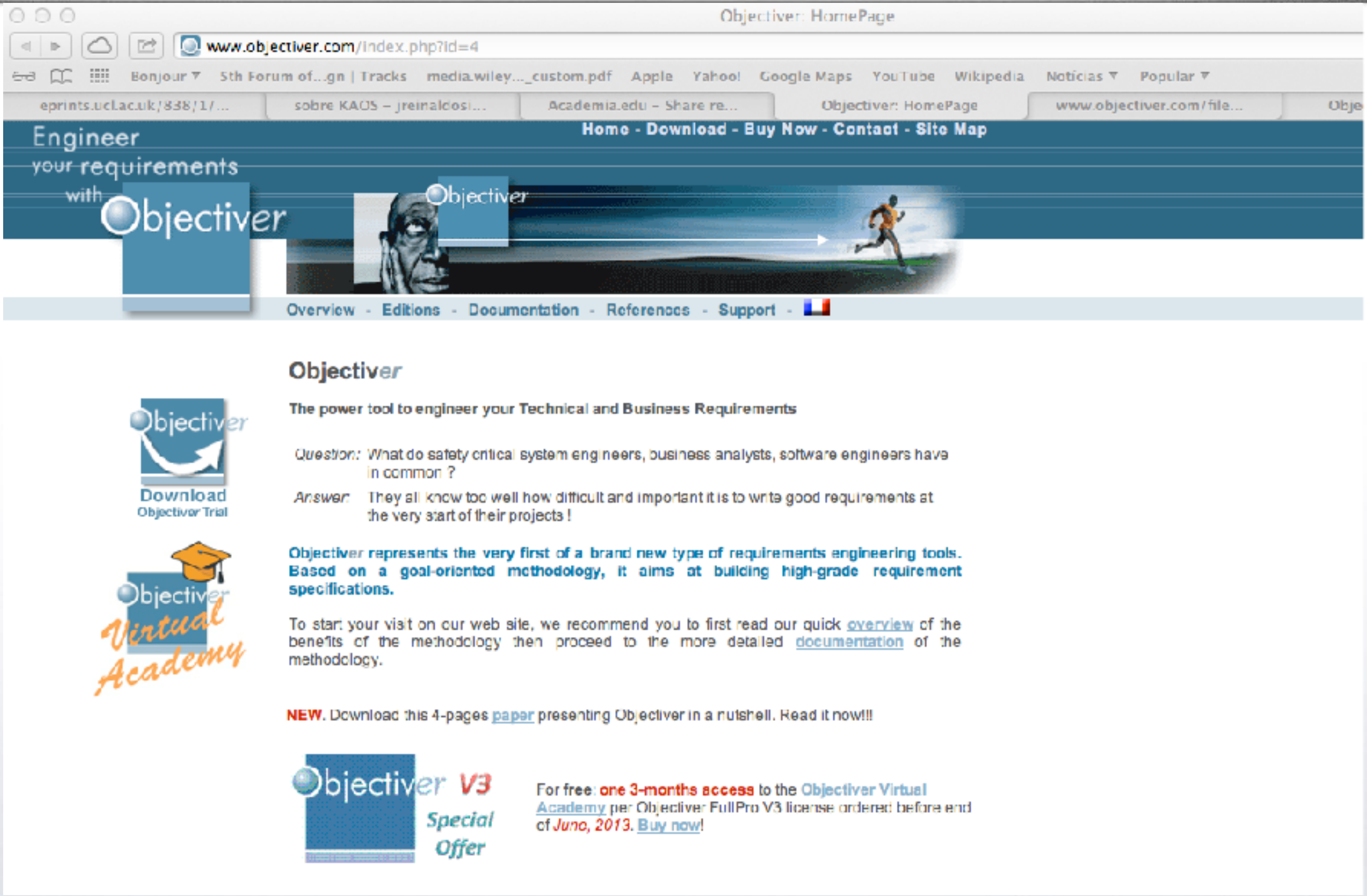
PMR 5020

Metodologia do Projeto de Sistemas

Aula 6: Requisitos & Conhecimento

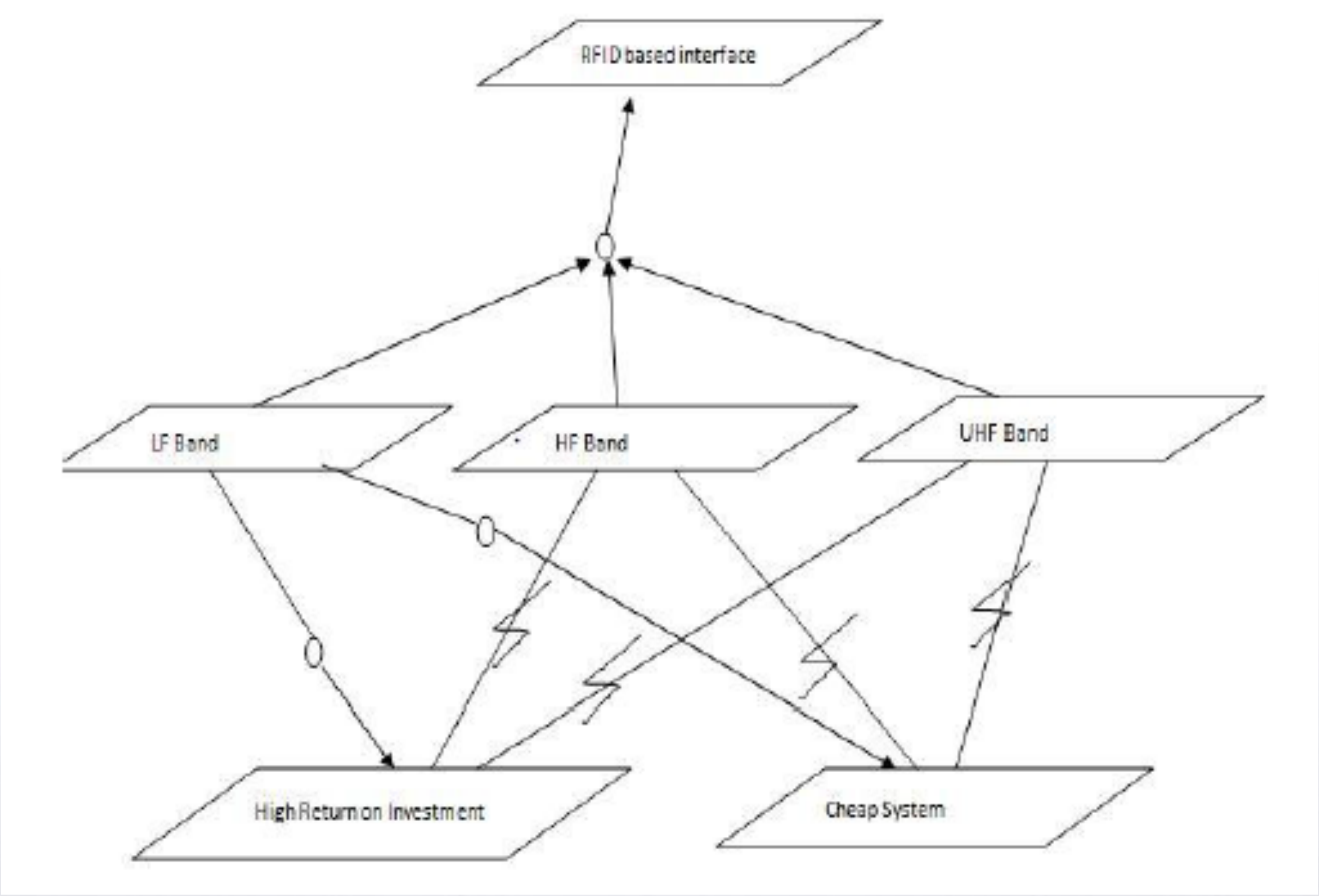
Prof. José Reinaldo Silva
reinaldo@poli.usp.br


Objectiver: HomePage
www.objectiver.com/index.php?id=4
Bonjour
5th Forum of...gn | Tracks
media.wiley..._custom.pdf
Apple
Yahoo!
Google Maps
YouTube
Wikipedia
Noticias
Popular
eprints.ucl.ac.uk/838/1/...
sobre KAOS – jreinaldosi...
Academia.edu – Share re...
Objectiver: HomePage
www.objectiver.com/file...
Home - Download - Buy Now - Contact - Site Map
Engineer
your requirements
with Objectiver
Overview - Editions - Documentation - References - Support
Objectiver
The power tool to engineer your Technical and Business Requirements
Question: What do safety critical system engineers, business analysts, software engineers have in common ?
Answer: They all know too well how difficult and important it is to write good requirements at the very start of their projects !
Download
Objectiver Trial
Objectiver represents the very first of a brand new type of requirements engineering tools. Based on a goal-oriented methodology, it aims at building high-grade requirement specifications.
To start your visit on our web site, we recommend you to first read our quick overview of the benefits of the methodology then proceed to the more detailed documentation of the methodology.
Objectiver Virtual Academy
NEW. Download this 4-pages paper presenting Objectiver in a nutshell. Read it now!!
Objectiver V3
Special Offer
For free: one 3-months access to the Objectiver Virtual Academy per Objectiver FullPro V3 license ordered before end of June, 2013. Buy now!

Design of a Smart Stick Prototype Using Goal Oriented Requirements Engineering Methodology

ARTICLE *in* INTERNATIONAL JOURNAL OF COMPUTER APPLICATIONS · FEBRUARY 2010

DOI: 10.5120/578-750 · Source: DOAJ


RFID based interface
LF Band
HF Band
UHF Band
High Return on Investment
Cheap System

di.unipi.it

A Q | Emerald | J... | Entrada (2 ... | On-the-fly T... | https://hal.a... | traceability e... | Petri Net Ap... | Engineering... | Publication | F | www.di.uni...

Evaluating the Effectiveness of a Goal-Oriented Requirements Engineering Method

Huzam S. F. Al-Subaie
King's College London
Department of Computer Science
UK
huzam.al-subaie@kcl.ac.uk

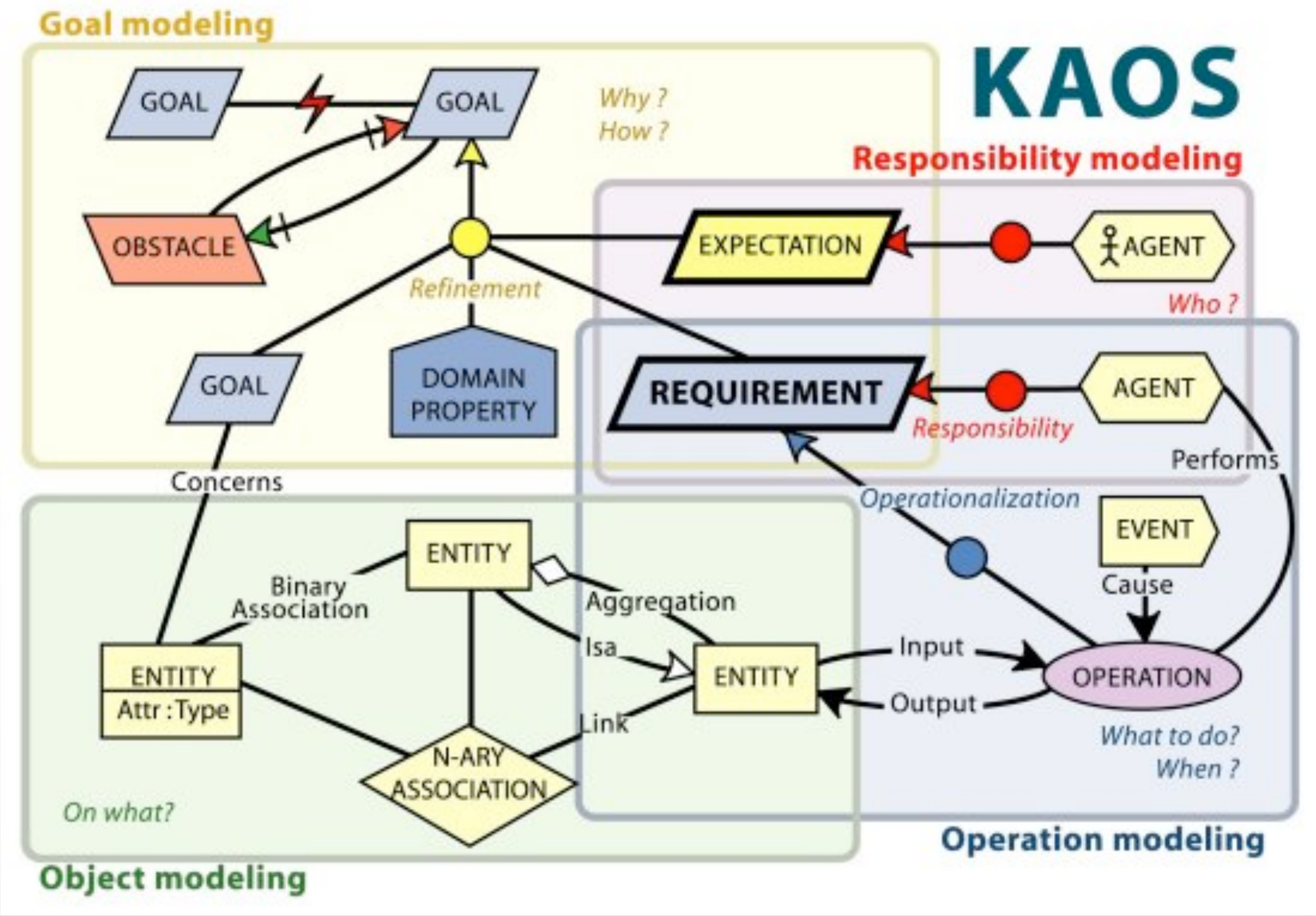
Tom S. E. Maibaum
McMaster University
Department of Computing and Software
Canada
tom@maibaum.org

Abstract

As an attempt to answer the need for methods and tools in requirements engineering (RE) which are domain specific and can address the main RE objectives (REOs), and the growing interest in the goal oriented requirements engineering (GORE)

the identification of system goals and the transformation of these goals into requirements; it addresses concerns of why a certain goal is required, how it can be achieved and who is responsible for it in the system and/or the environment [2] [3] [22]. To address these issues, the research we are undertaking proposes to evaluate KAOS, a GORE method, and

KAOS metamodel



Knowledge Engineering

Edward Feigenbaum
"Pai" dos Sistemas Especialistas
Sanford University

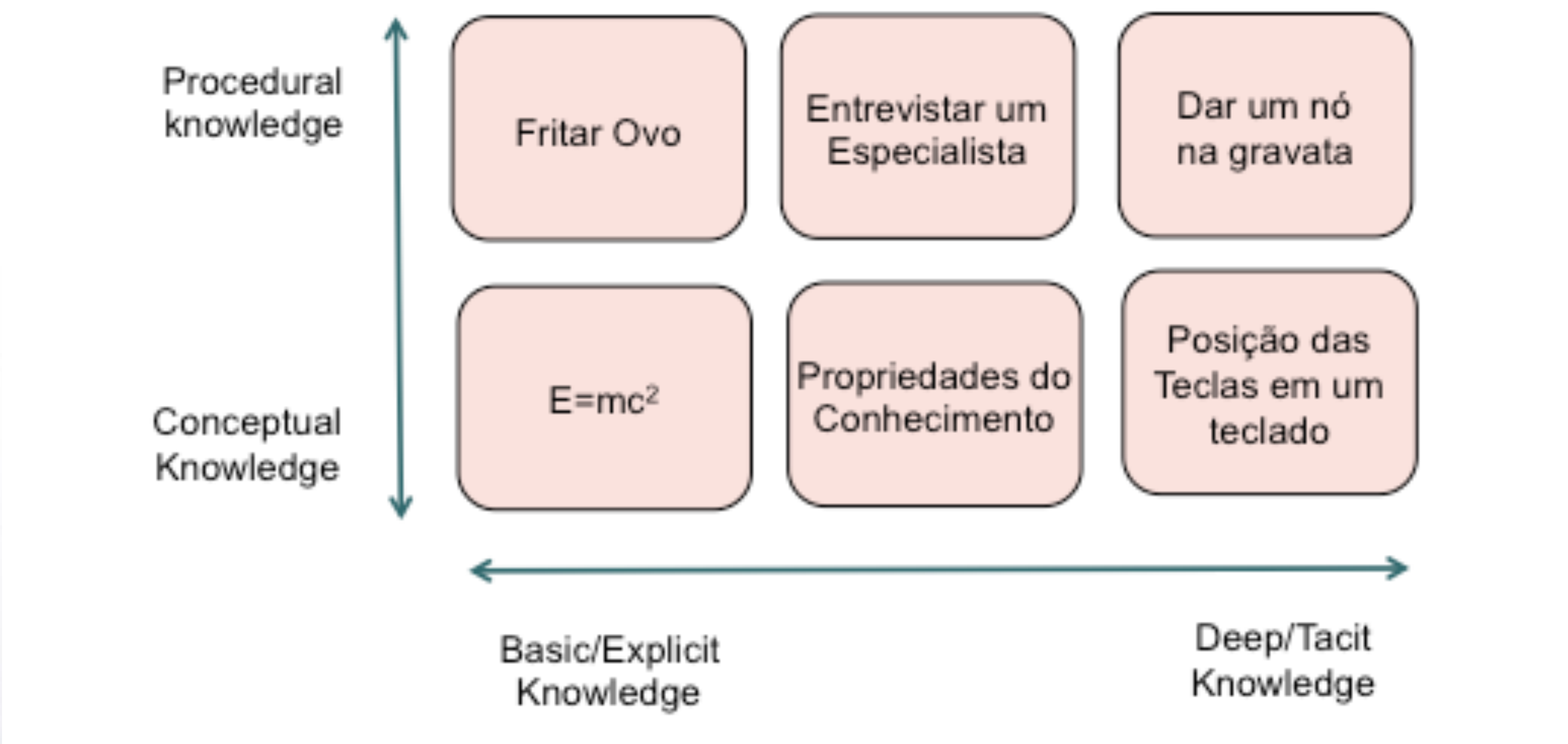
KE is an engineering discipline that involves integrating knowledge into computer systems in order to solve complex problems normally requiring a high level of human expertise

Knowledge Engineering

Knowledge Acquisition
Knowledge Modeling and Analysis
Knowledge Validation
Knowledge Base Building

→ KNOWLEDGE ENGINEERING

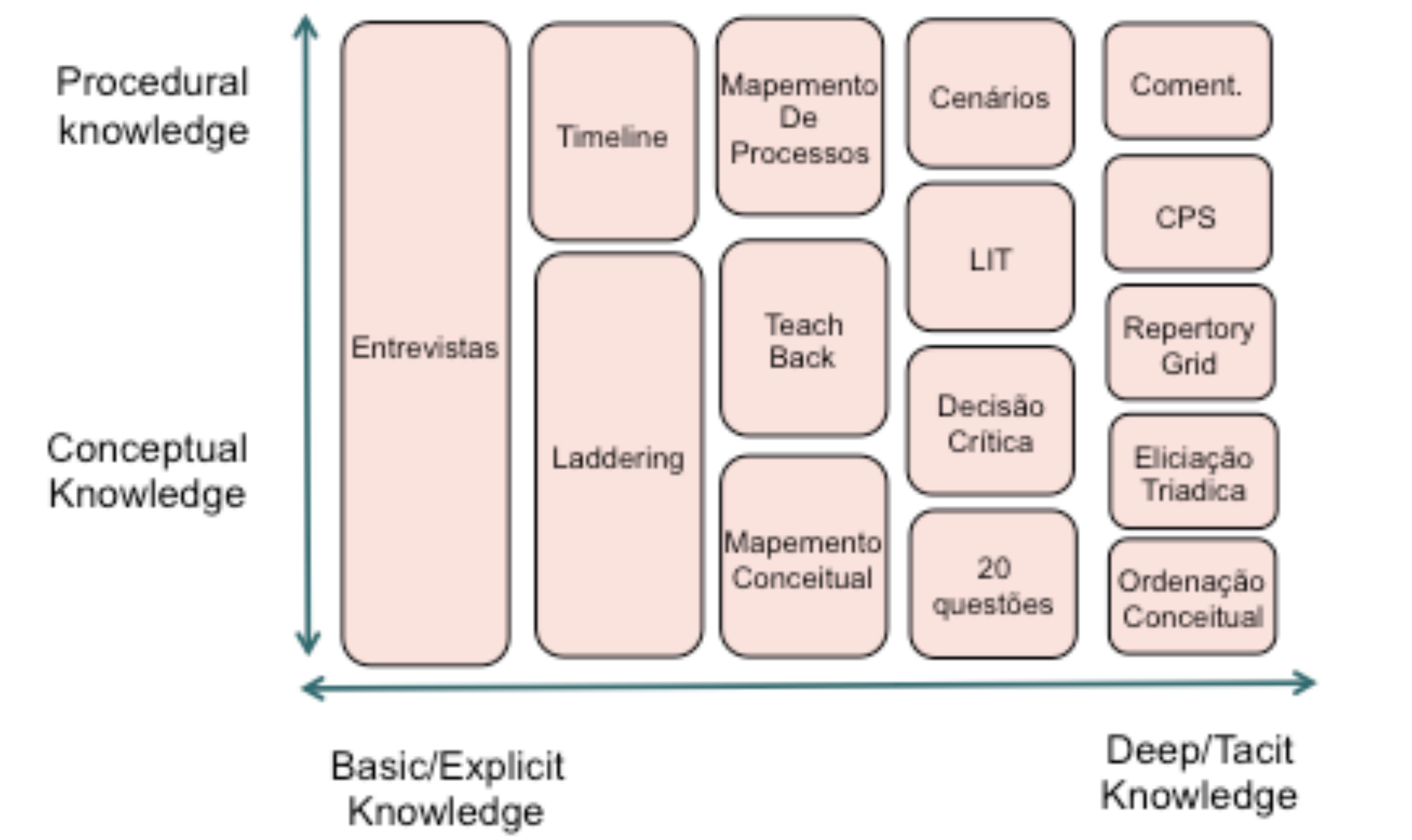
Knowledge Types



Milton, N. R; Knowledge Acquisition in Practice, Springer-Verlag, 2007

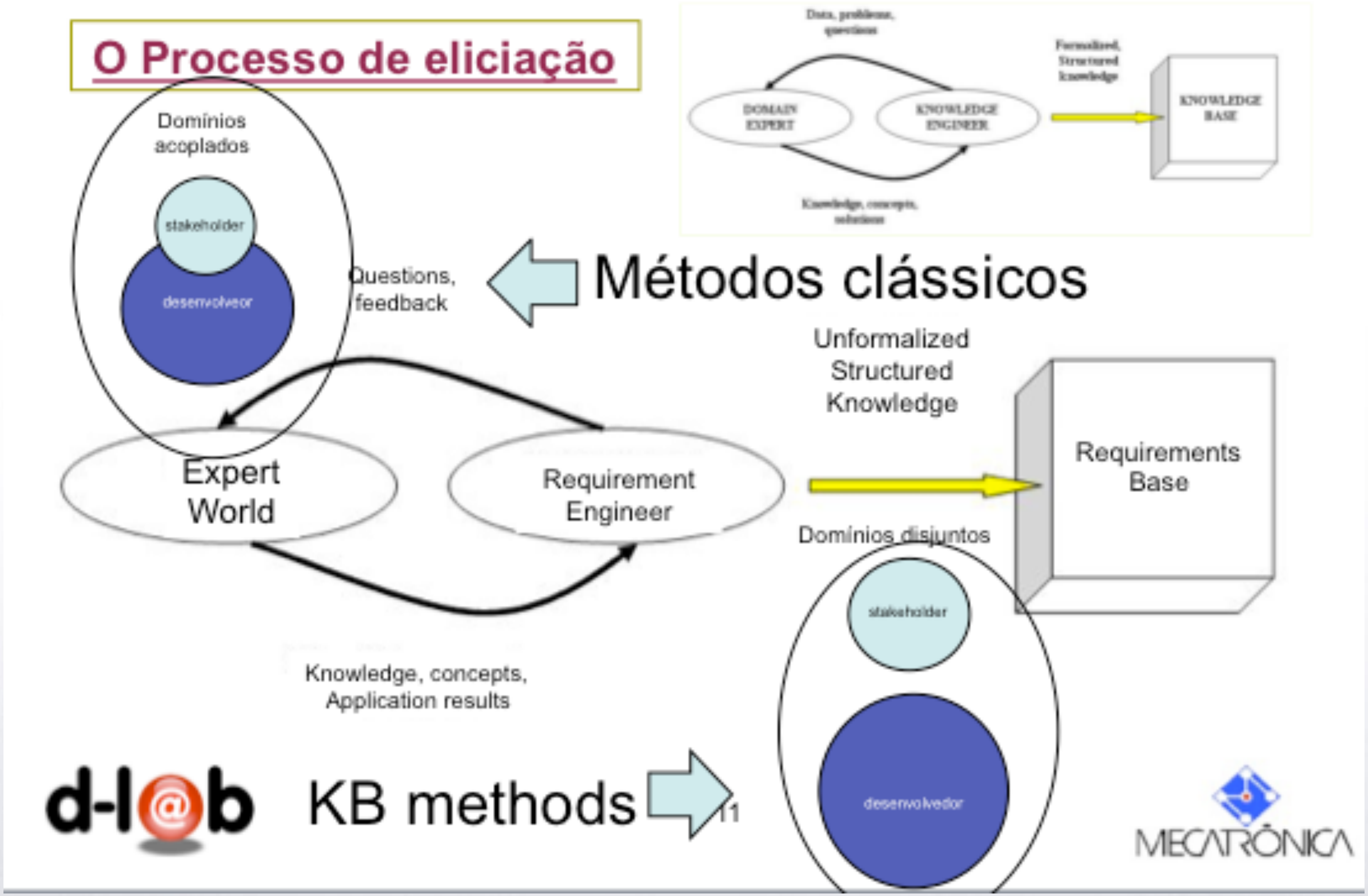
5

Knowledge Elicitation

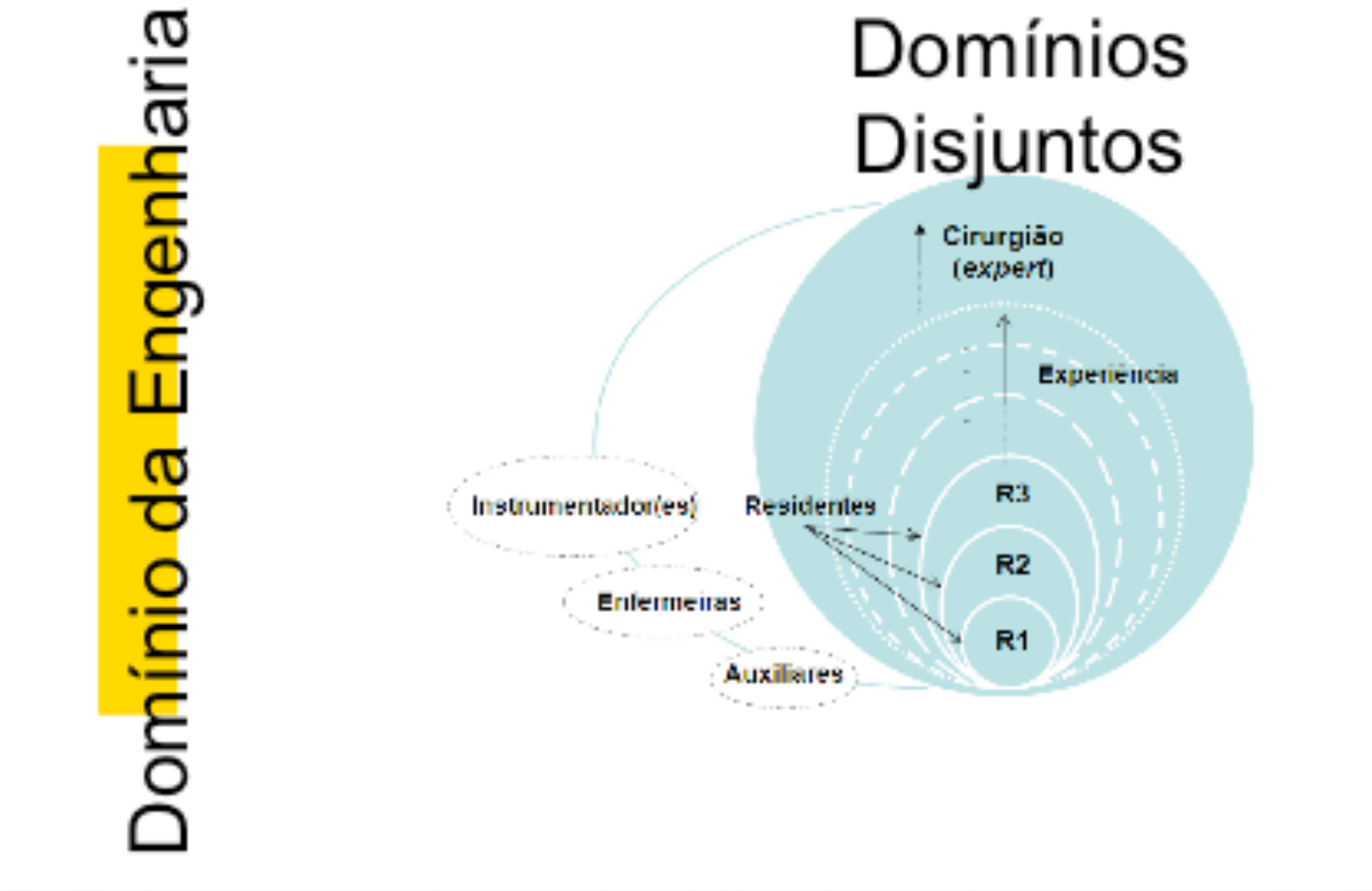


Milton, N. R; Knowledge Acquisition in Practice, Springer-Verlag, 2007

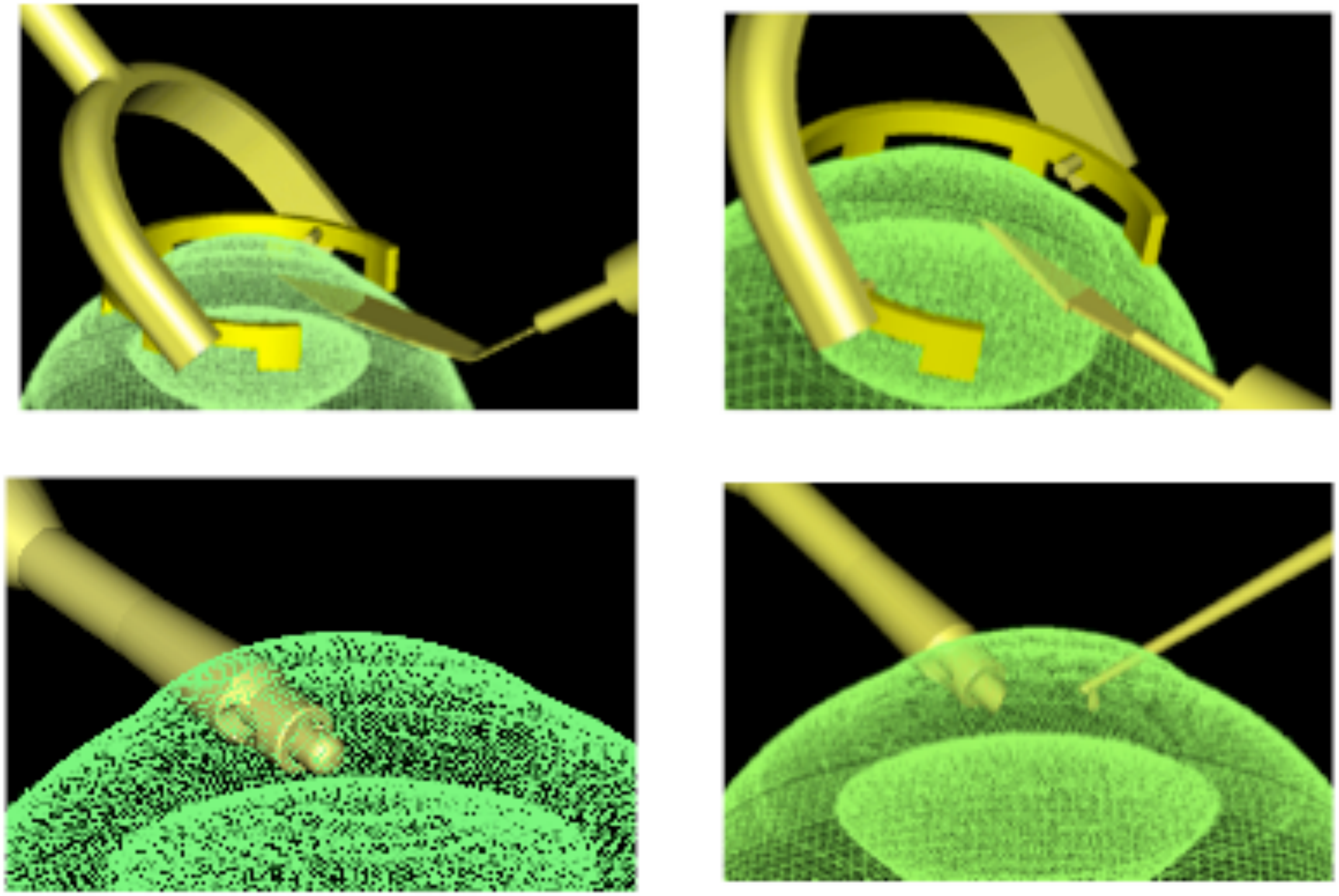
O Processo de eliciação



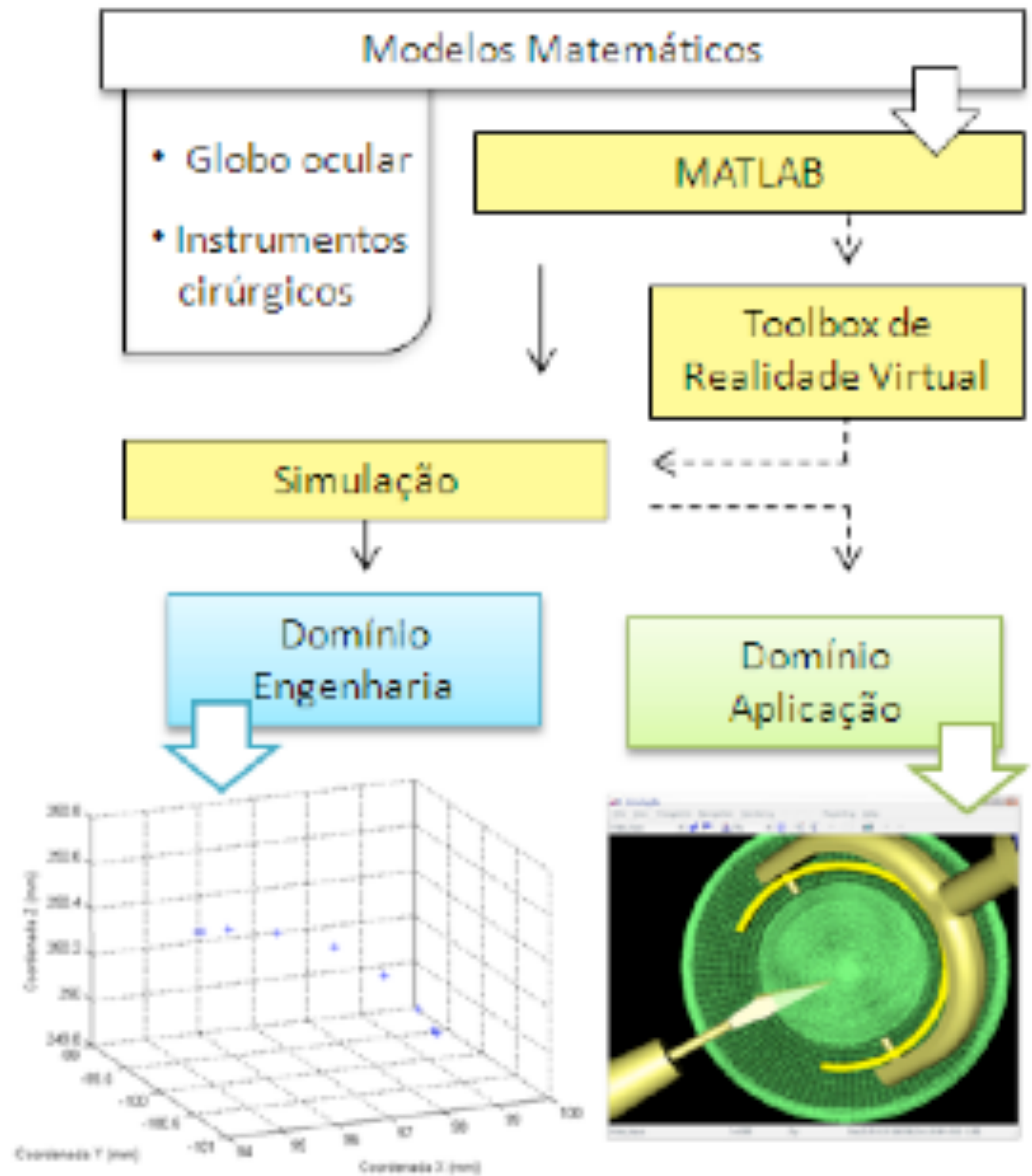
Um exemplo: automação de procedimento cirúrgico



validação?



Queiroz, R.A.A., and Silva, J.R.; Eliciação e Comunicação de Requisitos em Domínios Disjuntos: Estudo de Caso para a Área Médica, submetido à revista Controle e Automação, Sociedade Brasileira de Automática.



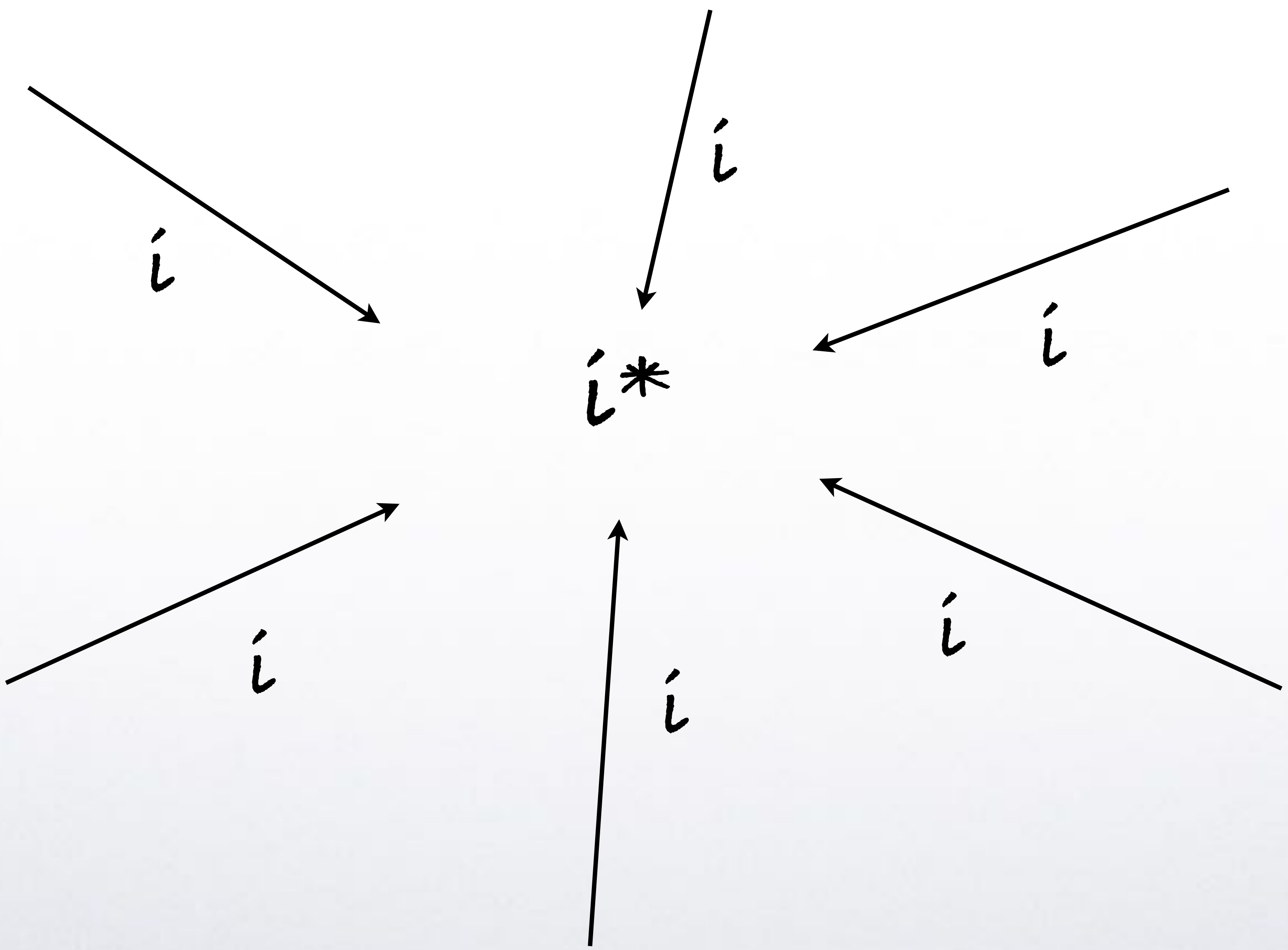
Entendendo o momento atual da ER

No final dos anos 80 surgiu um movimento liderado por alguns dos maiores pesquisadores do mundo em Engineering Design (Paul J.W. ten Hagen, Paul Veerkamp, do CWI e Tetsuo Tomiyama e Hiroyuke Yoshikawa, da Univ. de Tokyo que se chamou iCAD, ou inteligent CAD cuja meta era estudar a inserção de conhecimento no processo de Design.

GORE: Goal Oriented Requirements Engineering

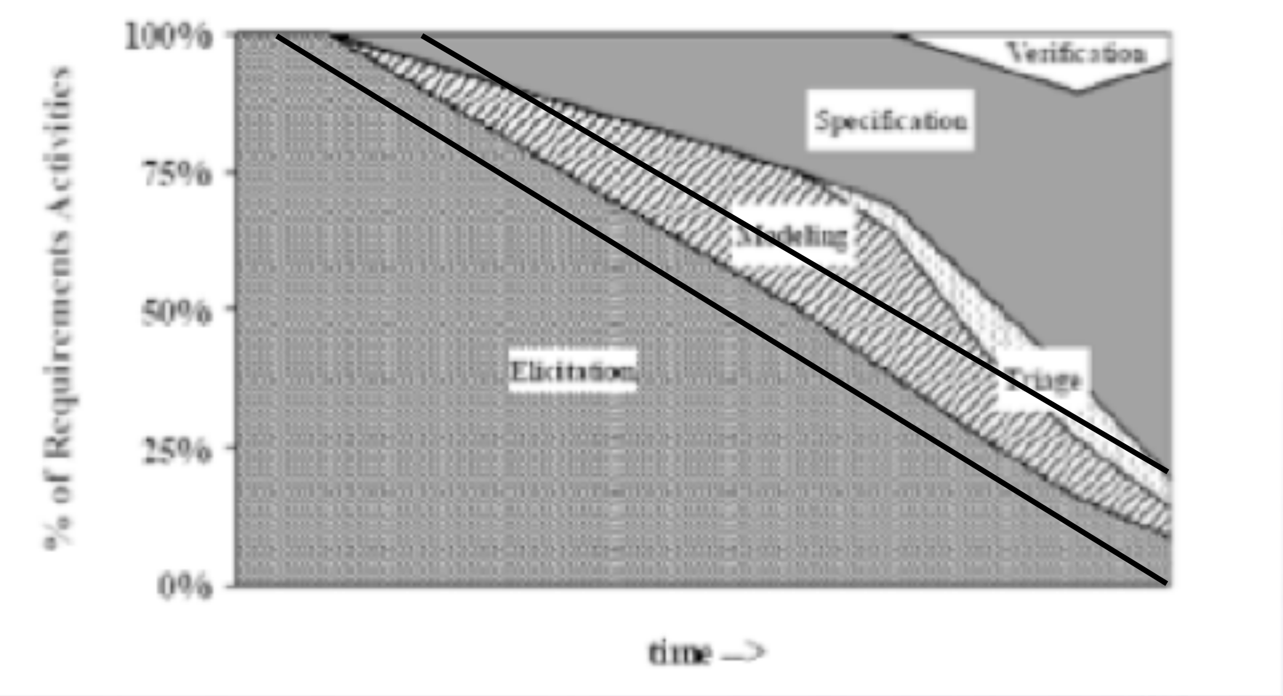
As técnicas GORE (Goal Oriented Requirements Engineering) tem como base o aumento do volume de conhecimento especialmente na fase inicial do processo de design, retirando muito do peso voltado exclusivamente para a modelagem funcional. Até o momento vimos somente o KAOS como abordagem que também aponta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



Formalismo X disciplina

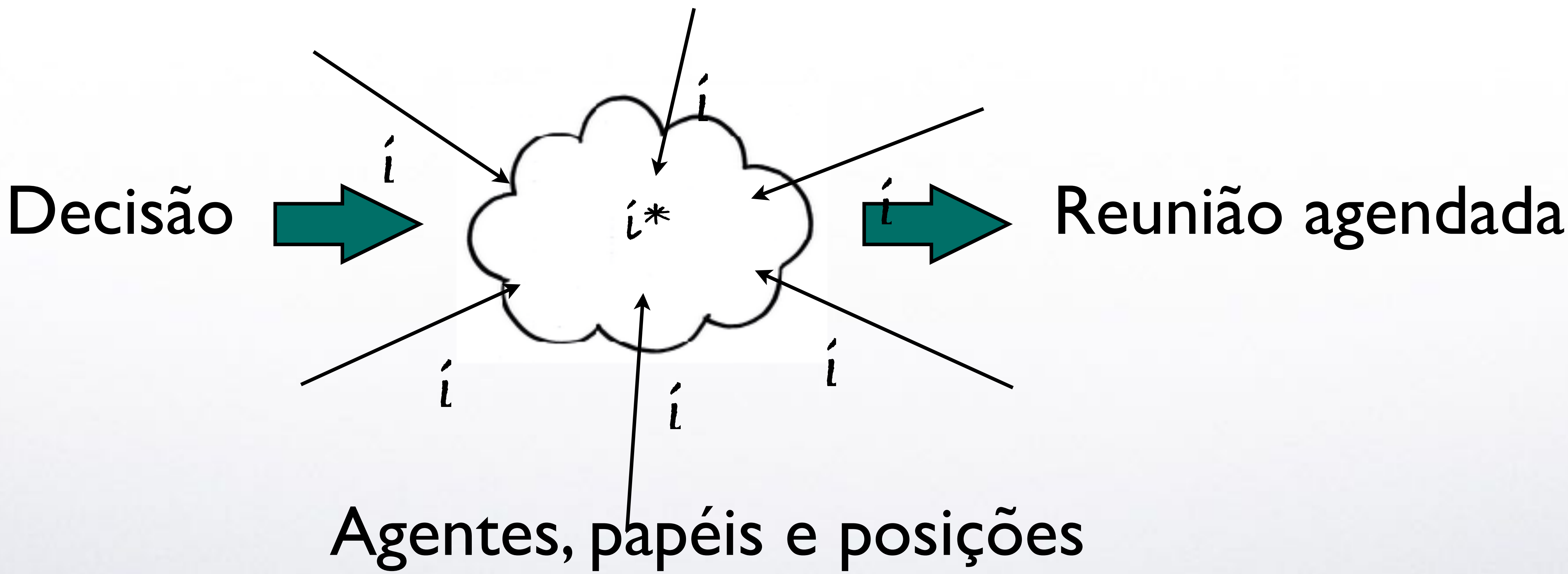
A proposta de Eric Yu é que as atuais propostas para representação de requisitos pertinentes à fase inicial (mais próxima da eliciação) são inadequadas porque estão de fato mais próximas da fase final do processo de requistos, isto é, pertinente à formalização e baseada em conceitos, e na completeza dos processos de verificação automática.

Portanto, para se livrar desta carga (formal) e capturar a verdadeira essência dos requisitos seria necessário ter uma representação preliminar, baseada em intenções. Neste caso os diversos viewpoints (que podem aparecer como conflitantes no KAOS) poderiam aparecer como discrepância entre as intenções de diferentes agentes.

Representação diagramática

No i* todo o modelo estaria baseado em dois diagramas: o diagrama de dependencia estratégica (strategic dependency diagram, SD), e o diagrama de "strategic rationales", SR que tem como objetivo a captura da intencionalidade de todos os agentes.

Um exemplo simples: agendamento de reuniões



Prof. Eric Yu

O método i* foi criado na tese de doutorado de Eric Yu na sua tese de doutorado apresentada na Universidade de Toronto em 1995: Modeling Strategic Relationships for Process Reengineering. John Mylopoulos, um nome famoso na área de Engenharia de Requisitos foi o seu orientador.
Desde a defesa Eric Yu vem trabalhando neste tema e tem já alguns resultados surpreendentes, tendo se tornado uma referência na área.

Professor Eric Yu – Home Page

www.cs.toronto.edu/~eric/

UofT Home
ischool@Toronto

UNIVERSITY OF TORONTO
FACULTY OF INFORMATION

Eric Yu Ph.D.

Professor

"Social Modeling for Requirements Engineering" - Hot off the press!

November 15, 2010

From MIT Press: "This book offers a new approach to the requirements challenge, based on modeling and analyzing the relationships among stakeholders. The *i** framework conceives of software-based information systems as being situated in environments in which social actors relate to each other in terms of goals to be achieved, tasks to be performed, and resources to be furnished. The book includes Eric Yu's original proposal for the *i** framework as well as research that applies, adapts, extends, or evaluates the social modeling concepts and approach."

Chapter 2 is a reprint of Eric Yu's doctoral dissertation from 1995. It is followed by 18 chapters authored by researchers from around the world who have applied, adapted, or extended the i* framework in various ways, and for diverse application contexts – from business processes to knowledge management to air traffic control, from information security to software development.

Sneak preview - Chapter One.

MIT Press. Amazon.com. Google books.

The *i** framework is now part of an international standard!

November 13, 2008

The User Requirements Notation (URN) received final approval as an international standard today in Geneva, Switzerland, as ITU-T Recommendation Z.151. URN consists of the Goal-oriented Requirements Language (GRL), based on Professor Eric Yu's *i** modelling framework, and Use Case Maps (UCM), a scenario modelling notation. GRL provides a notation for modelling goals and rationales, and strategic relationships among social actors. It is used to explore and identify system requirements, including especially non-functional requirements. Thanks to the many students, research team members, colleagues, and international collaborators who contributed directly or indirectly. ITU is the UN agency for information and communication technologies.

www.cs.toronto.edu/km/GRL/

UNDER CONSTRUCTION

- GRL Introduction
- GRL Ontology
- GARE Methodology
- Case Studies & Tutorial
- Discussions
- Tool
- References & Links

University of Toronto, CANADA

GRL - Goal-oriented Requirement Language

GRL (Goal-oriented Requirement Language) is a language for supporting goal-oriented modeling and reasoning of requirements, especially for dealing with non-functional requirements. It provides constructs for expressing various types of concepts that appear during the requirement process. There are three main categories of concepts: intentional elements, links, and actors. The intentional elements in GRL are goal, task, softgoal, and resource. They are intentional because they are used for models that allow answering questions such as why particular behaviors, informational and structural aspects were chosen to be included in the system requirement, what alternatives were considered, what criteria were used to deliberate among alternative options, and what the reasons were for choosing one alternative over the other.

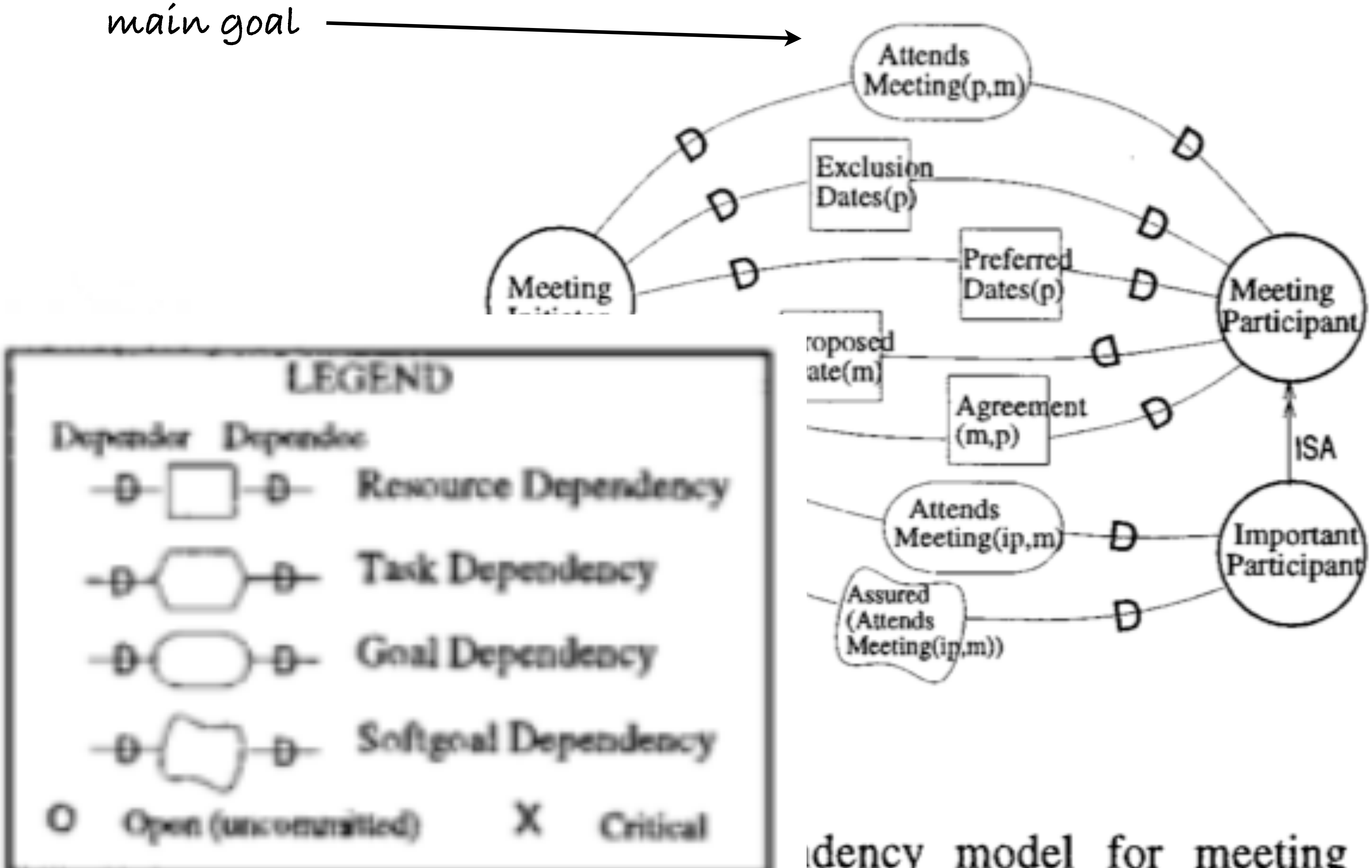
This kind of modeling is different from the detailed specification of what is to be done. Here the modeler is primarily concerned with exposing "why" certain choices for behavior and/or structure were made or constraints introduced. The modeler is not yet interested in the "operational" details of processes or system requirements (or component interactions). Omitting these kind of details during early phases of analysis (and design) allows taking a higher-level (sometimes called a strategic stance) towards modeling the current or the future software system and its embedding environment. Modeling and answering "why" questions leads us to consider the opportunities stakeholders seek out and/or vulnerabilities they try to avoid within their environment by utilizing capabilities of the software system and/or other stakeholders, by trying to rely upon and/or assign capabilities and by introducing constraint how those capabilities ought to be performed.

GRL supports the reasoning about scenarios by establishing correspondences between intentional GRL elements and non-intentional elements in scenario models of URN-FR. Modeling both goals and scenarios is complementary and may aid in identifying further goals and additional scenarios (and scenario steps) important to stakeholders, thus contributing to the completeness and accuracy of requirements.

A proposta da URN

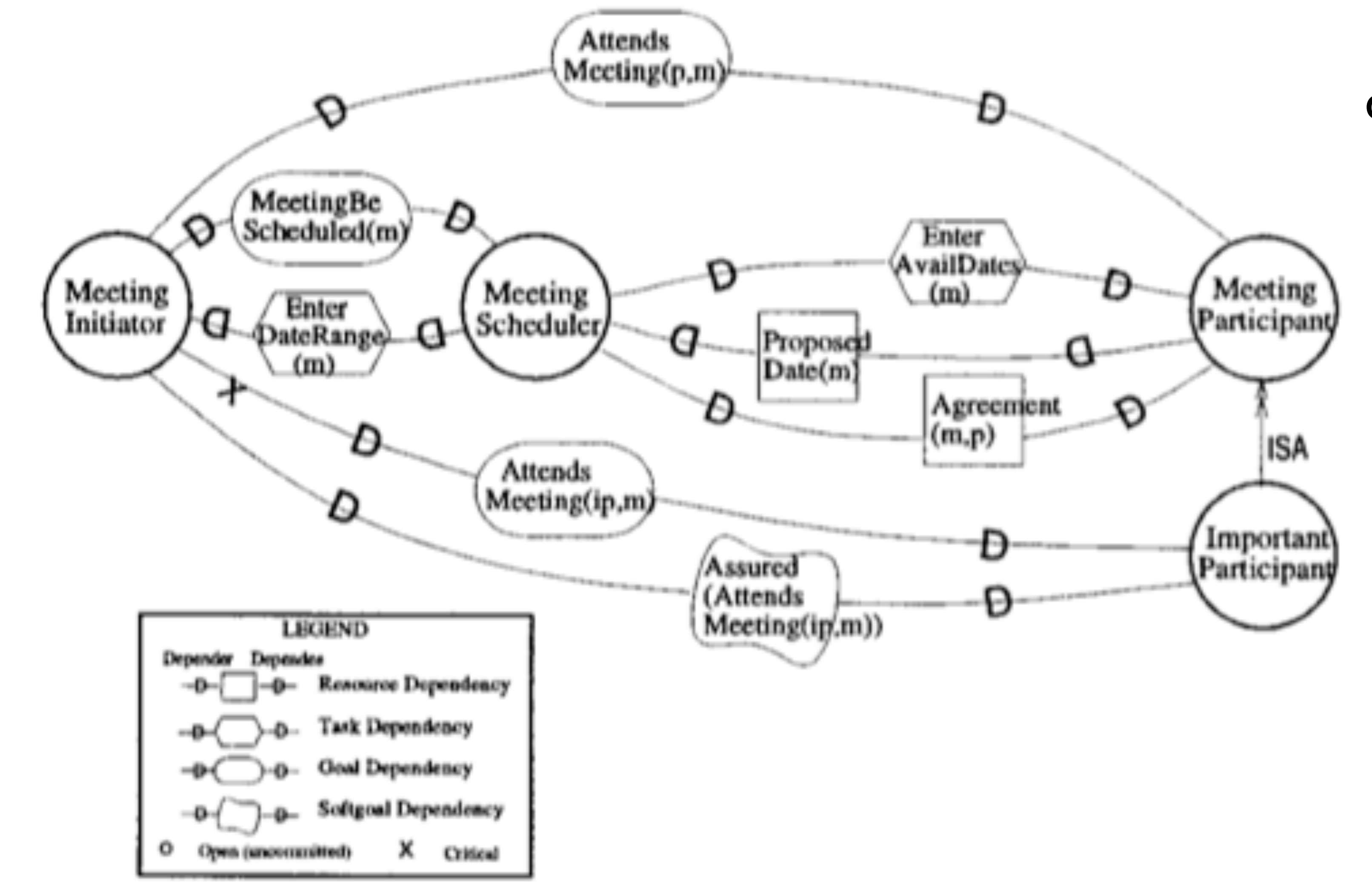
O i* deu origem a uma representação chamada URN (User Requirement Notation) que foi submetida como proposta de padrão para a fase inicial de requisitos (eliciação e análise) e foi aceita, tendo já uma recomendação. Este fato vem colocar novos argumentos na discussão sobre o uso da UML e da modelagem orientada a objetos.

main goal



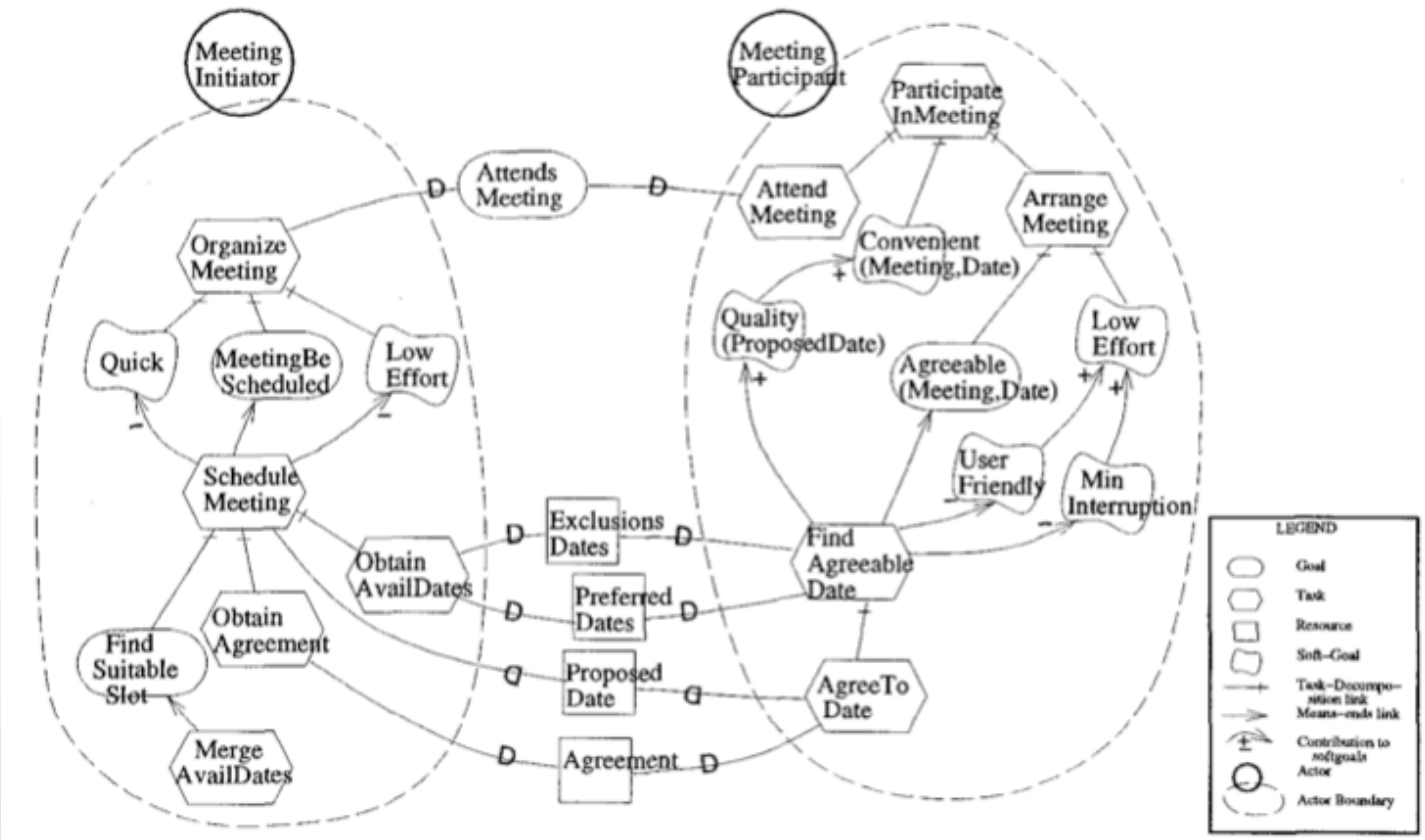
...dency model for meeting ...-based scheduler

Inserindo a automação

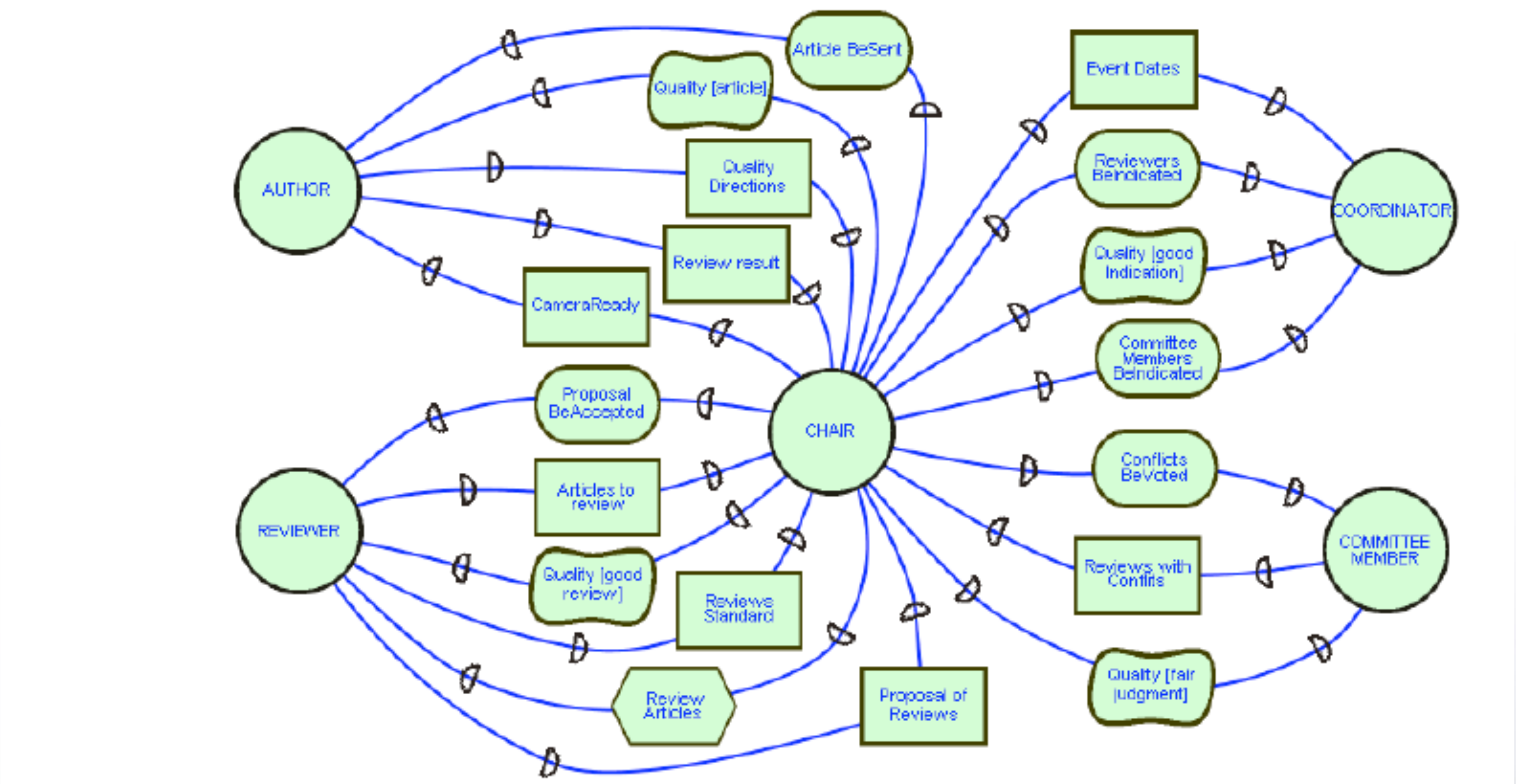


A mesma ação de agendamento pode agora ser feita com o auxílio de uma ferramenta computadorizada (meeting scheduler) que passa a ser outro agente (máquina) que na verdade tem um papel a desempenhar (que não é de fato a sua "intenção", embora possa ser interpretado como tal).

Capturando intenções

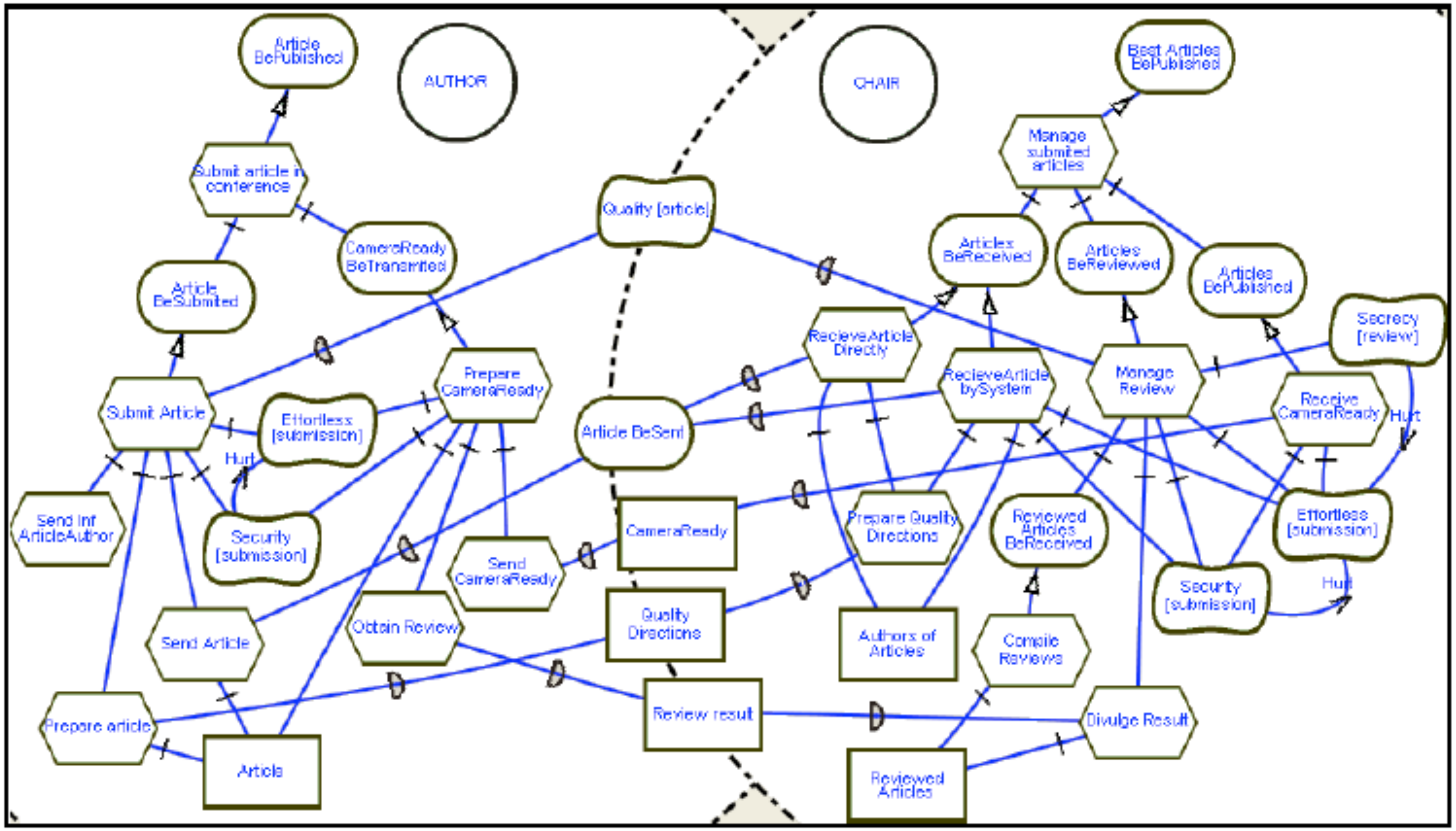


Um outro exemplo: revisão de artigos



Werneck, V.M.B., Oliveira, A. P.A., Leite, J. C. P.; Comparing GORE frameworks: i-star and KAOS, *Ibero-American Workshop of Engineering of Requirements, Val Paraiso, Chile, (July 2009*)

Strategic Rationale Diagram



troposproject.org

Home | Research | Tools | Papers | Presentations | People | Links | Projects | Seminars

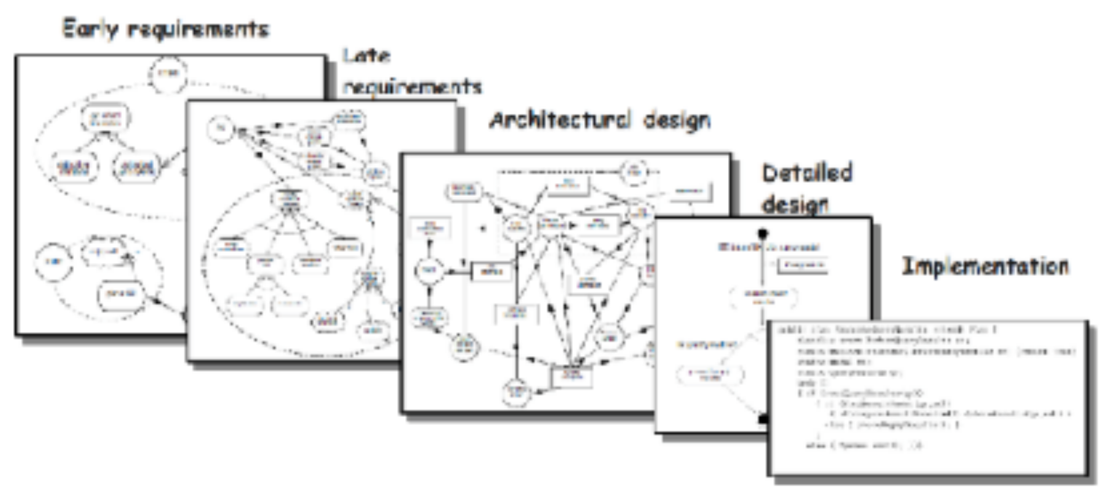
- Methodology
- Requirements
- Security and Compliance
- Tropos for Self-*
- Empirical Studies

Check out the Socio-Technical Security modeling language (STS-ml) and its support tool, STS-Tool (website).

UNIVERSITY OF TRENTO - Italy

Tropos

Tropos is a software development methodology, where concepts of the agent paradigm are used along the whole software development process. Notions of agent, goal, task and (social) dependency are used to model and analyze early and late software requirements, architectural and detailed design, and (possibly) to implement the final system. In this web site, you can find details of ongoing research, developed tools, industrial projects and Tropos related events. Tropos is derived from the Greek τροπος, which means "way of doing things"; also τροπή, which means "turn" or "change".



tropos project.org

Home | Research | Tools | Papers | Presentations | People | Links | Projects | Seminars

- Methodology
- Requirements
- Security and Compliance
- Tropos for Self-*
- Empirical Studies

Check out the Socio-Technical Security modeling language (STS-ml) and its support tool, STS-Tool (website).

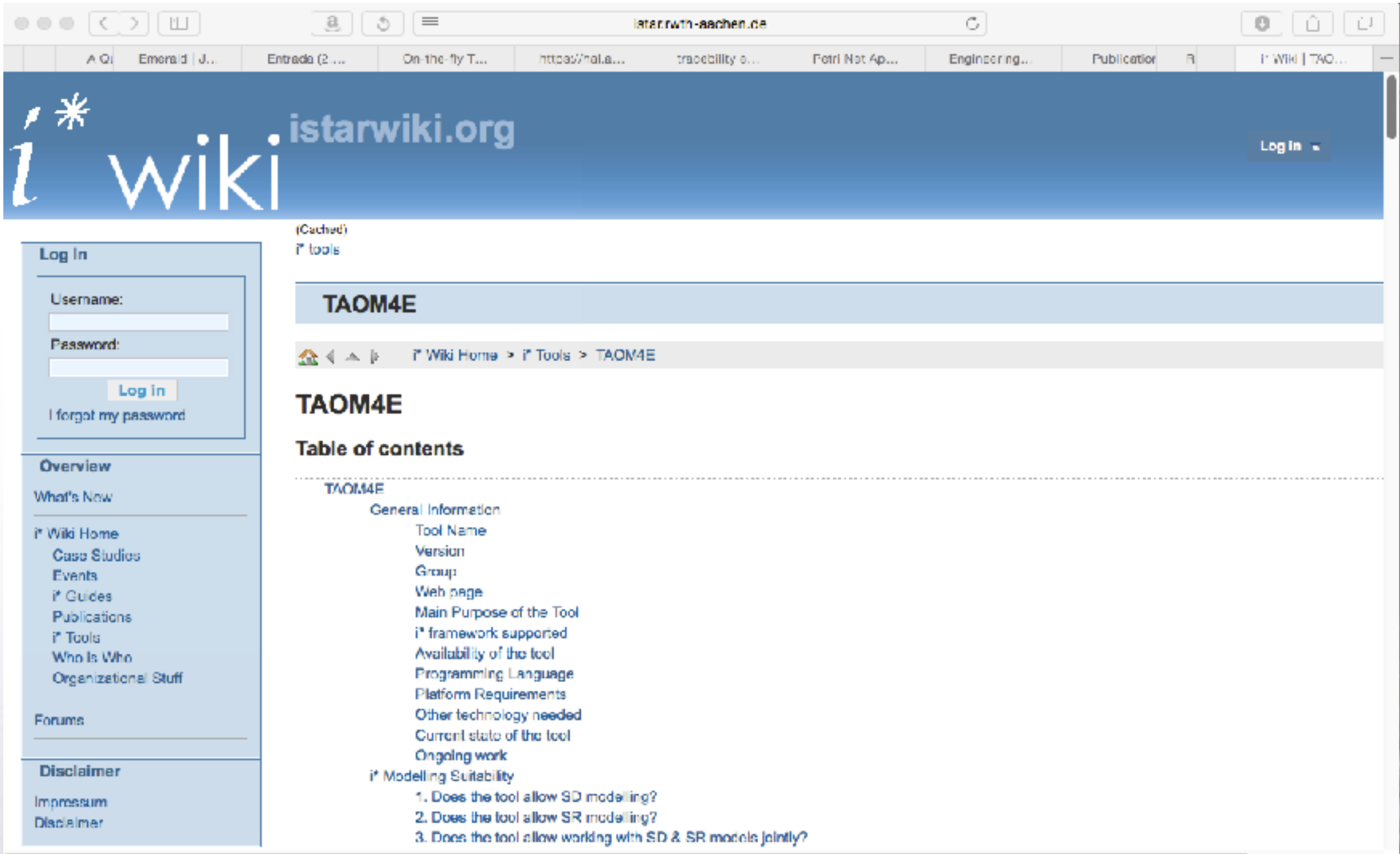
UNIVERSITY OF TRENTO - Italy
Information Engineering

Tools

People working on Tropos developed a number of prototype tools that have been used and tested in several projects. Here the list of current available tools.

Tool	Maintainer	Notes
STS-Tool	DISI - University of Trento	Support tool for the Socio-Technical Security modeling language (STS-ml)
SI* Tool	DISI - University of Trento	Secure Tropos tool. It supports risk reasoning and planning
TAOM4E	Fondazione Bruno Kessler	Support tool for the Tropos Methodology. Code generation to JADEX
GR-Tool	DISI - University of Trento	Goal Reasoning tool
T-Tool	DISI - University of Trento	Formal Tropos tool
DW-Tool	DISI - University of Trento	Data Warehouse design tool
OpenOME	University of Toronto	General purpose tool based on i*
DESCARTES	Université catholique de Louvain	Design CASE Tool for Agent-Oriented Repositories, Techniques, Environments and Systems
SecTro	University of East London	Automated modeling tool that provides supports for the Secure Tropos methodology

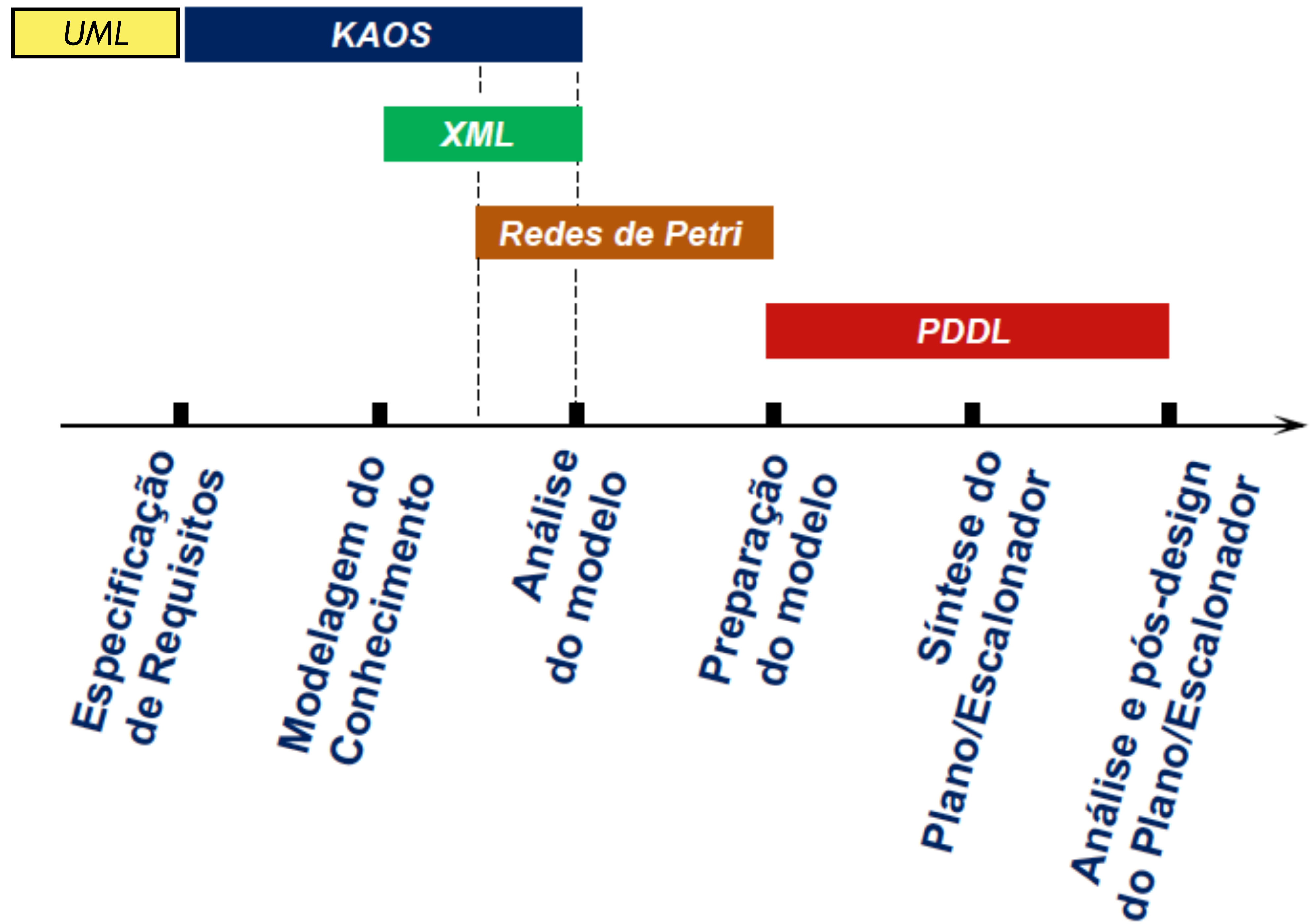
Back to top



http://istar.rwth-aachen.de/tiki-index.php?page=TAOM4E

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.


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

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 represen-tation	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.

Leitura da Semana

werneck.pdf (page 1 of 12)

Comparing GORE Frameworks: i-star and KAOS

Vera Maria Bejamim Werneck[1], Antonio de Padua Albuquerque Oliveira[1], Julio Cesar Sampaio do Prado Leite[2]

[1] *Universidade do Estado do Rio de Janeiro – UERJ-IME*
Rua São Francisco Xavier 524, 6o Andar Bloco B – Rio de Janeiro – RJ, Brazil

[2] *Pontifícia Universidade Católica do Rio de Janeiro – PUC-Rio*
Departamento de Informática, Rua Marques de São Vicente 225 – RJ, Brazil,

{vera, padua} @ime.uerj.br, www.inf.puc-rio.br/~julio

Abstract

Goal-Oriented Requirements Engineering (GORE) is an approach to requirements engineering dealing with intentionality in accordance with the relations among different actors. KAOS and i (i-star) frameworks have been receiving many references as being important GORE proposals. This paper presents an conceptual analysis comparing characteristics of these methods giving examples related to actors' relations definition, goal organizational model, tasks representation, risk analysis, and non-functional requirements. The aim of this work is to show both frameworks benefits and drawbacks. We believe that this analysis helps the understanding of the core concepts of GORE as well as it draws attention to key representation issues for both KAOS and i*.*

1. Introduction

The increasing demand for complex software applications requires methods that are able to deal with intentionality. The RE sub-area Goal-Oriented Requirements Engineering (GORE) [1] comes to meet these requirements in order to improve current approaches, either Object Oriented or Aspect Oriented, in the development of complex software applications. Using Lamsweerde classification [1], several methods can be considered as belonging to GORE: i* [11], TROPOS [19], The goal/strategy Map [20], GLR [7]. Among these methods, KAOS and i* have been the most cited.

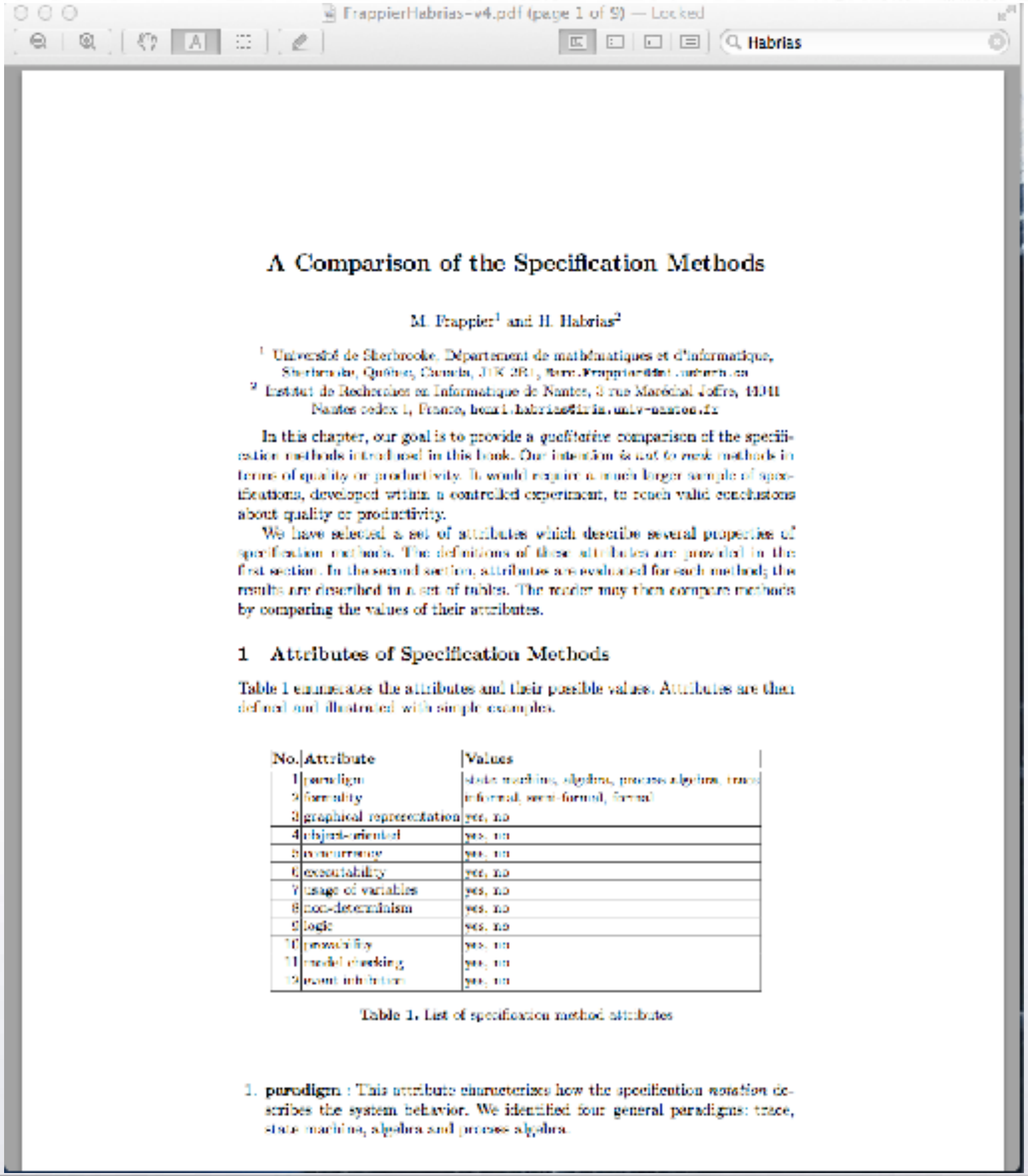
Our work analyzes KAOS [1] and i* [2], discussing qualitatively their coincidences and differences as well as their benefits and drawbacks in the representation of five characteristics (actors representation, goals model, NFRs, tasks and risk analyses) which we consider central in Goal-Oriented Requirements Engineering. Contrary to Matulevicius and Heymans [16], which analyzed i* and KAOS usage, we are analyzing the languages concepts and structure. We also stress the importance of properly representing goals and avoiding the common confusion with actions [5].

In this article we describe, using an example, the most important characteristics of KAOS and i*. Kaos and i* are presented by means of two UML meta-models as described in Section 2. We explain the comparison process, and provide the comparison results in Section 3. We conclude by mentioning the benefits of both frameworks in Section 4.

2. Goal-Oriented Requirements Engineering (KAOS and i*)

First, Subsection 2.1, we describe the concepts used for the comparison. They are: the requirements activities, functional requirement (FR), non-functional requirement (NFR), softgoal and goal. Subsection 2.2

Leitura da Semana



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

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 Recherche 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	provability	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.

Obrigado

Reinaldo