Ontology-based Transformation Framework from Tropos to AORML

Renata S. S. Guizzardi[1] and Giancarlo Guizzardi[2]
[1,2]Computer Science Department, Federal University of Espírito Santo, Brazil
rguizzardi@inf.ufes.br
[2]Laboratory of Applied Ontologies (ISTC-CNR), Trento, Italy
guizzardi@loa-cnr.it

1. Introduction

Software Agents have gained a lot of attention in the past few years, attracting a number of research initiatives and gaining visibility in software development practice. Agent's popularity mainly resulted from the recognition that they represent a suitable abstraction both for conceptual modeling and system development. However, having the right abstraction is not enough for guaranteeing the development of adequate solutions. For that, a consistent software engineering methodology is needed. This need has led to the proposal of several agent-oriented methodologies.

Software analysts and designers can profit from these methodologies, each one having its own strengths and drawbacks, and exhibiting characteristics that make them more or less appropriate for specific domains and/or system types. Moreover, the concepts and processes underlying each one of them can be targeted at different activities of the software development cycle. Nowadays, with the increase in the complexity of domains and systems modeled using an agent-oriented approach, there are cases where a single methodology/modeling language is not sufficient to cover all activities and perspectives of the modeling process. In such cases, benefits may be achieved by using concepts and processes of two or more of such approaches in different development activities, thereby amplifying their gains while minimizing their limitations.

In this paper, we apply this idea by considering two modeling languages: the Tropos language on requirements analysis and the Agent-Object-Relationship Modeling Language (AORML) on system design. Moreover, we propose a methodology to transform a requirements model into a design model, by providing a systematic method for mapping one notation into the other. On one hand, the Tropos's modeling primitives have been designed to support requirements analysis and, hence, do not provide specific concepts for system design. On the other hand, AORML supports information, interaction and behavior modeling, which are crucial for system's design. However, this language is not specifically tailored for requirements analysis. Thus, the benefits of using both approaches become apparent, as one lacks what the other has to offer.

The transformation methodology is supported by a foundational ontology, which enables the evaluation, comparison and identification of correspondences between different agent-oriented modeling languages. This ontology is based on a number of theories from philosophy and cognitive science, and can serve as a well-founded basis for: (i) making explicit the ontological commitments of each modeling language; (ii) defining (ontological) real-world semantics for their underlying concepts; (iii) providing guidelines for the correct use of these concepts; (iv) relating concepts defined in different languages via their ontological semantics.

The remaining of this chapter is organized as follows: section 2 motivates our choice for transforming Tropos models into AORML ones and discusses the adopted strategy for performing this transformation; section 3 describes the foundational ontology used in this work; section 4 applies this ontology as a reference model to guide the evaluation of the two languages, making the necessary adjustments to facilitate the smooth transformation of models; section 5 presents the mapping between Tropos and AORML; and finally, section 6 concludes this chapter.

2. On the integration of the Tropos Language and AORML

Given the current state of research on the agent-oriented software engineering paradigm, we aim at granting analysts and designers with freedom to select the appropriate tools from a vast 'library' of methods and languages, depending on the specific case at hand. It is our belief that not one method or language possesses all the desired modeling features. Instead, these features can often be attained by integrating different approaches. This view is compliant with the efforts of the *method engineering* community which prescribes the reuse of fragments of different methods according to a given situation (Harmsen, Brinkkemper & Oei, 1994). Works on agent-oriented software engineering having analogous views can be found in (Henderson-Sellers, 2005; Juan, Sterling & Winikoff, 2004; Sabas, Delisle & Badri, 2002).

The characteristics of the targeted domain should be carefully considered before choosing methods and languages to be applied. The integration of Tropos and AORML was specifically proposed for the Knowledge Management domain (Guizzardi, 2006), giving rise to a methodology named Agent-oriented Recipe for Knowledge Management System Development (ARKnowD). A detailed description of ARKnowD's activities and life cycle, as well as a discussion on its applicability to Knowledge Management can be found in the chapter by Guizzardi, Perini, Dignum in this book. In the present chapter, however, we refrain from discussing details about methods and techniques, concentrating solely on modeling language issues and, in particular, on the suitability of the concepts underlying current agent-oriented modeling languages. More precisely, our focus here is to present a common conceptual framework that can be used to *evaluate, compare* and *identify correspondences* between the two modeling languages that are used in the ARKnowD methodology, namely, Tropos and AORML.

The differences between Tropos and AORML suggest that these two approaches can serve complementary rather than competing roles. Tropos may benefit from the following strengths of AORML: 1) the fact that 'mentalistic' concepts of agents, such as *beliefs* and *commitments*, are explicitly considered in system design supports the designer to reason about and to model the behavior of agents, both internally and in interaction with other agents of the system; 2) although norms and contracts are not directly supported by AORML, it provides deontic modeling constructs such as *commitments* and *claims*, which form the basis for the establishment of such norms and contracts; 3) it captures the behavior of agents with the help of *rules*. Besides these strengths, since AORML is an extension of UML, preserving its principles and concepts, it is an accessible language, and it allows the use of UML constructs whenever an extension is not provided, thus offering a comprehensive set of design tools. In a complementary manner, the explicit use of Tropos's *goals* and *plans* provides a rich conceptual framework for modeling the intentional dimension of the organization. This includes a preliminary view of how user's interact, without however adding unnecessary protocol details in the early stages of requirements analysis. Such concepts of goals and plans are missing in AORML.

Our approach for integrating Tropos and AORML is inspired by the OMG framework for *Model Driven Architecture (MDA)*, developed to enable flexible design of distributed software systems (Miller & Mukerji, 2003). Although MDA specifically focuses on the object-oriented paradigm, it shares some of our concerns, which makes it appropriate also for agent-orientation. In particular, as in our work, this initiative recognizes that each system development activity has its particular focus and needs, which constrain the applied constructs and strategies for the development of models for each one of them.

Instead of modeling activity, MDA talks of *viewpoint*, which is defined as a "technique for abstraction using a selected set of architectural concepts and structuring rules, in order to focus on particular concerns within that system." (Miller & Mukerji, 2003, p. 2-3). Using MDA terminology, the application of Tropos is mainly concerned with a *computation-independent viewpoint*, focusing on the domain of the system. At this point, the system's requirements are hidden or undetermined. Tropos contributes to make explicit and clear such requirements, presenting an initial model of the domain entities, along with a high level view of their interactions and behaviors. In contrast, AORML focuses on the *platform-independent viewpoint*, modeling the general functionality of the system, although not including the details that are specific of a given platform. At this point, the information, interaction and behavior aspects of the system become concrete. In other words, AORML details the structure of system's entities and relations, fully models the processes that involve such entities, and present a comprehensive description of their internal behavior.

When a system is developed, system requirements should be traceable to system functionalities, and vice-versa. For maintaining consistency between models, enabling a smooth transition from one viewpoint to another, MDA proposes the use of *transformation processes*, i.e. processes that convert from one model to another model of the same system. In this work, we apply the metamodel transformation technique described in the MDA reference guide (Miller & Mukerji, 2003), which requires a mapping from the modeling constructs of the source to the destiny language. However, before consistent transformation rules are developed, the semantics of the modeling concepts of each language should be well understood. Furthermore, the notations should be carefully evaluated to check for inconsistencies on their use individually or in integration with one another. In this respect, our work proposes an ontological approach.

An ontology is a theory that characterizes the kinds of entities which exist in a conceptualization of a certain domain of discourse. Moreover, it establishes a vocabulary and semantics for the terms referring to the entities constituting this conceptualization. Thus, for example, an ontology describing the domain of agents can be applied to create a common conceptual framework comprehending the concepts underlying both the Tropos language and AORML. This same ontology may be used as a reference model that should be fully covered by the set modeling constructs of these two notations, thus providing a consistent evaluation framework for the resulting language (i.e. the language underlying ARKnowD).

3. The UFO Ontology

We base our Agent Ontology on the UFO (Unified Foundation Ontology) defined by Guizzardi and Wagner (2005). According to these authors, a foundation ontology "defines a range of top-level domain-independent ontological categories, which form a general foundation for more elaborated domain-specific ontologies" (Guizzardi & Wagner, 2005, p. 346).

The UFO ontology is divided into three incrementally layered compliance sets: 1) UFO-A defines the core of UFO, excluding terms related to perdurants (i.e., occurrences, events) and terms related to the spheres of intentional and social things; 2) UFO-B defines, as an increment to UFO-A, terms related to perdurants; and 3) UFO-C defines, as an increment to UFO-B, terms related to the spheres of intentional and social things. This section briefly describes a subset of UFO-A and UFO-B, focusing only on the concepts that are directly relevant for the understanding of ontological distinctions proposed in this article. Following, we present UFO-C in details, extending it to create our ontology.

The ontologies are described here in natural language, and illustrated with the aid of UML class diagrams. Thus, UML is not intended here for formalization purposes but rather for facilitating the visualization of concepts. For an in depth discussion and formal characterization of UFO-A, one should refer to (Guizzardi, 2005). The formalization of UFO-B and UFO-C is planned as future work, once the semantics of the concepts comprising these ontologies is fully comprehended.

3.1 UFO-A: Endurants and Perdurants

Figure 1 shows an excerpt of UFO-A. UFO-A distinguishes between two kinds of individuals: endurants and perdurants. This distinction can be intuitively understood in terms of the distinction between “objects” and “processes”, respectively. An endurant does not have temporal parts, and persists in time while keeping its identity. Examples of endurants include a house, a person, a hole, the (objectified) color of an apple, and an amount of sand. A perdurant, conversely, is composed of temporal parts. A storm, a heart attack and a business process are three examples of perdurants.

Endurants are further specialized into substance individual and moment individual. The former refers to an endurant that possesses direct spatio-temporal properties and can exist by itself, i.e. substance individuals are not *existentially dependent* on other endurants, except possibly on some of its parts and constituents. A building, a person and a dog are examples of substance individuals. A moment individual, however, is an endurant that cannot exist by itself; that is, it existentially depends on other individuals (e.g. the age of a person, a belief of an agent). Making an analogy with the object-oriented software engineering domain, we can understand the difference between substance and moment comparing them respectively to object and (objectified) property.

A moment individual can be either an intrinsic moment or a relator (or relational moment). An intrinsic moment is a moment individual that is existentially dependent on one single individual (e.g., the color of an apple depends on the existence of the apple itself). Meanwhile, a relator is a moment individual that is existentially dependent on more than one individual (e.g., a marriage, an enrollment between a student and an educational institution). A relator is an individual capable of connecting or mediating entities. For example, we can say that John is married to Mary because there is an individual marriage relator that existentially depends on both John and Mary, thus, mediating the two. Likewise, we can say that Lisa works for Xerox because there is an employment relator mediating Lisa and Xerox.

Endurants bear moments, or inversely, moments inhere in endurants. The relation of *inherence* is a special type of existential dependence relation between moments and their bearers. Formally, besides existential dependency, inherence implies the so-called *non-migration principle* (Guizzardi, 2005), i.e., if a moment x inheres in an individual y, then there is no individual z distinct from y such that x inheres in z. In other words, inherence is a functional existential dependence relation. Fig. 1 particularly emphasizes that an intrinsic moments inheres in one single endurant. An externally dependent moment is a special kind of intrinsic moment which although inhering in a specific endurant, also existentially depends on another one. The employee identifier is an example of externally dependent moment, since although inherent to the employee, is also dependent on the organization where this employee works. A relator R mediating the individuals A and B inheres in the individual composed of A and B (the so-called mereological sum of A and B) and, due to the aforementioned non-migration principle, this individual cannot change. In other words, R inheres in (and, thus, is existentially dependent on) exactly that specific collection of individuals formed by A and B.

A substance individual is further specialized into amount of matter and physical object. A physical object satisfies a condition of unity, for which certain parts can change without affecting its identity (e.g. a house, a person, the moon). Conversely, an amount of matter is a substance individual that does not satisfy a condition of unity, typically referred to by means of mass nouns in natural language (e.g. a lump of clay, a pile of bricks, an amount of sand).

In Fig. 1, we emphasized that all specializations are disjoint, meaning that if an individual is an instance of one specialization class, it can not be an instance of another specialization class with the same parent. All specialization relations described in this section have this nature. Hence, we refrain from providing such details in the subsequent figures in order to simplify the models. The above information presented for the individual level may also be replicated for the type level. Figure 2 shows an entity may be either an individual or a type, the former instantiating the latter. So for example, the substance individuals John, Mary and Lisa instantiate the substance type Person.

Fig. 2 shows that for the category of substance types, UFO-A makes a further distinction based on the formal meta-properties of *rigidity* and *anti-rigidity*. In simple terms, a type T is said to be rigid if every instance x of T is necessarily (in the modal sense) an instance of T. In other words, x cannot cease to instantiate T without ceasing to exist. Conversely, a type T is anti-rigid if every instance x of T is possibly (in the modal sense) not an instance of T, i.e., if x can cease to instantiate T without ceasing to exist (Guizzardi, 2005). A stereotypical example highlighting this distinction is provided by the types person and employee, both instantiated by the individual Lisa in a given circumstance. Whilst Lisa can cease to be an employee of Xerox (and there were periods of time in which Lisa was not one), she cannot cease to be a person. A substance type that is rigid is named a Kind. In contrast, a substance anti-rigid type is named here a Role[i].

Besides highlighting this important difference within the category of substance types, Fig. 2 also presents other entities. A relation is a type whose instances are tuples of connected elements. For instance, taking Lisa's example presented above, the 'works at' relation connects Lisa to Xerox. We consider here two types of relations: formal relation and material relation. A formal relation holds between two or more entities directly, without any further intervening individual. Examples of formal relation include Lisa 'is older than' Mike, and John 'is taller than' Mary. As pointed out by Guizzardi (2005), the relata of these relations are in fact moments and not substance individuals. To say that Lisa 'is older than' Mike is to say that Lisa's age is greater than Mike's age. Moreover, this relation between Lisa and Mike exists without the need for any real connection between the two. To put it differently, these relations between substance individuals are reducible to purely formal relations between intrinsic moments of the involved relata. Instantiation, inherence and existential dependency are all types of formal relations.

Conversely, material relations are founded on the existence of a relator. Thus, Lisa ‘works at’ Xerox because there is an employment relator connecting the two. This employment can be composed, for example, of all commitments and claims associated with the role Lisa plays at that organization. Later in this section we provide a more extensive discussion on commitments and claims. Likewise, John ‘is kissing’ Mary because there is an individual kiss connecting the two. In summary, differently from formal relations, material relations are not reducible to relations between intrinsic moments of the involved relata.

3.2 UFO-B: an Ontology of Perdurants

Figure 3 presents UFO-B, where the concept of perdurant from UFO-A is further specialized into state and event. A state is a perdurant whose temporal parts belong to the same state type as the whole. An event, on the other hand, is a perdurant that is related to exactly two states (its pre-state and its post-state). Pre-state and post-state are shown in the relations between event and state in Fig. 3.

An event is specialized into atomic and complex events. The former refers to an event that is not further decomposed, for instance: an explosion, or a message reception. The latter is an event that is composed of other events by means of event composition operators. Examples of complex event comprehend a parallel occurrence of two explosions, a storm, a heart attack, and a work meeting. A process can be understood as a synonymous of complex event, i.e. an event that is composed of two or more events as shown in Fig. 3.

3.3 Extending UFO-C

Our extended version of UFO-C is depicted subsequently, starting from Figure 4 until Figure 7.

Fig. 4 shows that the UFO-A concept of physical object is here specialized into physical agent and non-agentive object. A physical agent is a physical object that creates action events, perceives events (possibly created by other physical agents), and to which we can ascribe mental states. Examples of physical agents include: a man, a cat, a robot. A non-agentive object is a physical object that is not a physical agent (e.g. a book and a tree). A non-agentive object can be a resource, meaning that such object is used by a physical agent with specific purposes, and typically owned or controlled by this or other physical agent (relation owns and controls outcoming from physical agent).

A distinction is made between human agent, artificial agent and institutional agent (all three sub-kinds of physical agent), to differentiate humans agents, computational agents, and agents representing organizations or organization sub-units (such as departments and divisions). Institutional agents are composed of several internal agents, which may be of any kind of physical agent (human, artificial or institutional).

Most agent-oriented approaches only focus on agents, disregarding the presence of objects in the modeled scenario. We consider this a limitation and thus, acknowledge the existence of these two distinct entities. In fact, the real world is composed of both active and passive entities, captured respectively by the concepts of agents and objects (in UFO-C, physical agent and non-agentive object)

Action event and non-action event are two types of event (concept from UFO-B). The former refers to an event that is created through the action of a physical agent, for instance, ‘writing a book’, and ‘reviewing a paper’. The latter is an event that is not created through an action of a physical agent (e.g. ‘a deadline is achieved’, and ‘it becomes dark’), although it may be perceived by him/her. This differentiation is essential in agent-oriented approaches as modeling the environment populated by agents is paramount. Therefore, non-action events are typically events generated by the environment itself and perceived by the agents living in it.

A plan execution can be defined as an intended execution of one or more actions, being in this way a special kind of action event. In other words, a plan execution may be composed by one or more ordered action events, targeting a particular outcome of interest to the agent. These action events may be triggered by both action and non-action events perceived by the agent. A plan execution is said to instantiate a plan (or plan type).

Analogously to an UFO-B atomic event, an atomic action event is an action event that is not further decomposed, such as ‘picking a book in a shelve’ and ‘sending a message’. Actually, ‘sending a message’ can also be seen as a subtype of an atomic action event, referred to as communicative action event. Physical agents both send and receive communicative action events. Communication is one of the most important aspects of agent-oriented systems as this triggers one agent to adopt goals or to execute action events on behalf of another. Unlike objects that simply execute actions when requested, an agent reasons over another agent’s request before agreeing on a particular course of action (Wooldridge, 1992). Communication may be also required to inform an agent about changes in one’s course of action or in the environment itself, thus altering the agent’s beliefs.

In Fig. 5, the intrinsic moment concept of UFO-A is specialized into mental moment, which denotes an intrinsic moment that is existentially dependent on a particular agent, being an inseparable part of its mental state. Examples of mental moments include a belief, a desire, and an intention. We can then say that a mental moment inheres in a physical agent (relation inheres in).

Belief regards information the agent has about the environment and about other agents. Both desire and intention refer to an agent’s goal. A desire expresses the will of an agent towards a particular state of affairs in reality, i.e., goals are considered here to be desired state of affairs. More than a desire, an intention represents an internal commitment of the agent to act towards that will. Thus, saying that an agent has an intention towards a certain state of affairs indicates that: (i) this state of affairs is desired by the agent (i.e., it is a goal of that agent); (ii) such agent has a plan to accomplish it. In other words, an intention is always associated with a plan type.

Social moment is a specialization of the UFO-A concept of externally dependent moment, including the concepts of commitment and claim. When two physical agents agree to accomplish goals to one another, a commitment/claim pair is generated between them. These deontic concepts are highly important to regulate the social relations between members of an organization. Agents may have several commitments and claims towards one another. For example, on one hand, a consultant might commit to his colleague to pass on some valuable information about a past case that he was involved with, which is similar to a present task of his colleague. On the other hand, the colleague can claim this knowledge transfer from the consultant. A pair commitment/claim composes a social relator, which is a particular type of UFO-A relator. Fig. 5 also shows that commitment and claim may refer to a goal (refers to between social moment and a goal). In other words, when a physical agent A commits to a physical agent B, this can imply that A adopts a goal of B. In this case, the social relator created between A and B state that B has the right to claim the accomplishment of this specific goal to A. Castelfranchi (1995) made an important contribution on the understanding of commitments. In one of his work, he cites Searle, who claims that “a commitment is a right producing act” (Castelfranchi, 1995), highlighting that it is much more complex for an agent to disengage from commitments towards other agents (social commitments, in Castelfranchi’s term) than to dismiss his own intentions (which Castelfranchi calls internal commitments).

Figure 6 emphasizes the difference between physical agent type and physical agent individual. Furthermore, it also depicts the difference between rigid and anti-rigid agent types, here physical agent kind and physical agent role, respectively. While person is an example of a physical agent kind, physical agent roles are specifically suited to model organizational roles (e.g., secretary, manager) as well as other roles performed by agents in specific situations that can be played independently of the position someone has in an organization (e.g., ‘coffee maker’ or ‘book reader’). As previously clarified in UFO-A, a person cannot cease to be a person (i.e., it is a rigid concept, a kind) while a secretary can be promoted into manager, or can assume another organizational position (thus being anti-rigid, a role).

Still aiming at clarifying the concept of agent role, Fig. 6 shows that an agent role is characterized by social moment types, which describe the set of general commitments and general claims a physical agent playing that role has. This is again based on the work of Castelfranchi (1995), who defines a general commitment as a commitment an agent makes towards a set of goals of the same type. For example, when agreeing to perform the organizational role of a 'secretary', one is automatically committing oneself to 'writing letters' and 'making appointments' on behalf of one's boss. Conversely, this person also has some claims a priori, such as receiving a certain salary in the end of the month and having a suitable working place. Bottazzi and Ferrario (2005) remind us that an agent's autonomy within an organization is restricted by the set of general commitments and claims he/she has, as a result of playing a specific role.

Figure 7 finally concludes our UFO-C extension, depicting the important distinction between the concepts of dependency and delegation. The first difference regards the fact that while a dependency constitutes a formal relation, a delegation consists of a material relation, following the definitions of UFO-A. Let us examine this difference in further detail. The figure shows that a dependency connects two physical agents (a depender and a dependee) and a dependum, which can be either a goal or a resource. An agent A (the depender) depends on an agent B (the dependee) regarding a goal G if G is a goal of agent A, but agent A cannot accomplish G, and agent B can accomplish G. Here, the fact that an agent cannot accomplish a goal may mean that this agent either does not have the capability to achieve it. Or else, it may denote that this agent's pursuit towards this goal may interfere with his other intentions, such that he/she decides not to pursue this goal after all. This may well be a reason why agent A decides to delegate such goal accomplishment to agent B. A delegation is thus associated with a dependency but it is more than that. As a material relation, it is founded on something more than its connected elements. In this case, the connected elements are two physical agents (delegator and delegatee) and a goal (delegatum), and the foundation of this material relation is the social relator (i.e. a commitment/claim pair) established between the two physical agents involved in this delegation. In other words, when agent A delegates a goal G to agent B, besides the fact that A depends on B regarding G, B commits himself to accomplish G on behalf of A. Goal and plan delegation refer to what Castelfranchi and Falcone (1998) define as open and close delegation, meaning that the former leaves the decision regarding the strategy towards goal accomplishment to the depender. The latter rather prescribes a specific strategy (i.e. a plan) the depender should adopt towards achieving the delegated goal.

To illustrate the difference between dependency and delegation, consider the following case. Suppose John is a program committee member of a certain conference and that he received from Paul (the conference program chair) an article X to review. Suppose that John cannot review this article by himself, since there are some aspects of the article that are outside his field of competence. Now, suppose that George is a colleague of John who is knowledgeable exactly in those aspects that John needs to review article X. In this case, we could say that John depends on George to review article X. Notice, however, that this relation between John and George can be reduced to relations between the goals and capabilities of these individual agents. Moreover, this relation does not even require that the related agents are aware of this dependence. This is certainly not the case for the relation between Paul and John. As the program committee chair, Paul depends on John to review article X. However, in this case, not only they are both aware of this dependence but there is the explicit commitment of John to Paul to review article X. In other words, the delegation of Paul to John to review article X cannot be reduced to relations between their intrinsic moments, but it requires the existence of a certain relator (a commitment/claim pair) that founds this relation.

Not explicit in the diagram is the concept of *socially can achieve*, or *socially can execute*. When we say above that a certain agent can achieve a goal, this means that such agent is able to do it himself or can delegate to another agent that can accomplish it on his behalf. In the example above, if John can review part of article X by himself and can delegate a remaining part to George, we could say that John socially can achieve the goal of reviewing article X.

Similarly to delegation, resource acquisition is also a material relation associated with the same concepts of dependency and social relator. We created this as a different concept because when agent A needs access to a resource R controlled by agent B, it is awkward to say that agent A delegates resource R to agent B. Moreover, this relation is differentiated as follows: an agent A acquires a resource R from agent B is equivalent to say that agent A needs to use resource R, agent A does not control resource R, agent B controls resource R, and agent B commits himself to give agent A access to resource R. In an alternative formulation we can say that if agent A acquires resource R from agent B then: (i) there is a resource dependence from A to B w.r.t. R; (ii) A and B are mutually aware of this dependency; (iii) B socially commits to give A access to R.

4. Evaluating ARKnowD Notation

When conceiving a novel modeling language, one should obviously be concerned with its quality. Two aspects that influence the quality of a modeling language are: (i) how well the language is able to represent phenomena in its domain of discourse (also referred as *domain appropriateness*); (ii) how clearly the language is able to communicate such phenomena to the eventual readers of its models (also referred as *comprehensibility appropriateness*) (Guizzardi, 2005). ARKnowD's language comprehends both the notation of Tropos and AORML. It is thus important to verify the quality of these languages individually, but especially the consistency in their integration to generate ARKnowD's language.

4.1. Evaluation Method

Guizzardi (2005) provides a framework for evaluating domain appropriateness and comprehensibility appropriateness of modeling languages. This framework is based on the construction of a domain ontology to describe the conceptual domain of discourse. This ontology is then used as a reference model for the modeling language, i.e. for verifying how well this modeling language is able to represent the concepts and relations represented in the ontology.

Given the ontology elaborated and described in the previous section, we intend to apply this method to evaluate ARKnowD's language. The evaluation criterion is based on a number of properties that should hold for the mapping between a representation (or system of representations) and the portion of reality it is supposed to represent. These properties are *lucidity, soundness, laconicity* and *completeness* if we operate at the level of individual models, i.e., analyzing the mapping between individual models and what they are supposed to represent. In a complementary way, we consider the properties of *construct overload*, *construct excess*, *construct redundancy* and (language) *completeness* when operating at the level of a system of representation, i.e., when considering the mapping between a language and a conceptualization of given domain. These eight properties are briefly described in the sequel. For an in depth presentation of this method one should refer to (Guizzardi, 2005).

A model is considered *lucid* according to a conceptualization if each of its constructs represents at most one entity of this conceptualization. Although not exactly the same, non-lucidity is closely linked to *construct overload*, i.e. having a single language construct representing two or more ontological constructs. As stated in (Guizzardi, 2005, p. 31), "Construct overload is considered an undesirable property of a modeling language since it causes ambiguity and, hence, undermines clarity. When a construct overload exists, users have to bring additional knowledge not contained in the specification to understand the phenomena which are being represented.”

Soundness refers to the property of a model of representing solely the entities of the domain conceptualization. Having a construct in a modeling language that does not map to any ontological construct is also known as *construct excess*. The presence of this extra construct should be avoided since it undermines the understanding of the specification. In other words, the clarity of a specification is improved if the reader is able to link the language constructs to the entities of the domain of discourse.

A model is said *laconic* if it possesses only one construct to represent each phenomenon in the domain or discourse (i.e., each entity in the domain ontology). Conversely, the same conceptual entity may be represented by two or more constructs in a specification, consequently adding confusion to the meaning of the model. A reader may ask himself, for example, if the two constructs are actually the same or if there is any semantic distinction between them. Non-laconicity is then related to *construct redundancy* at the language level, which besides turning more difficult the understanding of specifications adds unnecessary complexity to the modeling language.

A model is said to be *complete* if every concept in the represented domain conceptualization is covered by at least one element of this model. This is directly linked to expressivity or completeness at the language level. *Language incompleteness* entails lack of expressivity, i.e., there can be phenomena in the considered domain that cannot be represented by the language. Alternatively, users of the language can choose to overload an existing construct in order to represent concepts that originally could not be represented, thus, undermining clarity. Thus, unless some existing construct is overloaded, an incomplete modeling language is bound to produce incomplete models.

4.2 Evaluation

Taking the ontology presented in section 3 and based on the method briefly described above, we found a few problems in the current Tropos and AORML notations. Consequently, we here suggest a few adjustments in order to proceed with their use in ARKnowD. It is important to point out that these languages are here considered in integration with one another, so for example, if one comprehends a set of ontological concepts, the lack of these same concepts in the other is not considered incompleteness.

4.2.1. Evaluating the Tropos Language

In Tropos, there are one case of construct redundancy, one case of construct excess (and, hence, unsoundness), two cases of incompleteness, and one case of construct overload.

Construct Redundancy and Unsoundness. First, let us address the cases of construct redundancy and unsoundness together. In Tropos, besides the concept of agent and role, corresponding to our ontological concepts of physical agent kind and physical agent role, there are two other concepts: actor and position. Figure 8 depicts these concepts and their corresponding notations.

The concept of position is considered solely with the purpose of aggregating different roles. Let us analyze this concept a bit further. As stated in section 3.3, a physical agent role is characterized by the set of social moments types, i.e. general commitments and claims a physical agent playing such a role agrees to. As a role aggregation, a position is characterized by the union of the social moment types characterizing its aggregating roles. Regarding the meta-property of rigidity, a position like a role is an anti-rigid type. Therefore, a position as much as a Role can be seen as an anti-rigid type characterized by a set of social moment types. Hence, we conclude that there is no real distinction between the ontological concepts of role and position, thus not justifying the use of two different language primitives instead of one. To put it differently, the presence in Tropos of both role and position to represent the same ontological concept constitutes a case of construct redundancy.

Still referring to Fig. 8, we note that actor is a general concept that can refer to an agent, to a role or to a position. However, we find no reason why to consider such a concept. Some claim that in i* (Yu, 1995), agent is defined as *an instance of* the actor type. However, this is not clear in this picture, also present in the i* original proposal. Moreover, such definition would only add confusion to the modeling language, since there is no reason whatsoever to name an instance differently from its type.

Other researchers, who apply i* for requirements analysis combined with object-oriented design claim that the concept of agent is important because not all system actors from the analysis activity become agent-oriented systems, thus agent could be used to differentiate object-oriented and agent-oriented systems (i.e., representing the latter). However, we find this a misconception since the term actor already implies that such an entity is ‘one who acts’, thus it is an active entity and could not represent an object anyway. In our view, there is no inconsistency in applying agents to analyze a domain and then, design the system using object-oriented methodologies or others. In this case, agents are applied as modeling metaphors only during analysis and thus, their behavior is modeled as active entities, even if later, the chosen implementation technology will lead them to be designed as passive ones. Consequently, such an explanation does not justify the use of two concepts, i.e., actor and agent.

We conclude that the concept of actor leads to unsoundness, causing many of i* and Tropos literature not to make use of the four available concepts of actor, agent, position and role. Many times, authors limit themselves to one or two of these concepts or even worse, apply them inconsistently (for instance, while in one work, agent is considered an instance of an actor, in another, agent is applied to differentiate between agent-oriented systems and object-oriented systems).

Hence, our proposed solution considers only the concepts of agent and role. Dismissing the position and actor concepts, the analyst should identify from start, if a domain participant is an agent or a role (rigid and anti-rigid concepts, respectively). For representing an agent, ARknowD adopts Tropos's actor notation (empty circle) for being the simplest form, maintaining the current Tropos notation for roles.

Incompleteness and Construct Overload. Going forward with the evaluation of the Tropos language, we address the incompleteness and non-lucidity issues by considering the case illustrated in Figure 9.

Fig. 9 depicts the following situation: the department manager of an organization relies on the department secretary to make an appointment for a meeting with all the employees of the department. For that, the secretary, on her turn, depends on a specific calendar system named eDate. The secretary is always checking for free new versions of the eDate system in the developer's website, aiming at profiting from new functionality and enhancements in this system. Department manager, department secretary, eDate, and eDate system developer are examples of Tropos agents, which correspond to the UFO-C concept of physical agent.

In part A), however, it is not possible to differentiate between physical agent type and physical agent individual. For instance, it is not clear if we talk about a specific secretary and a specific system, or general ones. Representing both ontological concepts using only one language construct is understood as construct overload. This may be in the way for a clear understanding of the modeled setting. We therefore provided in B), a way to differentiate these two entities. Inspired by UML, we chose to underline the name of physical agent individuals to point out the difference between them and physical agent type, thus imitating the way UML differentiates between instances and classes. Our choice provides a suitable connection between Tropos and AORML, which already adopts this UML strategy. Following such strategy, part B) depicts eDate as an individual and all other agents as types. The choice of making the others as agents is to maintain a level of generality for the model, for instance this case would typically hold even if the secretary were changed. The new one would continue to be responsible for setting up meetings on behalf of her boss, and using the same system to do so.

A case of incompleteness that can be noted in Fig. 9 A) refers to the lack of language expressivity of the Tropos language to model the concept of dependency. What Tropos usually terms dependency is actually a case of delegation according to our ontology. As seen in section 3.3, the latter is stronger than the former, as besides dependency, it also involves commitment from the dependee in relation to the depender. In part A) of Fig. 9, the delegation depicted between the secretary and the eDate system developer is actually only a dependency. The secretary does not know the developer, who on his turn has no way to commit specifically to her on releasing new system's versions. In other words, if the developer decides to stop providing free releases and rather to start charging for new eDate versions, the secretary does not have a claim towards him and will just have to accept this fact. To correct this expressivity problem, we created a new symbol to distinguish dependency from delegation. This is a similar arrow as used before, however empty headed to denote the lack of commitment. This new symbol is illustrated in Fig. 9 B). Please note that it is not our intention to determine whether both dependency and delegation are to be applied in all modeled cases. It may be that in some specific cases, only dependencies (or only delegations) make sense. However, with this proposal, we make sure that the language is expressive enough to capture both concepts since in some situations (as the one just illustrated), both dependency and delegation are present.

The other case of incompleteness, actually refers to the concept of resource acquisition, differentiated from resource dependency by the commitment of the acquisitee to deliver a given resource to the acquisitor. Figure 10 depicts the modeling constructs used for the three kinds of dependencies, as well as for resource acquisition, goal and plan delegations. The resource acquisition notation maintains uniformity regarding the ones used for goal and plan delegation, showing that analogously to these relations, a resource acquisition is a resource dependency added by a commitment (in this case, the commitment of the acquisitee to provide the acquisitor with access to the acquisitum).

4.2.2. Evaluating AORML

In AORML, we found one case of construct overload. This regards the notation used at the same time to model a non-agentive object and a belief. To correct this problem, we use stereotypes, an UML construct commonly used to extend this language, differentiating old and new entities. Such construct is already applied in AORML, for example to distinguish between human, institutional and artificial agents. Figure 11 shows our proposed solution, depicting a typical situation involving a library institutional agent and a borrower human agent. The library uses an information system to organize its book collection. The borrower borrows books, having his own internal beliefs related to these books. The figure differentiates the actual books from the agent's internal beliefs by stereotyping the belief class.

5. ARKnowD Transformations: Mapping Tropos into AORML

As previously explained, it is necessary to provide a transformation method to convert the notation of the models of the different viewpoints. In ARKnowD, this is done by mapping Tropos concepts to AORML concepts, now made semantically explicit by the presented UFO-C ontology. Table 1 depicts this mapping, previously presented in (Guizzardi, 2006).

In Tropos, an agent models an entity that has strategic goals and intentionality within the system or the organizational setting. This concept directly maps to one of the three types of agents in AORML: human, artificial or institutional agent, depending on its nature.

Tropos's plans may indicate paths for AORML's interaction modeling. In other words, for each plan in a Tropos model, there can be an AOR Interaction Sequence Diagram, modeling the interactions of the agents participating in this plan (i.e. agents having the plan, or being connected to it by a delegation link).

Capabilities in Tropos may be seen as a set of plans and, therefore, could be mapped to the set of interaction modeling paths (i.e. set of AOR Interaction Sequence Diagrams), representing the agent's plans.

Analogously, resources that represent physical or information entities in Tropos become objects according to AORML conceptualization.

Additionally, in Tropos, goal, plan and resource dependency between two agents indicate that one agent depends on the other in order to achieve some goal, execute some plan, or obtain some resource. Because such dependency link indicates a kind of relation between the two agents (depender and dependee), an association link may be depicted between these agents in an AOR Agent Diagram, typically used for information modeling. Here, we consider the differences between dependency, delegation and resource acquisition pointed out in section 3. As mentioned in that section, besides involving dependency between agents, delegation implies that the delegatee actually agreed to accomplish a goal or perform a task on behalf of the delegator. Thus, a commitment is established from the delegatee regarding the delegator (or a claim emerges from the delegator towards the delegatee). Therefore, goal and plan delegations leads to the establishment of AORML commitments/claims between agents, usually depicted in interaction modeling, using one or more types of AOR interaction diagrams. Resource acquisition is treated analogously to goal and plan delegation, since as previously discussed in section 3, these concepts have similar nature. In other words, also in the case of resource acquisition, an association link in the AOR Agent Diagram, and a commitment/claim link are assumed to exist between the two agents (the acquisitor and the acquisitee).

Note that one of the most important entities in Tropos, the concept of 'goal', is not mapped into AORML. This relates to the fact that ARKnowD applies goal modeling exclusively for requirements elicitation and analysis and architectural design. On detailed design, all goals have already been dealt with. Goals may have been fulfilled or abandoned. But most commonly, goal analysis leads to the delegation of unsolved goals to new or old agents, who are either part of the organization or a new information system. And finally, concrete plans are assigned to goals with the purpose of accomplishing them. Consequently, when the detailed design activity starts, plans should be modeled rather than goals. As observed in Table 1, plan modeling may be done through the use of AOR Interaction Sequence Diagrams, which detail the protocol of communication between agents to realize a specific sequence of actions.

6. Conclusion

This chapter described an ontology-based strategy to identify correspondences between two distinct agent-oriented modeling languages, namely, i.e. the Tropos language and AORML, enabling them to be used in two distinct development activities of the ARKnowD methodology. For making such correspondences, we proposed an MDA-inspired transformation method, which requires a mapping between the metamodels of the two notations. To guarantee the consistency of this mapping, we use an ontological approach aimed at making explicit the semantics of the applied agent-oriented concepts. The developed ontology also serves as a reference model for the evaluation of the adopted notations, allowing us to improve the clarity and consistency of the resulting modeling language.

The development of a philosophically well-founded upper level ontology is an important step towards the definition of real-world semantics for conceptual modeling and agent-oriented concepts. The partition of the Unified Foundational Ontology (UFO) employed here reflects a certain stratification of our world. However, it also reflects different degrees of scientific consensus: there is more consensus about the ontology of endurants (UFO-A) than about the ontology of perdurants (UFO-B), and there is more consensus about the ontology of perdurants than about the ontology of intentional and social things (UFO-C). In particular, the extension of UFO-C proposed in this article should be regarded as work in progress, and as a preliminary attempt towards the characterization of concepts constituting the reality of organizations and social relations.

References

1. Bottazzi, E., & Ferrario, R. (2005). A Path to an Ontology of Organizations. *Proceedings of the Workshop on Vocabularies, Ontologies, and Rules for the Enterprise 2005* (pp. 9-16). Enschede: Centre for Telematics and Information Technology.
2. Castelfranchi, C. (1995). Commitments: From Individual Intentions to Groups and Organizations. In *Proceedings of the First International Conference on Multi-Agent Systems* (pp. 41-48). AAAI-Press and MIT Press.
3. Castelfranchi, C., & Falcone, R. (1998). Towards a Theory of Delegation for Agent-Based Systems. *Robotics and Autonomous Systems*, (24), 141–157.
4. Guizzardi, G. (2005). Ontological Foundations for Structural Conceptual Models. Ph.D. thesis, Enschede: University of Twente.
5. Guizzardi, G., & Wagner, G. (2005). Some Applications of a Unified Foundational Ontology in Business Modeling. In P. Green and M. Rosemann (Eds.), *Business Systems Analysis with Ontologies* (pp. 345-367). London: Idea Group.
6. Guizzardi, R.S.S. (2006). Agent-oriented Constructivist Knowledge Management. Ph.D. thesis, Enschede: University of Twente.

7. Harmsen, F., Brinkkemper, S., & Oei, H. (1994). Situational method engineering for information system project approaches. In A.A. Verrijn-Stuart and T.W. Olle (Eds.), *Methods and associated tools for the information systems life cycle* (pp. 169-194). Amsterdam: Elsevier.
8. Henderson-Sellers, B. (2005). Creating a Comprehensive Agent-Oriented Methodology Using Method Engineering and the OPEN Metamodel. In B. Henderson-Sellers and P. Giorgini (Eds.), *Agent-Oriented Methodologies* (pp. 368-397). London: Idea Group.
9. Juan, T., Sterling, L., & Winikoff, M. (2004). Assembling Agent-Oriented Software Engineering Methodologies from Features. In P. Giorgini, J. P. Muller and J. Odell (Eds.), *Agent-Oriented Software Engineering IV* (pp. 198–209). Berlin: Springer-Verlag.
10. Miller, J., & Mukerji, J. (2003). MDA Guide Version 1.0.1, omg/2003-06-01. Reference guide published in MDA's site.
11. Sabas, A., Delisle, S., & Badri, M. (2002). A Comparative Analysis of Multiagent System Development Methodologies: Towards a Unified Approach. *Proceedings of the 16th European Meeting on Cybernetics and Systems Research* (pp. 599-604), Vienna, Austria.
12. Susi, A., Perini, A., Mylopoulos, J., & Giorgini, P. (2005). The Tropos Metamodel and its Use. *Informatica*, 29(4), 401-408.

13. Wooldridge, M.J. (1992). Intelligent Agents. In G. Weiss (Ed.) *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence* (pp. 27-78). Cambridge: MIT Press.

14. Yu, E. (1995). Modeling Strategic Relationships for Process Reengineering. Ph.D. thesis, Toronto: University of Toronto.

[i] In (Guizzardi, 2005), a much richer system of substantial types is presented. There, Roles are characterized both in terms of anti-rigidity, relational dependence, and the sortal/non-sortal distinction. Together these metaproperties can differentiate Roles from other anti-rigid substantial types such as *Role Mixins* or *Phases*.

Tropos Concepts	**AORML Constructs**
agent	agent
Plan	AOR Interaction Sequence Diagram
Capability	set of AOR Interaction Sequence Diagram
Resource	object
Dependency	AOR Agent Diagram association relation
Delegation	AOR Agent Diagram association relation/AOR commitment
resource acquisition	AOR Agent Diagram association relation/AOR commitment

Table 1. Mapping Tropos into AORML

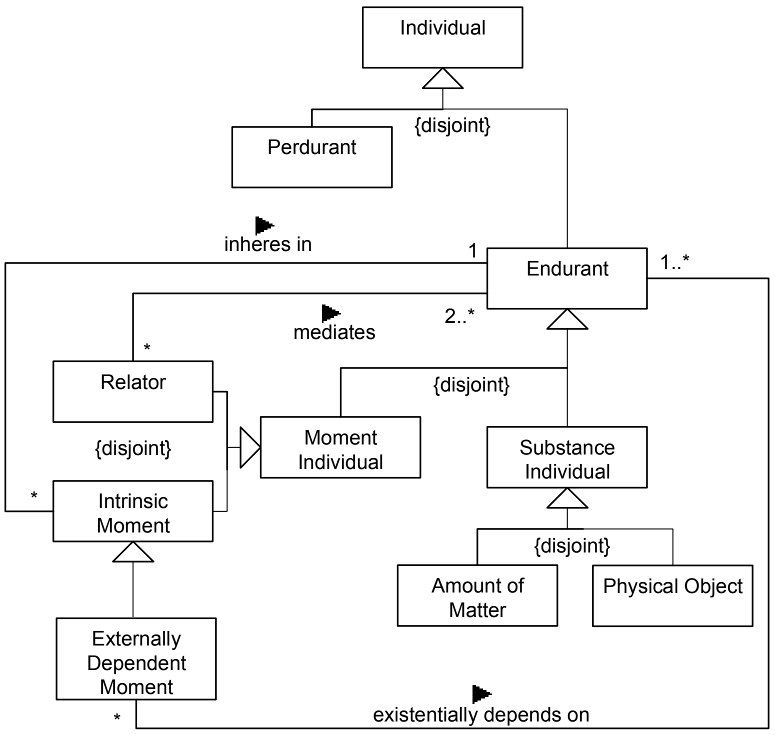

Figure 1. Different kinds of individuals in UFO-A

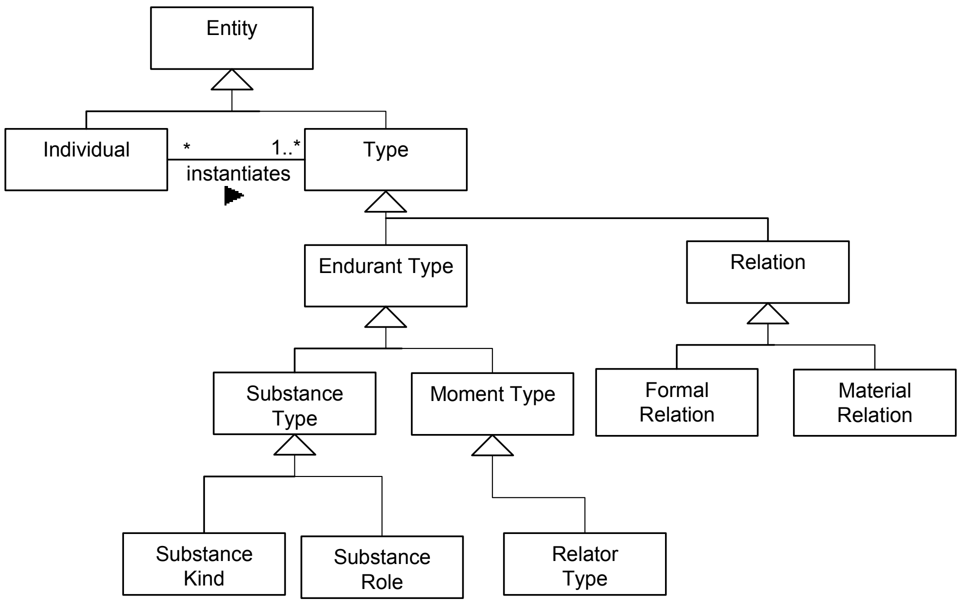

Figure 2. UFO-A: differentiating between Kind and Role

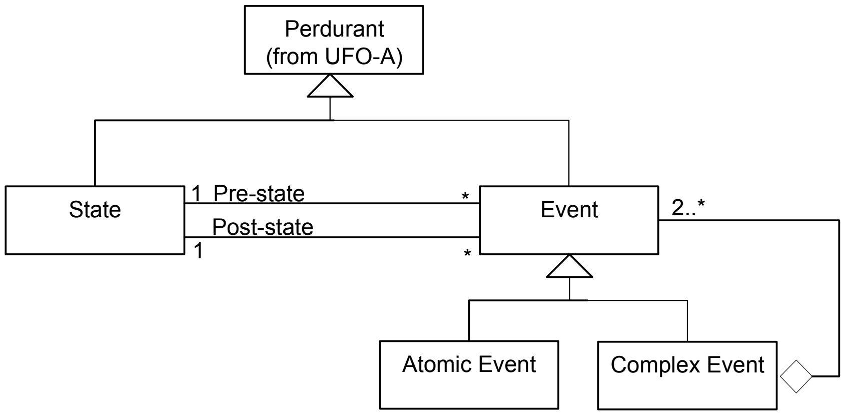

Figure 3. UFO-B: understanding perdurants in details

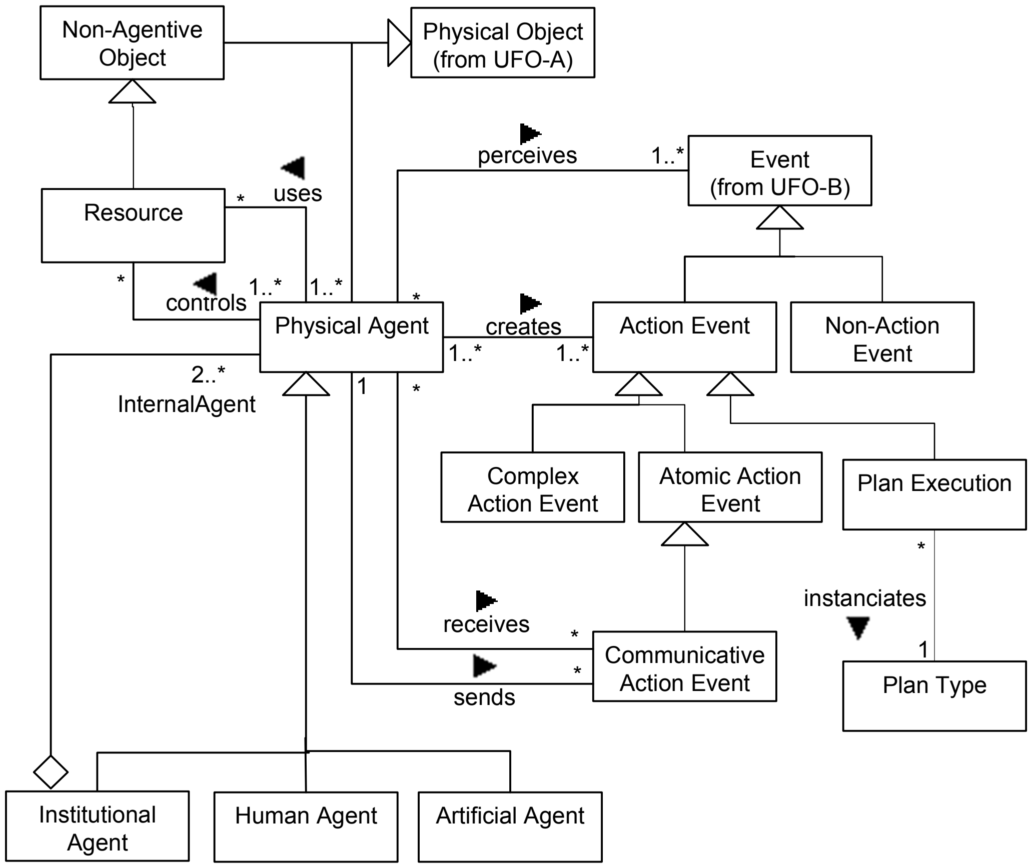

Figure 4. Extending UFO-C from the UFO-A concept of Physical Object and the UFO-B concept of Event

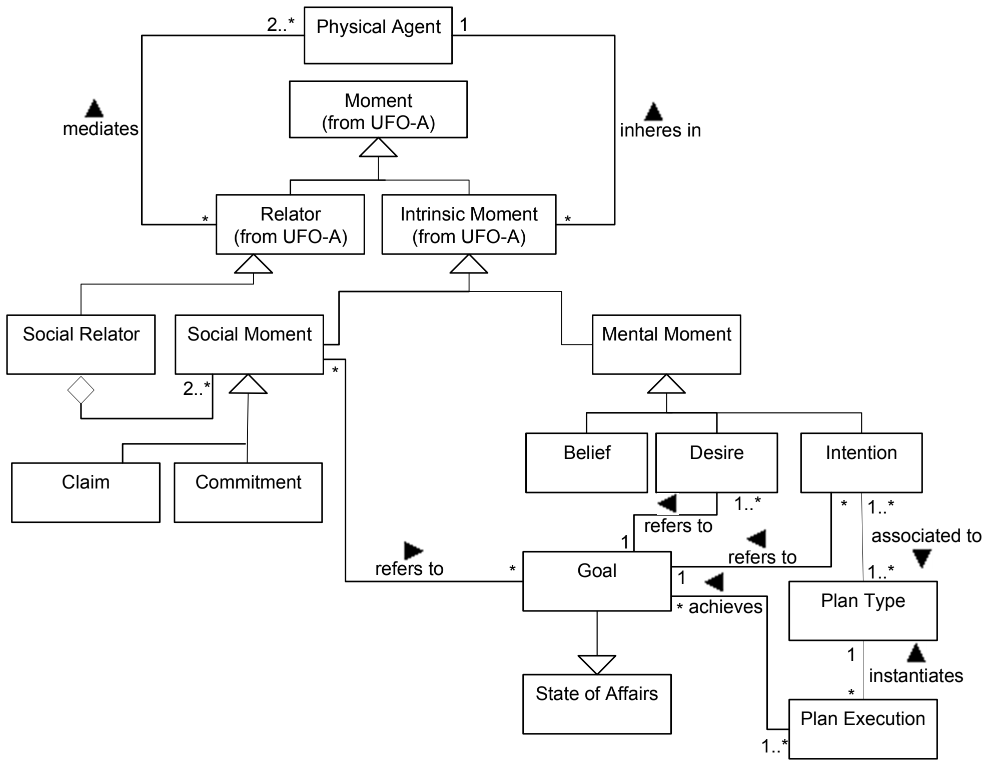

Figure 5. Extending UFO-C from the UFO-A concept of Moment

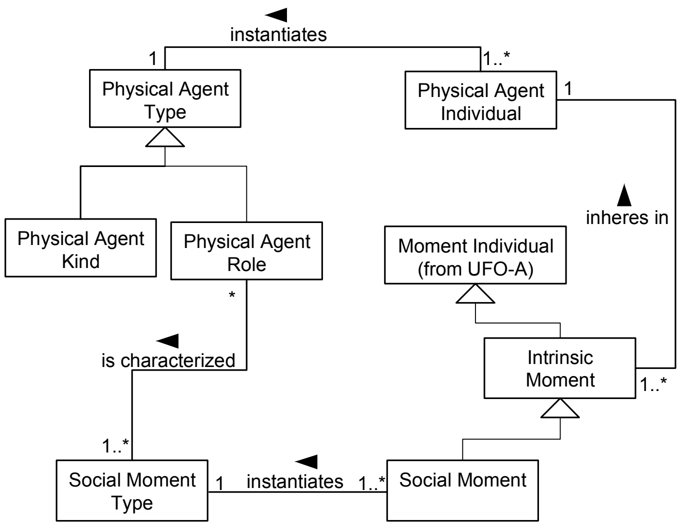

Figure 6. Pointing out the difference between Physical Agent Kind and Physical Agent Role

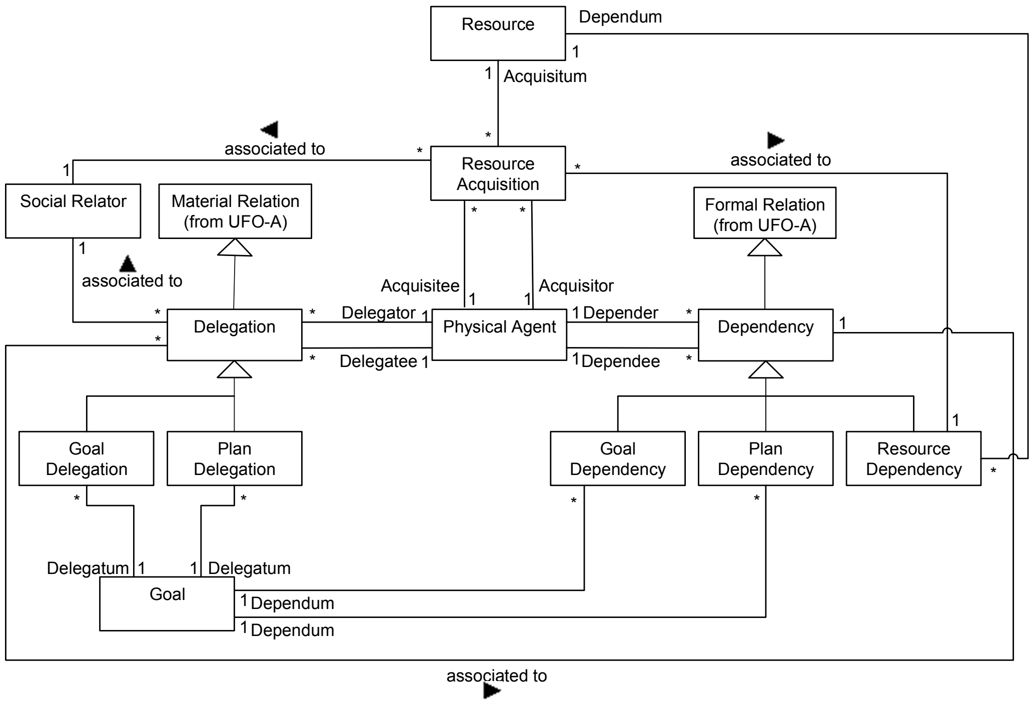

Figure 7. Distinguishing between Dependency, Delegation and Acquisition relations

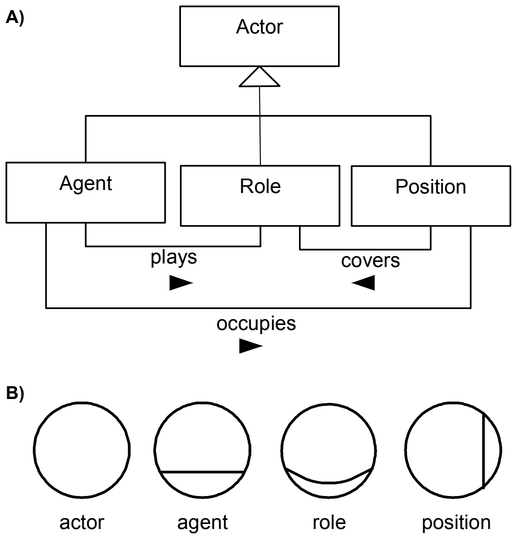

Figure 8. A) an excerpt of the Tropos's metamodel showing the concept of actor and its specializations (Susi, Perini, Mylopoulos & Giorgini, 2005) and B) corresponding notations

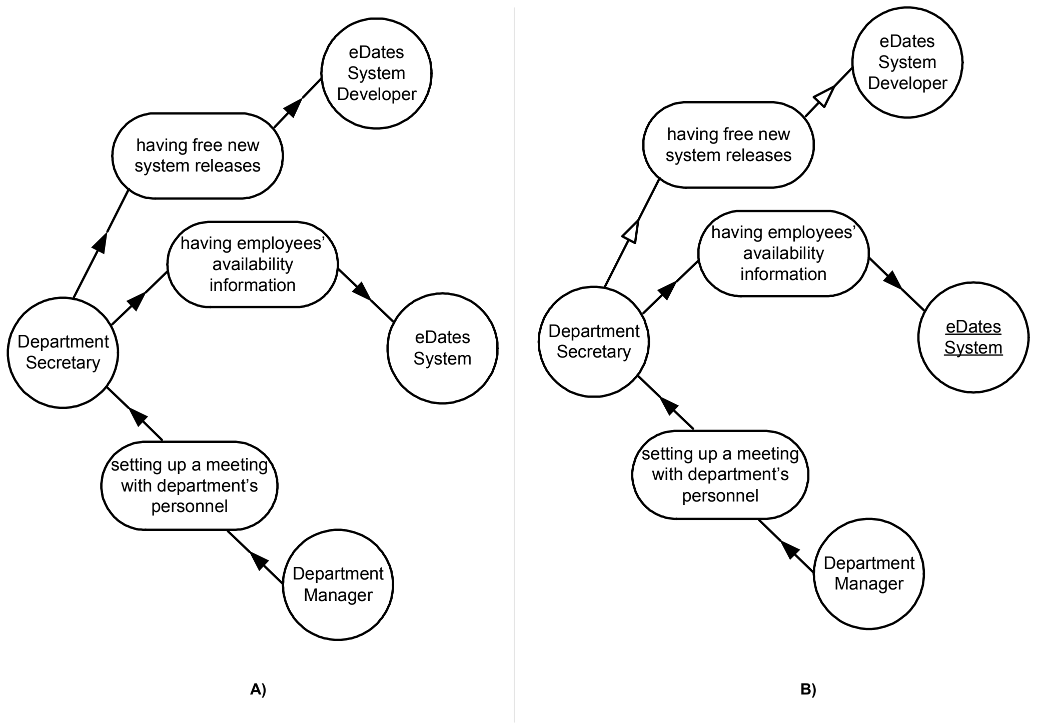
eDates System Developer
having free new system releases
having employees' availability information
Department Secretary
eDates System
setting up a meeting with department's personnel
Department Manager
A)
eDates System Developer
having free new system releases
having employees' availability information
Department Secretary
eDates System
setting up a meeting with department's personnel
Department Manager
B)

Figure 9. Correcting two cases of incompleteness

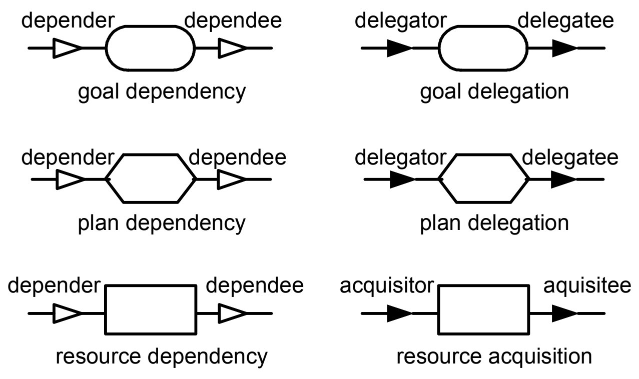

Figure 10. Differentiating the three types of dependencies, goal and plan delegation, and resource acquisition

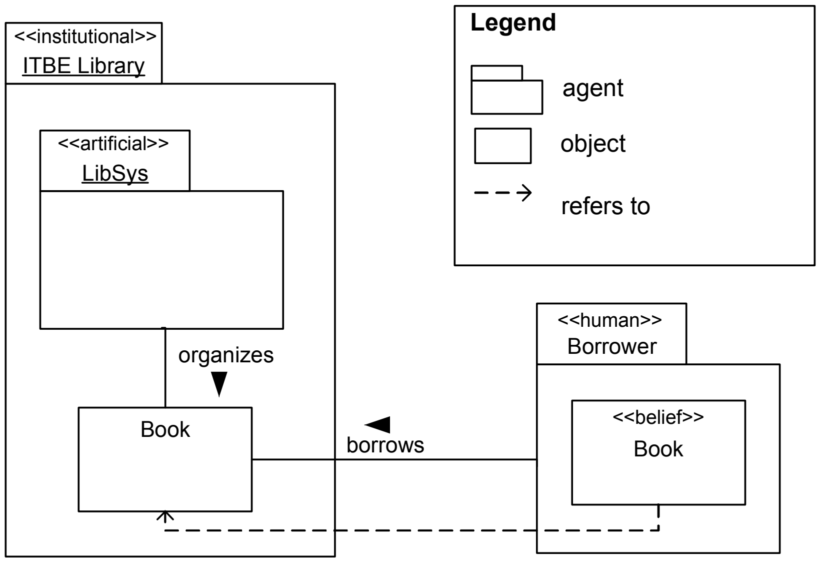

Figure 11. Distinguishing beliefs from non-agentive objects in AORML