

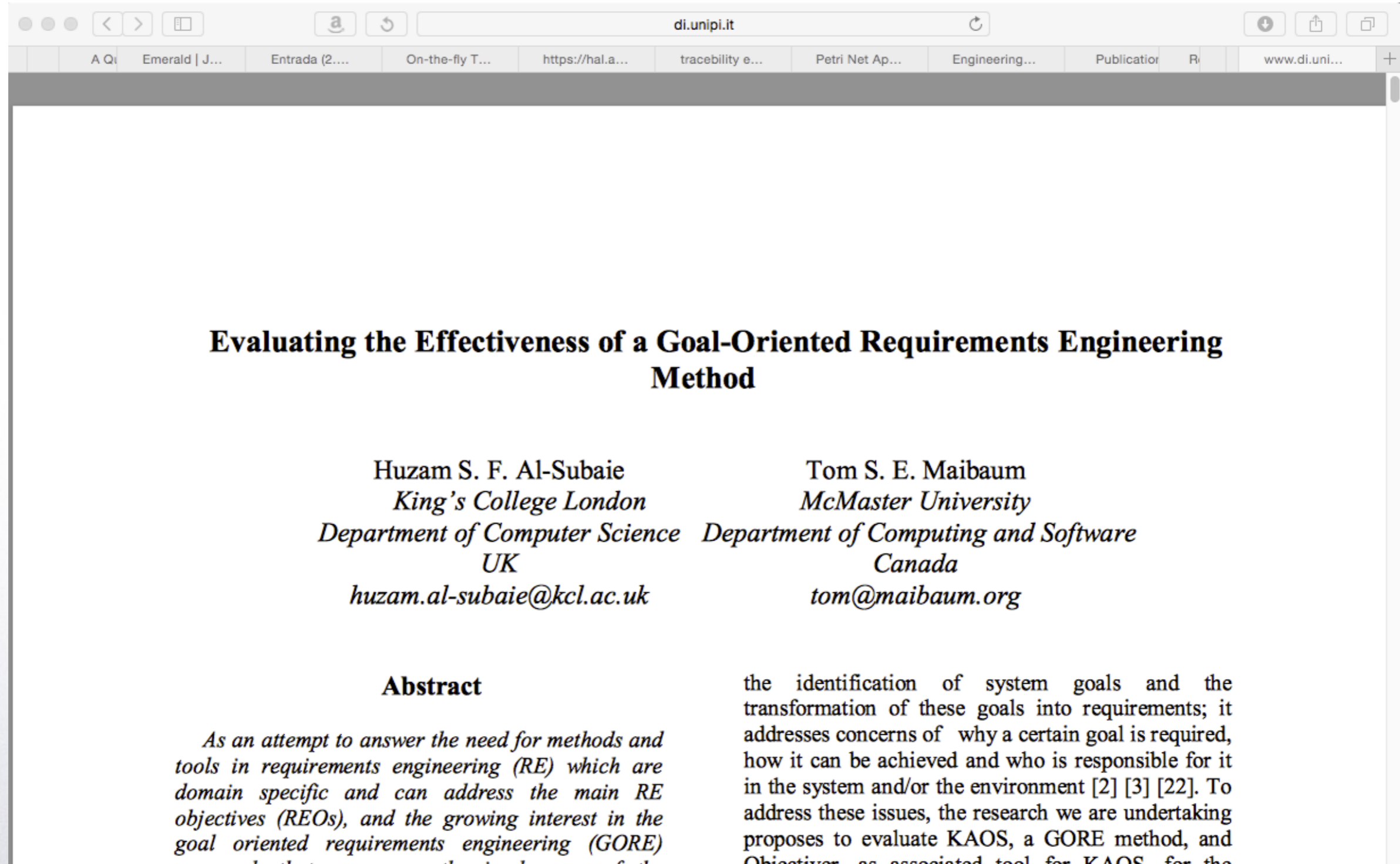
PMR 5020

Modelagem do Projeto de Sistemas

Aula 7: Uso de linguagens formais (no ombro de gigantes)

Prof. José Reinaldo Silva

reinaldo@usp.br



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 Objectiver, as associated tool for KAOS, for the

Knowledge Engineering

Knowledge Acquisition
Knowledge Modeling and Analysis
Knowledge Validation
Knowledge Base Building



KNOWLEDGE
ENGINEERING

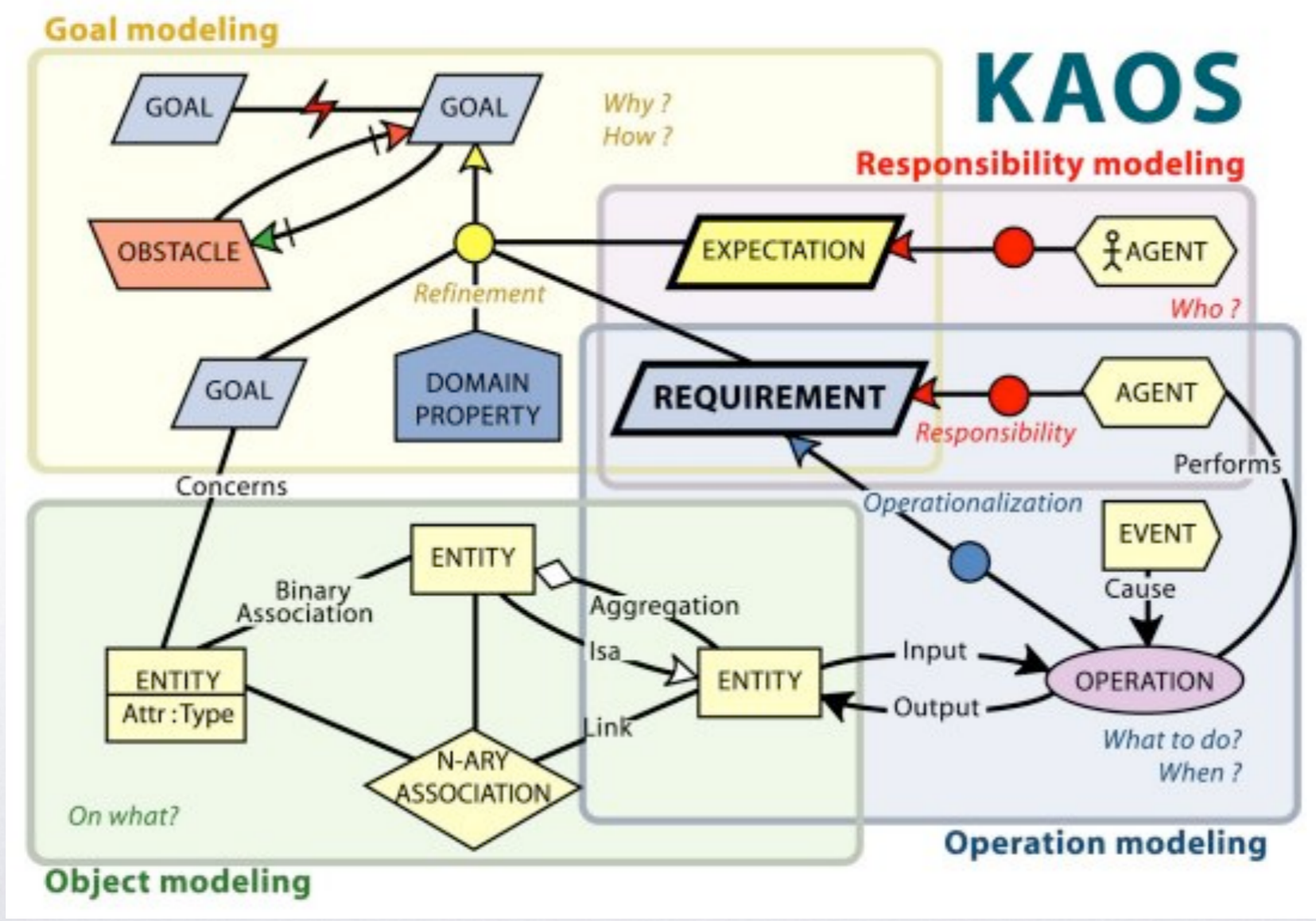
Requirements Engineering

Requirements Elicitation
Requirements modeling and analysis
Requirements validation/verification
Requirements documentation

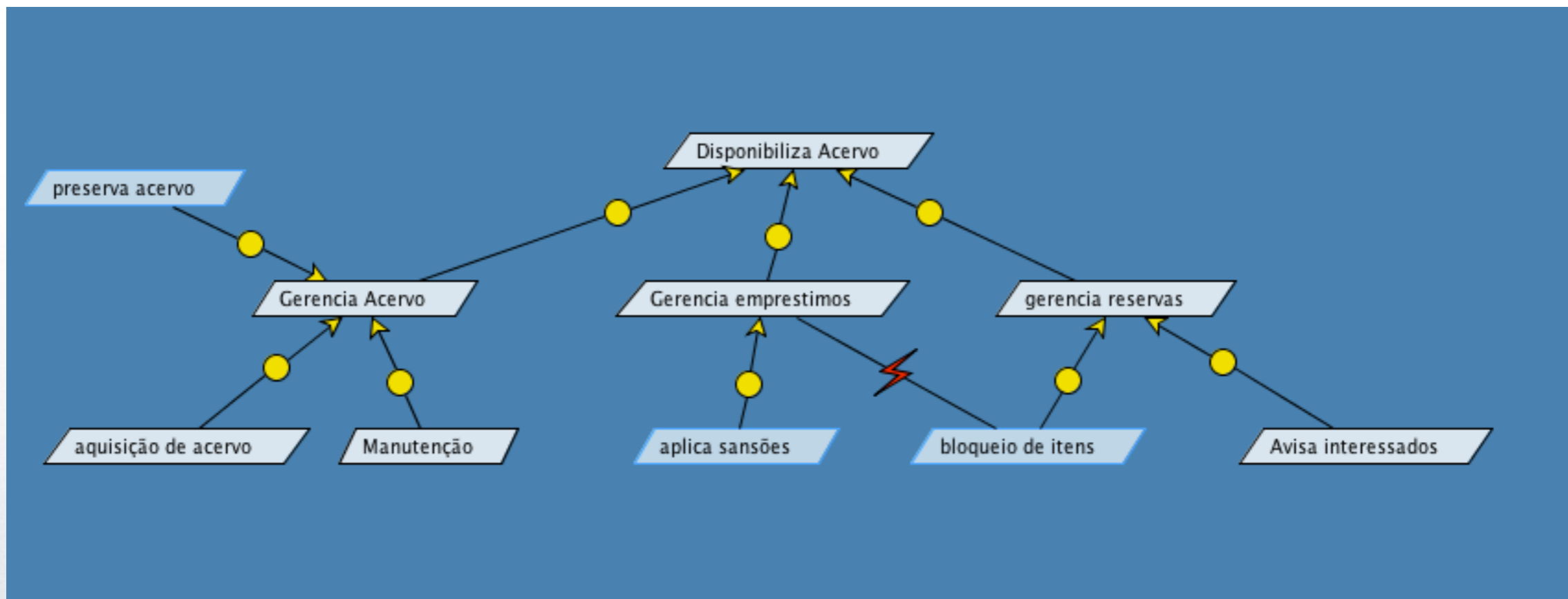


REQUIREMENTS
ENGINEERING

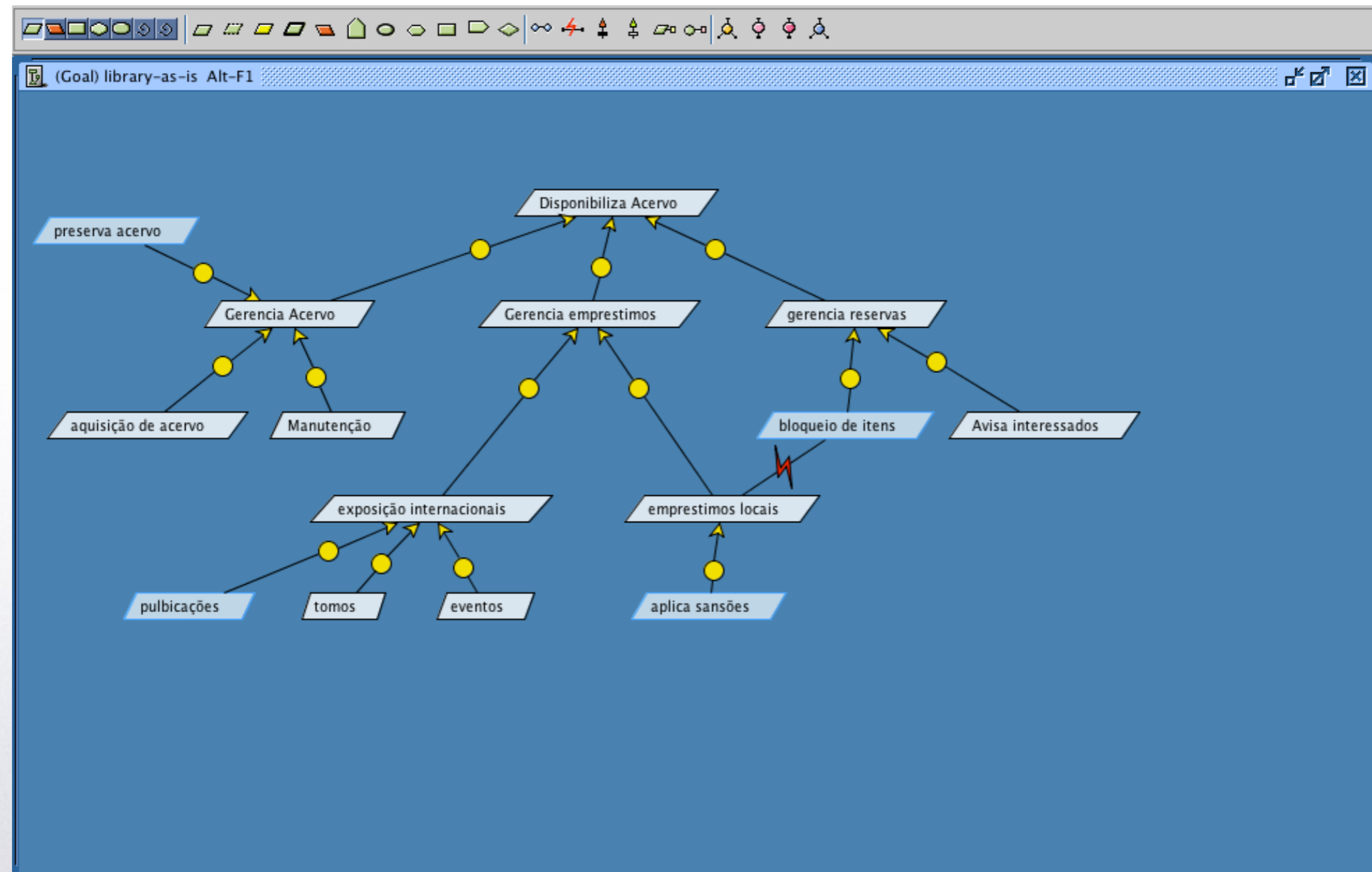
KAOS metamodel



Modelando o sistema de bibliotecas: system-as-is



Modelando o system-as-is



Evaluation of the Goal-Oriented Requirements Engineering Method KAOS

Frank Zickert
University of Frankfurt
mail@frankzickert.de

Roman Beck
University of Frankfurt
rbeck@wiwi.uni-frankfurt.de

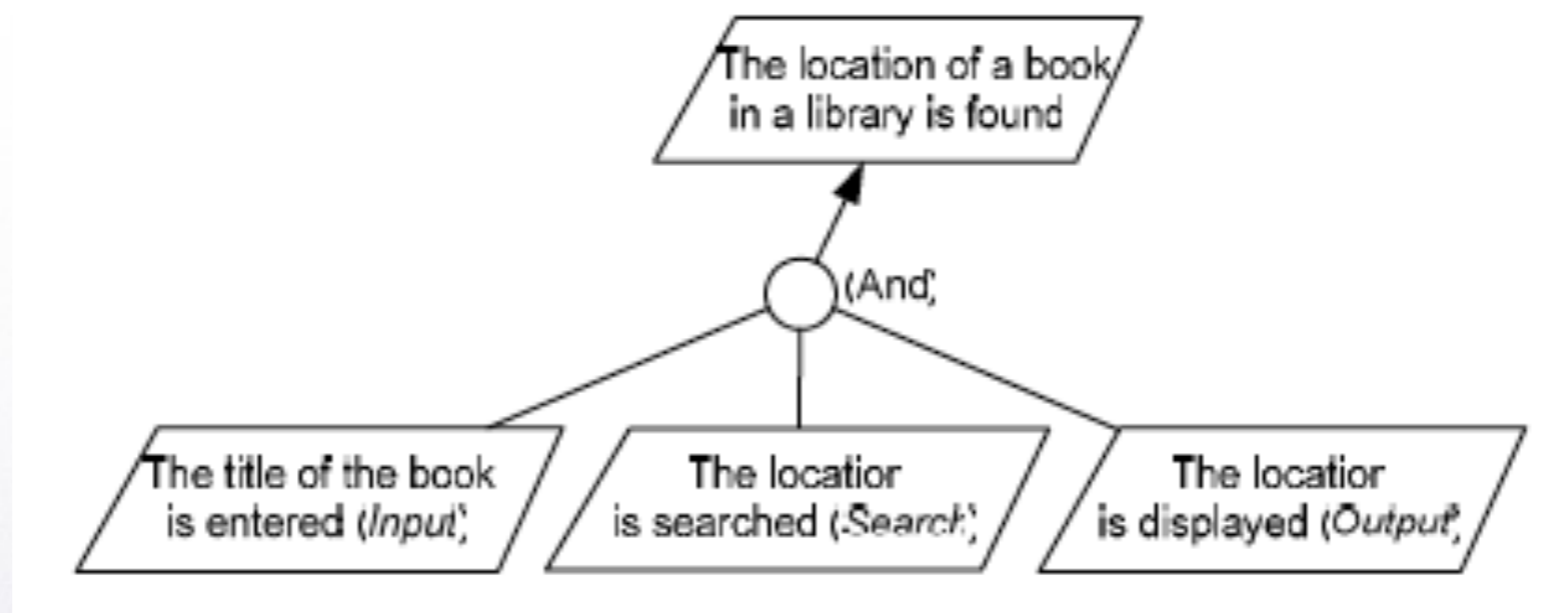
ABSTRACT Software engineering is a complex task. But although there is no silver bullet that guarantees accomplishing this task, appropriate methods can support the engineer by addressing the characteristics that make it complex. The objective of this paper is to evaluate whether and how the goal-oriented requirements engineering method KAOS addresses these characteristics of complex tasks and thereby, whether it effectively supports software engineering. For serving this purpose, we conduct a literature analysis, which discloses core concepts underlying to the KAOS method, and we apply KAOS in two software development projects, which provide insights into KAOS in use. Our results show that KAOS, despite of some shortcomings, addresses all characteristics, but that applying it can be work intensive. Consequently, while KAOS supports software engineering, provided support must be weigh up against invested work.

Keywords (Required) KAOS, system engineering, task complexity.

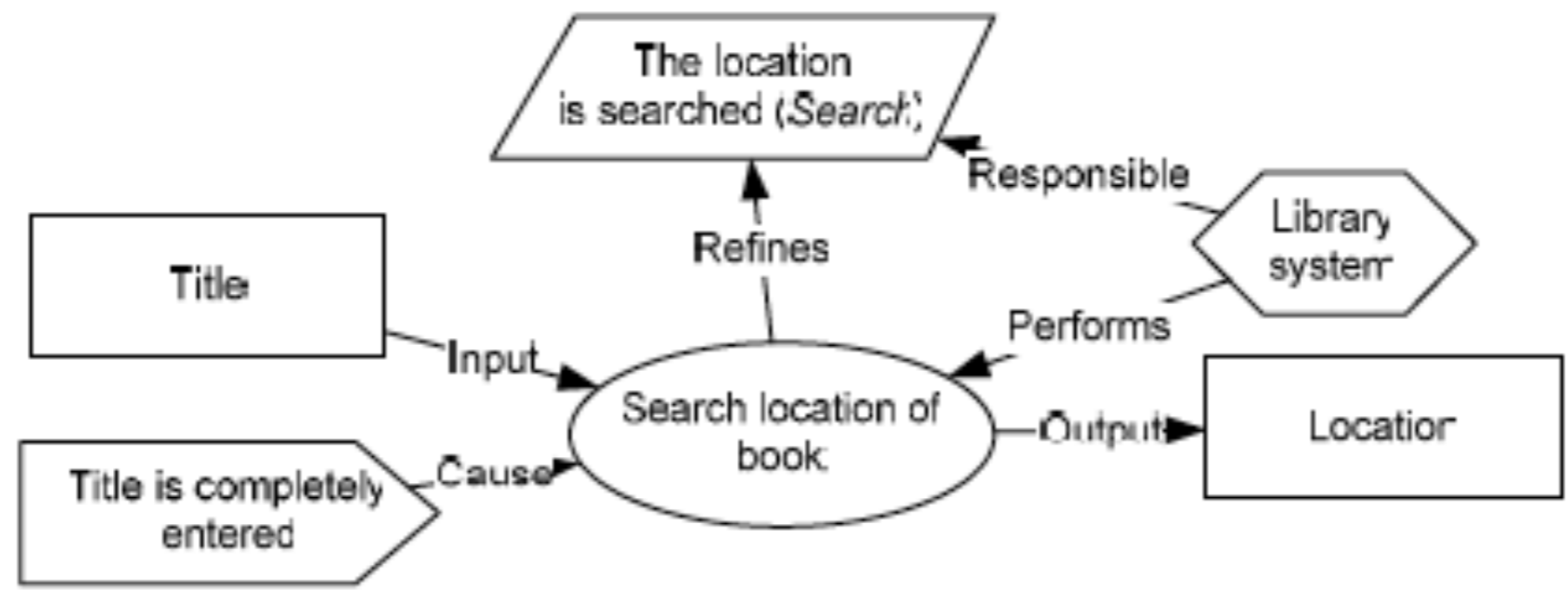
Zickert, Frank, "Evaluation of the Goal-Oriented Requirements Engineering Method KAOS" (2010). AMCIS 2010 Proceedings. 177.
<http://aisel.aisnet.org/amcis2010/177>



KAOS diagram



operation model



formal representation

Goal Achieve*[the location of a book in a library is found]*

Concerns *Title, Location*

RefinedTo *Input, Search, Output*

InformalDefinition The location of a book is found if it is displayed

FormalDefinition $\forall t: Title, l: Location: Output(l) \wedge SearchedTitle(t) \wedge IsLocationOfBook(l, t) \Rightarrow \circ BookFound(t)$

Association for Information Systems
AIS Electronic Library (AISEL)

AMCIS 2010 Proceedings

Americas Conference on Information Systems
(AMCIS)

8-2010

Evaluation of the Goal-Oriented Requirements Engineering Method KAOS

Frank Zickert

University of Frankfurt, mail@frankzickert.de

Zickert, Frank, "Evaluation of the Goal-Oriented Requirements Engineering Method KAOS" (2010). AMCIS 2010 Proceedings. 177.
<http://aisel.aisnet.org/amcis2010/177>

Authors	Year	Title	Citations	Rank
Dardenne, van Lamsweerde and Fickas	1993	Goal-directed requirements acquisition	1231	1
van Lamsweerde	2001	Goal-Oriented Requirements Engineering: A Guided Tour	759	2
van Lamsweerde	2000	Requirements engineering in the year 00: a research perspective	417	3
van Lamsweerde, Darimont and Letier.	1998	Managing Conflicts in Goal-driven Requirements Engineering	387	4
van Lamsweerde and Letier	2000	Handling Obstacles in Goal-Oriented Requirements Engineering	324	5
Darimont and van Lamsweerde	1996	Formal refinement patterns for goal-driven requirements elaboration	247	6

Class	Modeling concepts
Hierarchical decomposition	Refinement, patterns, architecture, operationalization, derivation, acquisition,
Goal-based reasoning	Verification, validation, interaction analysis, formal method, correctness, scenario, temporal logic, animation, model checking, runtime behavior
Obstacle management	Inconsistency, obstacle, conflict, divergence

**Is goal-oriented approach up to solve real
(engineering) complex problems?**

Before going further...

J. R. Silva

There are several methods and tools to support Requirements Engineering: NFR Framework, i^* Framework, KAOS, Problem Frames, and UML - to mention the most detached. We are focusing the discussion in KAOS and UML, and therefore it is also important to identify not only differences but also similarities.

UML for Systems

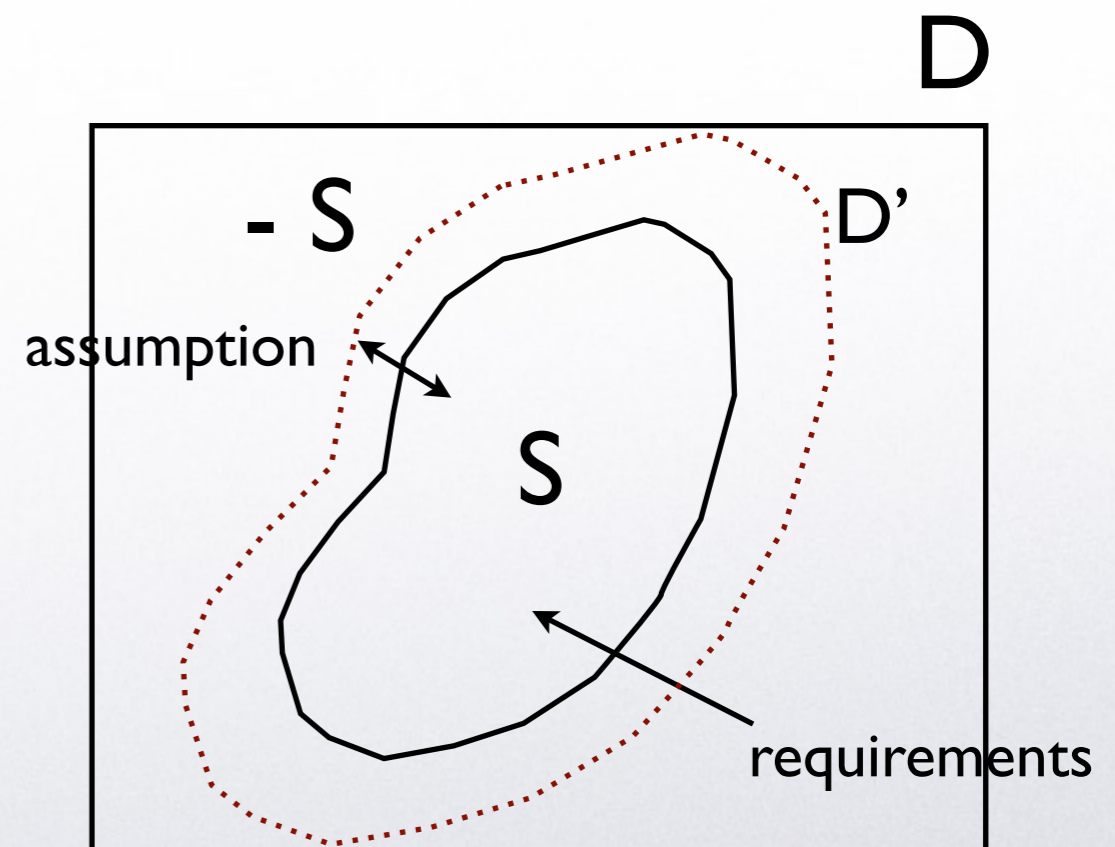
UML approach generally points to start with a meta-model of the system (to-be), followed by a domain model and a particular (functional and non-functional) requirements of the system instance - with all local knowledge.

KAOS for Systems

KAOS approach generally points to start with a meta-model of general goals and sub-goals of the system (to-be), followed by a domain model and a particular specific requirements to support goals - with all local knowledge.

We could say that the modeling "targets" are quite the same in UML and in KAOS

As mentioned in previous classes the main difference seems to be associated to the need to consider separately functional and non-functional requirements at the very beginning of the project.



However...

J. R. Silva

We experienced KAOS long enough to realize that goals **MUST** lead to requirements... and in this requirement form there is a distinction between functional and not functional. Thus, what is really the advantage?

UML for Systems

UML requirements must answer the question: which system-to-be must be built?

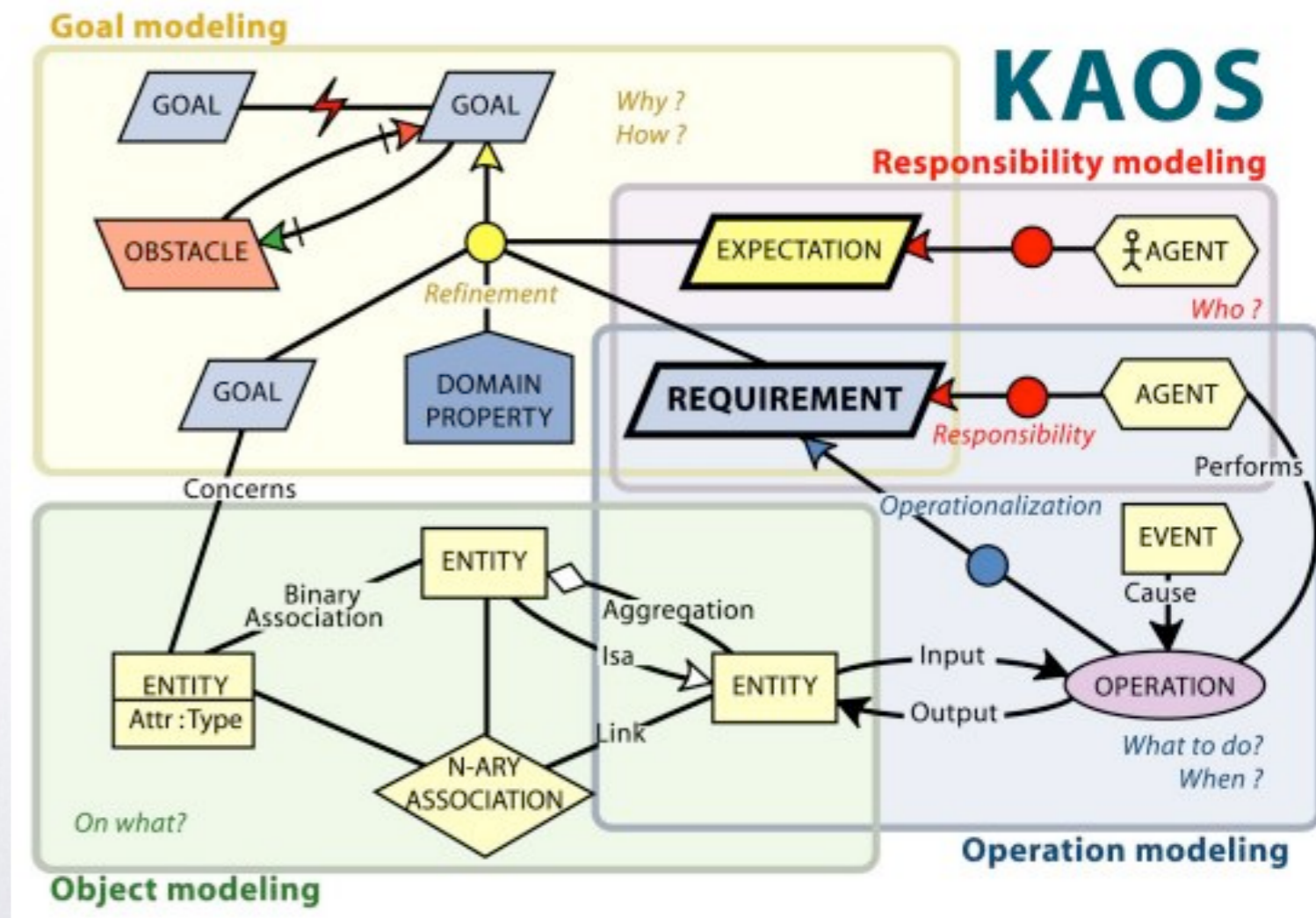
KAOS for Systems

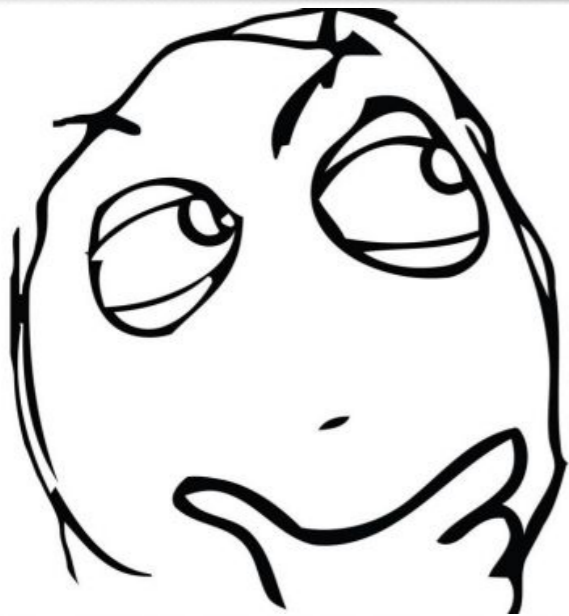
KAOS requirements must answer the question: if a system-to-be was built, how it would change the environment and serve the customers of the domain.

it would be "service-oriented"?

Thus, what is really the advantage?

The main difference is between “start with requirements” or “derive requirements that fit goals”. Without the reference of goals the identification, elicitation, capture, etc. of non-functional requirements become more difficult.





but... what are really non-functional requirements?

On Non-Functional Requirements in Software Engineering

Lawrence Chung¹ and Julio Cesar Sampaio do Prado Leite²

¹ Department of Computer Science, The University of Texas at Dallas

² Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro

www.utdallas.edu/~chung/, www.inf.puc-rio.br/~julio

Abstract. Essentially a software system's utility is determined by both its functionality and its non-functional characteristics, such as usability, flexibility, performance, interoperability and security. Nonetheless, there has been a lop-sided emphasis in the functionality of the software, even though the functionality is not useful or usable without the necessary non-functional characteristics. In this chapter, we review the state of the art on the treatment of non-functional requirements (hereafter, NFRs), while providing some prospects for future directions.

Keywords: Non-functional requirements, NFRs, softgoals, satisficing, requirements engineering, goal-oriented requirements engineering, alternatives, selection criteria.

A.T. Borgida et al. (Eds.): Mylopoulos Festschrift, LNCS 5600, pp. 363–379, 2009.
© Springer-Verlag Berlin Heidelberg 2009



Highly Influential

This paper has highly influenced 31 other papers.

[REVIEW HIGHLY INFLUENTIAL CITATIONS](#)



Highly Cited

This paper has 774 citations.

[REVIEW CITATIONS](#)

- ➔ ● Interface requirements: describe how the system is to interface with its environment, users and other systems. E.g., user interfaces and their qualities (e.g., user-friendliness).
- Performance requirements: describe performance constraints involving
 - time/space bounds, such as workloads, response time, throughput and available storage space. E.g., “system must handle 100 transactions/second.”
 - reliability involving the availability of components and integrity of information maintained and supplied to the system. E.g., “system must have less than 1hr downtime/3 months.”
 - security, such as permissible information flows.
 - survivability, such as system endurance under fire, natural catastrophes.
- ➔ ● Operating requirements: include physical constraints (size, weight), personnel availability, skill level considerations, system accessibility for maintenance, etc.
- Lifecycle requirements: can be classified under two subcategories:
 - quality of the design: measured in terms such as maintainability, enhanceability, portability.
 - limits on development, such as development time limitations, resource availability, methodological standards, etc.
- ➔ ● Economic requirements: immediate and/or long-term costs
- Political requirements

As briefly mentioned in Section 4, for example, the reference model states that (functional) requirements are satisfied through the collaboration between the functional behavior of the software system and the (functional) phenomena in the environment. KAOS [2] goes beyond these functional models and introduces general types of softgoals for the overall system, while addressing performance, accuracy and security concerns for the software system.

(17) (PDF) *On Non-Functional Requirements in Software Engineering*. Available from: https://www.researchgate.net/publication/215697482_On_Non-Functional_Requirements_in_Software_Engineering [accessed Apr 03 2019].

Evaluation of the Goal-Oriented Requirements Engineering Method KAOS

Frank Zickert
University of Frankfurt
mail@frankzickert.de

Roman Beck
University of Frankfurt
rbeck@wiwi.uni-frankfurt.de

ABSTRACT Software engineering is a complex task. But although there is no silver bullet that guarantees accomplishing this task, appropriate methods can support the engineer by addressing the characteristics that make it complex. The objective of this paper is to evaluate whether and how the goal-oriented requirements engineering method KAOS addresses these characteristics of complex tasks and thereby, whether it effectively supports software engineering. For serving this purpose, we conduct a literature analysis, which discloses core concepts underlying to the KAOS method, and we apply KAOS in two software development projects, which provide insights into KAOS in use. Our results show that KAOS, despite of some shortcomings, addresses all characteristics, but that applying it can be work intensive. Consequently, while KAOS supports software engineering, provided support must be weigh up against invested work.

Keywords (Required) KAOS, system engineering, task complexity.

Zickert, Frank, "Evaluation of the Goal-Oriented Requirements Engineering Method KAOS" (2010). AMCIS 2010 Proceedings. 177.
<http://aisel.aisnet.org/amcis2010/177>



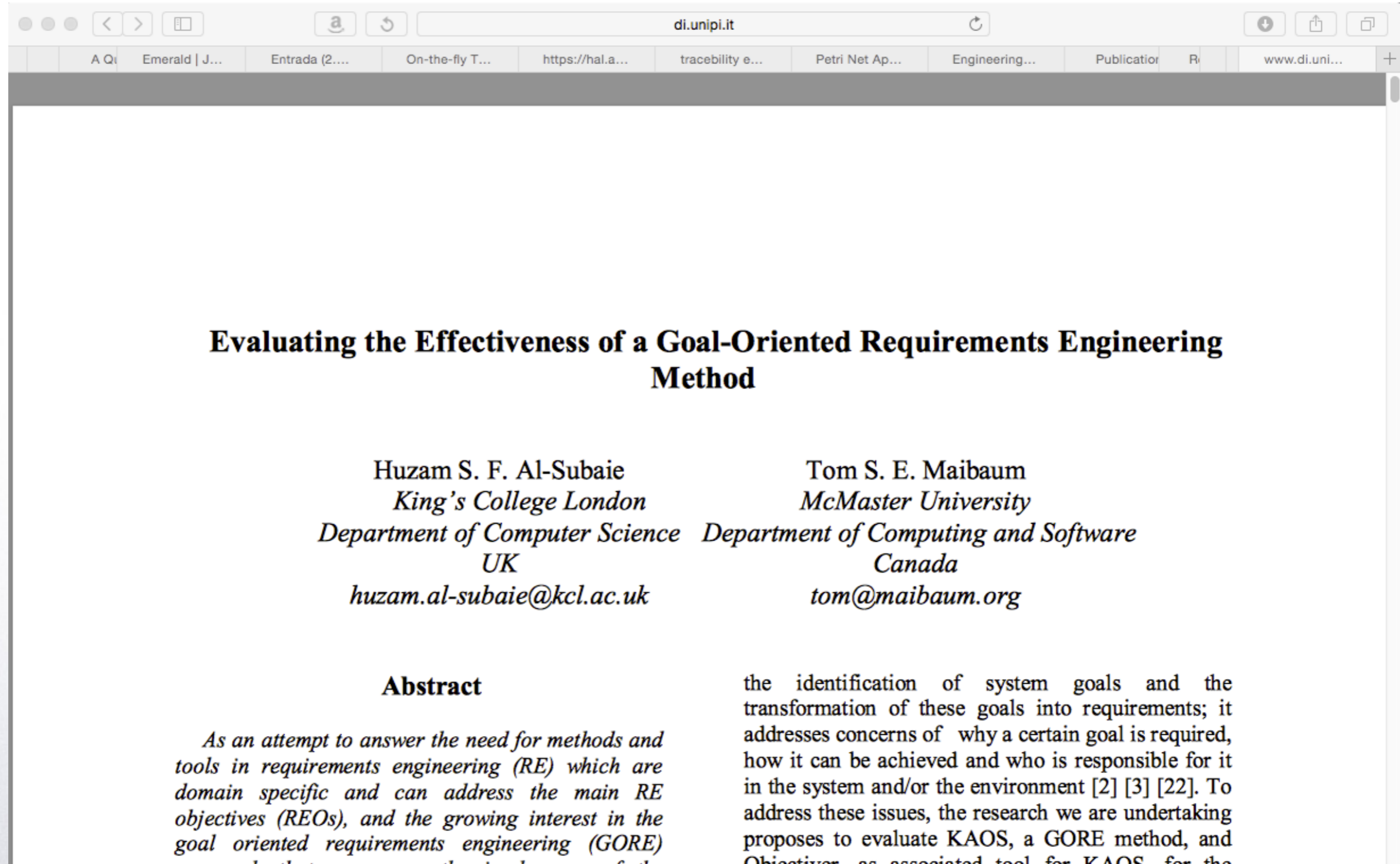
What makes a problem or task to be called "complex"?

Complexity characteristics	Short description
Multiple ways	In complex tasks, there is not a singular way of how the task must be done, but there are multiple ways, which may or may not result in the desired outcome. Multiple ways increase complexity, since the task-doer must decide, which way to follow.
Uncertain linkages	A task is complex, if the linkage between a way and its outcome is ambiguous. As a result, the way of achieving the goal cannot be planned upfront, because information does not become available until the task is performed.
Multiple outcomes	Complex tasks have multiple outcomes that require attention. Typically, each outcome entails a separate information processing stream, which increases the amount of information that has to be processed concurrently.
Conflicting interdependence among outcomes	A task is complex if achieving one desired outcome conflicts with achieving another desired outcome. As a consequence, ways that achieve both outcomes are less obvious and thus more difficult to find.

Campbell, D. (1988) Task complexity: A review and analysis, The Academy of Management Review, 13, 1, 40-52

KAOS approach to complex problems (systems)

Complexity characteristics	Modeling concepts used in KAOS	Observations in the cases
Multiple paths	- Hierarchical decomposition	- OR-relationships. Concurrent consideration and discussion of alternative solutions to the problem
Uncertain linkages	- Goal-based reasoning (verification) - Hierarchical decomposition (OR-relationships)	- Satisfaction of all goals were verified by using KAOS, whereas the engineer had no information about satisfaction of one out of seven goals
Multiple outcomes	- Goal-based reasoning (concurrent verification of high-level goals)	- Goals about the software under construction were considered - Resulting effort could not be assessed using KAOS, but it was an important factor in the cases
Conflicting interdependence among outcomes	- Obstacle management (resolving inconsistencies)	- KAOS supported finding a resolution for the wrong assumption



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

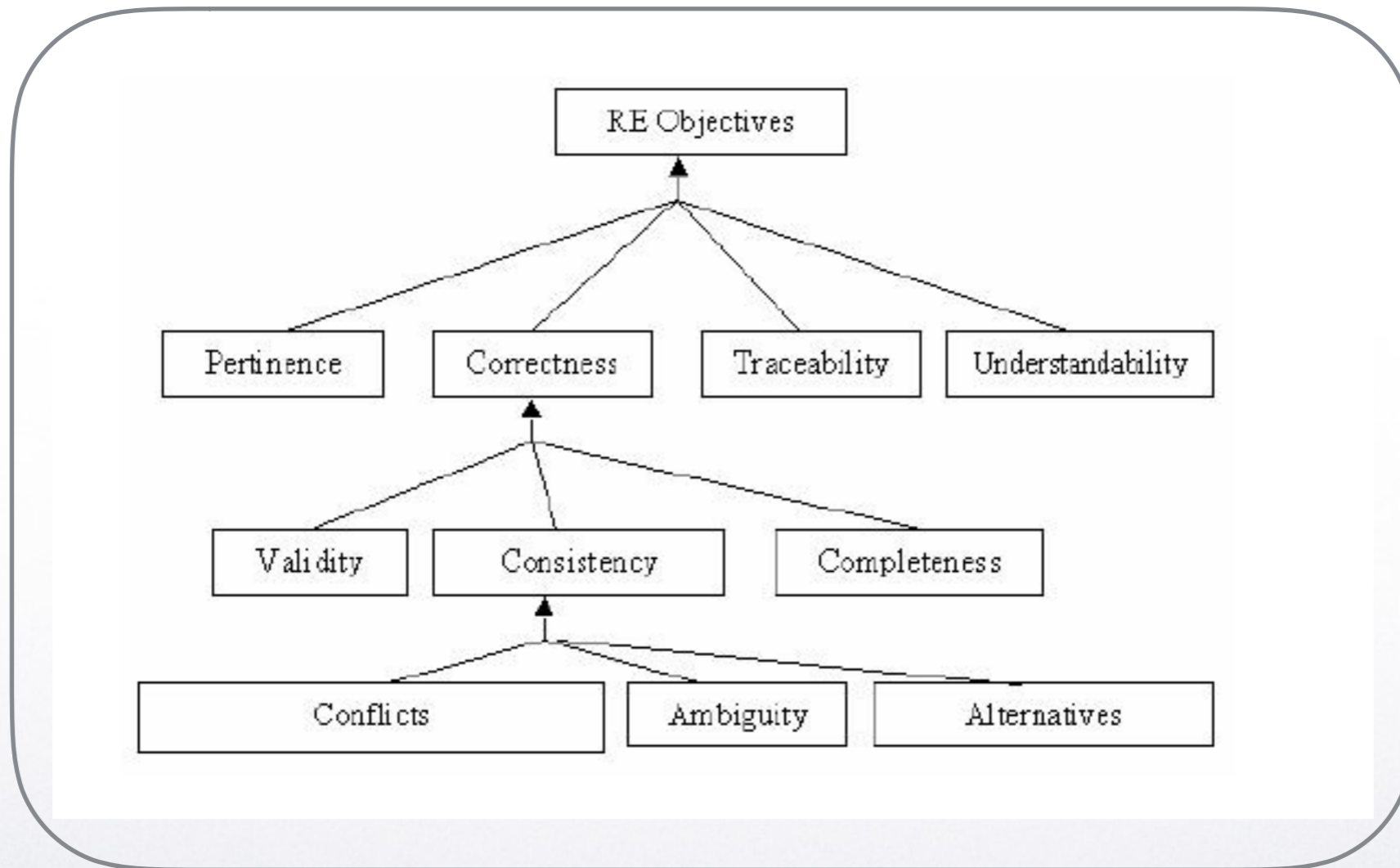
Tom S. E. Maibaum
McMaster University
Department of Computing and Software
Canada
tom@maibaum.org

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) method, this paper presents the evaluation of the effectiveness of the KAOS method, for 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 Objectiver, as associated tool for KAOS, for the

RE objectives (REOs)

The REOs are defined as those attributes that need to be achieved to produce complete, valid, correct, pertinent, consistent, traceable, unambiguous and understandable requirements



KAOS

KAOS is a prospective method based on REOs that claims to provide a complete model of requirements from an informal phase up to specifications once some basic rules of composition were respected. Formal modeling can also be generated base on the specification model.

Objectiver

ObjectivER is a computer application that claims to support a systematic process to conduct KAOS modeling, generate the formal specification and automatically synthesize the requirement documentation.

- A. Fully achieved:
1. The objective completely supported by:
 - Providing step-by-step assistance to the developer.
 - Providing explicitly detailed information in the literature.
 2. All aspects of the objective are covered:
 - No number of error reports mapped to this particular objective.
 - No number of negative observations mapped to the objective.
- B. Strongly achieved:
1. The objective largely supported by:
 - Providing step-by-step assistance to the developer.
 - Providing explicitly detailed information in the literature.
 2. All aspects of the objective are covered but the full achievement of the objective depends on the expertise and the talent of the developer:
 - Some error reports from the first target problem mapped to this particular objective but not from the second one.
 - Some negative observations to the objective from the first target problem mapped to this particular objective but not from the second one.

C. Partially achieved:

1. The objective partially supported by:
 - Providing some assistance to the developer.
 - Providing some information in the literature.
2. Part of aspects of the objective are covered but the full achievement of objective depends on the help of another tool or method:
 - Lightweight error reports mapped to this particular objective.
 - Lightweight negative observations mapped to this particular objective.

D. Slightly achieved:

1. The objective supported in very limited degree by:
 - Providing Limited assistance to the developer.
 - The literature does not cater for the objective.
2. Very limited achievement of objective:
 - Developer needs to extend the method to overcome its limitation.
 - Heavyweight error reports mapped to this particular objective.
 - Heavyweight negative observations mapped to this particular objective.

E. Fail to be achieved:

1. The objective unrealised
 - The objective is not addressed.
2. The objective totally left out.
 - The objective is not referred to in the literature.

UML for Systems

UML requirements must answer the question: which system-to-be must be built?

KAOS for Systems

KAOS requirements must answer the question: if the system-to-be was built how it would change the environment and serve the customers.

it would be "service-oriented"?

We can reach a partial conclusion (a more conclusive statement still demands more research and comparison) that KAOS (as a representative of GORE methods) provides a systematic approach to treat requirements in the same direction we treat knowledge, without anticipating decisions and avoiding subliminal references to "solutions" in the RE process - describing only the problem.

The (KAOS) method also inherit concepts proved to be successful on other conceptual methods, such as structured analysis, top down analysis, design reusability, separation of concerns (object orientation) while preserving the way to compose sound requirements that could be formally represented (using LTL in ObjectivER or using Petri Nets, in the current D-Lab approach).

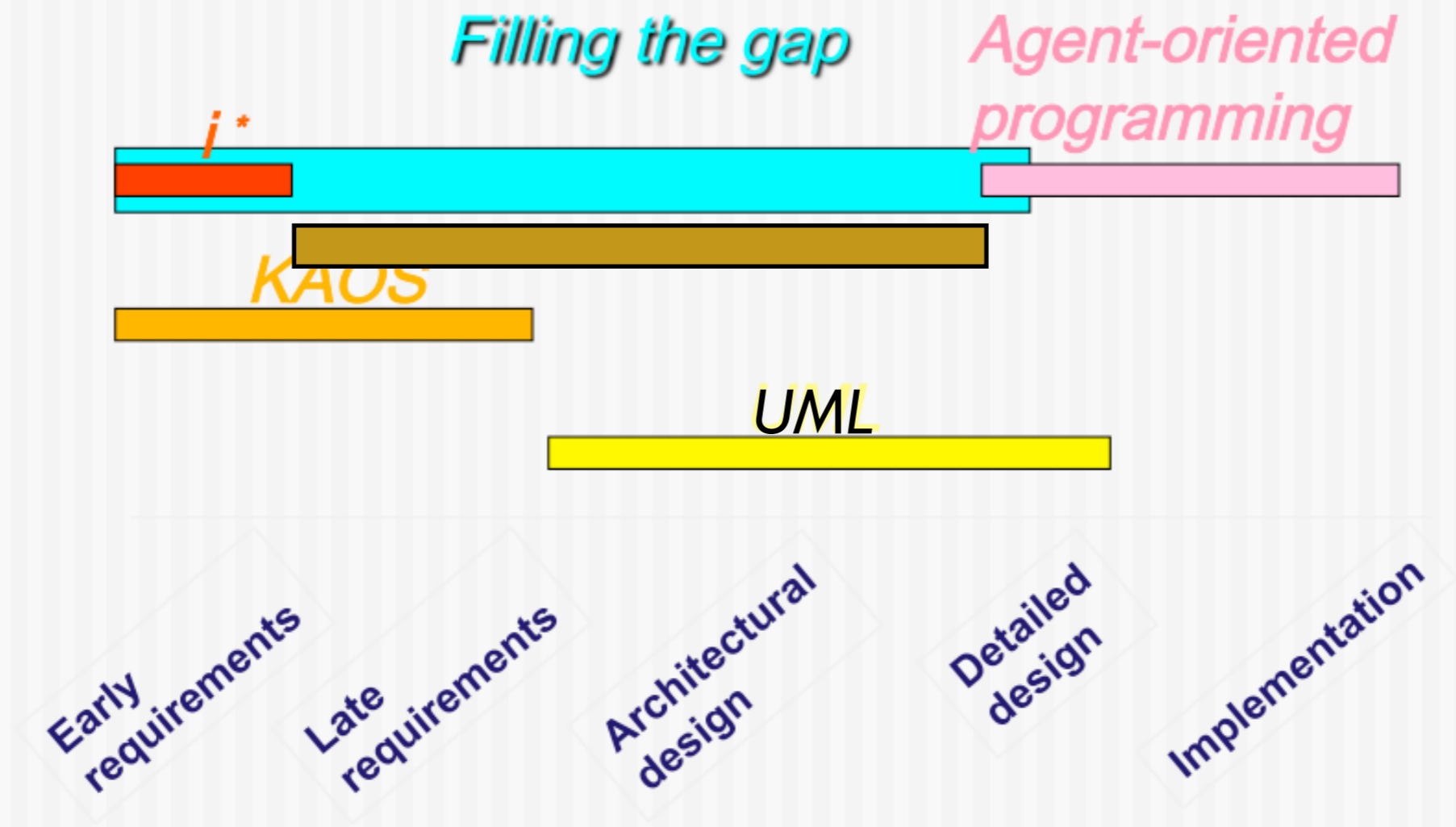
ObjectivER can implement and conduct the user through KAOS approach (addressing the target REOs) and can also generate documentation which can support maintenance and cover all life cycle of systems.

Remaining questions:

- i) besides complexity can Objectiver handle very big systems design?
- ii) can it handle systems of systems?

How Petri Nets comes in?

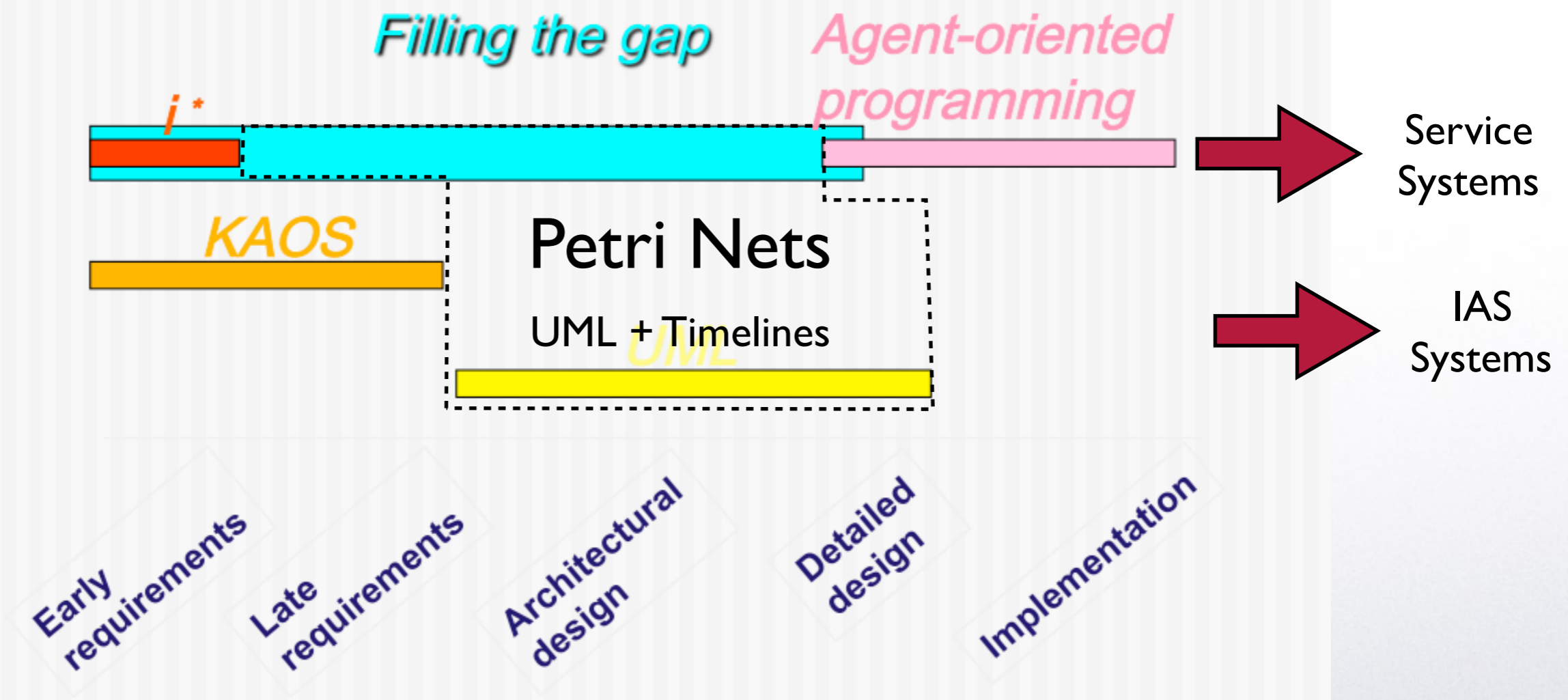
Tropos in Perspective



© P. Giorgini

A knowledge level methodology

Tropos in Perspective



A knowledge level methodology

© P. Giorgini





Obrigado

Reinaldo