A Comparative Analysis of *i*-Based Agent-Oriented Modeling Languages

Claudia P. Ayala, Carlos Cares, Juan P. Carvallo, Gemma Grau, Mariela Haya,
Guadalupe Salazar, Xavier Franch, Enric Mayol, Carme Quer
Universitat Politècnica de Catalunya (UPC)
C/Jordi Girona 1-3, UPC-Campus Nord (C6), Barcelona (Spain)
{cayala, ccares, carvallo, ggrau, mhaya, gsalazar, franch, mayol, cquer}@lsi.upc.edu
http://www.lsi.upc.edu/~gessi/

Abstract

Agent-oriented models are frequently used in disciplines such as requirements engineering and organizational process modelling. i* is currently one of the most widespread notations used for this purpose. Due to its strategic nature, instead of a single definition, there exist several versions and variants, often not totally defined and even contradictory. In this paper we present a comparative study of the three most widespread i* variants: Eric Yu's seminal proposal, the Goal-oriented Requirement Language (GRL) and the language used in the TROPOS method. Next, we propose a generic conceptual model to be used as reference framework of these three variants and we show its use for generating specific models for the three mentioned variants, as well as for other existing proposals.

1. Introduction

In the last years, the construction of agent-oriented models has become an extended practice in fields such as requirements engineering and organizational process modelling [1, 2, 3].

There exist several proposals of languages for the construction of agent-oriented models. Among them, we are interested in the *i** notation proposed by Eric Yu in the first half of the 90's [4, 5]. *i** allows for the clear and simple statement of actor's goals and dependencies among them. It also includes a graphical notation which allows for a unified and intuitive vision of the environment being modelled, showing its actors and the dependencies among them. Moreover, the *i** framework also provides an interactive support for an argumentative, but not fully automatic, style of reasoning about actors and their dependencies.

A characteristic that is soon discovered when starting to use *i** is that there is not a single definition of the language. This looseness is somehow intentional because, due to its nature and objectives, it was pretended to give the language some degree of freedom. But on the other hand, flexibility generates some doubts when using the notation. Furthermore, the existing definitions suffer from several pathologies that are well-known when defining formal languages, among them ambiguities, contradictions and incompleteness. As a result, there is a tendency of each research team to create its own customized *i**, resulting in multiple variants and hampering therefore the exchange of knowledge in the interested community. Some of the *i** variants in process of consolidation are the Goal-oriented Requirement Language (GRL) [6, 7] and the language of the TROPOS method [8, 9, 10]. As a matter of fact, these two variants raise other questions: when to use one or another, or Yu's seminal proposal?, and, which are the characteristics of each of these three proposals? The answers are not clear, especially when considering that there is more than one version for some of these proposals.

The objective of this paper is to clarify some of these questions, by means of the definition of a reference framework, to be used in the analysis and classification of the analyzed *i** variants. Sections 2, 3 and 4 briefly present the characteristics and our analysis of each of these variants. Additionally, we complement the study with observations following Meyer's [11] criteria for the analysis of informal specifications (since this is the predominant style in the description of these variants): *noises* (existence of irrelevant information), *silences* (information that is not mentioned), *contradictions* and *ambiguities*. Section 5 shows a comparison of the proposals. Section 6 proposes a conceptual model to be used as common reference framework and studies one of the analyzed variants, more precisely Yu's *i**, with respect to this framework. Section 7 briefly describes how other *i** variants can be integrated into the framework. Finally, section 8 includes the conclusions. The paper assumes a basic knowledge of *i** from the reader.

2. The *i** framework

The *i** framework defined by Eric Yu [2, 4] is the seminal proposal of the family of agent-oriented languages in which we are interested. Particularly, his doctoral thesis dissertation [4] is the most cited document describing the *i** language and therefore we have used it as main reference in this section.

The *i** framework proposes the use of two models, each one corresponding to a different abstraction level: a *Strategic*

Dependency (SD) model represents the intentional level and the *Strategic Rationale* (SR) model represents the rational level.

A SD model consists of a set of nodes that represent *actors* and a set of *dependencies* that represent the relationships among them, expressing that an actor (*depender*) depends on some other (*dependee*) in order to obtain some objective (*dependum*). The *dependum* is an *intentional element* that can be a *resource*, *task*, *goal* or *softgoal* (see [4] for a detailed description of their meaning). It is also possible to define the importance (*strength*) of the dependency for each of the involved actors using three categories: *open*, *committed* and *critical*.

A SR model allows visualizing the intentional elements into the boundary of an actor in order to refine the SD model with reasoning capabilities. The dependencies of the SD model are linked to intentional elements inside the actor boundary. The elements inside the SR model are decomposed accordingly to two types of links:

- *Means-end* links establish that one or more intentional elements are the means that contribute to the achievement of an end. The "end" can be a goal, task, resource, or softgoal, whereas the "means" is usually a task. There is a relation *OR* when there are many means, which indicate the different ways to obtain the end. The possible relationships are: *Goal-Task*, *Resource-Task*, *Task-Task*, *Softgoal-Task*, *Softgoal-Softgoal* and *Goal-Goal*. In *Means-end* links with a *softgoal* as end it is possible to specify if the contribution of the means towards the end is negative or positive.

- *Task-decomposition* links state the decomposition of a task into different intentional elements. There is a relation *AND* when a task is decomposed into more than one intentional element. It is also possible to define constraints to refine this relationship. The importance of the intentional element in the accomplishment of the task can also be marked in the same way that in dependencies of a SD model.

The graphical notation is shown in Figure 1 using an example about academic tutoring of students. On the left-hand side, we show the SR model of a tutor and the hierarchical relationships among their internal intentional elements. On the right-hand side, we show the SD dependencies between a student and a tutor.

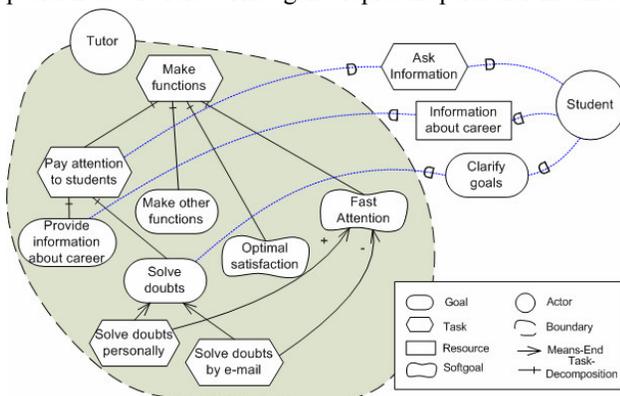*Actors* can be specialized into *agents*, *roles* and *positions*. A position covers roles. The agents represent particular instances of people, machines or software within the organization and they occupy positions (and as a consequence, they play the roles covered by these positions). The actors and their specializations can be decomposed into other actors using the *is-part-of* relationship.

SR models have additional elements of reasoning such as *routines*, *rules* and *beliefs*. A *routine* represents one particular course of action (one alternative) to attain the actor's goal among all alternatives. *Rules* and *beliefs* can be considered as conditions that have to fulfil to apply routines.

In Figure 2 we show an extract of the conceptual model in UML [12], corresponding to the *i** language. It integrates most of the described concepts, except those related to the additional reasoning elements. New concepts that are useful for modelling are: the class *Dependable Node*, which models the intentional elements for which it is possible define dependencies, that is, actors and intentional elements of the SR model; the *Root* association which represents the root of a SR decomposition inside an actor; the *Dependency Equivalence* that states equivalences among SD dependencies and their refinement in the corresponding SR models.

After the analysis of the *i** language based on the study of [4] we have identified some anomalous situations, mainly due to the (intended) incompleteness of the formalization of *i** in TELOS [13] included in the thesis. This incompleteness implies the need of an intensive study of the textual descriptions and the examples to complete the knowledge about *i**. Altogether leads to the following observations:

- **Noise**. The *is-a* relation (generalization/specialization) is used profusely in the examples as a simplification of the notation, but it is not defined as a constructor of the language. On the other hand, the routine concept is defined in the formalization and description of SD models. This situation induces to confusion, since in fact its use as a reasoning element is just noticeable in the SR model.

- **Silence.** The following situations have been detected: it is not indicated if it is allowed more than one root in the internal decomposition of an *actor*; it is not explicit if any type of *intentional element* can be root of a decomposition; it is not specified if an *actor* can decompose into other *actors* by means of an *is-part-of*; it is not detailed if a *dependum* can be related to more than one *depender*; the formalization and description of constraints of the *task-decomposition* link is incomplete; although definitions of the different types of nodes exist (*actors* and *intentional elements*), their criteria of use can be deduced only by analyzing the examples included in the text.

- **Ambiguities.** The importance (*strength*) of a *dependency* is interpreted differently depending on whether it is defined in the *depender* or in the *dependee* side, which seems to imply that a dependency can have different importance for each involved *actor*. Nevertheless, we have not found examples clarifying this point.

Figure 1. Example of an *i** model for an academic tutoring system.

Figure 2. Extract of the UML conceptual model for the *i** language.

3. The Goal-oriented Requirement Language

The *Goal-oriented Requirement Language* (GRL) is a language used in agent- and goal-oriented modelling and reasoning with non-functional requirements. It is strongly influenced by *i** and the NFR framework for specifying non-functional requirements [14]. GRL is part of URN (*User Requirements Notation*) [6] that has been proposed as standard of ITU-T (*International Telecommunication Union-Telecommunication Standardization Sector*) [15].

GRL distinguishes three main conceptual categories (as *i** does): *intentional elements, intentional relationships* and *actors* (specializations are not allowed). The main differences with respect to *i** are: GRL offers constructors for enabling relationships with external elements (*non-intentional elements* and *connection attributes*) and it has additional elements of argumentation and/or contextualization as *beliefs, correlations, contribution types* and *evaluation labels* for specifying satisfaction states, extending in this way the types and qualification ranges of the intentional relationships of *i**.

The observations from our analysis are:

- **Noise**. The existence of a triple syntactic specification (graphical BNF, textual BNF and XML) does not allow an obvious validation of the syntactic correctness of GRL expressions; therefore the effort for understanding this variant is higher.
- **Silence**. The syntactic specification allows the use of a variety of formulas that are not included in the natural language description.
- **Ambiguities**. The formal syntactic specification, accompanied with concise explications and simple

examples, prevent ambiguities in the construction of GRL expressions. However, there exist semantic ambiguities with respect to contributions. This is the case of contributions that have a qualification by means of binary operators (*AND, OR*) but that permit their construction with only one operand.

- **Contradictions**. It was detected a contradiction between the different syntactic specifications of GRL: the textual BNF determines a fixed set of values for contribution types with 11 terminal elements; on the contrary, the corresponding XML specification determines as a terminal element a basic data type (*CDATA*), which means that a value type is not necessarily within a fixed set of types. Another contradiction is that the tool for editing GRL models, OME [16], allows the definition of actor specializations, which does not match with any of the syntactic GRL specifications.

4. Tropos

Tropos is a project [10], whose mainly purpose is to define an agent-oriented software development method, using a variant of *i** [8, 9] as modelling language.

This method supports all the development stages from requirements analysis to implementation. Each stage adopts the concepts of *i** (i.e. *actor, dependency*) to show a framework of the model depending the stage.

In the requirements analysis stage, the *actors* are used to model stakeholders of the domain and the system to be constructed. Therefore, *dependencies* represent dependencies

between stakeholders and dependencies between them and the new software system.

In the design stage, the *actors* represent the components of the system architecture and the agents that should be implemented. The *dependencies* represent the data and control interchange between components and agents, and define the abilities or responsibilities of each one that must be implemented.

The differences between the language proposed in Tropos and *i** are related to the syntax of some concepts. For example, Tropos does not distinguish between SD and SR models. However, it proposes different views for each development stage: Tropos models explicitly and in a separated way aspects related to the domain and to the software system.

The observations from our analysis are:

- **Noise**: The Tropos literature is more focused on the method than on the language to be used.
- **Silence**: It exists a Tropos user guide [9] written in Italian. It details the language and explains some rules how to use it, but as far as we know it does not exist any English version.
- **Ambiguities**: Some papers describing Tropos method like [17] use Yu's *i** version or GRL instead of the Tropos own version of the language; this fact is very confusing.
- **Contradictions**: Tropos starts from the hypothesis that all goals and all tasks (named *plans* in Tropos) of the model must be assigned to an *actor*, but in the Tropos conceptual model included in [9] this aspect is optional. Also, the *intentional elements* that take part in the *contribution relationships* (*contributors* and *contributed*) are not the same in the meta-model and explanations found in the main sources of information [8, 9]; even more, in [9] the examples do not adhere completely to the explanations.

5. Comparative Analysis

In this section we present a comparative analysis of the three variants studied. With this purpose, we have identified fourteen criteria corresponding to two categories.

- Eight **structural criteria** consider the characteristics of the language constructors, and are related to models, actors, intentional elements, decomposition elements, additional reasoning elements and external model elements. This criteria form a semantic baseline upon which *i** variants may be assessed.
- Six **non-structural criteria** analyze the definition of the languages, its use, and also the elements that complement them, as can be formalizations, methodologies and software tools. Specifically, the definition analyzes the languages used to describe the syntax and semantics of the different variants, and the use considers the publications, standards and information found about the different languages through the Internet. In other words, these criteria are syntactical and therefore not as fundamental as the structural ones. However, they are relevant when considering understandability and accuracy of the notation.

Next, we give the details of the comparison for each of the fourteen criteria.

- **Types of models.** In Tropos and GRL do not exist any type of models. However, the level of abstraction of the SD and SR models of *i** (SD and SR) can be achieved in Tropos and GRL by drawing in their models the limits of the *actors*. As can be observed, we do not consider the *views* of Tropos as types of models because they represent different images of the same model that correspond to the different development stages.
- **Types of actors.** GRL can be distinguished from the two other variants since it does not allow the specialization of *actors*, although we have found contradictions among the different sources of information. On the other hand, it is worth to say that in TROPOS some specializations are specific of the design stage.
- **Intentional elements.** The *intentional elements* are the same in the three variants, although some of them differ in how they are named. Thus: the tasks of *i** and GRL are named *plan* in Tropos; Tropos names *hardgoals* the same concept that is named *goal* in the other variants, and it generalizes *hardgoal* and *softgoal* as *goal*. Note that, although GRL defines *beliefs* as *intentional elements*, it uses them as *reasoning elements* (see below).
- **Relationships among actors.** We can distinguish between *dependencies* and other types of relationships. GRL is the only variant that does not allow other types of relationships. On the other hand, the relationship *is-part-of* just exists in *i**.
- **Relationships among intentional elements.** The three variants support four types of relationships: *dependencies, means-end relationships, decompositions* and *contributions*. *Dependencies* are used in a same way in the three languages (they allow the same four types of *dependums*). Nevertheless, the other three types of relationships differ in: the lexicon used; the *intentional elements* allowed as source and destination of the relationships; the combination of the elements that take part in the *contribution*; the expressive power of the types of *contributions*. Table 2 shows these differences for *means-end relationships, decompositions* and *contributions*. Specifically, for each of the variants compared, we can see how the constructor is named, the valid combinations of intentional elements (the source at the left side of the "-" and at the target side at the right side of the "-"), and the way of combining the elements that take part in the constructor. It is important to emphasize that the concept of *contribution* in GRL has two constructors, *contributions* and *correlations*. To understand the valid combinations in the different constructors it is necessary to know the abbreviations we have used, by means of one or two letters of the word: **O**bjective, **N**on-**F**unctional requirement, **T**ask,

	Criteria	Yu's *i**	GRL	TROPOS
Structural	Types of models	SD and SR	None	None (views)
	Types of actors	1 generic 3 specific: role, position and agent	1 generic	1 generic 3 specific: role, position and agent
	Intentional elements	Goal, softgoal, task, resource	Goal, softgoal, task, resource	Goal (hardgoal), softgoal, task (plan), resource
	Relationships among actors	Dependencies among actors by means of intentional elements. Relationships among specific types of actors: "occupies", "covers" and "plays". Relationship "is part of"	Dependencies among actors by means of intentional elements	Dependencies among actors by means of intentional elements Relationships among specific types of actors: "occupies", "covers" and "plays"
	Relationships among intentional elements (see table 2)	Dependencies among actors Means-end relationships Decomposition relationships Contribution relationships	Dependencies among actors Means-end relationships Decomposition relationships Contribution relationships	Dependencies among actors Means-end relationships Decomposition relationships Contribution relationships
	Decomposition elements	Decomposition of actors unlimited	Decomposition of actors restricted	Decomposition of actors unlimited
	Additional reasoning elements	<u>Explicit</u>: Strength, Contribution, Constraints <u>Dynamic Reasoning</u>: Routine, Rule, Belief <u>Properties</u>: Workability, Ability, Viability, Believability	<u>Explicit</u>: Belief, Contribution Types, Correlation Types, Evaluation Labels, Criticality	<u>Explicit</u>: Belief, Contribution Types, Mode <u>Dynamic behaviour</u>: Capability, Events
	Relationships with external model elements	They do not exist	Attributes External models elements Topics of soft goal	They do not exist
Non structural	Stages of the life cycle	Early and late requirements	Early requirements	From early requirements to implementation
	Dissemination and standardization	Wide dissemination Diversity of information sources	Medium dissemination. Adopted as standard by ITU-T (in progress)	Emergent dissemination Existence of a user guide (in Italian)
	Tools	OME, REDEPEND	OME	None
	Additional elements	Examples of case studies in different domains	Additional functional language (UCM) Additional method of use (URN)	Software engineering method Formal TROPOS (with the tool FTT)
	Syntactic description	Natural language, Graphical notation	BNF Textual, Graphical BNF, XML	Natural language, Graphical notation
	Semantic description	Natural Language, Telos	Natural Language	Natural Language, UML (model, metamodel, meta-metamodel)

Table 1: Comparative analysis of the three main *i** variants

Resource, **B**elief and re**L**ationship, representing this last concept the three types of relationships that GRL consider as possible participant in the *contributions*. Also an abbreviation "*" has been used instead of O, NF, T and R altogether. For example, as can be seen in the table, just Tropos allows defining relationships *means-end* in which the means is an objective or non-functional requirement.

- **Decomposition elements.** In *i** the relationships *is-part-of* among *actors* facilitate the decomposition. In Tropos the decomposition is possible by allowing new *actors* to appear inside the limits of an *actor*.

- **Additional reasoning elements.** The types of *reasoning elements* are different in the three variants, and also the elements for each of the types. In Tropos there are some elements that are specific of the design stages.

- **Relationships with external model elements**. Although this feature seems necessary in any language for modelling systems, these relationships and the external elements exist, according to the documents that we have consulted, just in GRL.

- **Life-cycle stages**. From our point of view, the three variants may be used in any stage of the development of software. However, according to the documents consulted, *i** is concerned to be used in the early and late requirements

stages, GRL just in the early requirements stage, and Tropos from early requirements to implementation.

- **Dissemination and standardization.** The analysis of the use of the variants has been done by considering the amount of papers published in the main conferences and journals, and references to these papers found in these sources. On the other hand, the only language that is currently in process to be accepted as standard is GRL [15].

- **Tools.** We have not found any tool that supports Tropos, but there exists the tool T-Tool that supports Formal Tropos [18] that is one evolution of it. On the other hand, OME [16] offers the possibility of editing *i** and GRL models, but it allows elements that are not reported in the studied documents. This also happens in REDEPEND [19] for *i**.

- **Additional elements.** The additions are listed in the table. Methodologies are just provided by GRL and Tropos. On the other hand, languages related with them exist also for GRL (UCM), and Tropos (Formal Tropos). Additional documents of *i** offer examples of case studies that illustrate its use.

- **Syntactic definition.** There only exists a formal syntactic definition for GRL. The other two variants define its syntax by means of natural language and graphic notation.

- **Semantic definition.** There exist an incomplete semantic definition of i* using TELOS [13]. In the case of

		Yu's *i**	GRL	TROPOS
Means-end	Name	*means-end*	*means-end*	*means-end*
	Connected elements	* – T	(T, O, R) – T	(O, NF) – * T – (T, R)
	Operation	OR	OR	OR
Decomposition	Name	*task-decomposition*	*decomposition*	*AND/OR decomposition*
	Connected elements	T – *	T – *	(G, NF) – (G, SG) T – T
	Operation	AND	AND	AND, OR
Contribution	Name	*means-end*	*contribution, correlation*	*contribution*
	Connected elements	O – O NF – NF	*contribution*: (NF, C, L) – (NF, T, C, L) *correlation*: (NF, T) – (NF)	O, NF – *
	Operation	Does not exist	*contribution*: AND, OR *correlation*: Does not exist	Does not exist
	Attributes	+, -	Make, Break, Help, Hurt, Some+, Some-, Equal, Unknown	++, +, -, --

Table 2: Comparative analysis of the relationships among intentional elements in *i**

TROPOS there exists a rigorous definition of its model, meta-model and meta-meta-model in UML.

6. A reference conceptual model for *i**

In this section, we present a conceptual model (see figure 3) that has as aim to be a reference framework for the variants of *i** that we have analyzed in the paper, and for those that may appear in the future. We have constructed the conceptual model including those concepts common to *i**, GRL and Tropos, and those concepts not common to the three variants but so important for agent-oriented modelling that should be present in any other variant that could appear. The reference framework allows determining the differences of an *i** variant respect to the framework, and thus to know how much different a new variant that would appear would be from the core of *i**.

We propose to describe these differences by means of operations that make transformations on conceptual models, in a similar way as done in refactoring [20]. Therefore, to know the differences among a variant of *i** and the reference framework, it is necessary to determine these operations needed to obtain the conceptual model of the variant from the reference framework.

In the particular case of the *i** variant of Yu [4], the operations applied to obtain the conceptual model of Figure 2 from the reference framework are the following (for simplicity, we omit the OCL constraints, which would be expressed by means of notes):

- **Addition.** The attributes *depender_strengh* and *dependee_strengh* in the class *Dependency* are added. The derived associations */dependency_equivalence* (with their corresponding role names) and */plays* are also added.

- **Deletion**. The class *External Element* and its relationships with the classes *Dependum* and *Node* are deleted.
- **Renaming**. The class *Dependum* is renamed as *SR-Dependum* and the class *Internal Element* is renamed as *SR-Element*. The name *has-parts* is given to the role of the association named *is-part-of*.
- The most complex operation is the transformation of the associative class *Relationship* of the reference framework into the associations *Means-End* and *Task-Decomposition* of the *i** model, since the transformation implies flattening the hierarchy. For doing this transformation we apply the following operations: the subclasses *Contribution* and *Correlation* are suppressed; the derived subclass *SR-Task-Element* is added as subclass of the class *SR-Element* (with the corresponding attribute constraint); the subclass *Decomposition* is suppressed; the associative class *Task-Decomposition Link* is added between the classes *SR-Element* and *SR-Task-Element*, with the strength attribute; the class *Means-End* is generalized and replaces the class *Relationship*, it is renamed as *Means-End Link*, and the attribute contribution is added to it.

7. Analysis of other *i** variants

The existence of an *i** reference framework also allows analysing and comparing easily new proposals of languages based on *i**, new versions of the existing ones, and even extensions of *i** for concrete domains uses.

A first example of these applications could be to analyse the proposal of the tool REDEPEND [19], which extends *i** allowing new types of *Means-End* relationships, *Contribution* relationships, and other minor differences. Most of these

Figure 3. *i** reference framework

differences are included in GRL and Tropos, and hence they have been considered in constructing the reference framework.

A second example could be to analyse the Formal Tropos language [18]. Formal Tropos adds to *i** temporal specification primitives, including their elements in the language [21]. On one hand, it allows specifying *cardinality constraints* in the dependencies among *intentional elements*. On the other hand, it allows defining a new *dependency* type (*prior-to*) to specify temporal order between *intentional elements*. These two extensions can be added to the reference framework defining the cardinalities as attributes of a *dependency*, and by defining a new associative class, named *Dependency*, respectively.

As a last example, we could analyze existing works that use *i** with extensions and adjusts to adapt the language to their particular needs. For example, in [22] the interactions between a software system and its users are analyzed. Therein, the authors propose new types of *dependencies* among *actors* and *intentional elements*: *responsibility dependencies* between an *agent* and a *goal* or a *task*; *authority dependencies* between two *agents*; *audit dependencies* between an *agent* and a *goal* or a *task*; and *capability dependencies* of an *agent* respect to a *goal* or *task*. All these new *dependencies* can be easily included to our reference framework by adding new subtypes to the associative class *Dependency*, and the appropriate integrity constraints.

8. Conclusions

The goal of this paper is to make a deep analysis of the three most important variants of the *i** language that, nowadays, is one of the most spread agent-oriented modelling proposals. This work can be considered useful both to the novice that may get some support when learning the notation, and to the expert that may have a summary of the similitudes and differences of the existing proposals. The most relevant contributions of this paper are:

- A comparative study of the three most important variants of *i**. This study has been carried out in a rigorous way by constructing a data conceptual model in UML for each variant (we have shown one in the paper), and identifying 14 comparison criteria.
- An enumeration of the noises, silences, ambiguities and contradictions that exist in the available definitions of the three variants.
- The definition of a conceptual model that constitutes a reference framework for *i**-based languages, that includes the concepts belonging to the studied variants, and that helps to contextualize them and others that could appear in the future.
- The empiric observation that the reference framework allows also capturing other specific variations existent in literature.

There exist other works related with the comparative analysis, evaluation and review of agent- and goal-oriented models [23, 24, 25] but we do not know of any focused on clarifying or guiding the user concerning the doubts and misunderstandings that might arise from the existence and particularities of the different *i** variants, neither on formalizing a general framework for them.

Acknowledgements

9. References

[1] E. Yu. "Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering". *Proceedings of the 3rd IEEE Int. Symp. on Requirements Engineering*, Washington, USA, January 6-8, 1997, pp. 226-235.

[2] E. Yu, J. Mylopoulos. "An actor dependency model of organizational work: with application to business process reengineering". *Proceedings of the Conference on Organizational Computing System*, Milpitas, California, USA, November 1-4, 1993, pp. 258-268.

[3] A. van Lamsweerde. "Goal-Oriented Requirements Engineering: A Guided Tour". *Proceedings of the 5th IEEE International Symposium on Requirements Engineering*, Toronto, Canada, August 27-31, 2001, pp. 249-263.

[4] E. Yu. *Modelling Strategic Relationships for Process Reengineering*. PhD. thesis, University of Toronto, 1995.

[5] *i** web page, http://www.cs.toronto.edu/km/istar/, last accessed April 2005.

[6] D. Amyot, G. Mussbacher. "URN: Towards a New Standard for the Visual Description of Requirements". Proceedings of the Third International Workshop on Telecommunications and beyond: The Broader Applicability of SDL and MSC., Aberystwyth, UK, June 24-26, 2002, pp. 21-37.

[7] GRL web page, http://www.cs.toronto.edu/km/GRL/, last accessed April 2005.

[8] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, J. Mylopoulos. "Tropos: An Agent-Oriented Software Development Methodology". *Journal of Autonomous Agents and Multi-Agent Systems*, Kluwer Academic Publishers Volume 8, Issue 3, May, 2004, pp. 203-236.

[9] F. Sannicoló, A. Perini, and F. Giunchiglia. "The Tropos modelling language. A User Guide". Technical report DIT-02-0061, University of Trento, February 2002.

[10] TROPOS web page, http://www.troposproject.org/, last accessed April 2005.

[11] B. Meyer. "On Formalisms in Specifications". *IEEE Software*, 2 (1), January, 1985, pp. 6-26.

[12] Object Management Group. *Unified Modelling Language (UML) 2.0 Final Adopted Specification*. OMG document ptc/03-10-14. October 2003.

[13] J. Mylopoulos, A. Borgida, M. Jarke, M. Koubarakis. "Telos: Representing Knowledge about Information Systems". *ACM Transactions Information Systems*, 8 (4), October, 1990, pp. 325-362.

[14] L. Chung, B.A. Nixon, E. Yu, J. Mylopoulos. *Non-Functional Requirements in Software Engineering*. Kluwer Academic Publishers, 2000.

[15] Z.151 (GRL). International Telecommunication Union (ITU). September, 2003.

[16] OME3 web page, http://www.cs.toronto.edu/km/ome/, last accessed April 2005.

[17] J. Mylopoulos, M. Kolp, J. Castro, "UML for Agent-Oriented Software Development: The Tropos Proposal". *Proceedings of 4th International Conference on The Unified Modelling Language, Modelling Languages, Concepts, and Tools*, Toronto, Canada, October 1-5, 2001, pp. 422-441.

[18] Formal Tropos web page: http://dit.unitn.it/~ft/doc, last accessed April 2005.

[19] N. Maiden, P. Pavan, A. Gizikis, O. Clause, H. Kim, X. Zhu. "Integrating Decision-Making Techniques into Requirements Engineering". *Proceedings of the 8th International Workshop on Requirements Engineering: Foundation for Software Quality*, Essen, Germany, September 09-10, 2002.

[20] G. Sunyé, D. Pollet, Y. Le Traon, J.M. Jézéquel. "Refactoring UML Models". *Proceedings of the 4th International Conference <<UML>> 2001 – The Unified Modeling Language*, Toronto, Canada, October 1-5, 2001, pp. 134-148.

[21] A. Fuxman, L. Liu, J. Mylopoulos, M. Pistore, M. Roveri, P. Traverso. "Specifying and analizing early requirements in Tropos". *Requirements Engineering Journal*, vol. 9, num. 2, 2004, pp. 132-150.

[22] A. Sutcliffe, S. Minocha. "Linking Business Modelling to Socio-Technical System Design". *Proceedings of Advanced Information Systems Engineering, 11th International Conference* CAiSE'99, Heidelberg, Germany, June 14-18, 1999, pp. 73-87.

[23] E. Kavakli. "Modeling organizational goals: analysis of current methods". *Proceedings of the 2004 ACM symposium on Applied Computing*, Nicosia, Cyprus, March 14-17, 2004, pp. 1339–1343.

[24] C. Silva, R. Pinto, J. Castro, P. Tudesco. "Requirements for Multi-Agent Systems". *Proceedings of the Workshop em Engenharia de Requisitos*, Piracicaba-SP, Brasil, November 27-28, 2003, pp.198-212.

[25] R.B.K. Brown, A. Ghose. "Hierarchic Decomposition in Agent Oriented Conceptual Modelling". *Proceedings of the Fourth International Conference on Quality Software*, Braunschweig, Germany, September 8-9, 2004, pp. 240-249.