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 those 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* Framework [2], GBRAM [6] [8], NFR [9], KAOS [1] [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 covers the i* Framework and the third 2.3 presents the KAOS Framework.

2.1. Relating Activities and Goals to Requirements Engineering

There is no common definition for Requirements Engineering (RE) activities. Pohl [4] uses elicitation, negotiation, specification, and validation and Lamsweerde [3] uses elicitation, elaboration, organization, analysis, negotiation, documentation, and evolution as RE activities. In this work we adopted elicitation, modeling and analysis as the RE main activities. This is Leite´s [12] view, which we believe is both simpler and concise. Elicitation means understanding the Universe of Discourse and discovering the software requirements. Modeling means describing requirements information by means of specific models. Analysis means verifying and validating requirements models.

GORE considers organization and actors' goals as the source of requirements (both functional and non-functional, so we need to first elicit goals in order to elicit requirements. Different from the usual RE approaches, which model just the requirements, GORE approaches model goals, which are the reason why requirements are needed. For i* there are two kinds of goals: concrete goals, commonly called goals, and softgoals. i* Framework is precise when it defines goal: "A goal is a condition or state of affairs in the world that an actor would like to achieve" [2]. And softgoal is a kind of goal that is satisficed, rather than satisfied, (a term coined by Hebert Simon to denote lack of precision in the perception of satisfaction). KAOS understand goal as i*.

Authors commonly consider that an NFR is a softgoal, although we consider softgoals as different from NFRs. Softgoals are used for qualifying specific topics and how they are related to others [9] [2]. NFRs are used in a general way for designating a quality that the system must have. NFRs are related to software and softgoals and concrete goals are related to actors' interests and motivations. NFRs and softgoals represent quality attributes or constraints, though. Both goals and softgoals will be operationalized by functional requirements in the software system, if implemented.

In a general, GORE methods use a combination of structures either top-down/bottom–up or AND/OR graphs which are elicited from why and how questions. Goal-oriented methods ask why the functionality is necessary and how the functionality could be implemented. Working at the level of the "rationale" in models, GORE deals with software functionality considering different alternatives and criteria for selection of alternatives [4].

2.2. The i* Framework

i* Framework [2] was proposed in 1995 by Eric Yu's thesis and since 1995 it has been used by other methods. GLR, an ITU standard [7] and TROPOS [19], were based on i*. i* modeling framework [2] models organizational contexts based on the dependency relationships among actors. The central idea of i* is that: actors depend on each other for goals to be achieved, for resources to be provided, for tasks to be performed, and for softgoals to be satisficed[1]. i* deals with two kinds of models: the Strategic Dependency (SD) model and the Strategic Rationale (SR) model.

For i* "an actor is an active entity that carries out actions to achieve goals by exercising its know-how" [2] and an actor is considered as a "super class" for agent, position and role, Leite et al. [15].

The Strategic Dependency SD model depicts the organizational environment context of the system as a network of dependency relationships among actors. This network consists of a set of nodes and links where each node represents an actor and each link maps out one dependency between two actors. Figure 1 shows in the upper right side that one actor (called depender) depends on one or more actors (called dependees) through a dependency (either an end node or a component node called dependum). A dependum may be one of four types when it is an end (or when it is a component node): a softgoal (or a SoftgoalFor), a task (or a SubTask), a resource (or a ResourceFor), or a goal (or a Subgoal).

The Strategic Rationale (SR) model expresses organizational context and also internal relationships among the intentional elements within an actor's reasoning. Rationales are modeled through means-ends relationships, task decompositions, and softgoal contributions. Figure 1 shows these signed by three message tags. The means-ends relationship (means node and end node in the Figure 1) has a means that represents a softgoal (softgoal node) or a task (task node) and an end that can be a Task, a Softgoal, a Resource, or a Goal. The relation is called GT when the end is a GOAL and the means is a TASK, and so on defining five means-ends types {GT, TT, RT, ST, SS}. But, when the relations are either ST (TASK – SOFTGOAL) or SS (SOFTGOAL – SOFTGOAL) they are called "softgoal" contributions. A **contribution** may be of four **type**s: {break, hurt, ignored, help, make}.

---

1 "Softgoals are satisficed, rather than satisfied", NFR framework [9].

In an SR model an actor can have one or more goals. Recursive representations can occur too: (1) a softgoal may be a means and may be an end for another relationship; (2) a goal in a task-decomposition may be an end in a means-ends relationship. Figure 1 shows that two other relations occur in a task-decomposition: a Subtask restricts the task (task node as the means) and Softgoal qualifies the task (task node as the means).

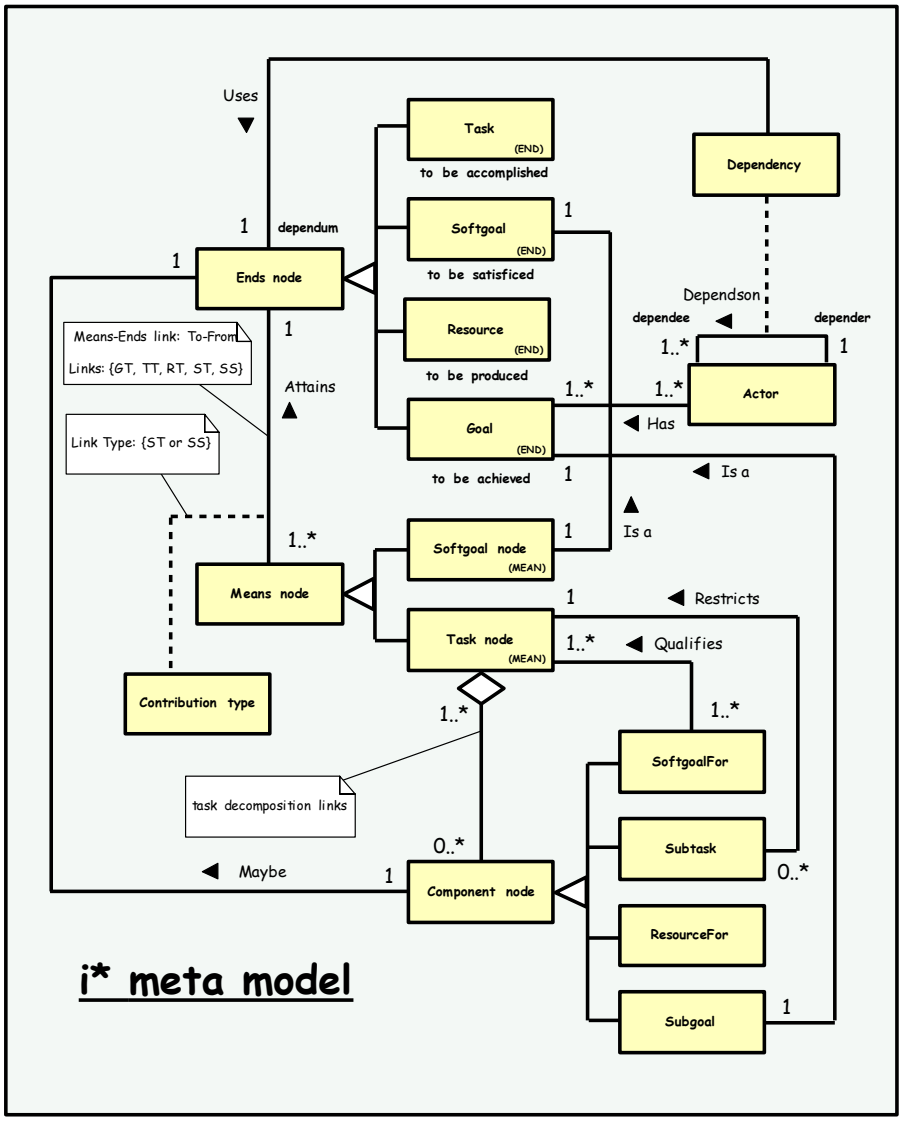

Figure 1. i* Framework meta model [5]

Figure 2 is an example of i* SD model for the EC - Expert Committee example. The model illustrates the strategic dependency relationships among actors. In this paper, we only focus on the relationship between chair and author. The SD model shows that the chair depends on the author to achieve the goal "Article BeSent", to satisfy the softgoal "Quality [article]" and to provide the resource "Camera Ready" furthermore, the author depends on the chair to get the resources "Quality Directions" and "Review result".

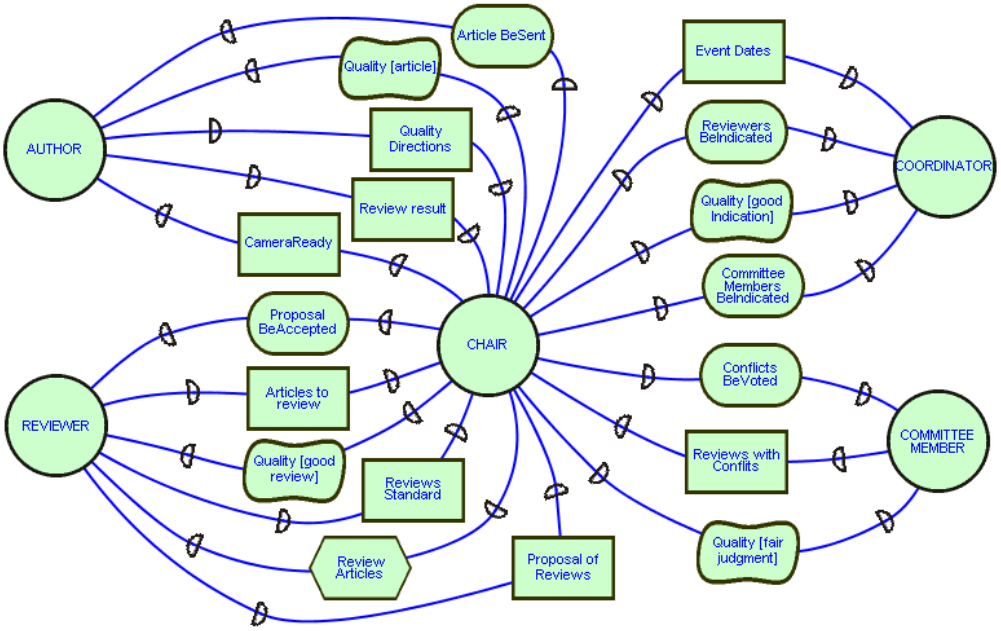

Figure 2. SD Model (EC - Expert Committee) adapted from [18]

Figure 3 portrays the SR model involving the actors author and chair. See for example chair when the chair is dealing with authors, chair's main goal is "Best Articles BePublished", which has one only task to achieve the goal. The diagram shows the task "Manage submitted articles" has three goals (or sub goals) "Articles BeReceived", "Articles BeReviewed", and "Articles BePublished". These associations, by task decomposition, mean that these goals are part of the task and only if the chair achieves all these goals ("and" association), the task will be concluded.

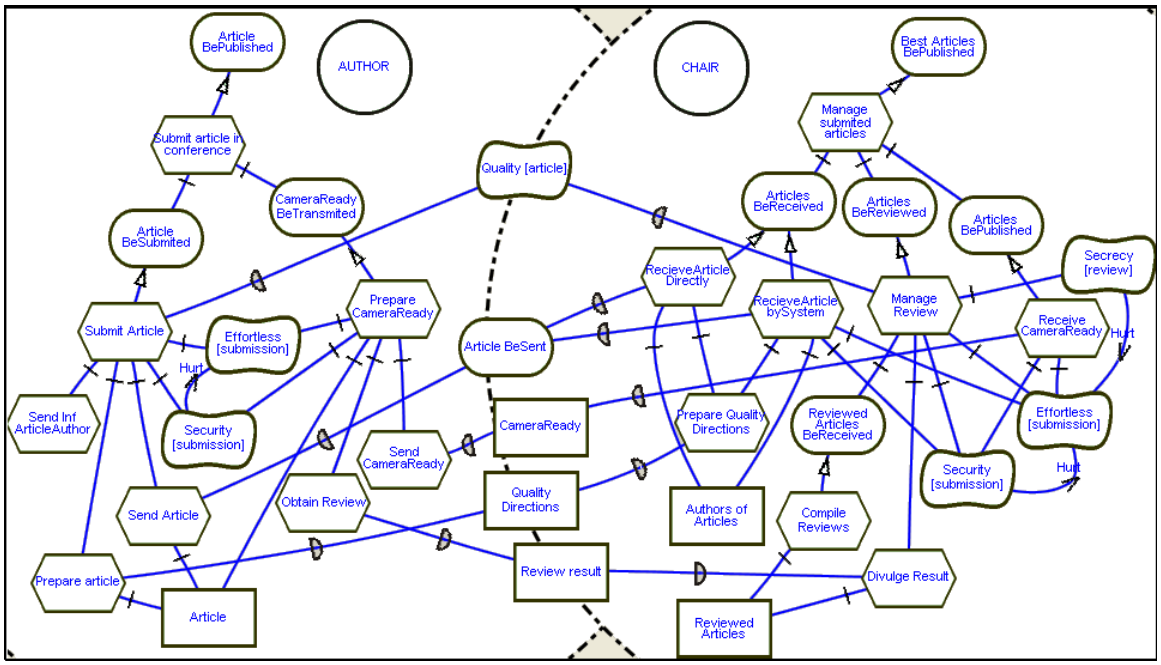

Figure 3. SR Model (EC - Expert Committee), this example, from [18], is part of the SR Model.

The diagram shows also that the goal "Articles BeReceived" has two alternatives ("xor" association), two tasks may be performed "Receive Articles Directly" or "Receive Articles bySystem". The goal dependency "Article BeSent" associated means a goal that must be achieved for both alternatives (tasks). "Articles BeReviewed" has a task "Manage Review".

The model shows that an essential task "ReceiveArticle BySystem" depends on "Article BeSent" from author; needs "Prepare Quality Directions" to authors, in order to give the quality specifications and directions; creates a resource "Authors of Articles" to other tasks and provides chair with some softgoals: "Effortless [submission]", "Security [submission]", and "Secrecy [review]". One can see in Figure 3 that both the softgoal "Security" and the softgoal "Secrecy" contribute "negatively" (hurt) to the softgoal "Effortless" because probably chair will have to enter with a password for this process. By showing these softgoals in the SR model the software engineer has the information about the concerns that must be operationalized.

2.3. The KAOS Framework

KAOS has been developed and refined for over 15 years of research with the development of tools and with experience in several industrial projects [1]. KAOS, according to Lamsweerde [1], [10] means Keep All Objects Satisfied. This method is considered a multi-paradigm model that allows a combination of different levels of expression and rationale: semiformal for modeling and structuring goals, qualitative for alternative selections and formal for the critical elements. In general, KAOS modeling has an external graphic semantic layer with concepts, attributes and relationship and an internal formal layer assisted by temporal logics. In Figure 4 we show a KAOS meta-model built based on the meta-model proposed by Heaven and Finkelstein (2004) [17].

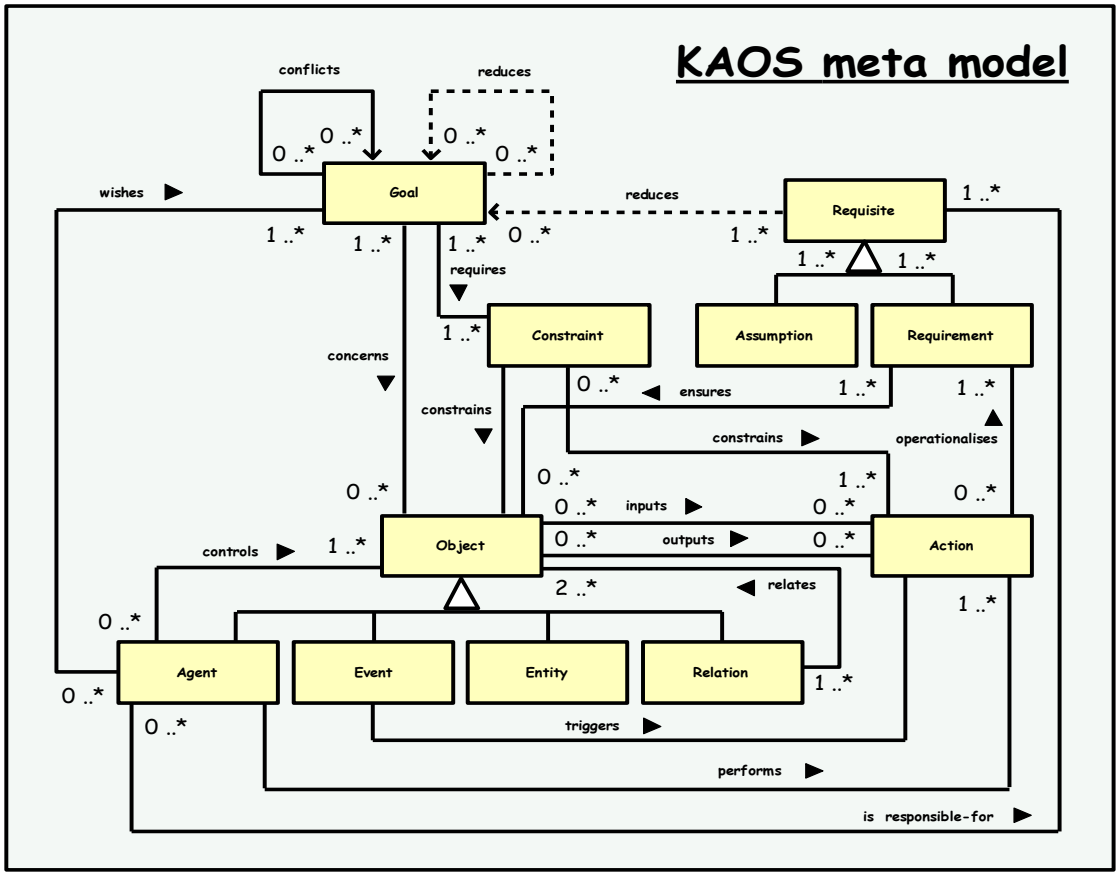

Figure 4. KAOS meta-model (adapted from Heaven and Finkelstein 04 [17])

The KAOS concept of goals (Figure 4) can be defined as a prescriptive intention statement about some system whose satisfaction, in general, requires cooperation of some agents that configure the system. Goals are reached by requisites that can be requirements that are operationalized into specification of software operations (actions) or assumptions that express behaviors performed by external agents. Software agents are active components that perform some operations (actions) which can reach requirement they are responsible for. Objects can be specified to describe the project structural model and they can be passive (entities, relations or events) or active (agents). Agents are related together through their interfaces made of object attributes that are controlled by those agents. Obstacles (constraints) and goals relations (conflict goals) are used to permit analyses scenarios where goals are obstructed and not satisfied, therefore contributing to identify vulnerabilities [10].

KAOS proposes four visions of the problem which are treated by the following models: Goal Model, Object Model, Responsibility Model and Operational Model. Those models are based on goals, requirements, agents, expectations, obstacles, domain propriety, operations, entities, event, and relationship and associations among those concepts.

Figure 5 was built based on the Objectiver tool documentation [13] and presents an example of the four models based on the Expert Committee example.

The Goal Model is a set of goal diagrams inter-related that contains goals, sub-goals, expectations, requirements, domain proprieties, agents, conflicts, obstacles, resolutions and refinements of obstacles. Goals are organized in a top-down AND/OR hierarchy. The refinements of goals end when a sub-goal is performed by an agent. Goals can refer to services (functional goals) and to the quality of services (non-functional goals). Each goal in the model is in general justified by another goal that explains why the goal was defined in the model. Each goal is refined as a set of sub-goals describing how the goal in a higher level can be reached. Requirements and expectations are modeled in a lower level and have to be associated to an agent. Obstacles (in red in the Goal Model of Figure 5, for instance: see System Invasion) can be introduced in the model to permit goal alternatives and analyses of vulnerabilities.

The goal oriented process of KAOS is developed through activities: (i) identification of goals, (ii) formalization of goals, (iii) modeling of objects and identification of state variables, (iv) detection and resolution of goal conflict levels, (v) refining of goals and identification of agent responsibilities, (vi) generation of obstacles and resolution to goal fulfillment and (vii) derivation of operation requirements from system goals.

The Responsibility Model represents the agents' responsibilities and interface describing for each agent the requirements under his responsibility and expectation assigned to the agent. This model contains all diagrams of responsibilities where each diagram shows all requirements and expectations under the agent's responsibility. The responsibility and interface attribution are defined on the Goal Diagram and the Responsibility Model is generated automatically.

The Operation Model describes all agents' behaviors that are necessary to reach their requirements. Behaviors are expressed in terms of operations and tasks performed by the agents. Those operations work with objects defined in the Object Model which can create, change their states and activate other operations. This model represents the functional vision of services that are assigned to the problem under study in which the following concepts can appear: operations, agents, entities and associations, events and restriction.

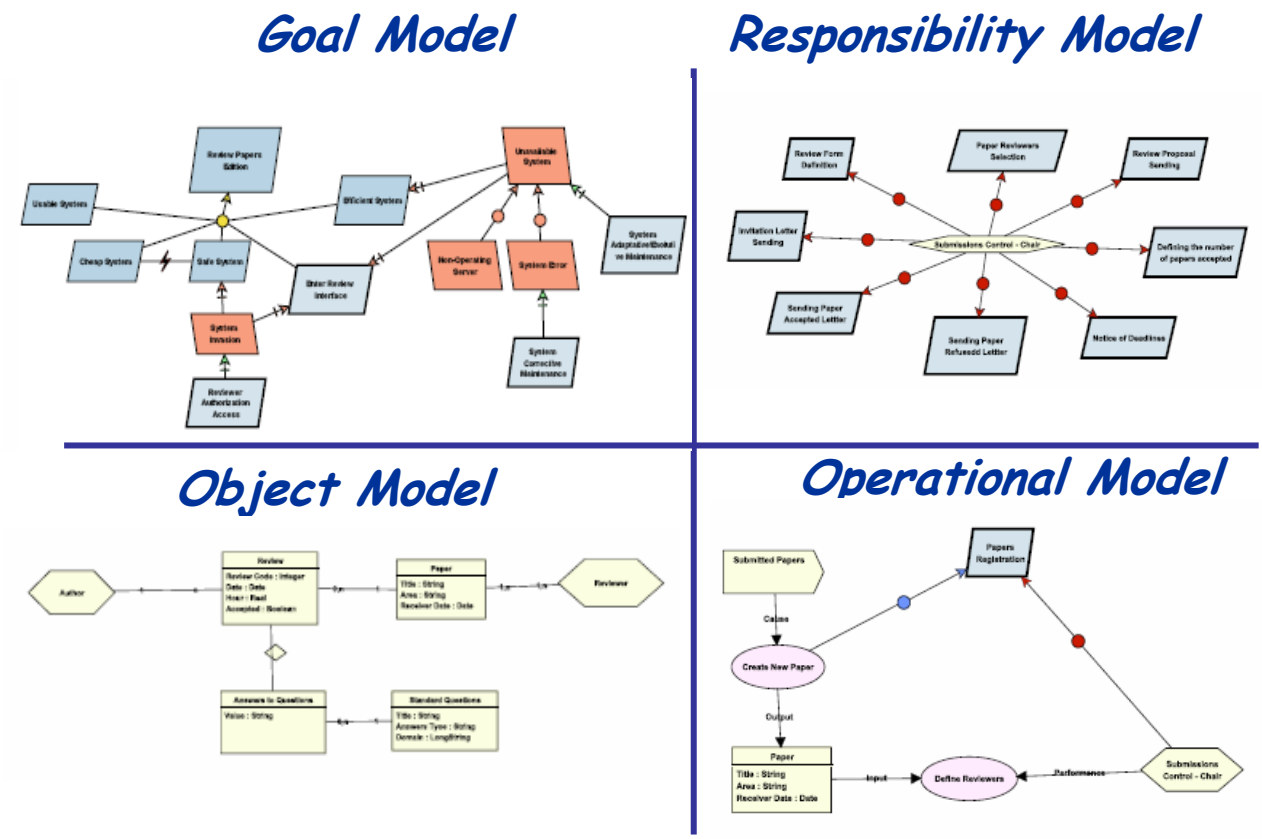

Figure 5. KAOS Models. Examples from Objectiver [13] show the four perspectives of KAOS Models.

3. The Comparison Process: Analyzing the Frameworks

This work was developed based on the following activities: (i) study of the methods KAOS and i*, (ii) modeling, using KAOS and i*, the classic case study of Conference Expert Committee System, (iii) identification of essential issues of GORE representation, (iv) representations analysis of each model, and (v) qualitative analysis of both methods.

KAOS modeling was supported by the Objectiver tool which helps the construction of the models and diagrams. Objectiver provides documentation support and an interface easy to use. Objectiver is a commercial tool, and there are limitations for the free academic license.

i* modeling used the OME. This tool is open-source and supports both SD and SR modeling. OME tool helps the edition of diagrams, but has limitations on editing operations. For instance, the OME tool lacks a support for “cut and paste”, which of course is a burden for designers.

Five issues were used to drive the qualitative analysis: (i) treatment of actors/agents´ relations in the target social context; (ii) the goal representation model; (iii) task modeling, which is responsible for showing the behavior of functional requirements, (iv) non-functional requirements and (v) risk analysis that allows the study of exceptions, pessimist and alternative scenarios considering the functional and non-functional requirements and their influences.

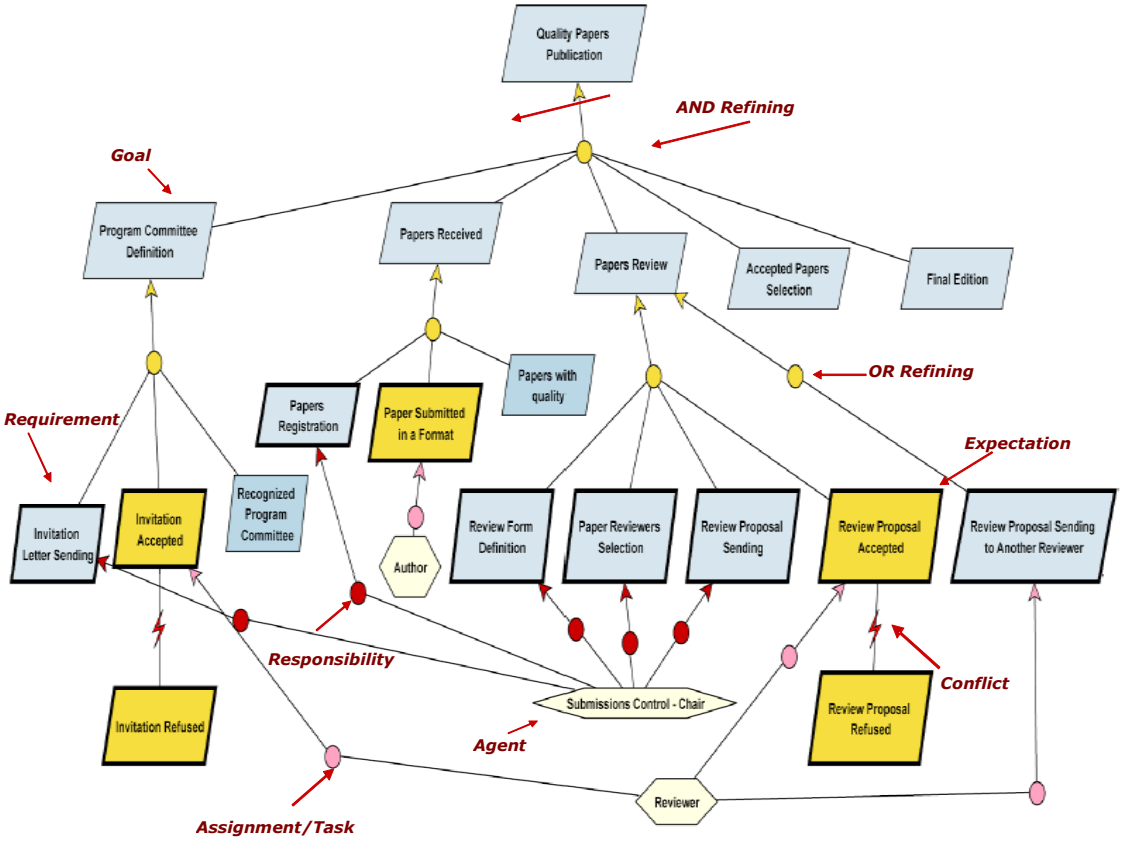

Figure 6. The EC example modeled in KAOS

KAOS uses four models but the main focus is the Goal Model, and the others are derived from it, while i* has only two diagrams interconnected through strategic dependencies. Although the requirement definition process in KAOS and i* uses similar concepts: goals, constraints/softgoals, operations/tasks, objects/ resources, they have different approaches to model and rationalize about these concepts. KAOS from the beginning of modeling is goal-oriented and its goals can be decomposed into other goals until they can be operationalized. The focus is on goal refinement. In i* the strategic dependency guides the SD Diagram modeling. In the SR Diagram the strategic dependencies are defined in detail explaining how each actor reaches the goals. Both methods define how system goals can be reached and the mechanism of alternatives can be analyzed.

In the following sections we are going to detail the similarities and disagreements between the two methods always trying to exemplify the discussion based on Figures 1 to 6.

3.1. Goal Model

In KAOS the goal model starts with the decomposition of goals into sub-goals until getting to a requirement or expectation where the software agents are responsible for the requirements and the expectations of the human agent are expressed to an interface. This model is top-down: it begins with the most strategic goal and refines it by sub-goals. In this model, conflict goals can be expressed as in Figure 6 (Review Proposal Accepted and Review Proposal Refused) or, for example, the goal Safe System can be in a conflict with the goal Cheap System (Goal Diagram of Figure 5). Some goals can be in conflict with other goals if neither can be satisfied simultaneously.

The KAOS concept of obstruction, obstacles, obstacle refining and resolutions allow for the definition of a failure or crash scenarios for the goals and the risks associated with them. For example, in Figure 5 the obstacle “System Invasion” is an obstruction to the goal “Safe System” and the requisite “Enter Review Interface”. The requisite “Reviewer Authorization

Access" is an obstacle resolution to the obstacle "System Invasion".

In i* we do not have a specific goal model for the conflict analysis, because SD focus is on the intentionality dependencies of actors (agent, a role or a position). In a SR Diagram goal details are defined in the intern rationale of the actor. The intentionality defines relations between different actors showing the responsibility and strategic dependencies among them. In this way it is possible to express scenarios where goals can be performed by different actors and relations can be explicit. In KAOS, these explicit relations cannot be represented directly.

The goals are satisfied through one or more tasks that can be decomposed in other goals, softgoals, task and resources. The focus of i* is not only on goals but on the concept of intentionality and strategic dependencies that are sufficient to cover the multi-agent scenarios because the cooperation relation among the agents can be explicit.

The framework i* also allows modeling of critical situations by using softgoals and their contributions attached to the SR model. This diagram does not rationalize about the conflicts, obstacles and resolutions but through positive or negative contributions to softgoals. For instance, in Figure 3, the option for the alternative task "Receive Article by System" has different impacts on Security, Secrecy and Ease of Use (Effortless).. So obstacles can be modeled as negative contribution to desired softgoals and resolution would require a different alternative..

3.2. Actors

KAOS has the agent concept, which can be software or human. They are represented in the Goal Model and in the KAOS Responsibility Model, defining requirements and expectations related to agents. Agents are atomic entities. .

i* uses the more general concept of actors, which can be expressed as agents, roles and positions. For example, in the KAOS Goal Model in Figure 5 the agent "Reviewer" has the responsibility of meeting the expectation of "proposed revision accepts" however it is not clear that the agent "Chair" depends on the "Reviewer" for the goal to be satisfied as it can be seen in SD Diagram in Figure 2.

Thus we can conclude that the KAOS agents as defined in the Goal Model and the Model Responsibilities do not directly show the relationships between the actors. If we i* actor taxonomy we can also represent that the actor "Chair" is a position that a member of the program plays.

In summary, KAOS works with the agents concept related to requirements or expectations in the Goal and Responsibility Model. i* has structured of actors allowing for a finer grain distinctions among different types of actors as well as their relationship and as such enriching the description of strategic dependencies.

3.3. Non-Functional Requirements

KAOS defines non-functional requirements in the same way as functional goals. Although the Objectiver tool [13] uses a different symbol for non-functional goal, they are still treated as goals. The conflict notion helps the representation of flexible goals (softgoals) conflicting, but there is no special treatment.

i* defines the non-functional requirements as softgoals that can contribute positively or negatively to another softgoal. In i* a task can be decomposed into a softgoals, as seen in Figure 3. This means that the task in question may have quality attributes related to it and that they will impact other goals, softgoals or tasks in the model. When a softgoal is a component in task decomposition, it serves as a quality attribute for that task, guiding (or restricting) the alternatives selection to the task decomposition. This representation allows a different contribution analysis of the softgoals that can be addressed simultaneously.

Such analysis can also be held in KAOS, but focusing on anti-requirements scenarios which do not explicit contributions from non-functional requirements.

3.4. Tasks

KAOS has an Operation Model that defines tasks as an operations concept relating them to events, agents, entities and requirements. So KAOS operationalizes a requirement by making it explicit and presenting the event, agent and involved entities. Tasks are related to agents and expectations.

In i* the task concept is present in the SD model, where actors can depend through a task which means that the task should be carried out by "depender". In the SR model, tasks can make goals operational and may also be decomposed to model alternatives. This representation can also link resources (entities) and it is defined in the context (the dotted circle of Figure 3) of an actor who is responsible for the task.

In short, both methods deal with tasks but KAOS allows tasks to be related to events and as such more detailed in an operational sense. On the other hand, an i* task can be decomposed and related to other tasks, goals, softgoals and resources. i* tasks, when linked by the means-ends link, serve as alternatives, which may be

linked to softgoals by decomposition, and as such being qualified and amenable to be used in a reasoning process.

3.5. Risk Analysis

In KAOS the risk analysis is done by detailing the obstacles. By exercising possible obstacle scenarios we can assess the risk of the system and propose means as to deal with such foreseeing obstacles. This analysis is especially important in the study of the application vulnerability analysis. In addition, the model can be analyzed with respect to conflicting goals. The formalization of obstacles and goals enable the formal treatment and the use of deduction mechanisms, thus improving the possibilities for automated verification.

In i*, risk analysis may be done by analyzing the reason (rationale) inside the actor. By detailing undesired situations (obstacles) as softgoals to be avoided and softgoals contributions as relationships to the undesired softgoal, it is possible to map the conditions (contributions type, see Figure 1) to tasks that resolve the undesired condition, by denying it. The relationship means-end allows the alternatives representation and the impact analysis of softgoals. This analysis is powerful and useful because it can examine solutions to conflicts in the stage of definition of requirements. This analysis is qualitative but also amenable for automated treatment.

Table 1. Five criteria comparison with drawbacks and potentialities.

	i-star (i*)	KAOS	Potentialities
Goal Model	The models represent both goals and softgoals . The models also show tasks and entities. Models may be very complex and huge.	The goal model does not represent tasks and entities. Models are divided into four different ones, which may be confusing. Models are very detailed.	There are some i* extensions in order to control complexity named Visions [21] and SDsituations.[18].
Actors	Does not explore role and position. Shows collaboration in a explicit way.	Does not provide a taxonomy for agents. Collaboration may be modeled, but is not central.	There is one i* extension that provides a diagram called Strategic Actors (SA) Model [15]. KAOS can model role and position as responsibilities.
NFRs	Represent NFR as softgoals, allowing for qualitative reasoning based on contribution types.	NFRs are mentioned as a different representation in the CASE tool, but it is mainly treated as an goal and refined as an obstacle or constraint.	i-star uses softgoals as task quality attributes and promotes softgoal strategic dependencies representation.
Tasks	Lacks the event concept, as well as the object concept, as such is less operational. The tasks may be modeled as alternatives driven by softgoals.	Tasks are represented in an Operational Model. Alternative is managed at goal/requisite level.	i* can represent events as communication subtasks.
Risk Analysis	Does not mention obstacles. Risk analysis must be done using denying of desired softgoals in a detailed level.	Obstacle analysis helps the introduction of risk analysis early on.	i-star can model obstacles using contributions types and strategic dependencies to show vulnerabilities/opportunities.

4. Conclusion

In this paper we compared KAOS and i*, using **high level** Goals Oriented Requirements Engineering **concepts**. Both methods have been explored by different research groups. Industry dissemination is at the beginning, but KAOS with the support of Objectiver has been more effective in terms of technology transfer.

In general, our observations can be summarized by Table 1 above. Our experience indicates that there are opportunities for contributions mainly in the process of modeling with those methods. The modeling process of i* has been studied and improved [5], but there is still opportunities for new achievements on that direction. The KAOS modeling process is very dependent on sparse examples, and its documentation is fragmented. However, Lamsweerd just edited a book where KAOS plays a central role[2].

The criteria we used is based on our interpretation of the important aspects of GORE modeling, but also in our experience with i* and to a lesser degree with KAOS. We refrain to quantify or to assert the qualities of one representation over another, this is not our goal. We understand that Table 1 provides insights we have observed in our previous work on both methods and in the modeling of the Expert Committee (EC). We did not exercise the follow up after the preliminary models, and as such, we did not use the formalization of goals in KAOS nor a detailed operationalization of tasks in i*.

As such, the criteria we have stressed in Table 1 are at a high level of abstraction, it provides the opportunity for a general view, but, of course, needs further study. We understand that the major difference among KAOS and i* can be summarized by three observations, as can be seen in the meta-models presented.

First, KAOS provide a detailed basis for the description of tasks (actions), see that a action is related to a constraint and to an object, which can be: an agent, an event, an entity, or a relation. In i*, on the other hand, a task is related to a task, a goal, a softgoal or a resource. As such, KAOS provide more support to a detailed description.

Second, i* uses softgoal (NFR) as a driver to decomposition by allowing means-end links to be evaluated (contribution types) and as such making explicit the design decisions in the model, whereas KAOS treat NFR as goals to be decomposed accordingly.

Third, KAOS provides a more explicit mechanism as to deal with risk analysis, with the concept of obstacle (constraint). Although we argued that i* could also reason with respect to alternatives to undesired goals, this requires a more detailed model.

Even limiting our comparison to qualitative criteria based on our experience, we understand that stressing our insights does contribute for a better understanding of the basic principles behind goal oriented requirements engineering.

A comprehensive quantitative analysis is very difficult since several systems, applying both frameworks, would have to be developed. Given that, we understand that there is still room for more detailed qualitative analysis, and a combination of concepts analysis, as we have performed, with the empirical qualitative analysis conducted by Matulevicius and Heymans [16] seems a reasonable path to follow.

[2] *Requirements Engineering: From System Goals to UML Models to Software Specifications*, 2009, Wiley by Lamsweerd, A.v..

5. References

[1] Lamsweerde, A. van; "Goal-Oriented Requirements Engineering: A Guided Tour", Proc. RE'01: 5th Intl. Symp. Req. Eng., Aug. (2001).

[2] Yu, E.; "Modelling Strategic Relationships for Process Reengineering", PhD Thesis, Graduate Department of Computer Science, University of Toronto, Toronto (1995).

[3] Lamsweerde, A. van; Engineering Requirements for System Reliability and Security in Software System Reliability and Security, M. Broy, J. Grunbauer and C.A.R. Hoare (eds.), NATO Security through Science Series - D: Information and Communicarion Security, Vol. 9. IOS Press, 196-238 (2007).

[4] Pohl, K. Process-Centered Requirements Engineering. Taunton, Somerset, England, Research Studies Press Ltd (1996).

[5] Oliveira, Antonio de Padua Albuquerque, Intentional Requirements Engineering: A Method for Requirements Elicitation, Modeling, and Analysis. 261 p. Doctoral Thesis – Computer Science Department, PUC-Rio - Rio de Janeiro. (2008).

[6] Anton, A; McCracken, W; Potts, C.; Goal Decomposition and Scenario Analysis in Business Process Reengineering. Proc. 6th Conference On Advanced Information Systems Engineering (CAiSE'94), Utrecht, Holland, June (1994).

[7] GRL - Goal-oriented Requirement Language, University of Toronto, Canada. At: <http://www.cs.toronto.edu/km/GRL/>. Access: Jan (2008).

[8] Anton, A; "Goal-Based Requirements Analysis", Proceedings 2nd IEEE International Conference on Requirements Engineering, (1996).

[9] Chung, L.; Nixon, B.; Yu, E.; Mylopoulos, J.; Non-Functional Requirements in Software Engineering – Kluwer Academic Publishers, Massachusetts, USA, (2000).

[10] Lamsweerde, A. van, Letier, E. (2003) From Object Orientation to Goal Orientation: A Paradigm Shift for

Requirements Engineering. Proc. Radical Innovations of Software and Systems Engineering, LNCS, (2003).

[11] Dardene, A., Lamsweerde, A. van; Fikas, S; "Goal-Directed Requirements Acquisition", Science of Computer Programming, 20, pp. 3-50, (1993).

[12] Leite, Julio C. S. P.; Oliveira, A de Padua A.; Client Oriented Requirements Baseline, Proceedings of the Second International Symposium on Requirements Engineering, RE95, IEEE Computer Society Press, pp. 108-115 (1995).

[13] Objectiver's documentation - general overview, Access: Aug (2007) http://www.objectiver.com/en/documentation/

[14] Letier, E.; Lamsweerde, A. van. Agent-Based Tactics for Goal-Oriented Requirements Elaboration Proceedings ICSE'2002 - 24th International Conference on Software Engineering, Orlando, May (2002).

[15] Leite, Julio; Werneck, Vera; Oliveira, A. Padua; Capelli, Claudia; Cerqueira, Ana Luiza; Cunha, Herbert; Baixauli, Bruno; "Understanding the Strategic Actor Diagram: An Exercise of Meta Modeling" The X Workshop on Requirements Engineering; Toronto, Canada – May (2007).

[16] Matulevicius, R., Heymans P.; Comparing Goal Modelling Languages: An Experiment, REFSQ 2007, LNCS 4542, pp. 18-32, (2007).

[17] Heaven, William; Finkelstein, Antony; - UML profile to support requirements engineering with KAOS. IEE Proceedings - Software 151(1): 10-28 (2004).

[18] Oliveira, Antonio de Padua A.; Cysneiros Luiz M.; Leite, Julio C. S. P.; Figueiredo, Eduardo M. L.; Lucena, Carlos José P.;: Integrating scenarios, i*, and AspectT in the context of multi-agent systems. 204-218 – CASCON (2006).

[19] Castro, J.; Kolp, M.; Mylopoulos, J. "Towards Requirements-Driven Information Systems Engineering: The Tropos Project." In: The 13th international conference on advanced information systems engineering, Oxford: Elsevier Science Ltd, v.27, n.6. (2002).

[20] Rolland, Colette and Salinesi, Camille; Modeling Goals and Reasoning with Them - Engineering and Managing Software Requirements - Springer Berlin Heidelberg 189-217 (2005).

[21] You, Zheng; Using meta-model-driven views to address scalability in i* models, Master of Science thesis, Graduate Department of Computer Science, University of Toronto, 2004, pp. 231.