Supporting Formal and Informal specification in Agent Oriented software development
Extended Abstract

AI*IA 2004 Workshop - Agenti e Vincoli: Modelli e Tecnologie per Dominare la Complessità

Anna Perini and Angelo Susi

ITC-irst, Via Sommarive, 18, I-38050 Trento-Povo, Italy
{perini,susi}@itc.it

A considerable effort in defining Agent-Oriented (AO) approaches to engineering distributed systems is going on based on the recognition that the agent paradigm, beside providing a useful technology to build software systems with an open architecture, offers appropriate abstractions for specifying and designing critical properties of these system, such as the dynamic evolution of their architecture and the interaction protocols of system components [5, 8, 9, 13].

Most of the proposed software engineering methodologies adopt visual modeling as a core process which drives the whole software development, from requirement analysis to implementation. The conceptual languages they use provide an effective graphical notation, but often lack a formal definition of their semantic [1]. This can result to subjective models which can hardly be refined in a straightforward way into a system design. Formal specification languages can solve some of the weaknesses of visual modeling languages, specifically, they permit to define models with a precise semantics, and facilitate their transformation into system designs. However, writing a formal specification usually require strong skills, and it is often ineffective for discussing with the stakeholders. Moreover, the formalization “a posteriori” of visual models expressed according to a conceptual modeling framework is not an easy task at all, due to the ambiguities in the meaning of the graphical notations.

In a previous work [12] we proposed a framework which rests on a light integration of informal and formal languages, adopting *Tropos* [3, 11], an Agent Oriented software development methodology which provides a conceptual modeling language that can be used to build both an informal specification or a formal one [6]. From a practical point of view, the methodology guides the software engineer in building a conceptual model that is incrementally refined and extended from an early requirements model, namely a representation of the organizational setting where the system-to-be will be introduced, to system design artifacts, according to a requirements-driven approach.

The *Tropos* modeling language allows to represent intentional and social concepts, such as actor and goal, and a set of relationships between them, such as actor dependency, goal decomposition, means-end and contribution relationships. A diagrammatic

[1] Some methodologies rest on UML, extending it as AUML [1], or defining appropriate profiles (e.g. PASSI, Process for Agent Societies Specification and Implementation [2]), and therefore suffer from the limit of UML of being a semiformal specific language.

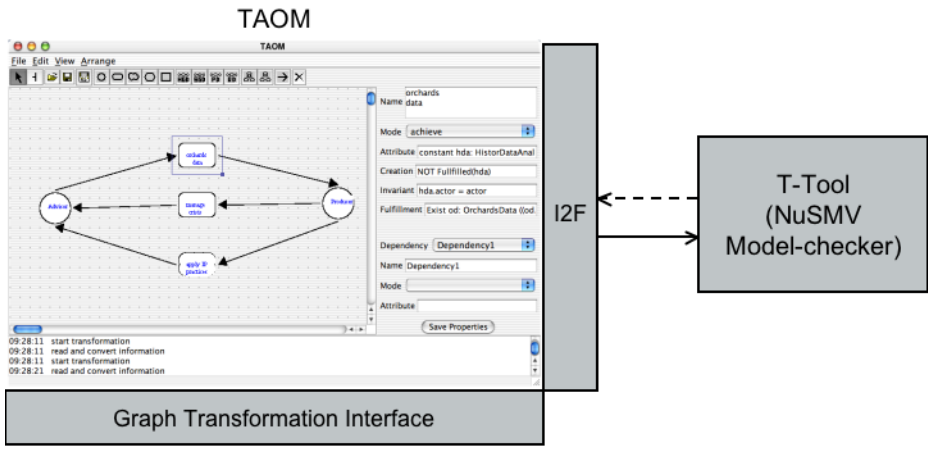
notation, derived from the *i** framework [14] allows to build views on the actor dependencies or on goal analysis conducted from the point of view of a specific actor. The language ontology has been given in terms of common sense (informal) definitions and a subset of it (called *FT*, Formal *Tropos*) is defined in terms of a typed first-order Linear Temporal Logic (LTL). *FT* specifications can be automatically verified by model-checking techniques.

Using *FT*, means extending a *Tropos* specification with annotations that characterize the valid behaviors of the model and are represented as temporal constraints on the evolutions of model instances. That is, an *FT* specification consists of a sequence of entity declarations such as actors, goals, and dependencies. Each declaration associates a set of attributes to the entity on which temporal constraints are given, they are expressed in LTL.

The possibility to transform an informal *Tropos* specification into an *FT* specification allows to exploit automatic verification techniques to support the analyst during model refinement. For instance, focusing on early stages activities in software development, an analyst performs tasks such as: annotating the *Tropos* visual model with properties that can be represented in *FT* (*i*); performing the assessment of the model against possible inadequacies, incompleteness and inconsistencies (*ii*); validating the resulting specification with the domain experts or the stakeholders in order to end up with an agreed set of requirements (*iii*); managing the model refinement and evolution steps (*iv*). In particular, during activities (*ii*—*iii*), the analyst can query a model-checker relatively to specific aspects of the model.

We are developing an AO development environment at support of modeling by interleaving formal and informal specification, as described above, taking into account suggestions and standards which are proposed by the Model-Driven Architecture (MDA) initiative by OMG [4, 7]. In particular, the MDA approach gives a standard to which the meta-models of the specification languages used in the modeling process must be compliant with, it is called Meta Object Facility (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) approach.

Our development environment includes a tool that supports visual modeling languages whose meta-model is MOF compliant. We adapted the *Tropos* language meta-model given in [3]. Figure 1, gives an overview of the modeling environment structure. The modeling tool, named TAOM, is currently able to represent the basic entities defined in this meta-model like actor, goal, plan, resource and the relationships between them like the dependency, the and-or decomposition, the means-end and the contribution. TAOM allows to represent new entities that could be included in the *Tropos* meta-model or in language variants, as well as to restrict the set of representable entities to a subset in an easy way. Moreover, a modular design has been adopted so that it is possible to add new components, a key requirement being to support the translations of the basic *Tropos* specification into other target specifications languages in order to exploit services like automatic verification. The prototype integrates the T-TOOL, a model-checker for the verification of formal properties via a software component which uses a visitor pattern to implement the transformation from the informal *Tropos* model to the *FT* one, it is labeled as I2F in Figure 1. A model developed with TAOM is saved as an XMI file con-



Name: orchards data
Mode: achieve
Attribute: constant hda: HistorDataAnal
Creation: NOT Fullfilled(hda)
Invariant: hda.actor = actor
Fulfillment: Exist od: OrchardsData ((od.
Dependency: Dependency1
Name: Dependency1
Mode:
Attribute:

```
09:28:11  start transformation
09:28:11  read and convert information
09:28:11  start transformation
09:28:21  read and convert information
```

Fig. 1. The modeling environment structure: the Agent-Oriented (AO) modeling tool (TAOM) is connected to other tools, such as a model-checker for the verification of formal properties of the model (T-TOOL) through the I2F module. The *Graph Transformation interface* component integrates a graph transformation techniques library.

taining the specification of the properties of the entities of the model and including the entity annotations which can be represented in LTL. The model's XMI file is given in input to the transformation module which produces the correspondent *FT* specification and gives it in input to the T-TOOL.

A third tool which has been integrated with the modeler is AGG, a library which implements graph transformation techniques that can be used to support model refinement.

The current version of the prototype implements a core subset of the requirements that have been identified. Work is in progress to implement further requirements such as the management of other software development artifacts and of multiple views of the model, the support of the process phases proposed in the *Tropos* methodology. Moreover, expected updates of the MDA standards relevant for expressing model transformations (e. g. MOF 2.0 Query/Views/Transformations) will be considered. Graph transformation techniques have been already pointed out as a promising technology to provide mechanisms for the automatic synchronization of different views in a model or for translating a model given in a specification language to a different one. Along this line we are pursuing a parallel research [10] and strengthening the integration of the AGG system in the environment.

References

1. http://www.auml.org, 2004.
2. Passi documentation, 2003. http://www.csai.unipa.it/passi.

3. 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.
4. Alan Brown. An introduction to Model Driven Architecture Part I: MDA and todays systems. *The Rational Edge*, January 2004.
5. P. Ciancarini and M. Wooldridge, editors. *Agent-Oriented Software Engineering*, volume 1957 of *Lecture Notes in AI*. Springer-Verlag, March 2001.
6. A. Fuxman, M. Pistore, J. Mylopoulos, and P. Traverso. Model checking early requirements specifications in Tropos. In *IEEE Int. Symposium on Requirements Engineering*, pages 174–181, Toronto (CA), August 2001. IEEE Computer Society.
7. T. Gardner, C. Griffin, J. Koehler, and R. Hauser. A review of omg mof 2.0 query / views / transformations submissions and recommendations towards the final standard. In *Meta-Modelling for MDA Workshop*, York, England, 2003. Also available as a position paper at OMG.
8. Paolo Giorgini, Jörg P. Müller, and James Odell, editors. *Agent-Oriented Software Engineering IV(AOSE 2003). 4th International Workshop AOSE03, Melbourne, Australia - July 2003*, volume 2935 of *LNCS*. Springer-Verlag, 2003.
9. F. Giunchiglia, J. Odell, and G. Weiß, editors. *Agent-Oriented Software Engineering III*. LNCS. Springer-Verlag, Bologna, Italy, Third International Workshop, AOSE2002 edition, July 2002.
10. Aliaksei Novikau, Anna Perini, and Marco Pistore. Graph Rewriting for Agent Oriented Visual Modeling. In *Proc. of the International Workshop on Graph Transformation and Visual Modeling Techniques, in ETAPS 2004 Conference*, Barcelona, Spain, 2004.
11. A. Perini, P. Bresciani, F. Giunchiglia, P. Giorgini, and J. Mylopoulos. A Knowledge Level Software Engineering Methodology for Agent Oriented Programming. In *Proceedings of Agents 2001*, Montreal CA, May 2001. ACM.
12. A. Perini, M. Pistore, M. Roveri, and A.Susi. Agent-oriented modeling by interleaving formal and informal specification. In *Agent-Oriented Software Engineering IV(AOSE 2003). 4th International Workshop AOSE03, Melbourne, Australia - July 2003*, LNCS 2935, pages 36–52. Springer-Verlag, 2004.
13. M.J. Wooldridge, G. Weiß, and P. Ciancarini, editors. *Agent-Oriented Software Engineering II*. LNCS 2222. Springer-Verlag, Montreal, Canada, Second International Workshop, AOSE2001 edition, May 2001.
14. E. Yu. *Modelling Strategic Relationships for Process Reengineering*. PhD thesis, University of Toronto, Department of Computer Science, University of Toronto, 1995.