The Tropos visual modeling language. A MOF 1.4 compliant meta-model.

Davide Bertolini[1], Anna Perini[1], Angelo Susi[1] and Haralambos Mouratidis[2]
[1]ITC-IRST, Via Sommarive 18, I-38050, Trento, Italy
{bertolini,perini,susi}@irst.itc.it
[2]School of Computing and Technology
University of East London, Barking Campus, Dagenham

Contribution for the AOSE TFG meeting, February 28th, Ljubljana, Slovenia, collocated with the Second AgentLink III Technical Forum (AL3-TF2) http://www.agentlink.org/activities/al3-tf/tf2/

Premise.
One of the two objectives of the AOSE TGF meeting to be held in Ljubljana, on February 28th is that of collecting, refining and possibly merging Agent-Oriented (AO) modeling languages meta-models. Our aim in this paper is to contribute to the achievement of this objective by describing the Tropos modeling language meta-model that we have implemented in a Tool for AO Modeling (**http://sra.itc.it/tools/taom/**).This meta-model follows the specification given in [Bre04] and in a master thesis [San01]. Moreover, we describe our approach in its implementation in the *Eclipse* [ECLIPSE] development platform.
In this work we refer to the Model Driven Initiative (MDA) recommendations and standards, in particular the Tropos meta-model specification here described is compliant to the Meta Object Facilities (MOF) directives that allow to specify, build and manage technology neutral meta-models.
In the following we recall the essential of the Tropos methodology and the main concepts of its modeling language, as described in [Bre04][1]. We then describe the modeling language meta-model and propose some solutions towards a possible integration with the "unifying meta-model" presented in [Ber04], to be discussed at the meeting. Finally, we describe briefly how we implemented the Tropos meta-model.

1. The Tropos methodology

The **Tropos** methodology is intended to support an AO approach for the analysis and design activities of the software development process; from the application domain analysis down to the system implementation. In particular, Tropos rests on the idea of building a model of the system-to-be and of its environment, which is incrementally refined and extended, by providing a common interface to the various software development activities, as well as a basis for documentation and evolution of the software.

1.1 The Tropos development process

The set of activities (or *disciplines*, according to the terminology adopted in the Unified Process) for requirements analysis in the Tropos methodology, are **Early Requirements** and **Late Requirements** analysis. During Early Requirements, the analyst focuses on the understanding of a problem domain by studying an *existing organizational setting* where the system-to-be will be introduced. Social actors and software systems that are already present in the domain are modeled as actors with their individual goals and with mutual, intentional dependencies.
Late Requirement analysis focuses on the system-to-be, which is introduced as a new actor into the model. The system actor is related to the social actors in terms of dependencies and its goals are analyzed. This will eventually lead to revise and add new dependencies involving a subset of the social actors (the users).

On the other hand, the **Architectural Design** and the **Detailed Design** focus on the system specification, according to the requirements resulting from the above phases. In Architectural design the system's global architecture is specified in terms of subsystems, which are represented as

[1] Let's recall that several research groups are currently contributing to the definition of methods and techniques that can be exploited in the Tropos methodology. For more information see http://www.troposproject.org/.

actors, who they are assigned subgoals and/or subplans of the goals and plans assigned to the system. The Detailed design activity, takes into account the target implementation platform that has been chosen and provides a detailed design of the system components (agents) that will map directly to the code.

Finally, the **Implementation** activity produces an implementation skeleton according to the detailed design specification. Code is then added to the skeleton using the programming language supported by the implementation platform.

The methodology has been illustrated by several case studies [Per03,Giu01, Cas01].

1.2 The Tropos modeling language

The Tropos modeling language rests on the i* notation [Yu95]. It provides a set of concepts derived from agent paradigms and a visual notation. In the following we quote some of the key concepts definitions given in [Bre04][2], namely:

- **Actor**, which models an entity that has strategic goals and intentionality within the system or the organizational setting. An actor represents a physical or a software agent as well as a role or position. While we assume the classical AI definition of software agent, that is, a software having properties such as autonomy, social ability, reactivity, proactivity, in Tropos we define a **role** as an abstract characterization of the behaviour of a social actor within some specialized context or domain of endeavour, and a **position** as a set of roles, typically played by one agent. An agent can occupy a position, while a position is said to cover a role.
- **Goal**, which represents actors' strategic interests. We distinguish hard goals from softgoals, the second having no clear-cut definition and/or criteria for deciding whether they are satisfied or not. Softgoals are typically used to model non-functional requirements.
- **Plan**, which represents, at an abstract level, a way of doing something. The execution of plan can be a means for satisfying a goal or for satisficing a softgoal.
- **Resource**, which represents a physical or an informational entity.
- **Dependency** between two actors, which indicates that one actor depends, for some reason, on the other in order to attain some goal, execute some plan, or deliver a resource. The former actor is called the depender, while the latter is called the dependee. The object around which the dependency centres is called dependum. In general, by depending on another actor for a dependum, an actor is able to achieve goals that it would otherwise be unable to achieve on its own, or not as easily, or not as well. At the same time, the depender becomes vulnerable. If the dependee fails to deliver the dependum, the depender would be adversely affected in its ability to achieve its goals.
- **And/Or Decomposition.** Each goal can be analyzed from the point of view of the individual actor considering: possible sub-goals. In the case of an AND decomposition, the sub-goals has to be all fulfilled in order to fulfil the goal that represents the root of the decomposition; in the case of an OR decomposition the sub-goals represent alternative ways to achieve the root goal.
- **Means/end Analysis**. Given a goal, the means/end relationship specifies a means (in terms of a goal, a plan or a resource) to satisfy the goal.
- **Contribution.** Given a goal, the contribution relationship specifies the goals or plans or resources that can contribute positively or negatively to its achievement. The measure of the positive or negative degree of contribution is expressed via the qualitative metrics +, ++, -, --. In particular, if the goal g1 contributes positively to the goal g2, with metric ++ then if g1 is satisfied, so is g2. Analogously, if the plan p contributes positively to the goal g, with metric ++, this says that p fulfills g. A + label for a goal or plan contribution represents a partial, positive contribution to the goal being analyzed. With labels --, and - we have the dual situation

[2] We refer to that paper also for a description of the visual notation and for examples of the various Tropos diagrams.

representing a sufficient or partial negative contribution towards the fulfilment of a goal. Contribution analysis applied to softgoals is often used to evaluate non-functional (quality) requirements.

2. The modeling language meta-model

The abstract syntax of the language has been given in terms of a UML meta-model [San01].
A Tropos model is a directed labelled graph whose nodes are instances of meta-classes of the meta-model, namely actor, goal, plan and resource, and whose arcs are instances of the meta-classes representing relationships between them, dependency, means-end analysis, contribution and AND/OR decomposition.
In this section we describe a revision of the UML meta-model given in [Per04] which is compliant with the MOF 1.4 defined by OMG. For the sake of readability, we consider portions of the Tropos meta-model referring to the main language concepts.

The concept of Actor.
In the Tropos metamodel, an actor is represented as a UML class as shown in the UML class diagram of Figure 1. Each actor can have 0…n goals, and each goal is wanted by 1[3] actor as specified by the UML association relationship[4]. It is worth mentioning that the Goal UML class is used to represent both hard and soft goals. As mentioned above, actors also have dependencies. In our metamodel, an actor dependency is a quaternary relationship represented as a UML class. A dependency relates respectively a depender, a dependee, and the dependum (as defined earlier), and it also provides an optional reason for the dependency (labelled why).

The concept of Goal.
The concept of goal is represented by the class Goal in the UML class diagram depicted in Figure 2. The distinction between hard and softgoals is captured through a specialization of Goal into subclasses Hardgoal and Softgoal, respectively. Goals can be analyzed, from the point of view of an actor, performing means-end analysis, contribution analysis and AND/OR decomposition (listed in order of strength). Let us consider these in turn.
Means-end Analysis is a ternary relationship defined among an actor, whose point of view is represented in the analysis, a goal (the end), and a Plan, Resource or Goal (the means). Means-end analysis is a weak form of analysis, consisting of a discovery of goals, plans or resources that can provide means for reaching a goal.
Contribution Analysis is a ternary relationship between an actor, whose point of view is represented, and two goals. Contribution analysis strives to identify goals that can contribute positively or negatively towards the fulfilment of a goal. A contribution can be annotated with a qualitative metric, denoted by +,++,-,--.
AND/OR Decomposition is also a ternary relationship which defines an AND- or OR-decomposition of a root goal into subgoals.

[3] Note that in [Per04] this cardinality was 1..n..
[4] As already stated belief are not yet represented in our proposal, while they are in [Per04]

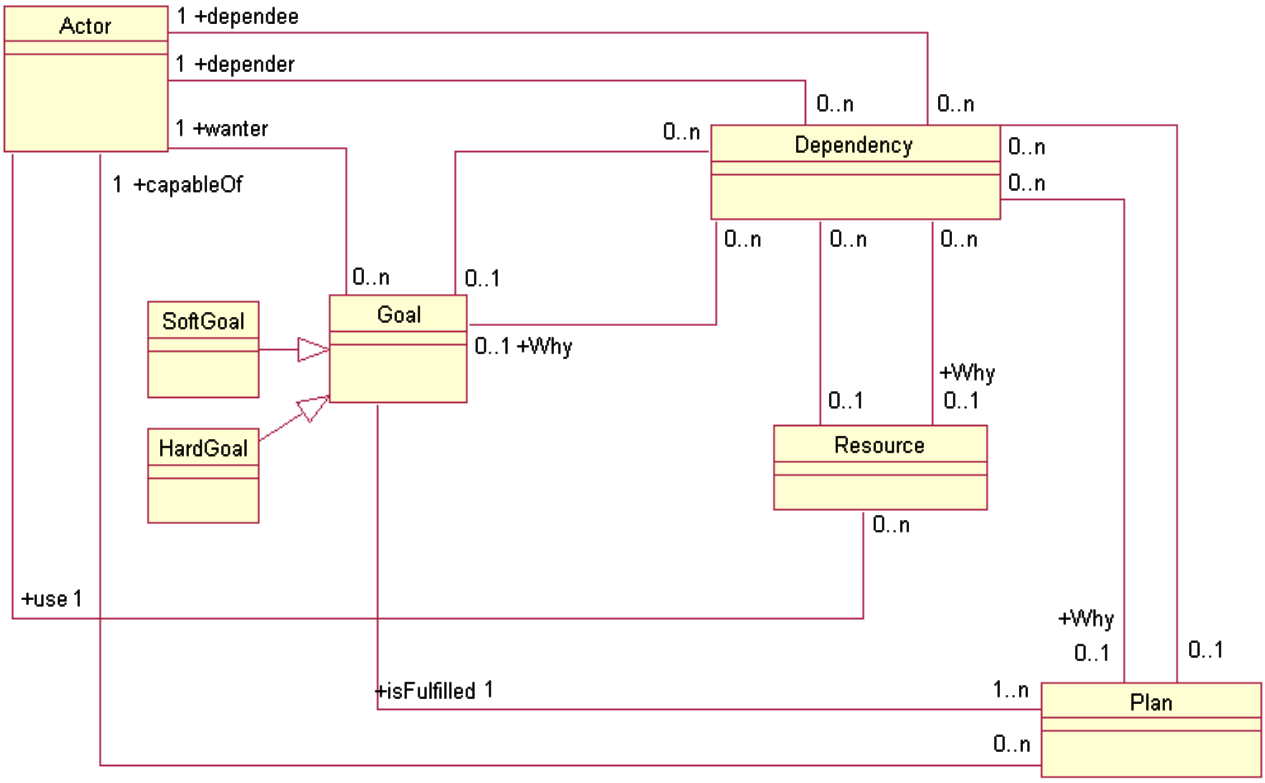

Figure 1: The UML class diagram specifying the actor concept in the Tropos metamodel.

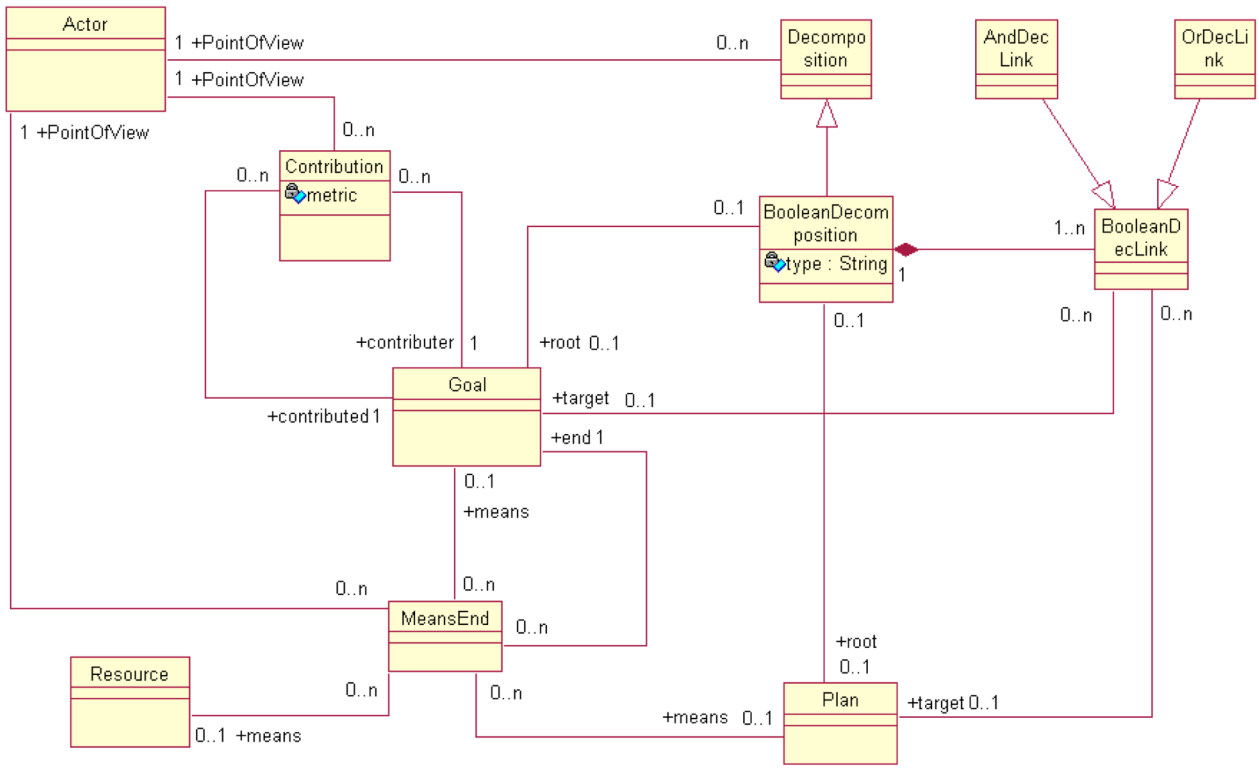

Figure 2: The UML class diagram specifying the goal and plan concepts in the Tropos metamodel.

The concept of Plan.
The concept of plan in Tropos is specified by the class diagram depicted in Figure 2. Means-end analysis and AND/OR decomposition, defined above for goals, can be applied to plans also. In particular, AND/OR decomposition allows for modeling the plan structure.

2.1 Discussion on the Tropos meta-model

In this section, we discuss the Tropos meta-model with respect to the aspects proposed by [Ber04]. According to this, four main aspects were identified: Agent Structure, Agent Interactions, Agent Society and Organisational Structure, and Agent Implementation.

- **Agent Structure**

  In comparison with the other meta-models analysed in [Ber04], Tropos adds a higher level of abstraction according to which, the concept of an actor is employed as a generalisation of an agent. Initially, during the early and late requirements analysis, actors are identified, which are then translated to possible agents during the architectural design. Moreover, Tropos defines the concept of a role as an abstract characterisation of the behaviour of a social actor and the concept of position to represent a set of roles. Therefore, an agent in Tropos can occupy a position, while s position is said to cover a role.
  A possible integration with the "unifying meta-model" presented in [Ber04] is illustrated in Figure 3. The classes with names in italic fonts refer to the "unifying meta-model" elements, while the others to the elements of the Tropos meta-model. In this way all the entities related to the concept of Actor could be included in the "unifying meta-model" provided that we map the Tropos concept of "plan" to the concept of "*task*" in the "unifying meta-model". In particular, notice that in this way we will introduce explicitly in the "unifying meta-model" the concept of actor's goal. Moreover, as also mentioned below, the relationship between the concept of Actor in Tropos and that of "*Organization*" in the "unifying meta-model" is to be analyzed in more details.

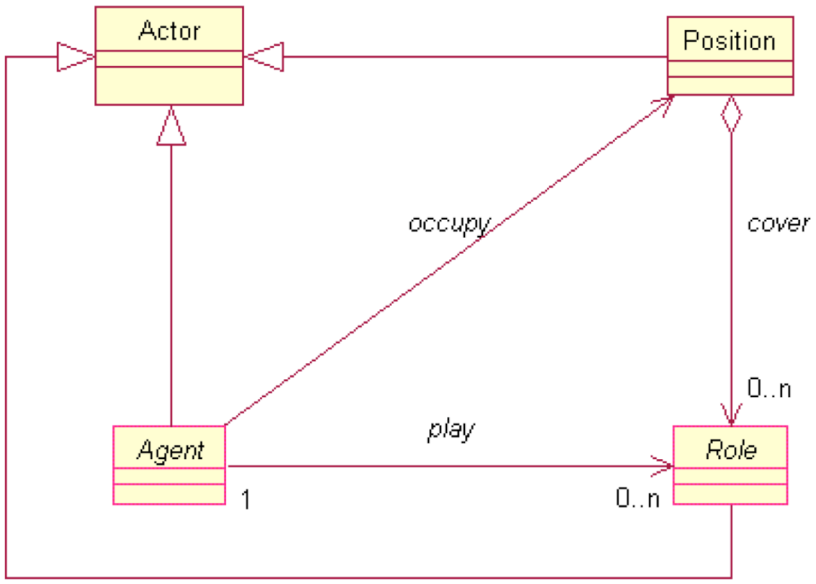

Figure 3: integrating Tropos Actor concept

Concerning entities, such as agent beliefs and capabilities, which are typically used to specify agent rationality, we think that a deeper understanding of how the "unifying meta-model" supports the modeling of these agent features is required before proposing an effective integration/extension.

The following aspects are related to the analysis process and do not necessarily require the introduction of new elements in the meta-model (we refer here to the hypothetic integration of the "unifying meta-model" with the Tropos meta-model proposed in the previous point).

- **Agent Interactions.**
  Initially, interactions are modelled in terms of social dependencies, which mainly denote interactions of an actor with its environment (other actors, agent, roles, positions and/or systems). In particular, an agent might depend on other agents in order to attain some goal, execute a plan, or deliver a resource. Later on the development process, dependencies are translated into structural patterns of interactions between the agents of the system. Such a structural pattern of interactions is captured in Tropos by emphasizing the chronological sequence of communications. For this reason, Tropos employs agent interaction protocols to describe the communication patterns among the agents as well as constraints on the content of the message they exchange. Moreover, agents are able to communicate with their environment in terms of events that describe a triggering condition for an agent's actions. The interaction protocol concept mentioned in [Ber04] could provide all the entities necessary to specify Tropos agent interaction protocols.

- **Agent Society and organisational structure**
  Organisational structures are in the very heart of the Tropos development process. During the early requirements analysis stage developers are concerned with the understanding of a problem by studying an existing organisational setting. This involves the identification of the domain stakeholders and their modelling as social actors. Therefore, an organisational model is produced in which the structure of the organisation is modelled in terms of actors, their intentions (goals) and their relationships (dependencies). From these considerations it could appear that the relationship between Actor and *Organization* could be modeled as the "*belong*s" relationship that is currently between the concept of *agent* and *organization* in Fig. 4 of [Ber04]. Moreover, dependencies describe an agreement (called dependum) between two actors, the depender (the depending actor) and the dependee (the actor who is depended upon). This organisational model forms the basis for the system's structure. For this, the Tropos ontology includes social patterns, such as mediator, broker and embassy [Kol02], as well as a set of organisational styles inspired by organisation theory, which allows defining precisely the structure of the system in terms of interconnecting agents.

- **Agent implementation**
  The implementation activity follows step-by-step the detailed design specification on the basis of the establish mapping between the implementation platform constructs and the detailed design notions. Although it is possible to use any agent platform during the implementation stage, the Jack Intelligent Agents platform provides a set of constructs, such as capabilities and plans, which match the concepts of Tropos. Agents are programmed with a set of plans in order to make them capable of achieving their goals. Agents are defined in terms of their capabilities, the types of events and messages it responds to, and the plans it uses to achieve its goals. In particular, a capability contains plans, events, beliefs and an agent can be assigned with one or more capabilities.

3. A meta-model implementation. The TAOM-Tool for Agent Oriented visual Modeling.

In order to support the use of the methodology we are developing an agent oriented modeling CASE tool (TAOM) that supports the analyst when building an informal specification using the Tropos methodology and a component that allows for its automatic transformation into a formal

specification which can be verified by a model-checker. The modeling environment supports the adoption of a framework, which rests on a light integration of informal and formal languages.
In developing this tool we are taking into account emerging guidelines and standards from the OMG' **Model-Driven Architecture** (MDA) [MDA] initiative, as discussed in [PER04], and in particular the **Meta Object Facility** (MOF) [MOF], and a set of requirements for the transformation techniques that will be applied when transforming a source model into a target model, this is referred as the **Query/View/Transformation** (QVT) [OMG] approach.
As described in the previous section, the Tropos metamodel has been according to the MOF (Meta Object Facilities) directives that allow to specify, build and manage technology neutral meta-models. For the model implementation we adopted the **Java Metadata Interface** (JMI) [JMI], which enables the implementation of a dynamic, platform-independent infrastructure to manage the creation, storage, access, discovery, and exchange of metadata. Finally, the persistence of the model has been assured via the representation of the model in **XMI** [XMI], the OMG standard for serializing model and meta-data in XML, that allow also the sharing of the model between the environment components.

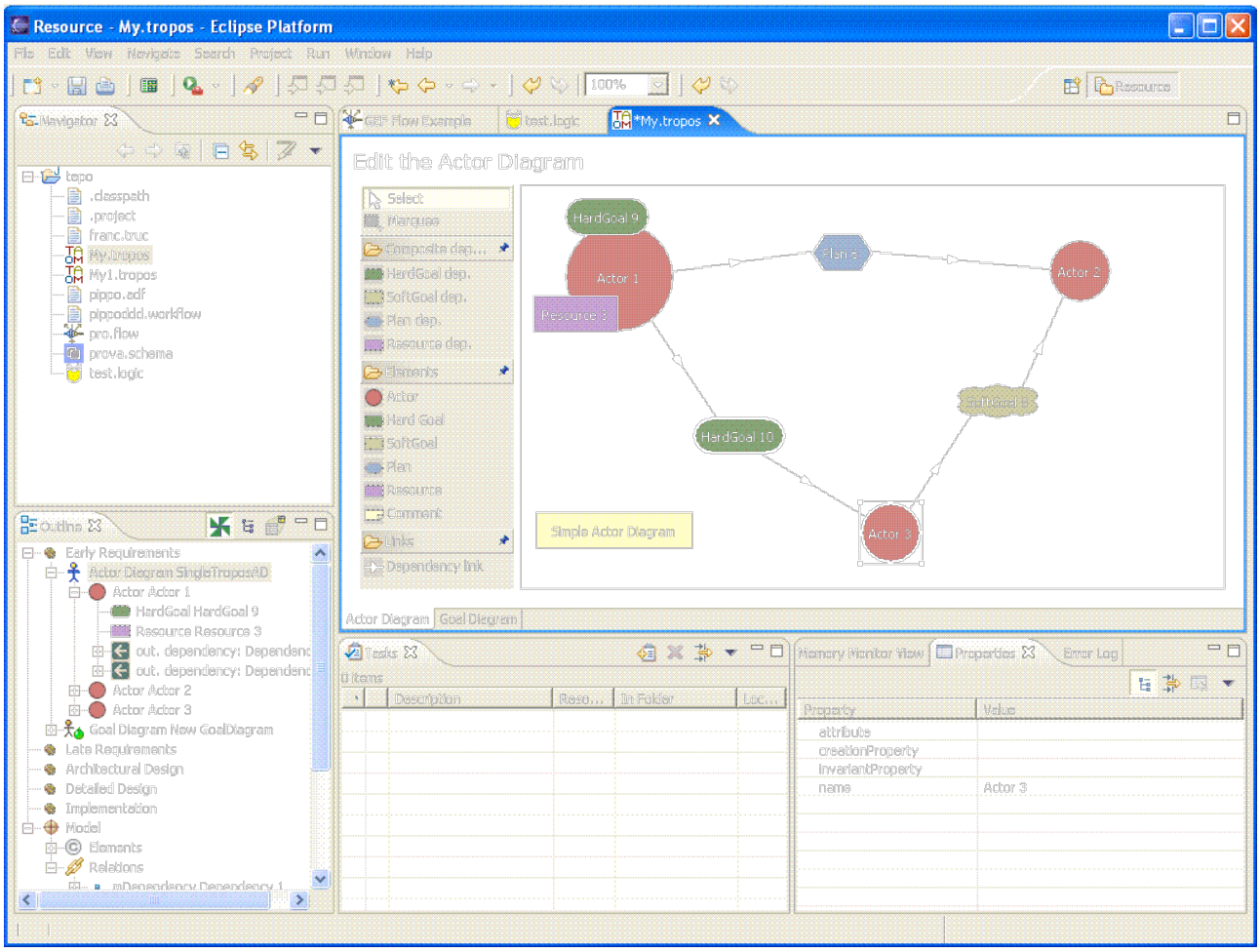

Figure 4: A snapshot of the TAOM4E modeling tool showing the editing of an actor diagram

We are currently porting and extending our tools into the ECLIPSE platform (**TAOM4E**) using the already stated standard and related technologies plugins available. The ***ECLIPSE*** Project [ECLIPSE] is an open source initiative that allows the integration of different tools into a single "application". Eclipse is a kind of universal tool platform - an open extensible IDE for anything and nothing in particular. New tools are integrated into the platform trough plug-ins that provides new functionalities to the environment.

Finally, it is important to note that the meta-model implemented in the tool has been designed to be flexible and extensible in order to allow easy support of variants and addictions to the language and the management of different project artefacts.

4. Conclusions

In this paper we have described the Tropos metamodel and we have discussed some of its aspects with reference to the "unifying meta-model" presented in [Ber04]. Moreover, we have briefly described the TAOM modelling tool, which represents an implementation of the Tropos metamodel. However, further worked is required. In particular, although entities such as agent belief and capabilities have been defined in Tropos, work is in progress to better refine them in the Tropos development process. As a result, such entities have not beed considered in the meta-model described in this paper. Moreover, important entities related to the description of trust and security in distributed systems have been added in Tropos and described in details in [Mou03]. Therefore, we are working on extending our meta-model in order to specify them.

References

[Ber04] C. Bernon, M. Cossentino, M-P. Gleizes, P. Turci, F. Zambonelli. A Study of some Multi-agent Meta-Models, in Proceedings of the 5th International Workshop on Agent Oriented Software Systems (AOSE'04), N.Y.-USA, July 2004.

[Bre04] P. Bresciani, P. Giorgini, F. Giunchiglia, J. Mylopoulos, and A. Perini. Tropos: An Agent-Oriented Software Development Methodology. *Journal of Autonomous Agent and Multi-Agent Systems*, 8(3):203 – 236, May 2004.

[Cas01] J. Castro, M. Kolp, and J. Mylopoulos. A Requirements-Driven Development Methodology. In Proceedings of the 13th International Conference on Advanced Information Systems Engineering (CAiSE'01), Interlanken, Switzerland, June 2001.

[Giu01] F. Giunchiglia, A. Perini, and F. Sannicolo'. "Knowledge Level Software Engineering". In Proceedings of ATAL 2001, Seattle, December 2001, Springer Verlag.

[Mou03] H. Mouratidis, P. Giorgini, and G. Manson, Modelling Secure Multiagent Systems, in Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multiagent Systems, Melbourne-Australia, 2003

[Kol02] M. Kolp, P. Giorgini, and J. Mylopoulos. Information Systems Development through Social Structures, In Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering (SEKE'02), Ishia, Italy, July 2002.

[Per03] A. Perini, A. Susi. "Developing a decisioon support system for integrated production in agriculture". In Environmental Modeling and Software, Journal 19(9), 2004.

[Per04] A.Perini and A. Susi. Agent-Oriented visual modeling and model validation for engineering distributed systems. MATES 04

[San01] F. Sannicolo', A. Perini, and F. Giunchiglia. The Tropos modeling language. A User Guide. Technical Report 0204-13, ITC-irst, December 2001.

[Yu95] Yu, E., 1995. Modelling strategic relationships for process reengineering. Ph.D. thesis, University of Toronto, Department of Computer Science, University of Toronto.

[MDA] http://www.omg.org/mda/

[MOF] http://www.omg.org/technology/documents/formal/mof.htm

[OMG] http://www.omg.org/

[JMI] http://java.sun.com/products/jmi/index.jsp

[XMI] http://www.omg.org/technology/documents/formal/xmi.htm

[ECLIPSE] http://www.eclipse.org/