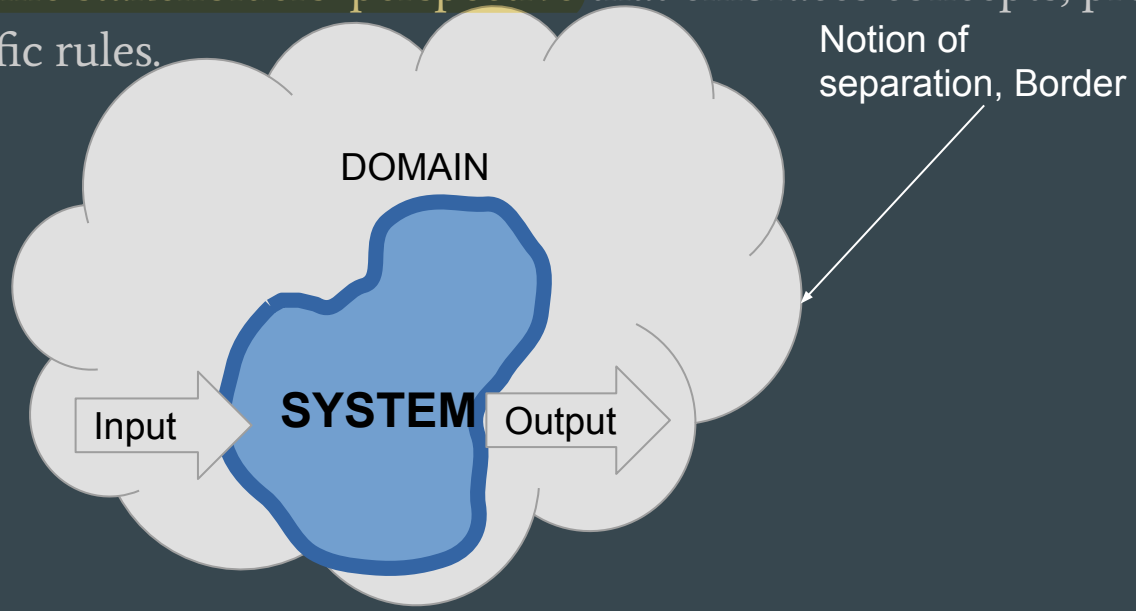
What is a Domain?

"An Area of knowledge or activity characterized by a set of concepts and terminology understood by practitioners", UML glossary



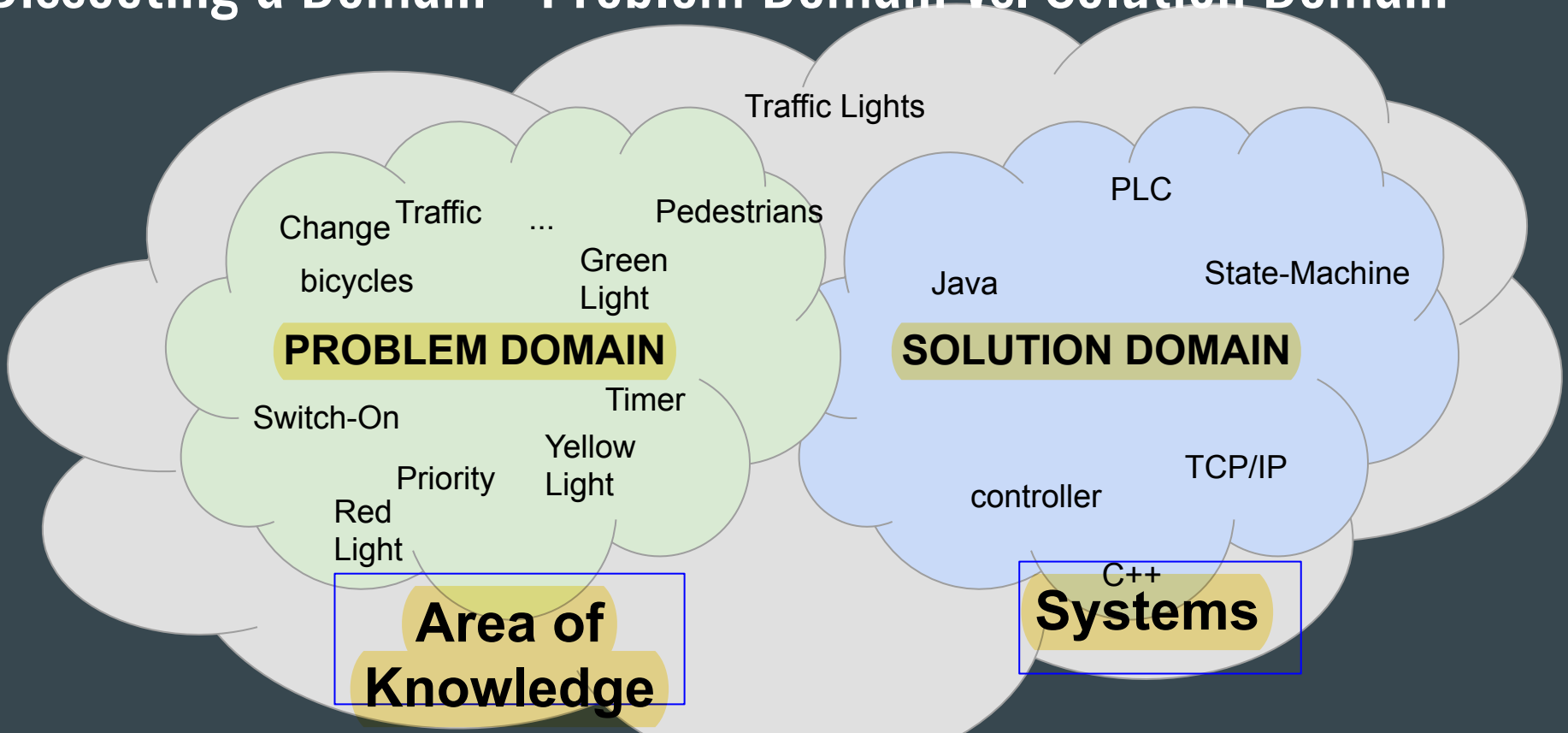
What is a Domain? System perspective

An area of knowledge or activity, typically giving context to a set of problems to be solved by a system, relevant to some stakeholders' perspective that embraces concepts, processes, terminology, and specific rules.

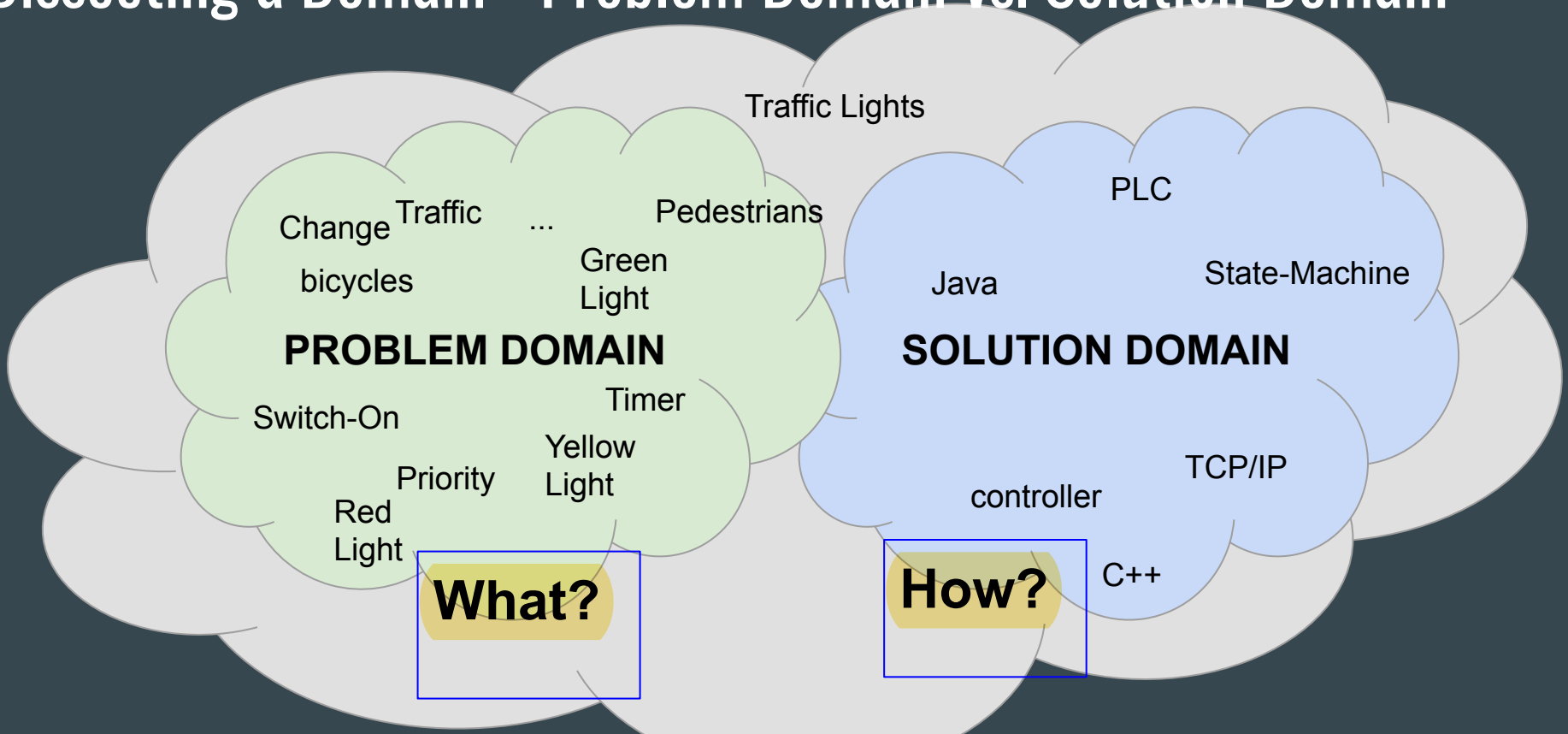


Dissecting Domain - Problem Domain vs. Solution Domain

Dissecting a Domain - Problem Domain vs. Solution Domain



Dissecting a Domain - Problem Domain vs. Solution Domain



Scoping a Domain

Must be scoped to maximize the agreement of the stakeholders regarding the inclusion of their goals

Must include:

- concepts,
- terminology
- and rules understood by the stakeholders in the domain

Includes knowledge of how to build software systems (or parts of software systems) in the area

Horizontal vs. Vertical

These terms have no clear meaning in the community, usually:

Horizontal Domain Scope - Classes of parts of systems. if it is technically oriented, with broad business domains of application. applied to a large group of applications that share the same technical characteristic

(ex. spreadsheets)

Vertical Domain Scope - pertains to a particular business, usually an in-house solution.

(ex. Wastewater treatment system)

Relationship between Domains

[Czarnecki et. al]

Let consider Domains A and B. We can relate A and B saying that:

A is contained in B - All knowledge in A belongs to B (B is **subdomain**)

A uses B - Knowledge in A references knowledge in B (B is a **support domain**)

A is analogous to B - B is a high degree of similarity to A, but it is not necessary to express one in terms of the other

What is a Domain model?

Different research areas, different interpretations:

OO - is the UML class model

IS - is the ERD model

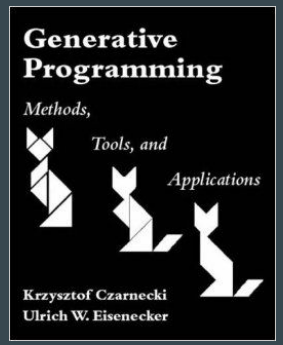
AI - is the Domain Ontology

...

Domain Model

"Is an explicit representation of the 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 and Eisenecker, "Generative Programming Methods, Tools, and Applications"



Domain Model

Domain Definition - Scope of the domain, stakeholders involved. Can give 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,...)

What is Domain Specific Modelling?

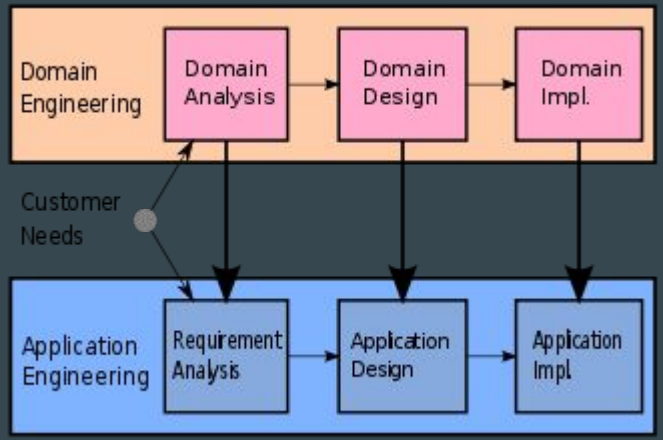
Aims to :

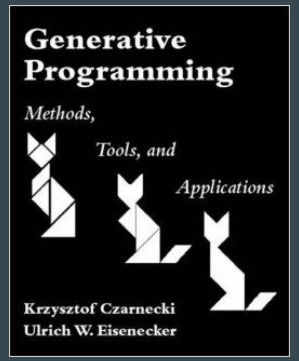
Raise the level of abstraction beyond programming languages and source code

by modelling the solution in a language that directly uses concepts and rules from a specific problem domain

Domain Engineering

“Is the activity of collecting, organizing, and storing **past experience** in **building systems or parts of systems** in a particular domain in the form of **reusable assets** (i.e. reusable work products) as well as providing an adequate means for reusing these assets (i.e., retrieval, qualification, dissemination, adaptation, assembly, and so on) when building new systems.” Czarnecki and Eisenecker

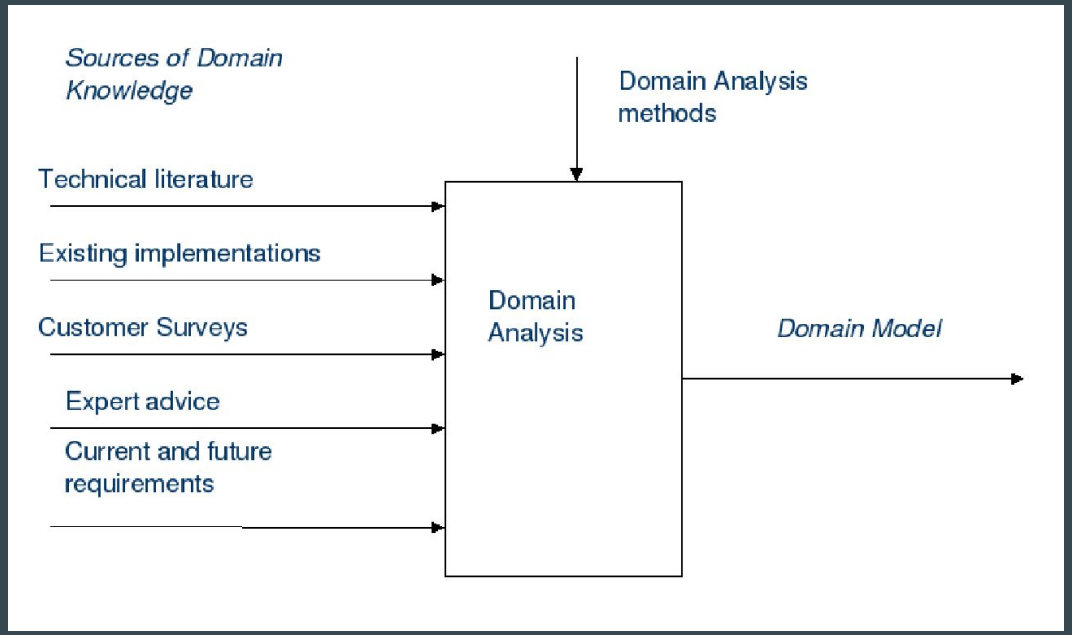




Domain Engineering

Focus:

- Engineering Reusable Software
- Knowledge Management



Adapted from, Scott Thibault Phd Thesis, 1998

Domain Analysis and 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)
- ...

Feature Oriented Domain Analysis

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)

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

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 Modeling

Abstract, concise and explicit representation of variability present in the software.

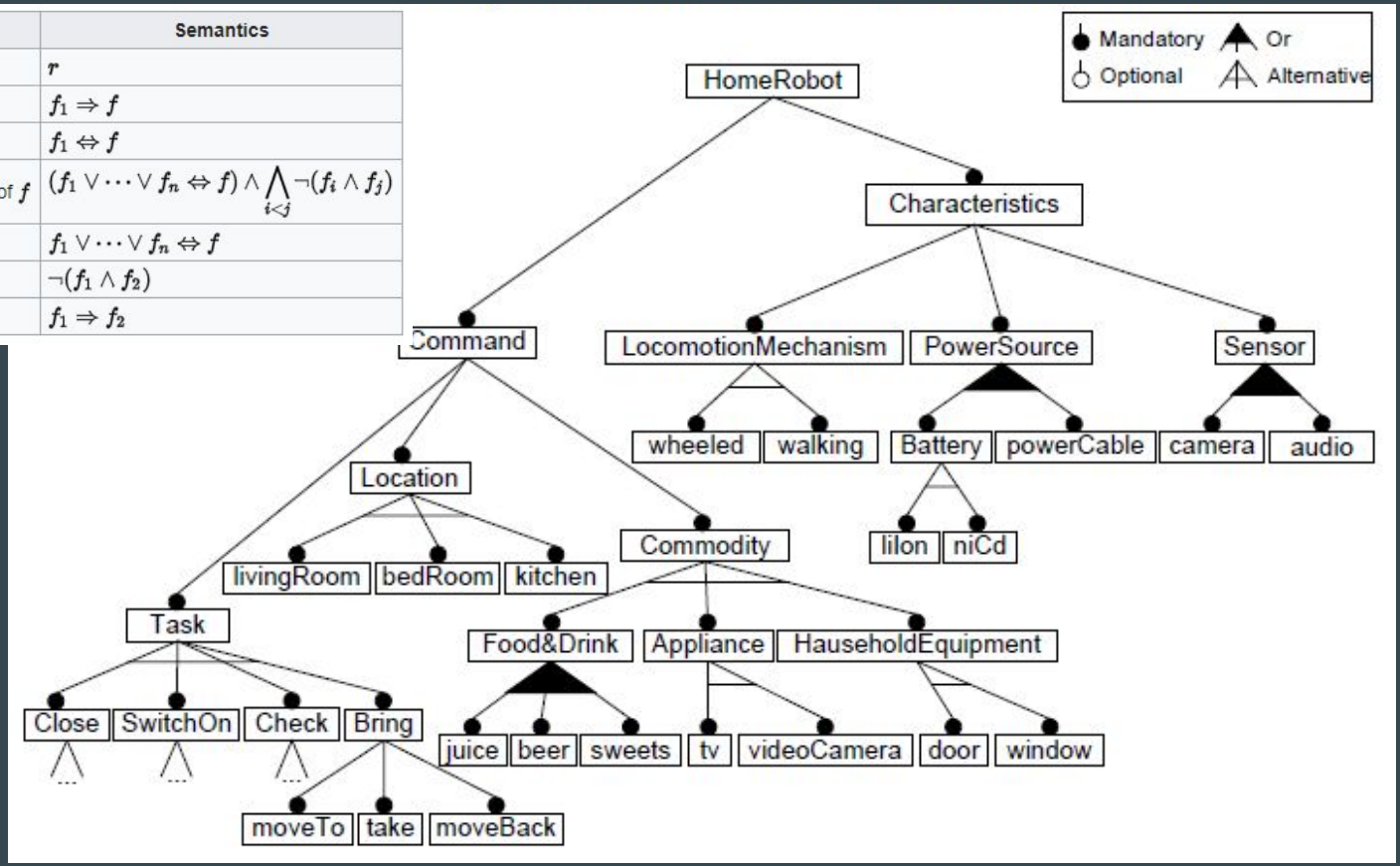
Consists of the the activity of modelling the common and the variable properties of concepts and their interdependencies organized into a coherent model called feature model.

Feature = Concept != Class

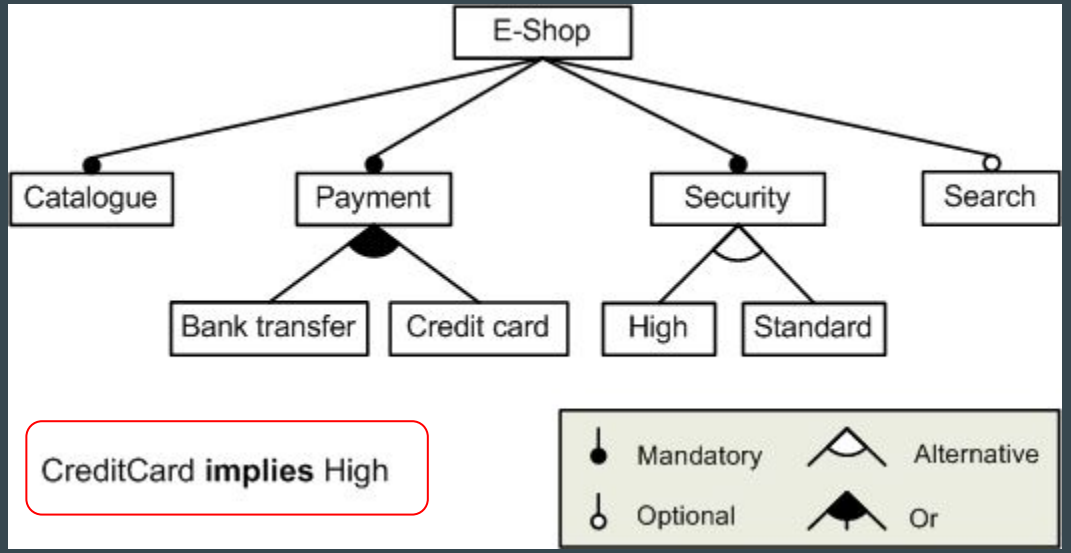
At any level: requirements, architectural, subsystem and component level.

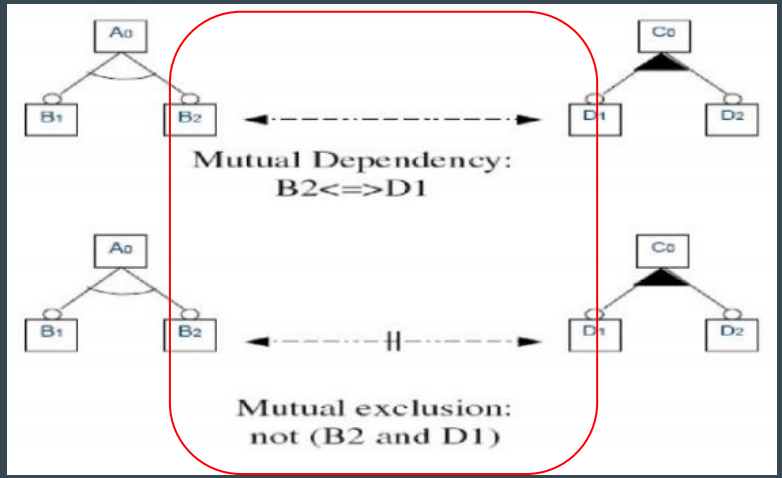
Feature Model

Feature Diagram Primitive	Semantics
r is the root feature	r
f_1 optional sub-feature of f	$f_1 \Rightarrow f$
f_1 mandatory sub-feature of f	$f_1 \Leftrightarrow f$
$f_1, \ldots, f_n$ alternative sub-features of f	$(f_1 \vee \cdots \vee f_n \Leftrightarrow f) \wedge \bigwedge_{i<j} \neg(f_i \wedge f_j)$
$f_1, \ldots, f_n$ or sub-features of f	$f_1 \vee \cdots \vee f_n \Leftrightarrow f$
f_1 excludes f_2	$\neg(f_1 \wedge f_2)$
f_1 requires f_2	$f_1 \Rightarrow f_2$



Dependencies (requires/implies, excludes)

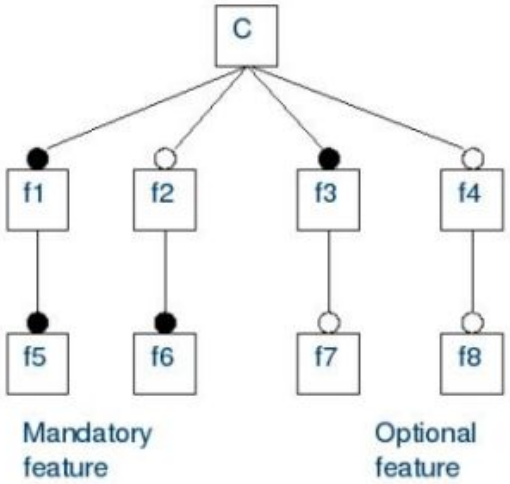




Well-formedness rules:

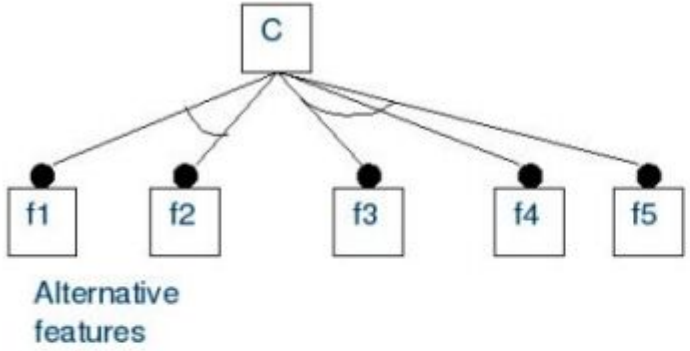
W1- It has only one root feature
W2- No feature becomes one of its own super-features
W3- No feature is an Island

Feature instances



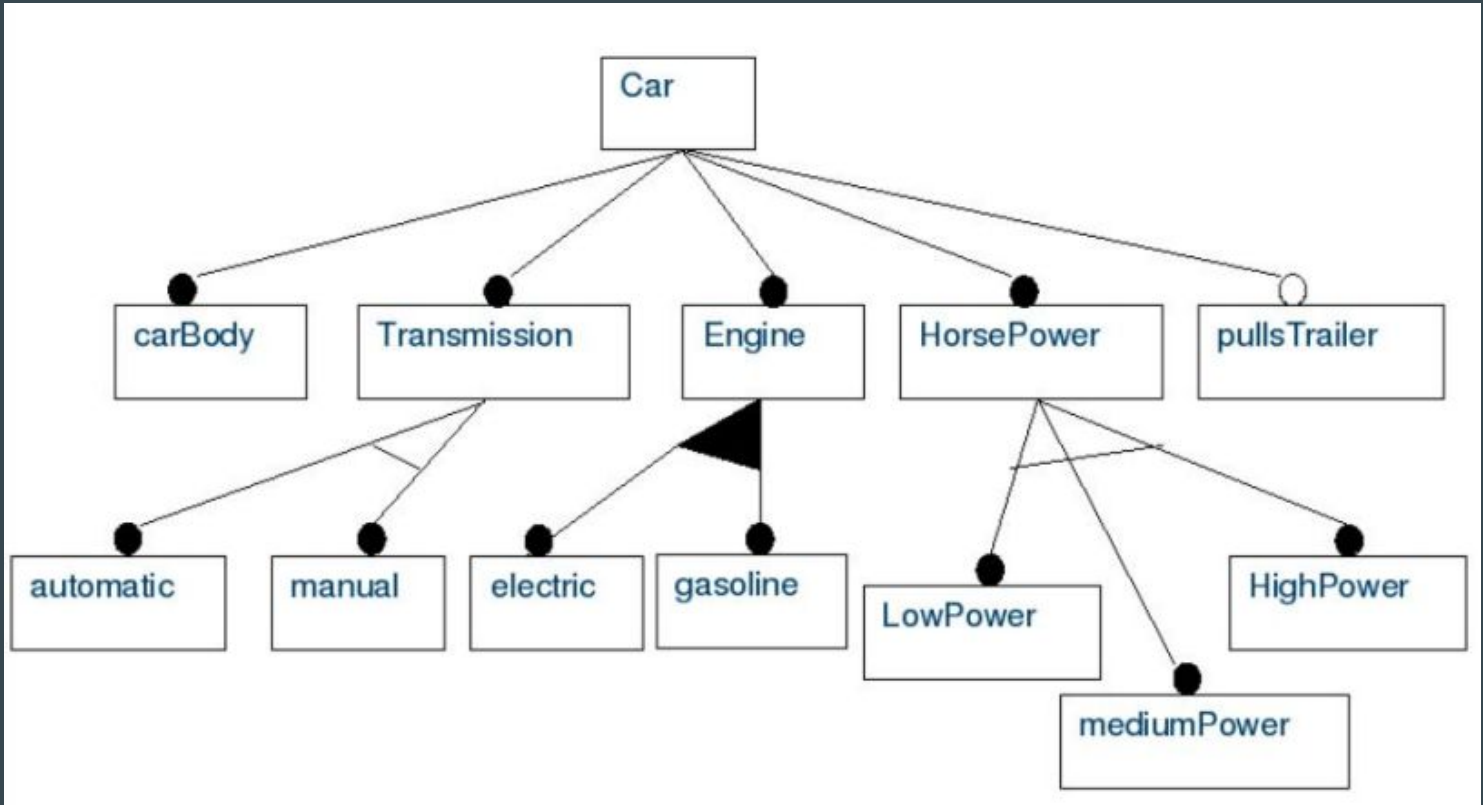
{C, f1, f5, f2, f6, f3} {C, f1, f5, f2, f6, f3, f4} {C, f1, f5, f2, f6, f3, f7}
{C, f1, f5, f3, f4, f8} {C, f1, f5, f3}...

Feature instances

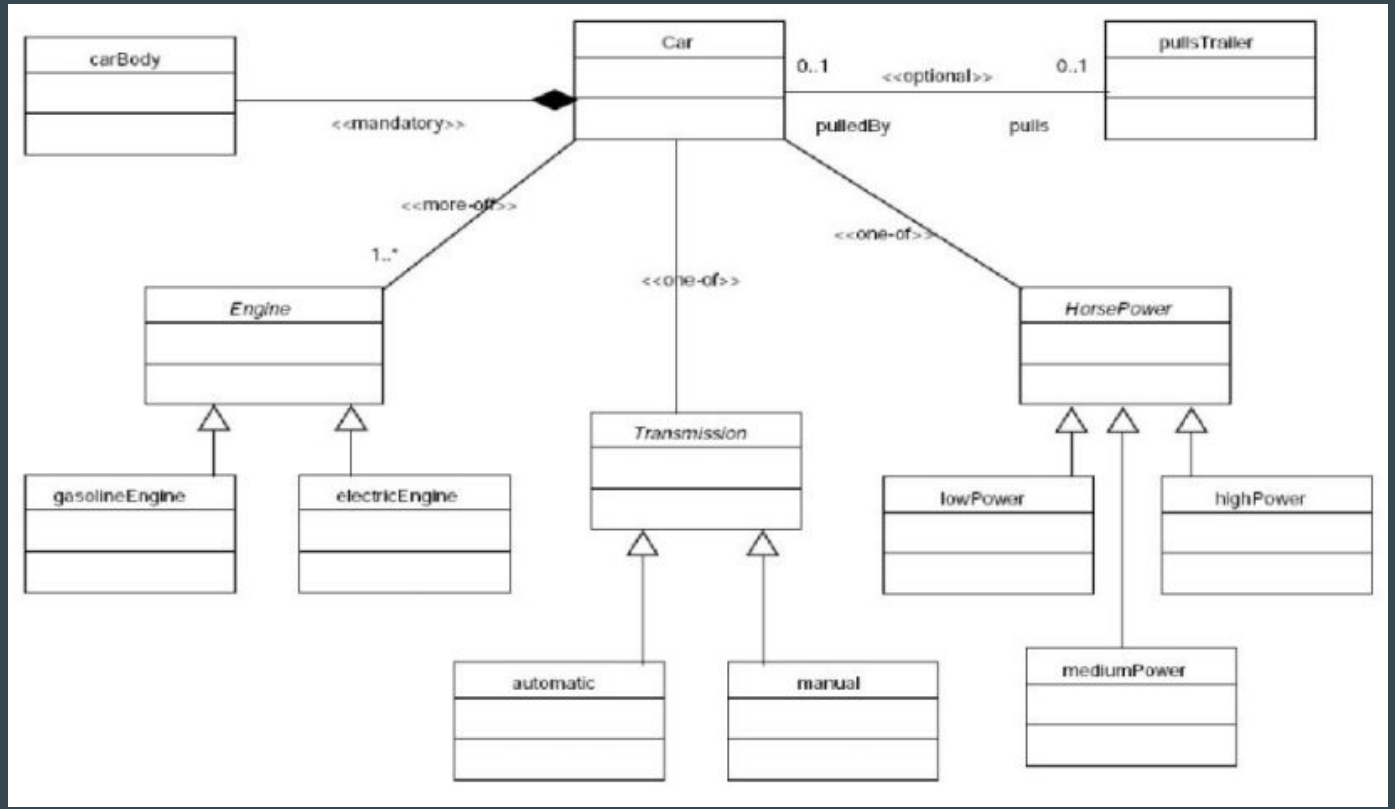


{C, f1, f3} {C, f1, f4} {C, f1, f5} {C, f2, f3} {C, f2, f4} {C, f2, f5}

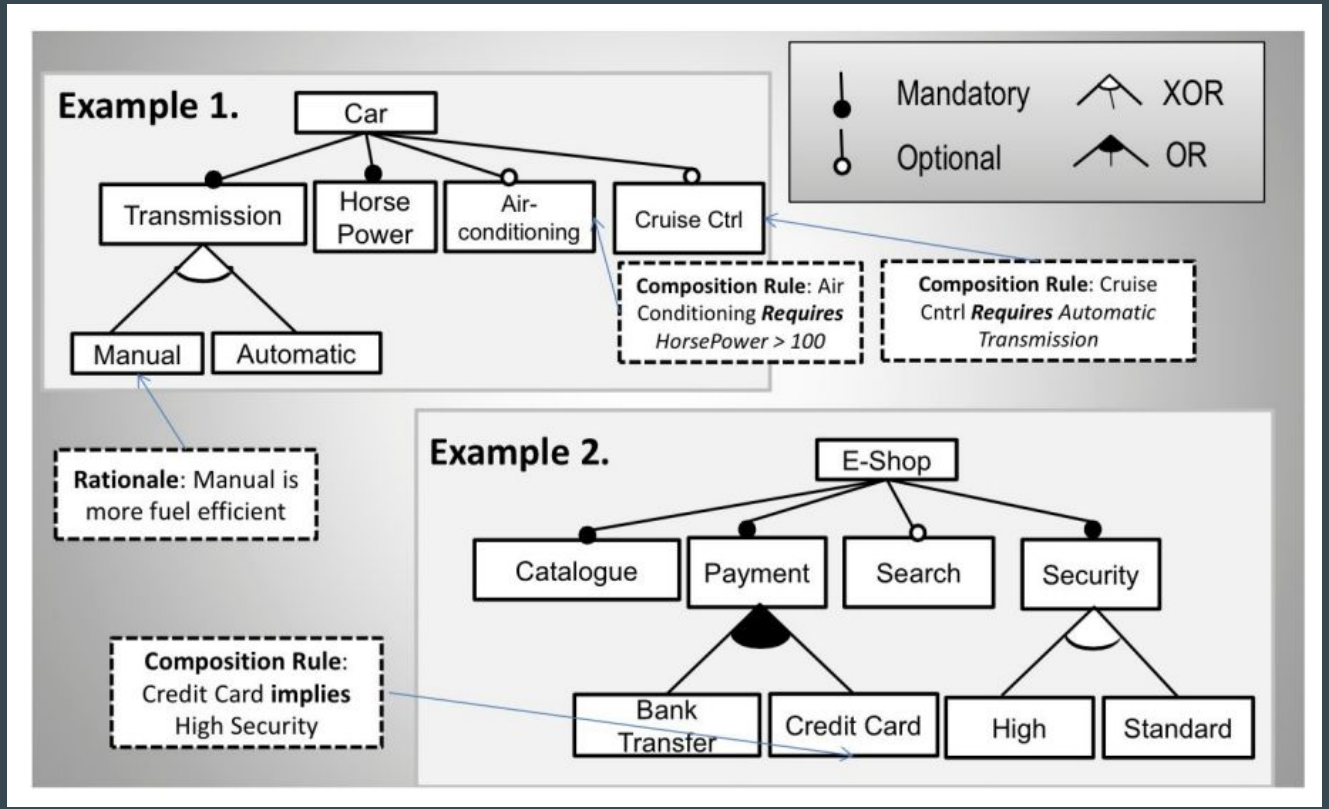
Could we represent the same with another language?



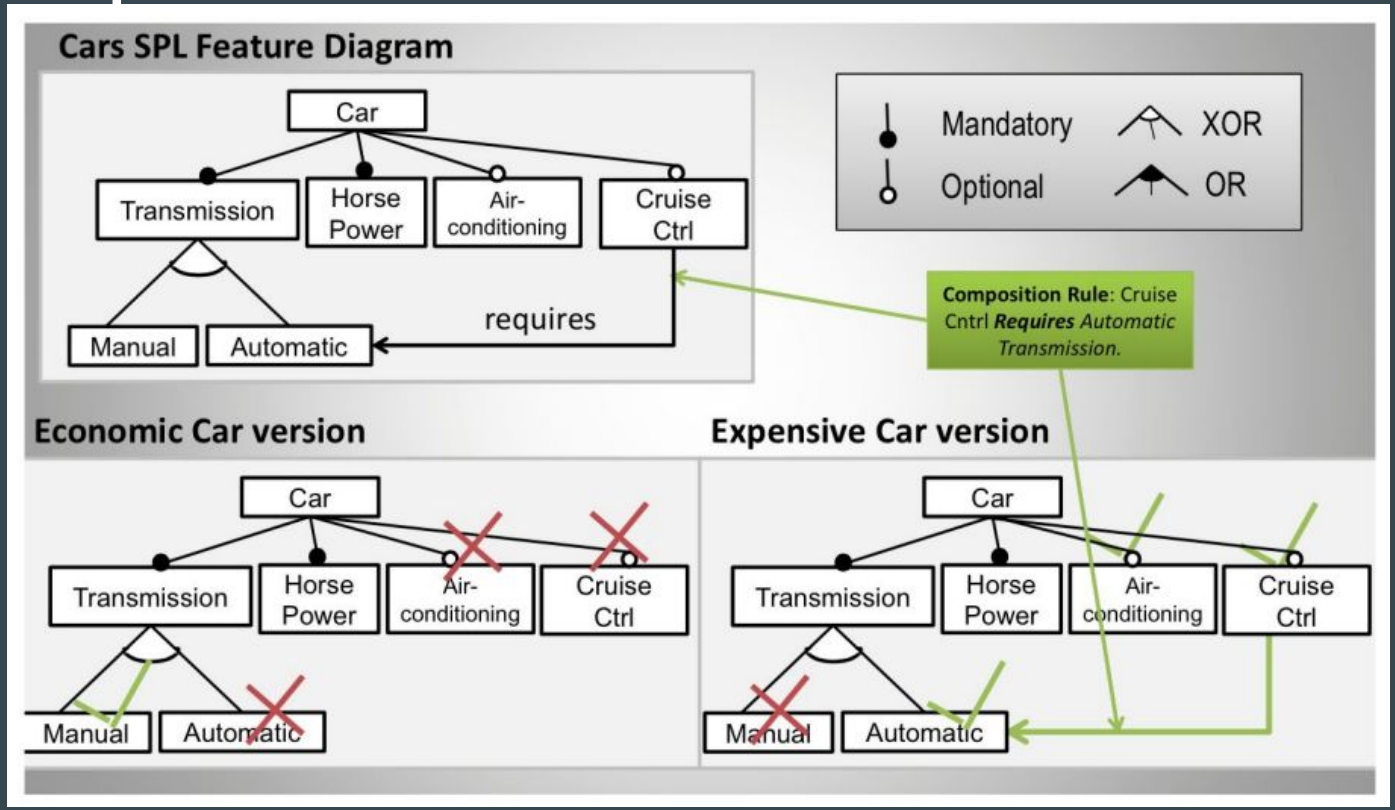
Could we represent the same with another language? YES



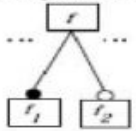
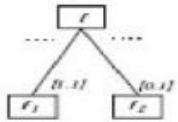
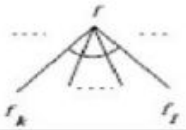
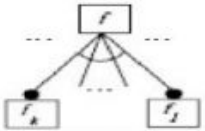
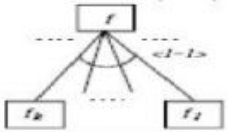
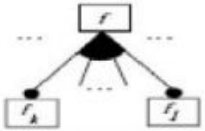
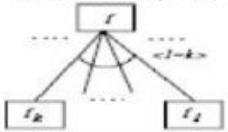
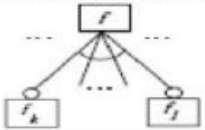
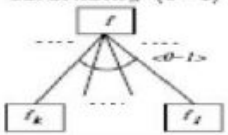
More examples



More examples



Other FM notations

FODA notation (Kang et al., 1990)	Extended notation (Czarnecki, 1998; Czarnecki and Eisenecker, 2000)	Cardinality-based notation
mandatory and optional subfeatures 	*mandatory and optional subfeatures* 	*mandatory and optional subfeatures* 
alternative subfeatures 	*exclusive-or group* 	*group with cardinality* ⟨1 – 1⟩ 
n/a	*inclusive-or group* 	*group with cardinality* ⟨1 – k⟩ 
n/a	*exclusive-or group with optional subfeatures* 	*group with cardinality* ⟨0 – 1⟩ 

How do we formalize the Domain Model?

Domain Model - Textual description - 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

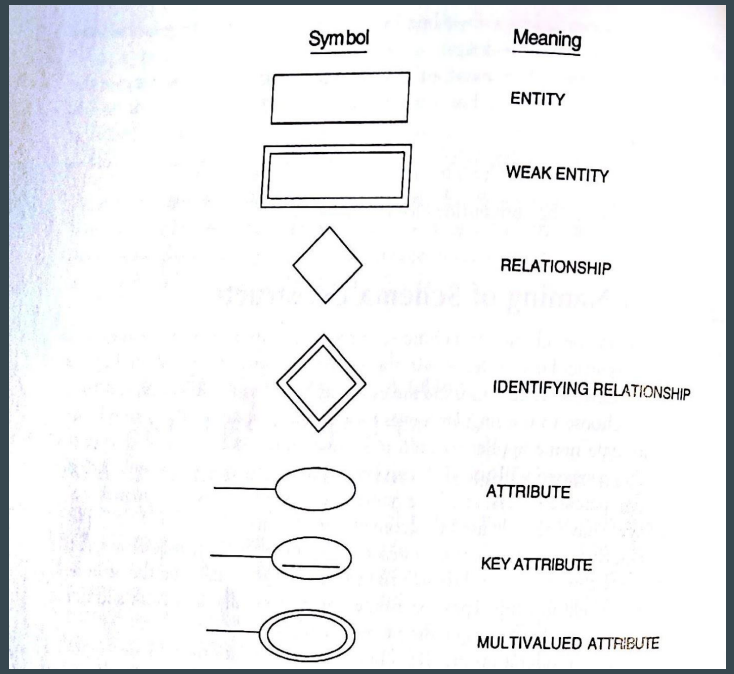
Entity-Relationship Diagrams (ERD) + Data Flow Diagrams (DFD)

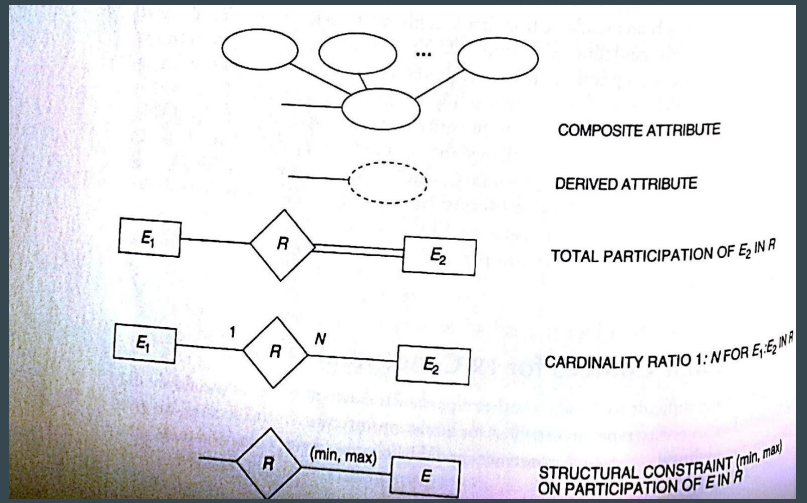
ERD - Developed by Peter Chen in the 70's. Help to describe inter-related things of interest in a specific domain of knowledge.

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

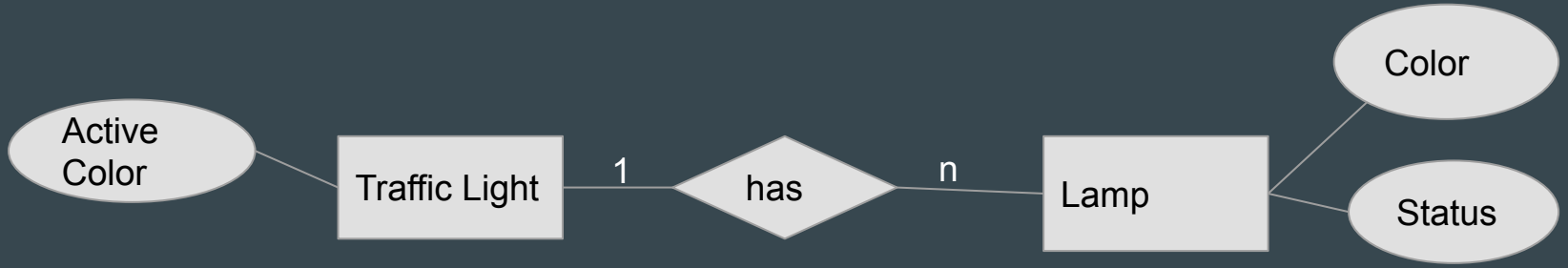
- how the system will operate,
- what the system will accomplish,
- and how the system will be implemented.

ERD

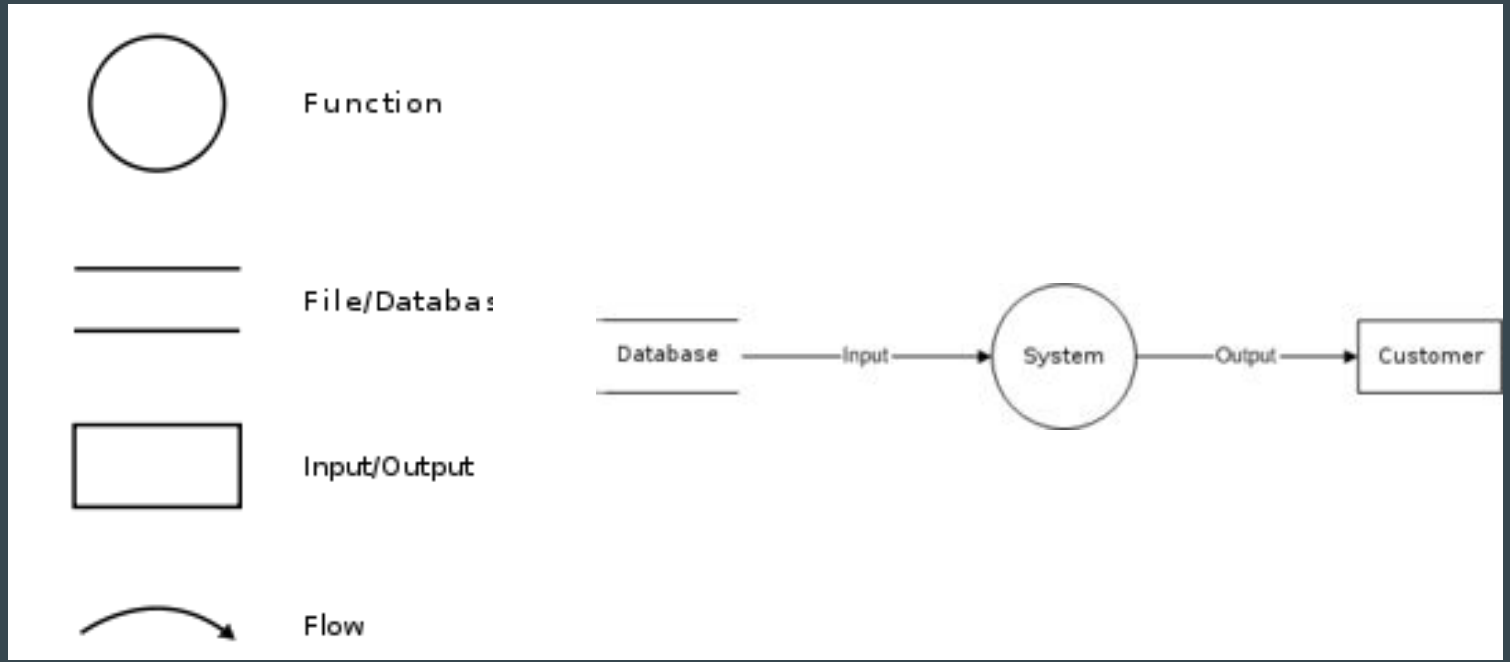

Symbol
Meaning
ENTITY
WEAK ENTITY
RELATIONSHIP
IDENTIFYING RELATIONSHIP
ATTRIBUTE
KEY ATTRIBUTE
MULTIVALUED ATTRIBUTE


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

Traffic Lights - ERD

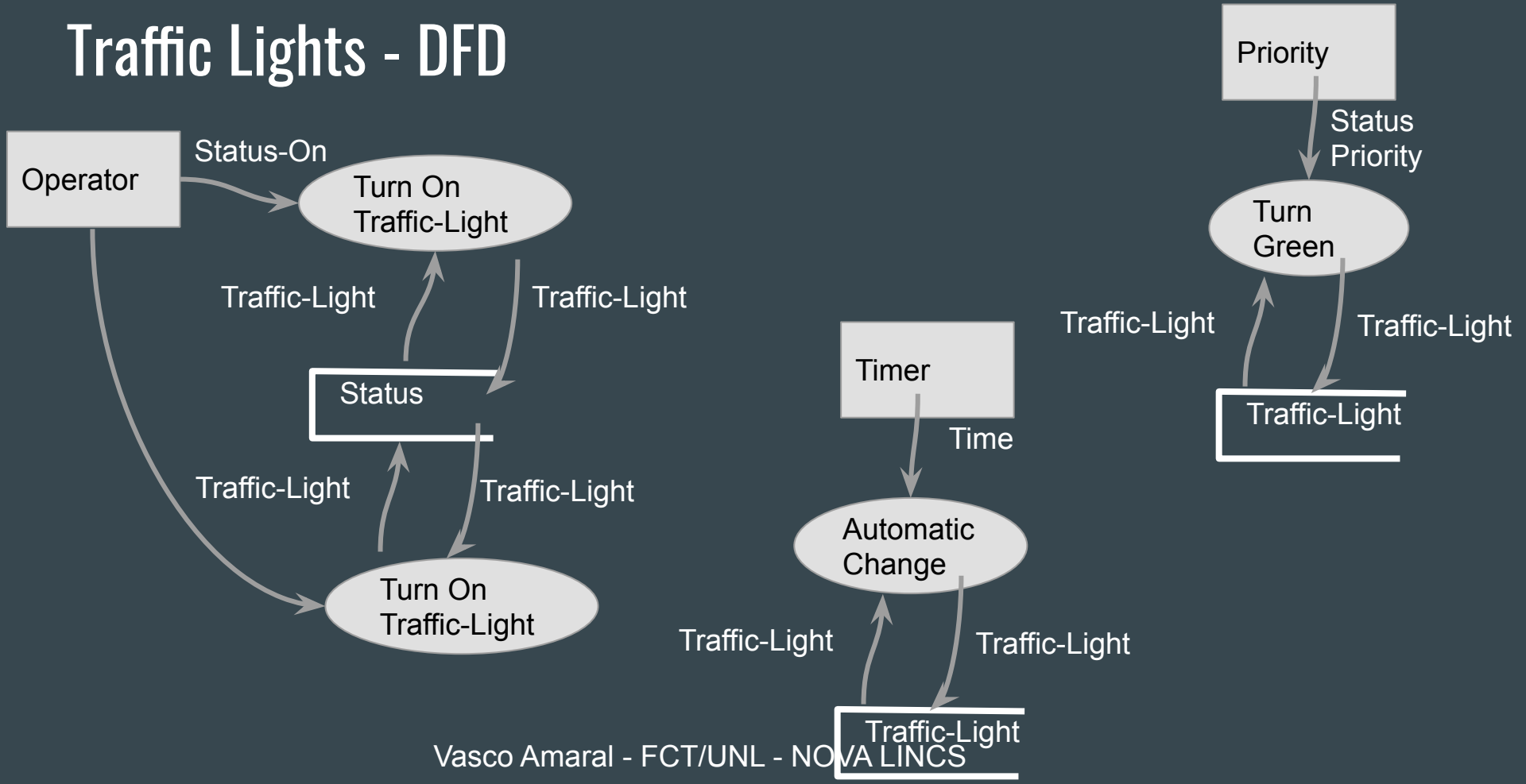


DFD



Taken from wikipedia

Traffic Lights - DFD



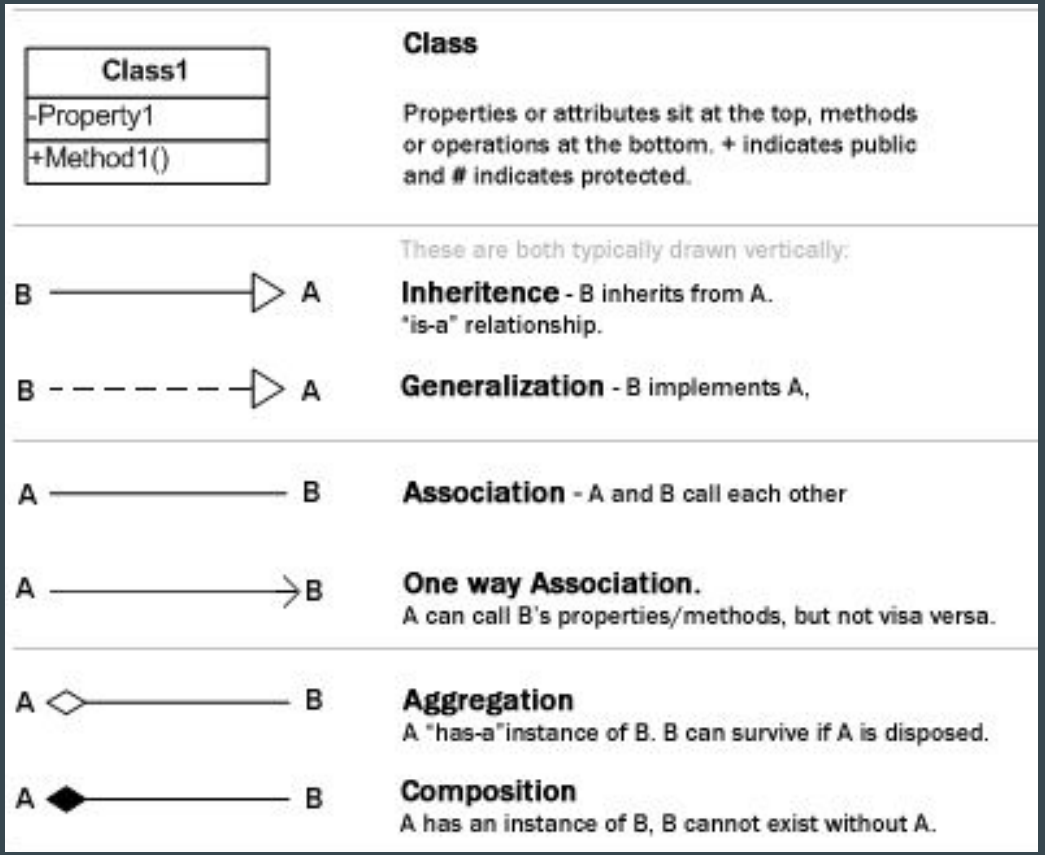
UML Class Diagrams + Statecharts + OCL

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

State Charts - Proposed by David Harel

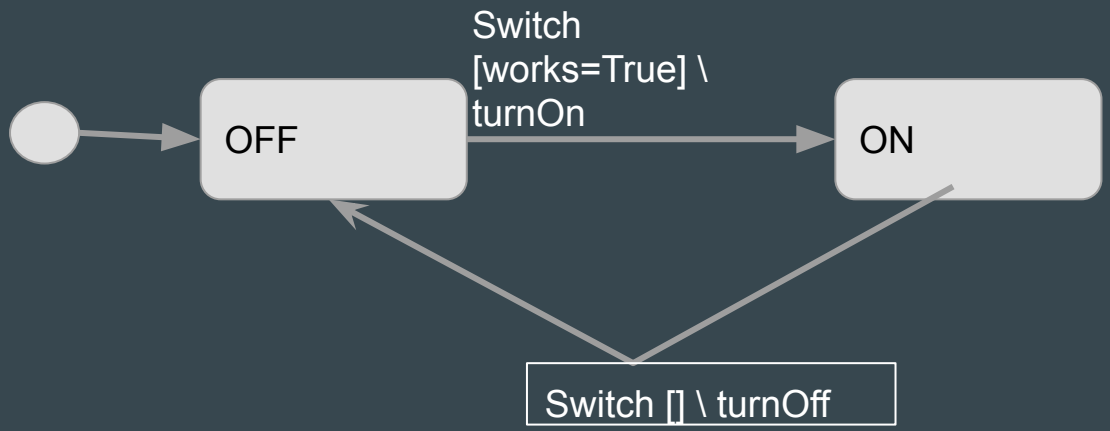
OCL - Proposed by Jos Warmer and Anneke Kleppe

Class diagrams

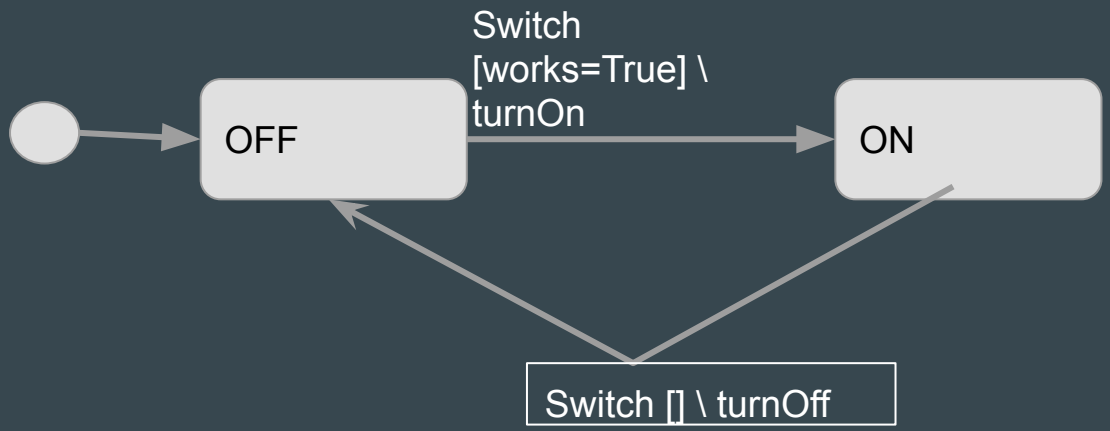


Class Diagrams

Statecharts



Statecharts



OCL Constraints

Context Traffic-Light

inv Only-One Color:

self.Light (l | l.status = On)->count() = 1

What is an Ontology?

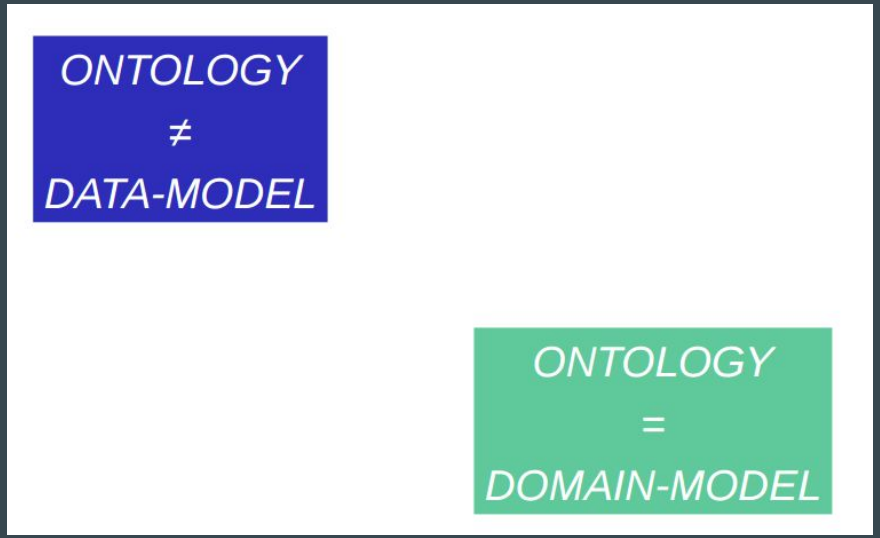
A model that represents knowledge

as a set of concepts within a given domain

and the relationships between these concepts.

It is a specification of a conceptualization in

the context of knowledge description.



Domain Modeling with ontologies

Advantage: ability to reason
(make inferences)

- Giraffe only eat leaves
- Leaves are parts of trees
- Trees are plants
- Plants and parts of plants are disjoint with animals and parts of animals
- Vegetarians eat all things that are not animals nor parts of animals

=> Giraffes are Vegetarians

Description Logics (DL)

Represents knowledge of an Application Domain in terms of:

- Concepts (unary predicates/formulae with one free variable)
  E.g., Person, Doctor, HappyParent, (Doctor t Lawyer)
- Roles (binary predicates/formulae with two free variables)
  E.g., hasChild, loves, (hasBrother ± hasDaughter)
- Individuals (constants)
  E.g., John, Mary, Italy
- Operators (for forming concepts and roles) restricted so that:
  Satisfiability/subsumption is decidable and, if possible, of low complexity

Description Logics (DL)

Axioms are of the types:

Terminology Box (TBox) - intensional Knowledge or collection of declarations (Axioms) with general properties of concepts and roles. Uses IS-A relationship in between concepts.

E.g. Birds are animals: “Bird $\subseteq$ Animal”

REASONING:

Satisfiability checking: if a concept is satisfiable (non contradictory).

Subsumption checking: if one description is more general than another.

Description Logics (DL)

Assertional Box (ABox) - extensional knowledge - expresses knowledge about the individuals

E.g. Tweety is a bird: TWEETY:Bird

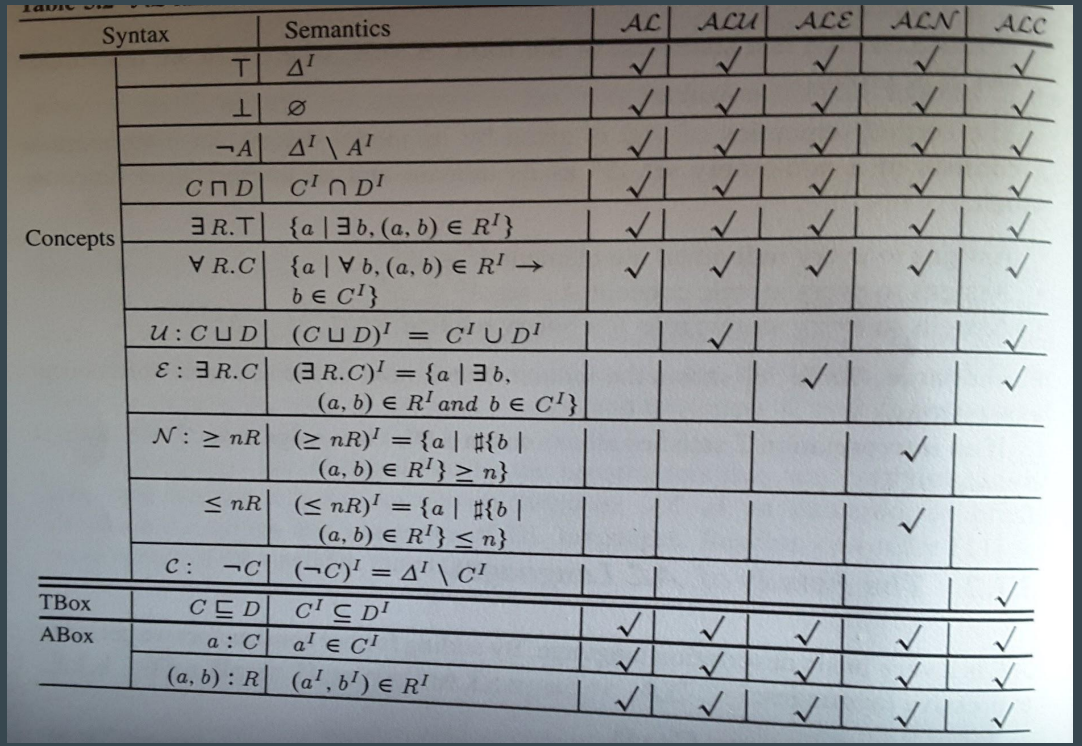
Alice is mother of TOM: hasMother (TOM, ALICE)

REASONING:

Consistency checking: If there is a model for the Ontology

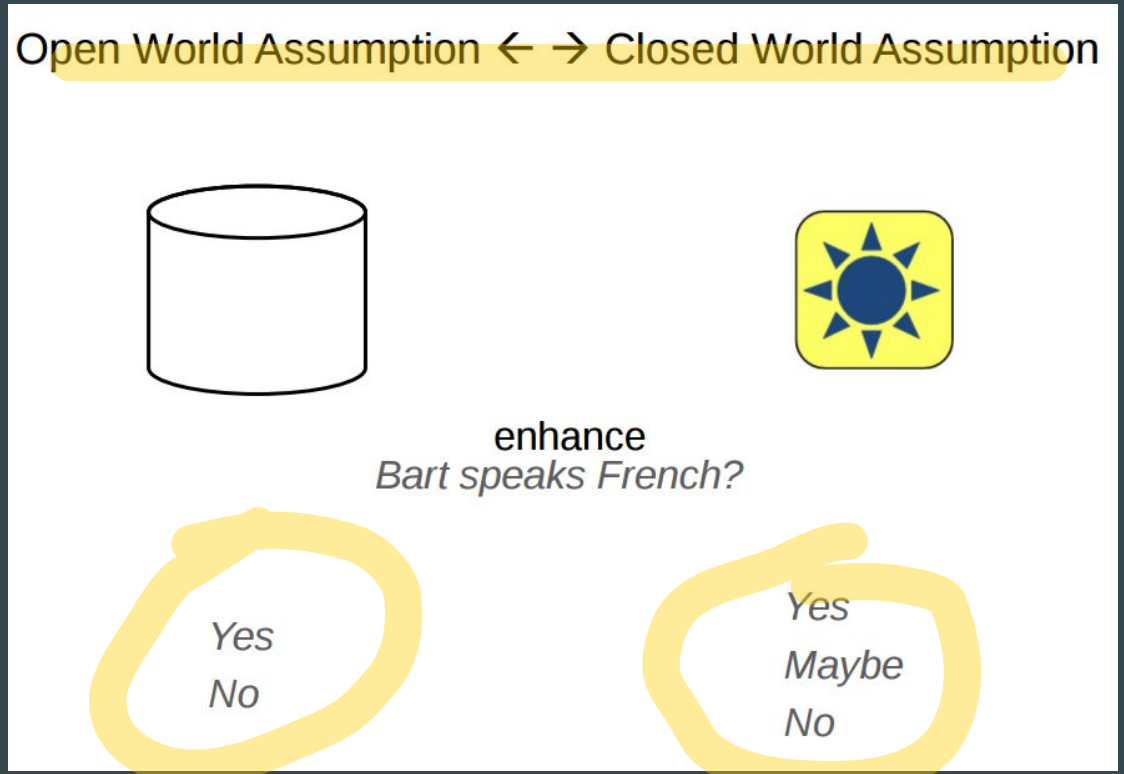
Instance retrieval: if a particular individual is an instance of a given concept.

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



Open World vs. Closed world assumption

A and B are concepts in TBox , if we declare A(x) x as Instance of A

Closed World - we can conclude that x is instance of A and not instance of B

Open world - we can conclude that x is instance of A and regarding B we don't know, unless we declare not(B(x))

- Absence of information must not be valued as negative information
- No unique name assumption : differences between classes must be expressed explicitly

OWL - Web Ontology Language

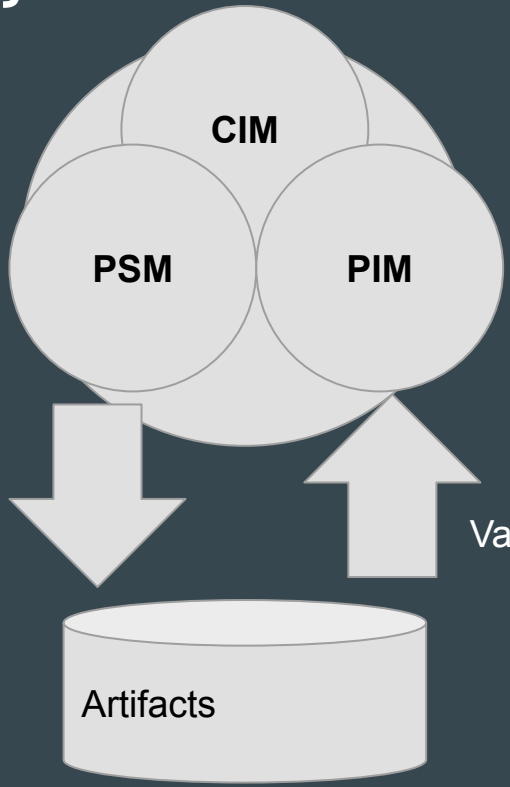
State of the art ontology language for the Semantic Web

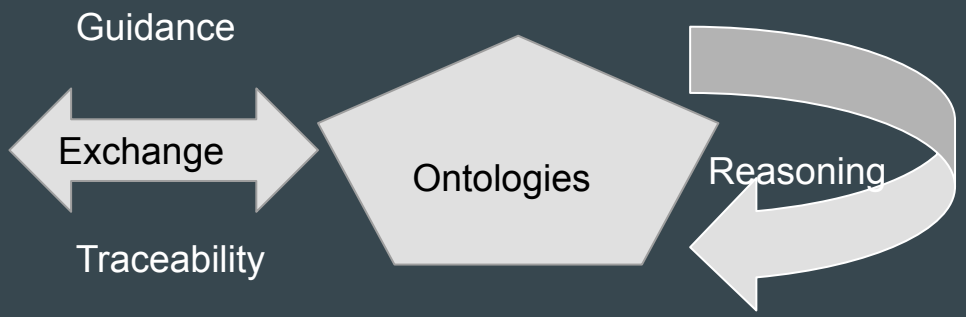
WC3 recommendation status

OWL2 is an extension of OWL

Machine readable, usable in software programs and in SOA environments

Ontology Driven Software Development





Adapted from Jeff Z. Pan et al, Ontology-Driven Software Development, Springer, 2013

Several approaches,
No holistic one,
Pick a consistent
and Adequate for a specific intention....

Summary

Domain Model

Domain Modelling

Domain Engineering

Thank you!