PMR 5020

Metodologia do Projeto de Sistemas

Aula 6: Uso de linguagens formais (no ombro de gigantes)

Prof. José Reinaldo Silva
reinaldo@usp.br

Knowledge Engineering

Edward Feigenbaum
"Pai" dos Sistemas Especialistas
Sanford University

KE is an engineering discipline that involves integrating knowledge into computer systems in order to solve complex problems normally requiring a high level of human expertise

Knowledge Engineering

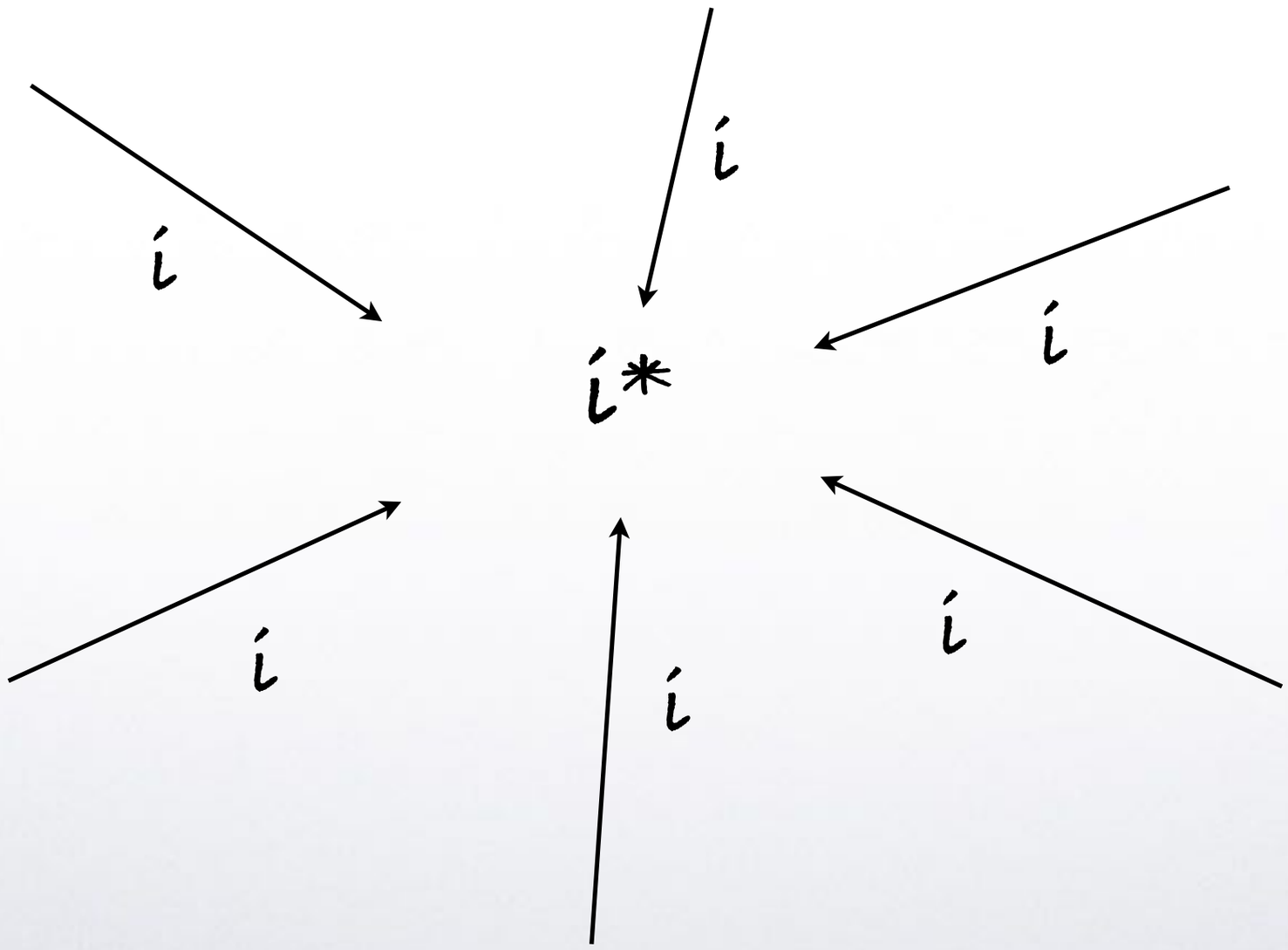
Knowledge Acquisition
Knowledge Modeling and Analysis
Knowledge Validation
Knowledge Base Building

→ KNOWLEDGE ENGINEERING

GORE: Goal Oriented Requirements Engineering

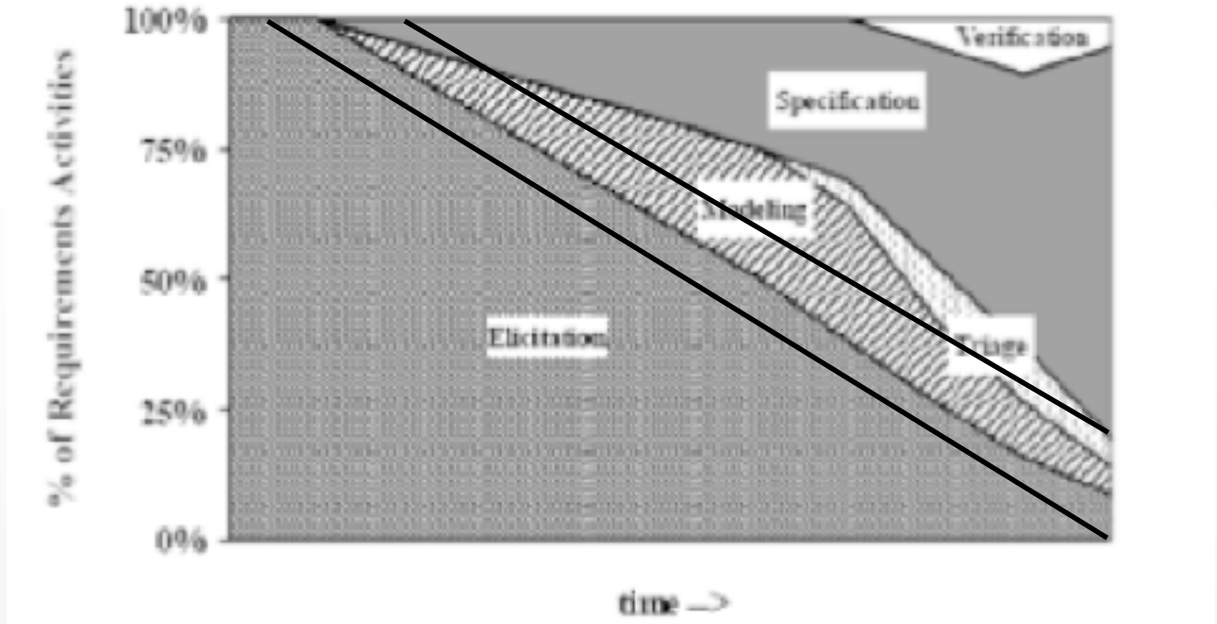
As técnicas GORE (Goal Oriented Requirements Engineering) tem como base o aumento do volume de conhecimento especialmente na fase inicial do processo de design, retirando muito do peso voltado exclusivamente para a modelagem funcional. Até o momento vimos somente o KAOS como abordagem que também aponta na direção do desenvolvimento voltado a modelos.

Uma outra possibilidade, comentada brevemente na aula passada, é o i*, criado por Eric Yu na sua tese de doutorado em 1995 (leitura da semana).


i
i
i*
i
i
i

*A essência do i**

A base do i* é que a fase inicial na verdade é composta de duas fases: uma onde o formalismo é inviável e onde se lida com as intenções dos diversos agentes com seus respectivos viewpoints, e outra que se encaixa mais diretamente nos métodos formais e na representação formal.



The connection between requirements, specification and design

Still, a great problem in this process is getting a proper documentation and (formal) representation of the design requirements, of the design rationales, further from the specifications and in the following the design of the artifact or system-to-be. A good question is how are the attributes of the different candidate languages for this process and how could we manage to select the proper one given the project characteristics.

Follow the yellow brick road!

É sem dúvida muito mais fácil trabalhar - especialmente quando alguma criatividade é demandada - quando se tem uma boa referência do que deve ser feito e como. É o que acontece com o formalismo, especialmente com as linguagens e representações formais.

Algumas linguagens conhecidas

PSL – Program Statment Language
U. Michigam, ISDOS Inc.

SADT – Structured Analysis and Design Technique
MIT

EDDA - (formalização do SADT e comercialização)

Algumas linguagens conhecidas

SAAM – Systematic Activity Modelling Method

Boeging Computer Services Co.

HOS – High Order Software

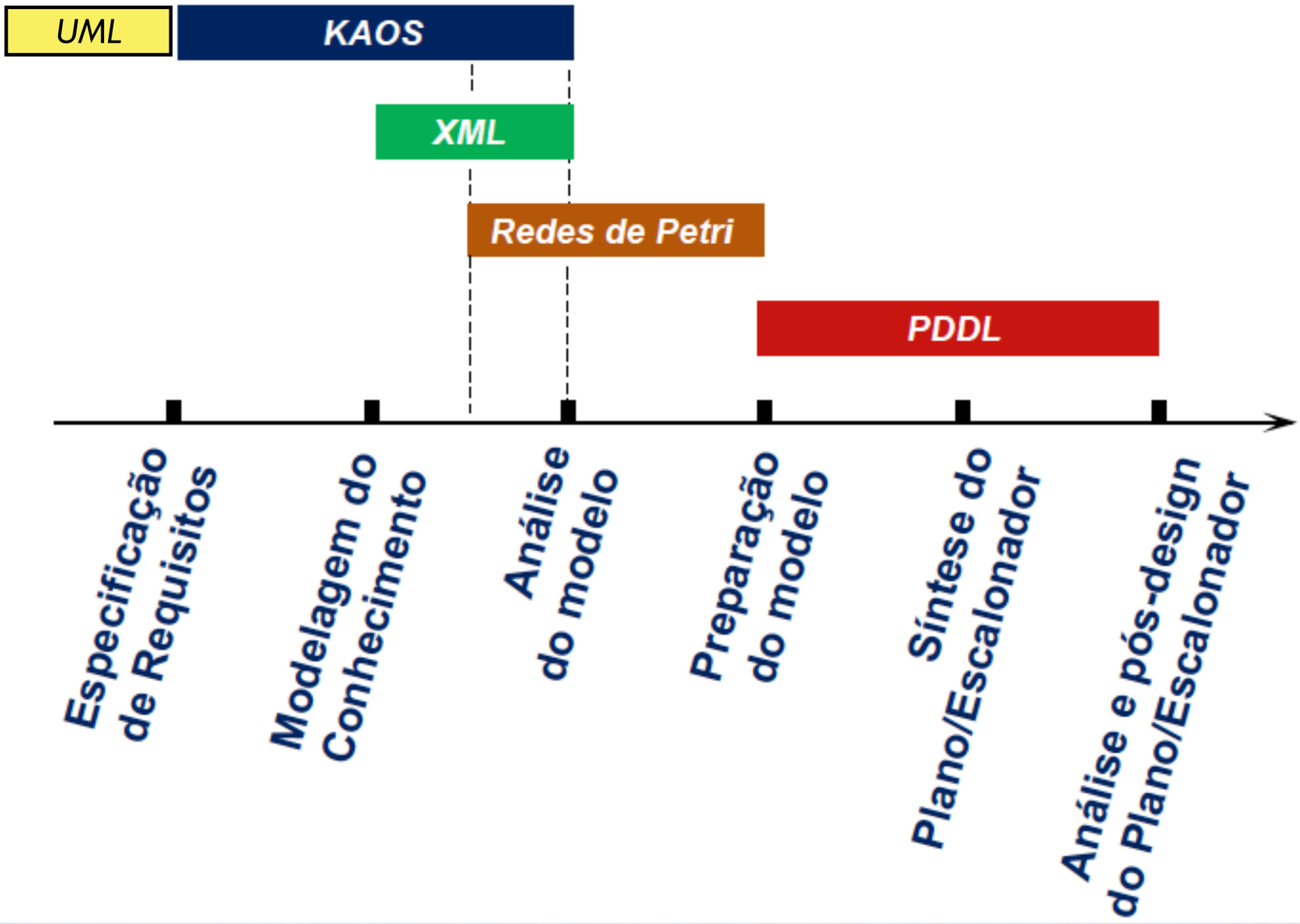
High Order Software Inc.

No.	Attribute	Values
1	paradigm	state machine, algebra, process algebra, trace
2	formality	informal, semi-formal, formal
3	graphical representation	yes, no
4	object-oriented	yes, no
5	concurrency	yes, no
6	executability	yes, no
7	usage of variables	yes, no
8	non-determinism	yes, no
9	logic	yes, no
10	provability	yes, no
11	model checking	yes, no
12	event inhibition	yes, no

method name	paradigm	formality	graphical representation	object-oriented
Action Systems	state transition	formal	no	no
B	state transition	formal	no	no
CASL	algebra	formal	no	yes
Cleanroom & JSD	traces & process algebra	formal	yes	no
COQ	state transition	formal	no	no
Estelle	state transition	formal	no	no
LOTOS	process algebra	formal	no	yes
OMT & B	state transition	formal	yes	yes
Petri Nets	state transition	formal	yes	no
Petri Nets with Objects	state transition	formal	yes	yes
SART	state transition	informal & semi-formal	yes	no
SAZ	state transition	semi-formal & formal	yes	no
SCCS	process algebra	formal	no	no
SDL	state transition	formal	yes	yes
UML	state transition	informal & semi-formal	yes	yes
VHDL	state transition	formal	no	no
Z	state transition	formal	no	no

method name	concurrency	executability	usage of variables	non-determinism
Action Systems	no	yes	yes	yes
B	no	yes	yes	yes
CASL	no	yes	yes	no
Cleanroom & JSD	no	yes	yes	yes
COQ	no	yes	yes	yes
Estelle	yes	yes	yes	no
LOTOS	yes	yes	yes	yes
OMT & B	no	yes	yes	yes
Petri Nets	yes	yes	no	yes
Petri Nets with Objects	yes	yes	yes	yes
SART	yes	no	no	yes
SAZ	no	yes	yes	yes
SCCS	yes	yes	yes	yes
SDL	yes	yes	no	yes
UML	yes	no	no	no
VHDL	yes	yes	yes	no
Z	no	yes	yes	yes

Portanto a questão persiste se devemos insistir na busca por uma representação (formal) para todo o processo de design ou deixar fluir uma proposta mais eclética de ter várias representações e concentrar mais no paradigma. Seja qual for a abordagem esta deve levar a um volume maior de conhecimento em todo o processo.


UML
KAOS
XML
Redes de Petri
PDDL
Especificação de Requisitos
Modelagem do Conhecimento
Análise do modelo
Preparação do modelo
Síntese do Plano/Escalonador
Análise e pós-design do Plano/Escalonador

Tropos in Perspective

Early requirements

Late requirements

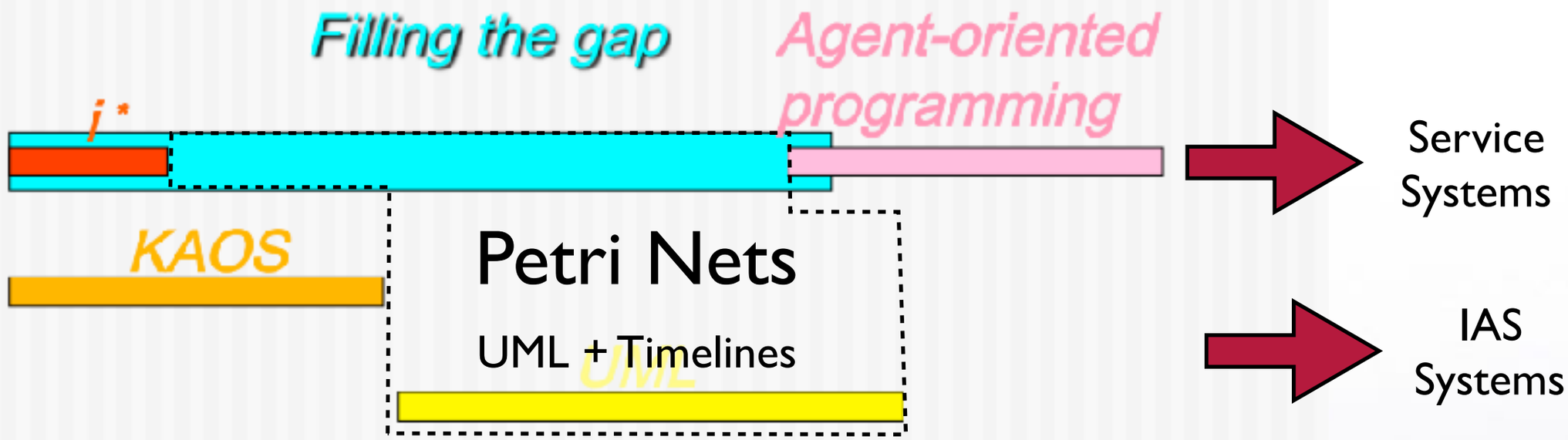
Architectural design

Detailed design

Implementation

A knowledge level methodology

Tropos in Perspective

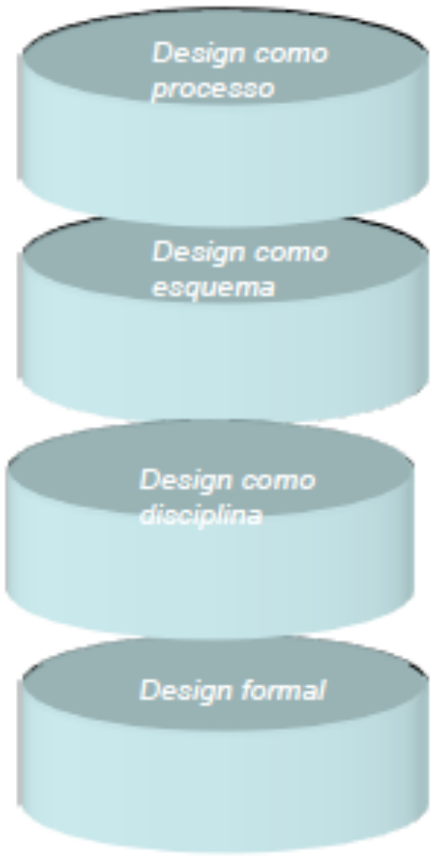


A knowledge level methodology

O que está faltando?

Temas de pesquisa

1. *Reutilização de designs*
2. *Escalabilidade e representação esquemática de sistemas complexos*
3. *Análise de Requisitos*
4. *Aplicações*



reutilização

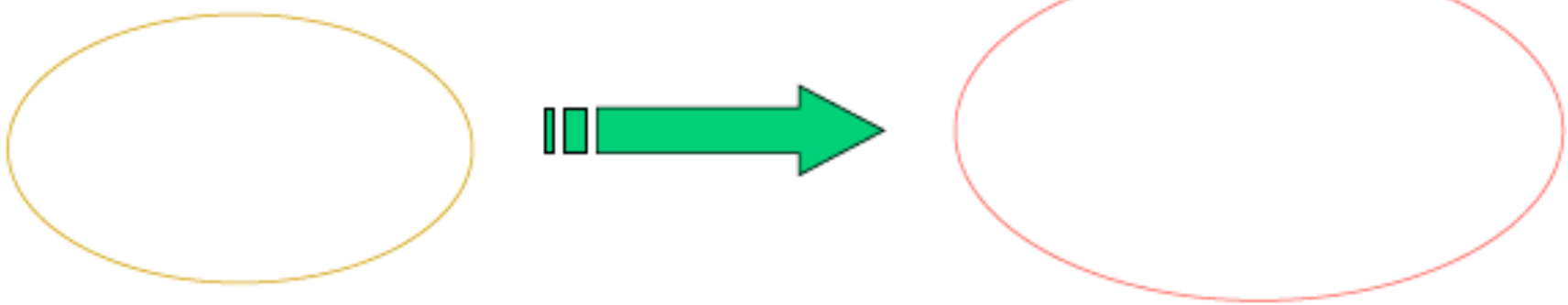
- *experiência*
- *eficiência*
- *regularidade*
- *consistência*

Transferência semântica: metáforas

1. *Reutilização de designs*
2. *Escalabilidade e representação esquemática de sistemas complexos*
3. *Análise de Requisitos*
4. *Aplicações*

Domínio alvo

Domínio fonte



Transferência semântica: metáforas

1. *Reutilização de designs*
2. *Escalabilidade e representação esquemática de sistemas complexos*
3. *Análise de Requisitos*
4. *Aplicações*

1. Achar uma componente reutilizável

2. Garantir a eficiência (reusar X refazer)

3. Fazer a transferência da componente reutilizável para o domínio da componente em desenvolvimento

4. Checar consistência do novo dominio.

Abordagem em lógica de predicados de 1a. ordem

[Bipin Indurkya, 1987]Elementos para uma teoria formal de metáforas

Def 1.] Um domínio é uma tuple <V, f, Sd, S> onde

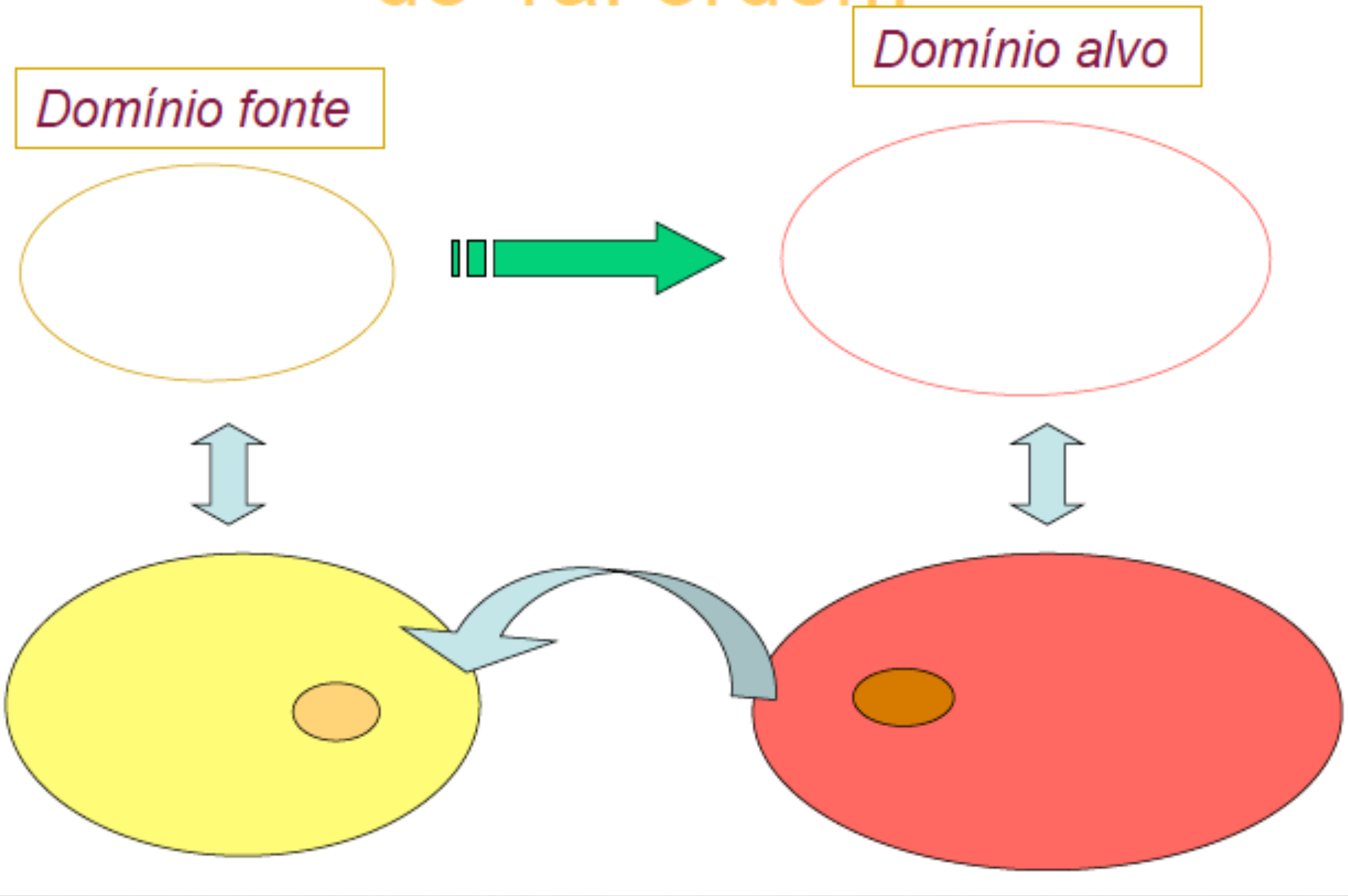
V é um conjunto de símbolos
f : V → W é um mapeamento sobre um conjunto de tipos W
Sd é um conjunto de derivações que definem termos, e
S é um conjunto de fbf sobre o vocabulário V =<V, f >

Abordagem em lógica de predicados de 1a. ordem

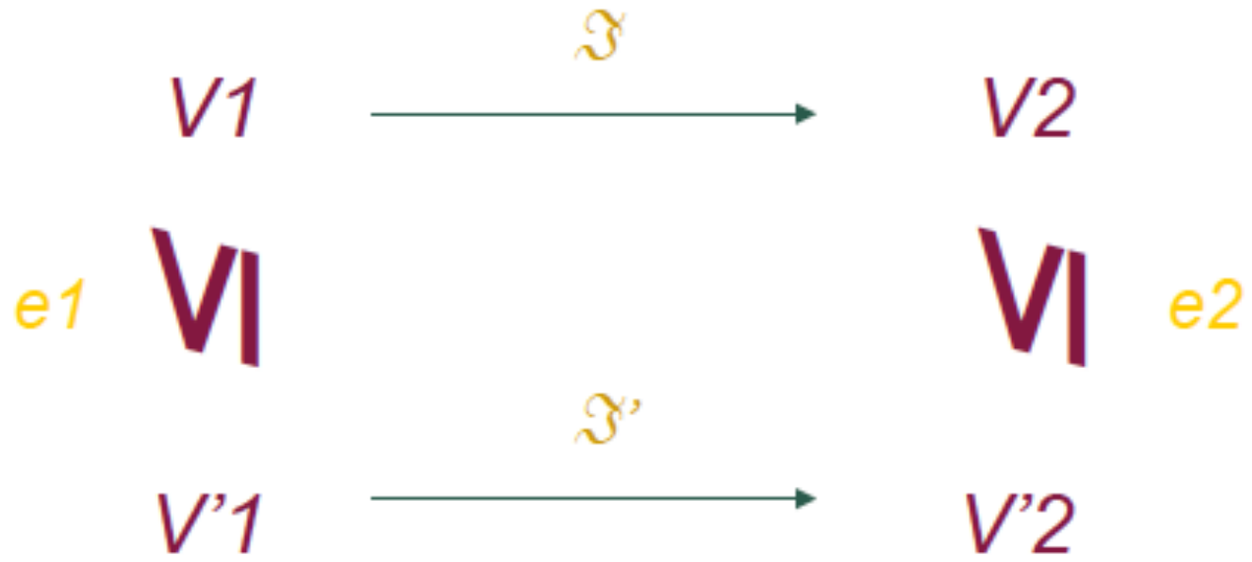
Def.2] Dados dois domínios, D1 = <V1, f1, Sd1, S1> e D2 = <V2, f2, Sd2, S2> e um mapeamento admissível f : V1 ® V2. Uma metáfora é um par <f, S > onde S ⊆ S1 é um conjunto de sentenças transformáveis.

Def. 3] Uma metáfora <f, S > é dita coerente se S"=f(S) ∪ S2 é consistente.

Abordagem em lógica de predicados de 1a. ordem



Abordagem utilizando interpretação entre teorias

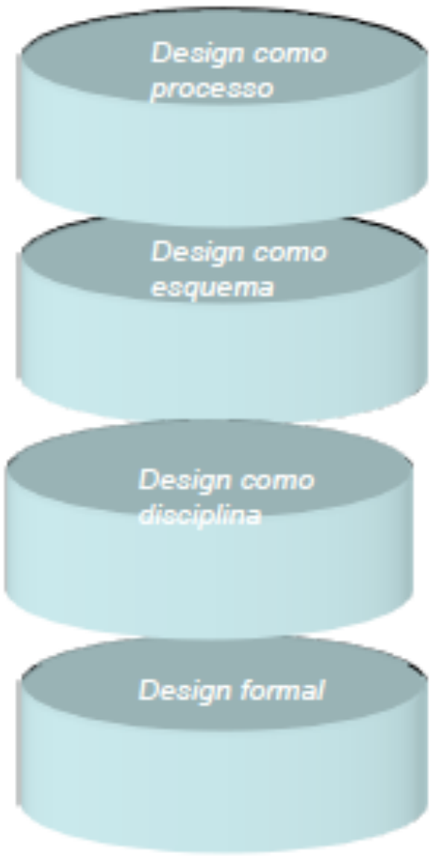


Uma metáfora é uma n-upla <V1, V2, ℑ, ℑ-1, e2>

O que está faltando?

Temas de pesquisa

1. *Reutilização de designs*
2. *Escalabilidade e representação esquemática de sistemas complexos*
3. *Análise de Requisitos*
4. *Aplicações*



reutilização

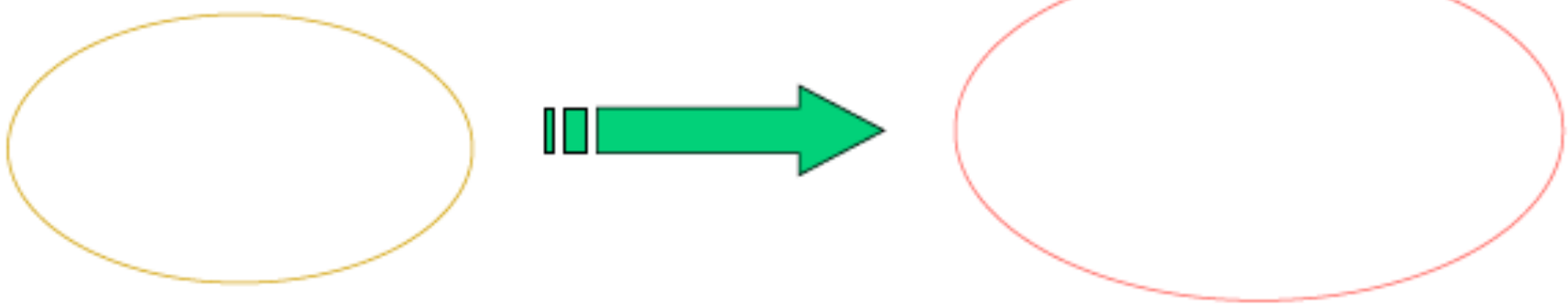
- *experiência*
- *eficiência*
- *regularidade*
- *consistência*

Transferência semântica: metáforas

1. *Reutilização de designs*
2. *Escalabilidade e representação esquemática de sistemas complexos*
3. *Análise de Requisitos*
4. *Aplicações*

Domínio alvo

Domínio fonte



Transferência semântica: metáforas

1. *Reutilização de designs*
2. *Escalabilidade e representação esquemática de sistemas complexos*
3. *Análise de Requisitos*
4. *Aplicações*

1. Achar uma componente reutilizável

2. Garantir a eficiência (reusar X refazer)

3. Fazer a transferência da componente reutilizável para o domínio da componente em desenvolvimento

4. Checar consistência do novo dominio.

Abordagem em lógica de predicados de 1a. ordem

[Bipin Indurkya, 1987]Elementos para uma teoria formal de metáforas

Def 1.] Um domínio é uma tuple <V, f, Sd, S> onde

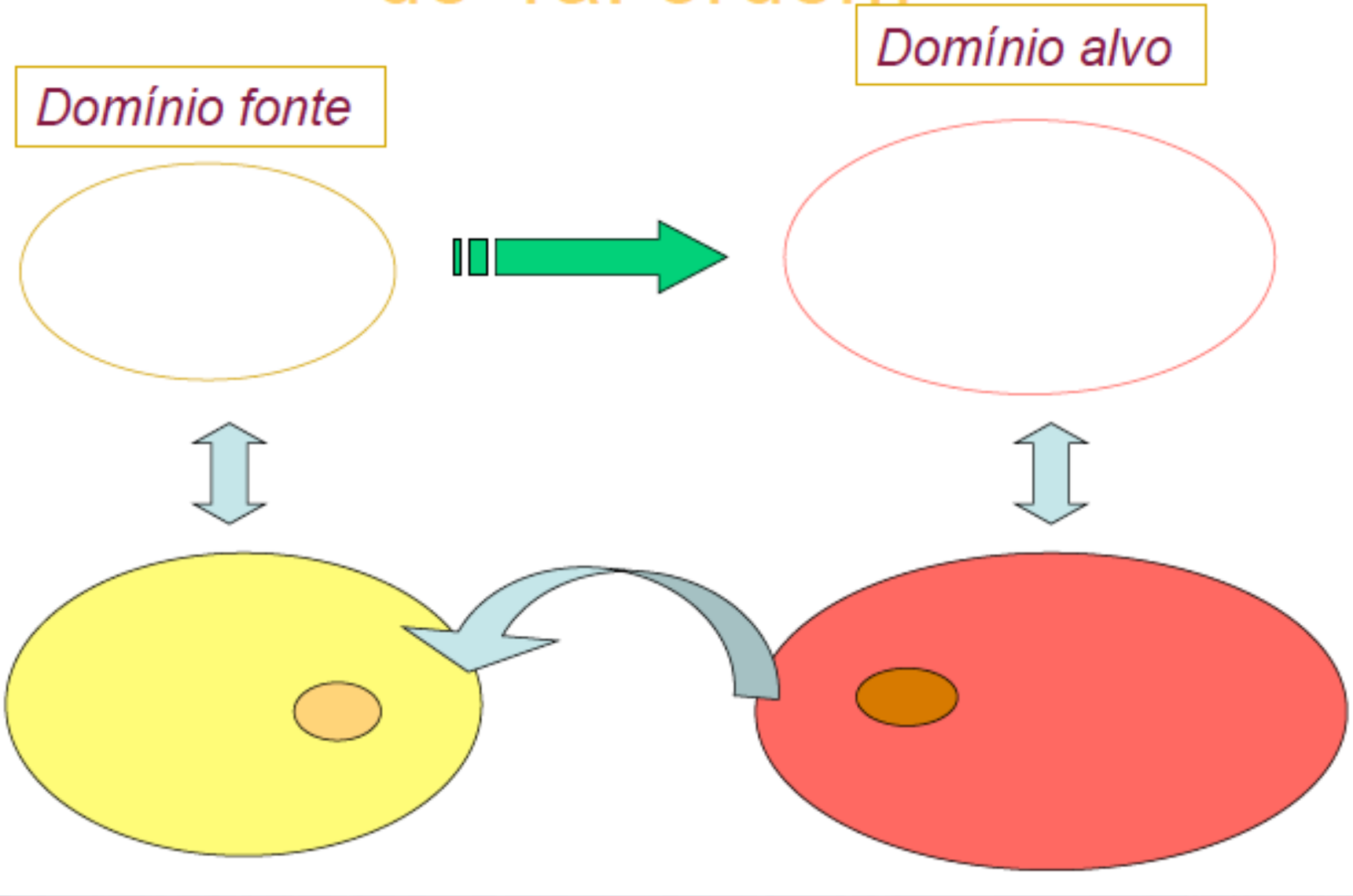
V é um conjunto de símbolos
f : V → W é um mapeamento sobre um conjunto de tipos W
Sd é um conjunto de derivações que definem termos, e
S é um conjunto de fbf sobre o vocabulário V =<V, f >

Abordagem em lógica de predicados de 1a. ordem

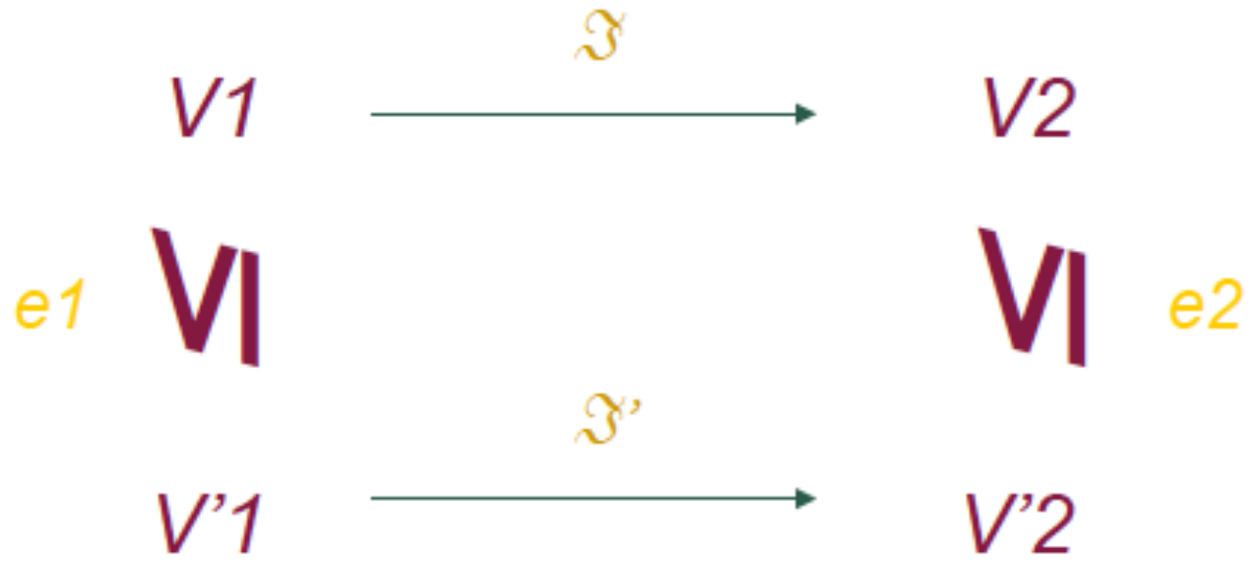
Def.2] Dados dois domínios, D1 = <V1, f1, Sd1, S1> e D2 = <V2, f2, Sd2, S2> e um mapeamento admissível f : V1 ® V2. Uma metáfora é um par <f, S > onde S ⊆ S1 é um conjunto de sentenças transformáveis.

Def. 3] Uma metáfora <f, S > é dita coerente se S"=f(S) ∪ S2 é consistente.

Abordagem em lógica de predicados de 1a. ordem

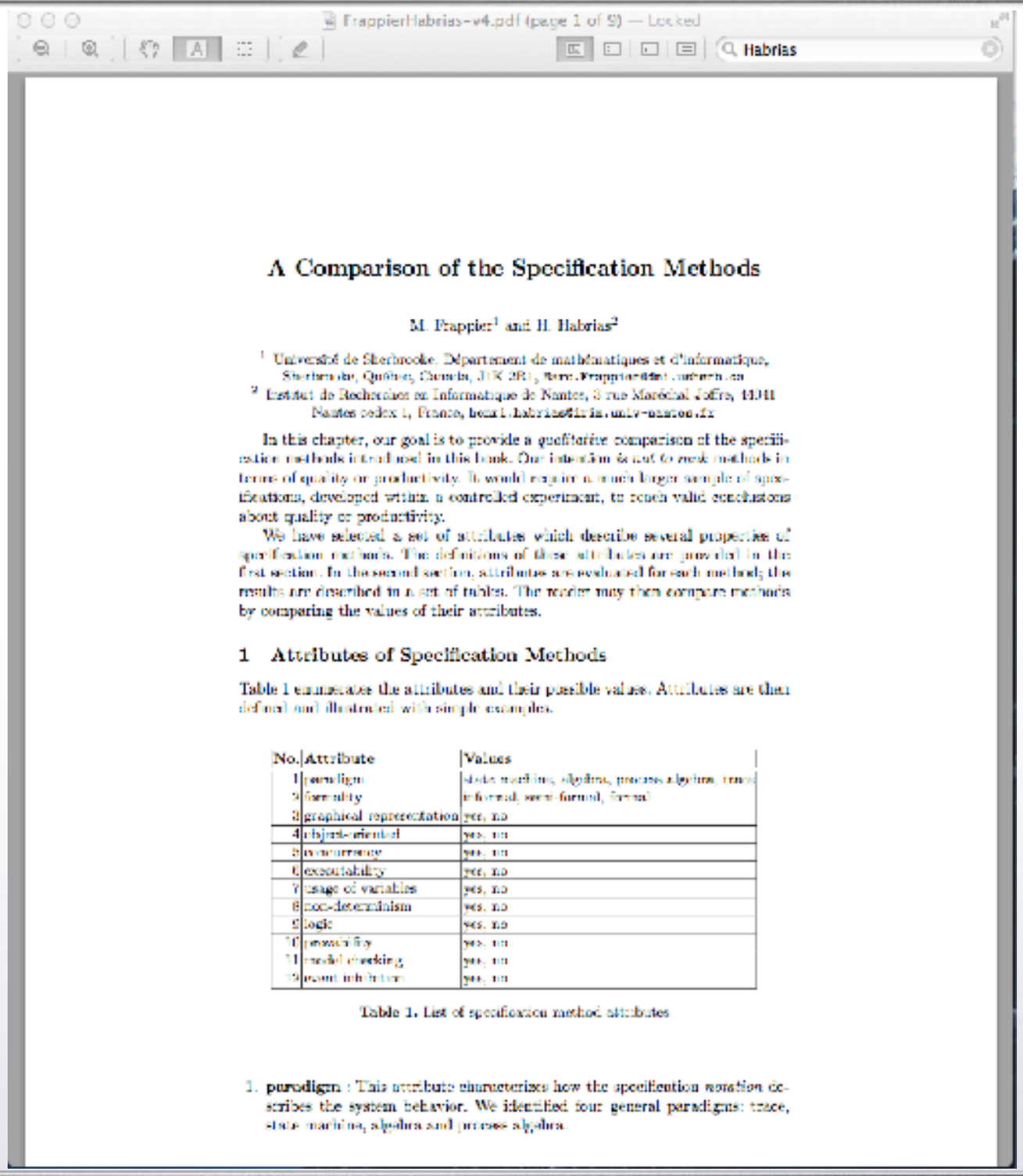


Abordagem utilizando interpretação entre teorias



Uma metáfora é uma n-upla <V1, V2, ℑ, ℑ-1, e2>

Leitura da Semana



FrappierHabrias-v4.pdf (page 1 of 9) — Locked

Habrias

A Comparison of the Specification Methods

M. Frappier[1] and H. Habrias[2]

[1] Université de Sherbrooke, Département de mathématiques et d'informatique, Sherbrooke, Québec, Canada, J1K 2R1, marc.frappier@dmi.usherb.ca
[2] Institut de Recherches en Informatique de Nantes, 2 rue Maréchal Joffre, 44041 Nantes cedex 1, France, henri.habrias@irin.univ-nantes.fr

In this chapter, our goal is to provide a qualitative comparison of the specification methods introduced in this book. Our intention is not to rank methods in terms of quality or productivity. It would require a much larger sample of specifications, developed within a controlled experiment, to reach valid conclusions about quality or productivity.

We have selected a set of attributes which describe several properties of specification methods. The definitions of these attributes are provided in the first section. In the second section, attributes are evaluated for each method; the results are described in a set of tables. The reader may then compare methods by comparing the values of their attributes.

1 Attributes of Specification Methods

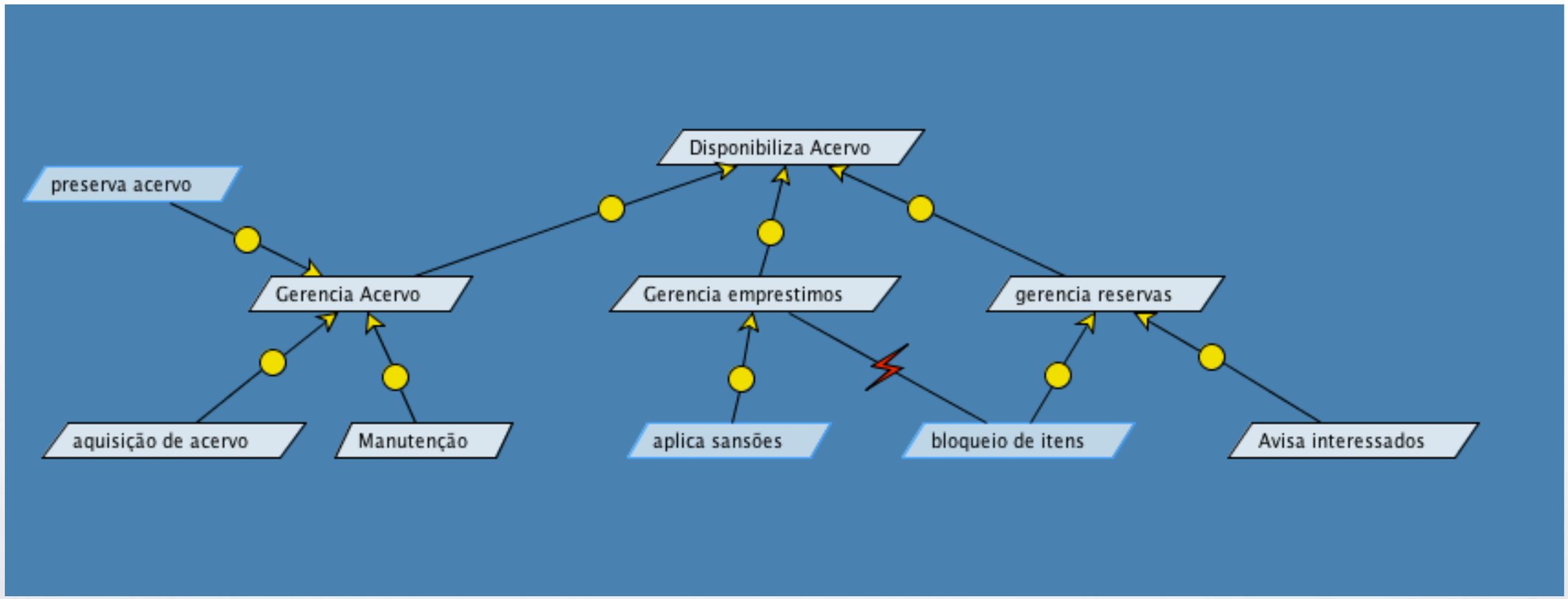
Table 1 enumerates the attributes and their possible values. Attributes are then defined and illustrated with simple examples.

No.	Attribute	Values
1	paradigm	state machine, algebra, process algebra, trace
2	formality	informal, semi-formal, formal
3	graphical representation	yes, no
4	object-oriented	yes, no
5	concurrency	yes, no
6	executability	yes, no
7	usage of variables	yes, no
8	non-determinism	yes, no
9	logic	yes, no
10	genericity	yes, no
11	model checking	yes, no
12	event inhibition	yes, no

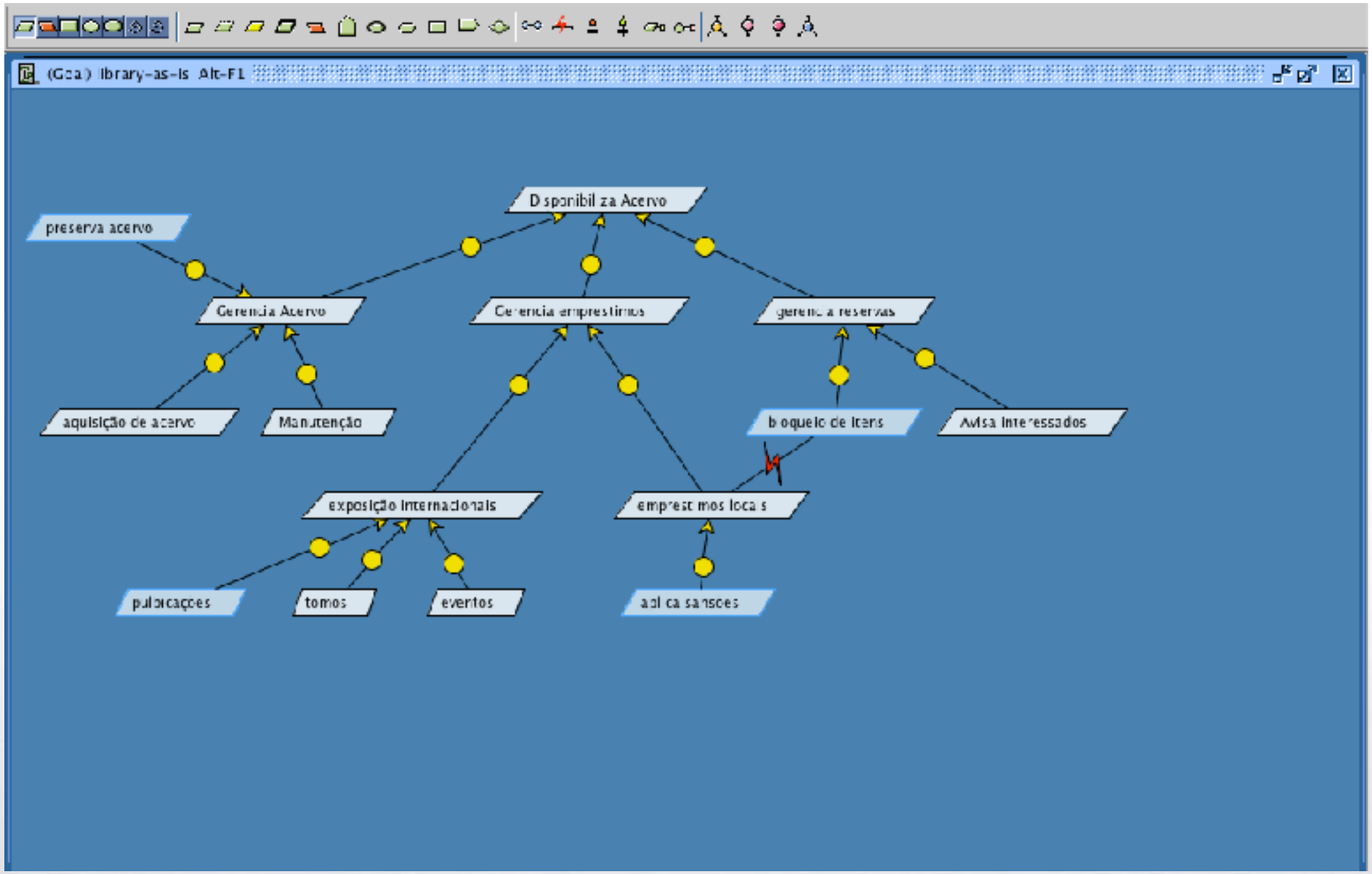
Table 1. List of specification method attributes

1. **paradigm** : This attribute characterizes how the specification notation describes the system behavior. We identified four general paradigms: trace, state machine, algebra and process algebra.

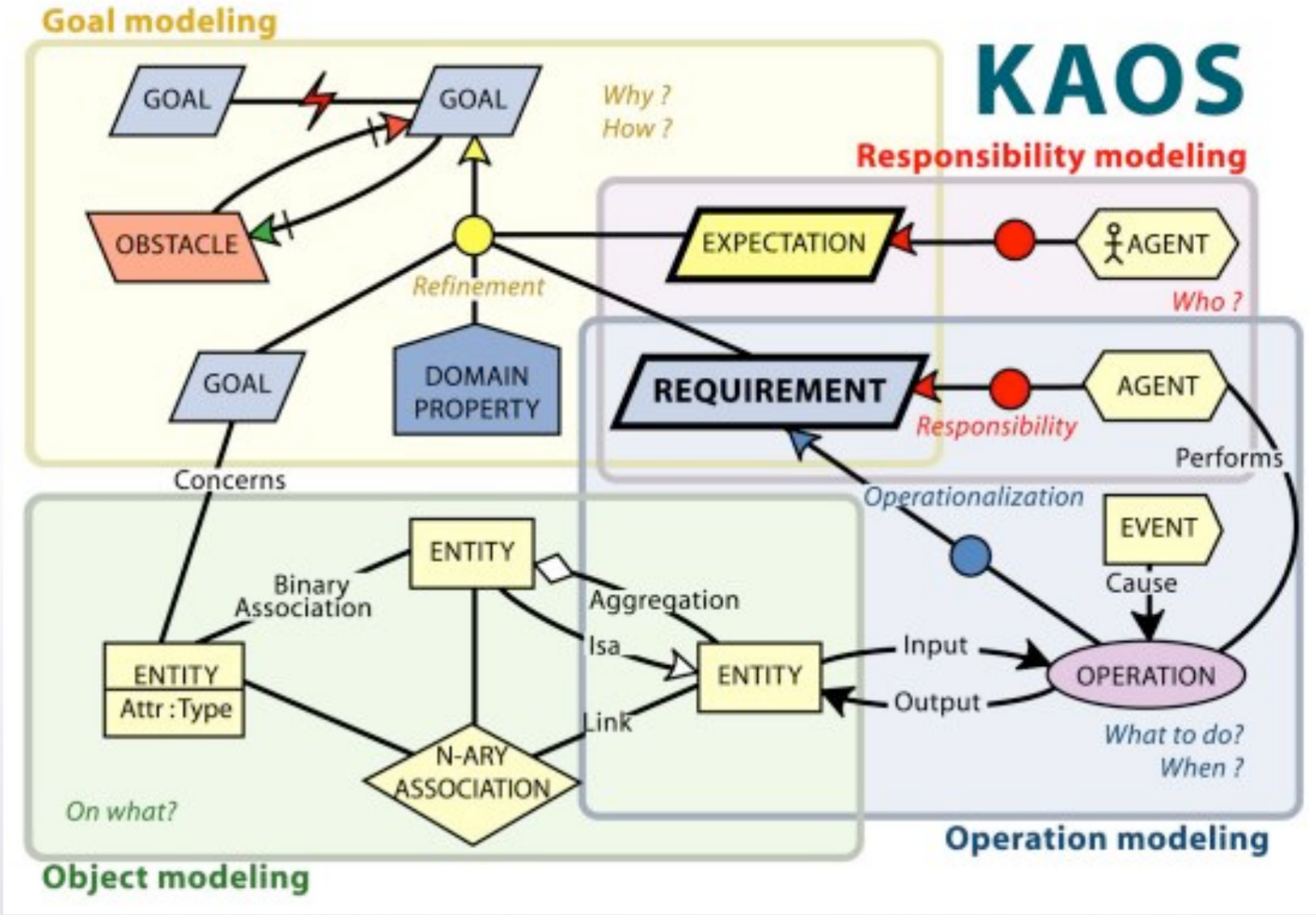
Modelando o sistema de bibliotecas: system-as-is



Modelando o system-as-is



KAOS metamodel



Obrigado

Reinaldo