PUC

ISSN 0103-9741

Monografias em Ciência da Computação
n° 09/05

A Model Driven Approach to Develop Multi-Agent Systems

Anarosa Alves Franco Brandão
Viviane Torres da Silva
Carlos José Pereira de Lucena

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO
RUA MARQUÊS DE SÃO VICENTE, 225 - CEP 22453-900
RIO DE JANEIRO - BRASIL

Monografias em Ciência da Computação, No. 09/05 ISSN: 0103-9741
Editor: Prof. Carlos José Pereira de Lucena March, 2005

A Model Driven Approach to Develop Multi-Agent Systems*

Anarosa Alves Franco Brandão Viviane Torres da Silva
Carlos José Pereira de Lucena

{anarosa, viviane, lucena}@inf.puc-rio.br

Abstract: The Object Management Group's (OMG) Model Driven Architecture (MDA) initiative has been the focus of research in academia and industry because it raises a fast and consistent software development through the use of software models. In this context we propose a model driven approach to the development of multi-agent systems beginning with an ontology for the multi-agent systems (MAS) domain. The approach aims to combine these two emerging research areas in order to overcome the complexity associated with the development of MAS.

Keywords: Model Driven Development, Ontology, Multi-Agent Systems.

Resumo: A pesquisa em arquiteturas dirigidas a modelos (MDA) tem se mostrado um campo fértil tanto na academia quanto na indústria. Isto se deve principalmente ao fato de que o uso de MDA para desenvolvimento de sistemas promete rapidez e consistência das aplicações resultantes. Neste contexto, este artigo propõe uma abordagem dirigida a modelos para o desenvolvimento de sistemas multi-agentes (SMAs), que parte da definição de uma ontologia para o domínio de SMAs e utiliza suas instâncias para iniciar a geração de modelos. A abordagem proposta tem por finalidade amenizar a complexidade associada ao desenvolvimento de SMAs.

Palavras-chave: Desenvolvimento Dirigido a Modelos, Ontologia, Sistemas Multi-Agentes.

---

* This work has been sponsored by the Ministério de Ciência e Tecnologia da Presidência da República Federativa do Brasil)

In charge for publications:

Rosane Teles Lins Castilho
Assessoria de Biblioteca, Documentação e Informação
PUC-Rio Departamento de Informática
Rua Marquês de São Vicente, 225 - Gávea
22453-900 Rio de Janeiro RJ Brasil
Tel. +55 21 3114-1516 Fax: +55 21 3114-1530
E-mail: bib-di@inf.puc-rio.br

1 Introduction

In recent years we could see the emergence and growth of the World Wide Web and its associated technologies. In fact, the distribution of information and associated technologies show that the use of open and distributed architectures is becoming a new reality for the development of software systems [31]. As the complexity associated with these systems is increasing at a quickening pace, it is no surprise that agent-orientation raises a new paradigm of software engineering. Nevertheless, to deal with the complexity associated with the development of agent systems, the research community has developed new methodologies based on agent concepts. In this context, there are results directed to the requirements phase, such as TROPOS [5], the creation of conceptual frameworks such as TAO [31], modeling languages, methods and methodologies, such as MAS-ML[32], AUML [1], Gaia[41], MaSE [40], AORML [38], architectural patterns such as FIPA [11] and RETSINA [33], and infrastructure such as JADE [2,3], RETSINA [33], DECAF [13], ZEUS [21] etc.

Moreover, as the major results in agent-oriented software engineering research are addressed to modeling activities and infrastructure, the basic tools needed for the use of a model driven approach (models and platforms/infrastructure) are already available. The Object Management Group (OMG) [24] initiative in model driven development, called model driven architecture (MDA) [18], has been the focus of research in academia and industry because it promises to be a fast and consistent process of software development through the use of software models. The promise of MDA consists in the definition of machinery readable application and data models that allow long term flexibility of implementation, integration maintenance, testing and simulation [19]. With this promise, the use of an MDA approach suggests the coexistence of a diversity of systems that are not based on the same operating system, programming language, and architecture but that agree on models that can be translated one into another.

Therefore, the use of a model driven approach to develop MAS is proposed in order to surround the natural complexity associated with the development of MAS. This surrounding must be done by using models to treat and encapsulate the complexity and transformations between models whose outputs are other models in a lower level of abstraction.

Furthermore, by adopting the ontology defined in TAO (Taming Agents and Objects) conceptual framework [31], we propose a model driven approach to the development of MAS. Our approach seeks to provide a (semi)automatic transformation from a domain model into a platform independent model described in MAS-ML (Multi-Agent System Modeling Language) [32]. We also suggest the use of the JADE [3,2] as the platform to specify our example of a platform specific model.

This work describes the approach proposal and outlines part of a development example. It is organized as follows: Section 2 provides an overview of the model driven approach and sketches our proposal, Sections 3, 4, and 5 outline the process development phases by levels, from CIM to PIM and finally to PSM, respectively. Section 6 describes some related work and Section 7 presents our conclusions and future work.

2 Model Driven Approach Overview

The MDA is a model driven framework for software development proposed by the OMG. Its development process is directed by modeling activities that allow the specification of the whole system in a high level of abstraction by using models that are platform independent and that can be mapped into models addressed to different platforms [12].

There are three basic models in an MDA development process: a computational independent model (CIM), a platform independent model (PIM) and a platform specific model (PSM). A CIM is a model that describes the application domain and has no details about the structure and processing of the system. The CIM models are analysis or conceptual models of applications. A conceptual model is derived from reality for the purpose of gaining a better understanding of such reality [9]. A PIM is a model that presents a specific degree of platform independence in order to be suitable for use with a number of different platforms of similar type [19]. According to the MDA guide [19], a platform is a set of subsystems/technologies that provides a coherent functionality set through interfaces and specified usage patterns. This makes it possible for any subsystem that depends on the platform to use it without being concerned about the details of how the functionality provided is implemented. Therefore, the degree of platform independence is a relative measure. The PIM models are application design models. A PSM is a model that instantiates a PIM, providing it with details that are specific to a particular type of platform.

At present, a substantial portion of the research effort in this field addresses PIM to PSM transformations. Model transformations are the key to model driven development, since the models are simply inputs and outputs of them. Model transformation is the process of converting one model to another model of the same system [19]. High-level models are transformed into low-level models. For instance, the idea of generating one model from another in an automatic/semi-automatic manner intends to provide a simple and fast way of software development.

Our approach describes the outlines of a multi-agent system development process that follows the MDA approach (Fig. 1). We propose the use of the TAO ontology to describe the application in the CIM level, the use of MAS-ML to model the application in the PIM level and the use of the JADE platform to implement the application in the PSM level. In this context, the application domain model plays a key role, since it guides the (automated/semi-automated) development. Our approach will be described in later sections

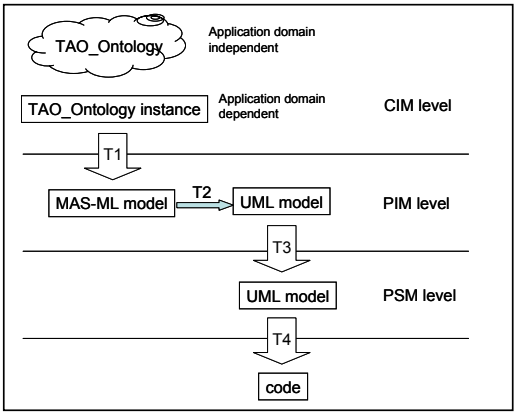

Fig. 1 The approach phases

3 CIM level

At the CIM level, application domains are described independently of computation models. Such domains can be described using ontologies. Apart from the philosophical concept of the term, an ontology can be thought of as being a set of basic terms and relations that describe a specific knowledge domain and that can be shared among people/machines to allow effective communication within that domain. In this paper we propose the use of the TAO_Ontology to describe MASs. The TAO_Ontology is defined based on the TAO conceptual framework. In order to better describe the CIM level, we begin with the description of the TAO conceptual framework and then we outline the transformation that will be used to generate the first PIM of our approach.

The goal of the TAO (Taming Agents and Objects) conceptual framework [31] is to define a core set of MAS abstractions. The core set of abstractions used in TAO has been developed based upon our investigation of existing agent-based and object-oriented methodologies [6,10,17,37,38,41,43], languages [29,39,15] and theories [30, 7, 27]. TAO groups together the abstractions that are frequently described in the literature for MASs and it defines the structural and dynamic aspects of MAS. While describing the structural aspects of MAS, TAO defines the entities that may be described in MASs, their properties and the relationships associated with them. While describing the dynamic aspects of MASs, TAO defines the creation and destruction of entities and also defines other domain-independent behavior.

Considering the aforementioned and the entity-relationship characteristic of TAO, it is natural to define an ontology based on TAO abstractions and relationships in order to describe the MAS domain. Thus, this ontology can be instantiated to generate conceptual models for MAS for specific application domains. Following this idea we define the TAO_Ontology using OWL, a Web ontology language [25]. We chose OWL as the ontology description language due to the objectives of the request for proposal from OMG for the definition of an Ontology Definition Metamodel that could be mapped to the UML Ontology Profile [22]. Fig. 2 shows a fragment of the TAO_Ontology, describing the Agent concept.

Instances of TAO_Ontology are built using ontology editors, such as Protégé 2000 [28] and Oiled [23]. The instances work as CIMs in a model driven approach to the development of MAS for specific application domains. An ontology instance is composed of the ontology plus the individuals (instances of the ontology concepts and relationships between them).

Fig. 3 shows a fragment of an instance of the TAO_Ontology in an MAS example. The MAS domain modeled using the ontology is the submission and revision of scientific papers problem domain. The example shows a system where authors can submit papers, reviewers can download papers to evaluate and post evaluation results, and the conference chair can organize review activities, such as distributing papers to reviewers, the sending of deadline alert messages, etc. Fig. 3 illustrates the use of the TAO_Ontology to describe an agent called UserAgent.

A CIM is a conceptual/analysis model, described using the TAO_Ontology that identifies the MAS entities and that can also list their properties and their relationships. However, it does not specify the interdependency among properties, the internal execution of the agents and their interactions, which are details depicted in the design models. A CIM model needs to be refined in order to better model an application. Therefore, CIM conceptual/analysis models are refined into PIM design models, described in the PIM level. We propose to represent the PIM design models using an

MAS modeling language where the MAS entities, relationships, properties, internal execution and interactions are defined.

```
<owl:Class rdf:ID="Agent">
    <owl:disjointWith rdf:resource="#Object"/>
    <owl:disjointWith rdf:resource="#Event"/>
    <owl:disjointWith>
      <owl:Class rdf:about="#Environment"/>
    </owl:disjointWith>
    <owl:disjointWith rdf:resource="#Role"/>
    <owl:disjointWith>
      <owl:Class rdf:about="#Organization"/>
    </owl:disjointWith>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:allValuesFrom rdf:resource="#AgentRole"/>
        <owl:onProperty>
          <owl:ObjectProperty rdf:about="#plays"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:about="#isSpecializationOf"/>
        </owl:onProperty>
        <owl:someValuesFrom rdf:resource="#Agent"/>
      </owl:Restriction>
    </rdfs:subClassOf>
     <rdfs:subClassOf>
      <owl:Restriction>
        <owl:someValuesFrom rdf:resource="#Agent"/>
        <owl:onProperty>
          <owl:ObjectProperty rdf:about="#isAggregatedTo"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:about="#isAssociatedWith"/>
        </owl:onProperty>
        <owl:someValuesFrom rdf:resource="#Object"/>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
```

Fig. 1 Agent description in TAO_Ontology

To define the first transformation (T1) from CIM to PIM, we use the TXL [35] programming language. TXL is a language specifically designed for source transformation tasks. TXL is known as a good language for designing recovery and architecture extraction from source [26]. Basically, TXL transforms an input text, described according to a specified grammar, to an output text, generated according to the grammar using a set of transformation rules.

T1 transforms instances of TAO_Ontology into modeling diagrams using an MAS modeling language such as MAS-ML and AUML. The TAO_Ontology grammar is the basis for the transformations from CIM into PIM. In fact, the TAO_Ontology grammar is used to allow the generation of the parse tree that will be the input for the T1 transformation. Fig. 4 shows a fragment of the TAO_Ontology grammar described using TXL. To complete T1, transformation rules are defined depending on the choice of the modeling language in order to transform instances of the TAO_Ontology into modeling diagrams.

To define the first transformation (T1) from CIM to PIM, we use the TXL [35] programming language. TXL is a language specifically designed for source transformation tasks. TXL is known as a good language for designing recovery and architecture extraction from source [26]. Basically, TXL transforms an input text, described according to a specified grammar, to an output text, generated according to the grammar using a set of transformation rules.

```
<Agent rdf:ID="UserAgent">
    <inhabits rdf:resource="#ExpertCommitteeEnv"/>
 <hasAction>to_editForm</hasAction>
    <hasAction>to_evaluatePaper</hasAction>
    <hasAction>to_putPaper</hasAction>
    <hasAction>to_rejectPaper</hasAction>
    <hasAction>to_joinPC</hasAction>
    <hasAction>to_getPaper</hasAction>
   <hasAction>to_acceptPaper</hasAction>
  …
 <hasPlan>Submit_a_paper</hasPlan>
 <hasPlan>Organize_a_conference</hasPlan>
    <hasPlan>Be_part_of_PC</hasPlan>
    <hasPlan>Review_a_paper</hasPlan>
…
 <hasGoal>review</hasGoal>
 <hasGoal>organize</hasGoal>
    <hasGoal>submit</hasGoal>
  …
    <plays rdf:resource="#PCMember"/>
    <plays rdf:resource="#Author"/>
    <plays rdf:resource="#Reviewer"/>
    <plays rdf:resource="#PCChair"/>
  …
    <isAssociatedWith rdf:resource="#Paper"/>
    <isAssociatedWith rdf:resource="#PageHTML"/>
    <isAssociatedWith>
      <Object rdf:ID="Proceedings">
        <inhabits rdf:resource="#ExpertCommitteeEnv"/>
      </Object>
    </isAssociatedWith>
  </Agent>
```

Fig. 3 – A fragment of an instance of TAO_Ontology – UserAgent

T1 transforms instances of TAO_Ontology into modeling diagrams using an MAS modeling language such as MAS-ML and AUML. The TAO_Ontology grammar is the basis for the transformations from CIM into PIM. In fact, the TAO_Ontology grammar is used to allow the generation of the parse tree that will be the input for the T1 transformation. Fig. 4 shows a fragment of the TAO_Ontology grammar described using TXL. To complete T1, transformation rules are defined depending on the choice of the modeling language in order to transform instances of the TAO_Ontology into modeling diagrams.

```
...
define bAgenttag
   '< 'Agent [spaces] rdf':'ID'= '"[name] '" [spaces] '>
end define

define eAgenttag
   '</ 'Agent [spaces] '>
end define

define Agent
   [bAgenttag]
        [AgentContent]
   [eAgenttag]
end Agent

define AgentContent
   [spaces] [repeat hasGoal] [NL]
   [spaces] [repeat hasBelief] [NL]
   [spaces] [repeat hasPlan] [NL]
   [spaces] [repeat hasAction] [NL]
   [spaces] [list AgentRelationship]
end define

define AgentRelationship
     [inhabits]
   | [plays]
   | [isAssociatedWith]
   | [isSpecializationOf]
   | [isAggregatedTo]
end define
...
```

Fig. 2 - Fragment of TAO_Ontology grammar

4 PIM level

When transforming a CIM model into a PIM model, several pieces of information can be added to the PIM model in order to describe application characteristics that were not described in the CIM model. For instance, a PIM model should model the relation between the properties of an entity and the interactions between MAS entities. Such interactions and relationships are not described in the CIM model, since it does not describe details about the structure and processing of the system.

We propose to use MAS-ML to represent PIM models. CIM models described in TAO_Ontology (T1 input) are transformed into PIM models represented using MAS-ML (T1 output). Since MAS-ML is also based on the TAO conceptual framework, T1 uses direct transformation rules to generate the output parse tree described using the MAS-ML grammar.

MAS-ML [32] is a modeling language for multi-agent systems that extends UML. The MAS-ML meta-model is defined by extending the UML meta-model according to the MAS concepts defined in the TAO conceptual framework. The MAS-ML goal is to model all the structural and dynamic aspects defined in TAO. MAS-ML defines structural and dynamic diagrams to represent all TAO aspects. The structural diagrams defined by MAS-ML are the class, the organization and the role diagrams. It is possible to model the structural aspects of all the entities defined in TAO using these three structural diagrams.

The dynamic diagram defined by MAS-ML is an extended UML sequence diagram. The UML sequence diagram was extended to model the interaction between the MAS entities, to model their internal execution and to model agent interaction protocols. Fig. 5 partially illustrates the MAS-ML grammar using TXL. It depicts the part of the grammar that defines an agent, the set of properties associated with agents and the interdependency among these properties. Fig. 6 describes an agent using the grammar presented in Fig. 5. The output of T1 is MAS-ML models textually described using the MAS-ML grammar.

```
define agent
   'AGENT [NL] '(
         [agentID][IN][NL]
         [goal*][NL]
         [belief*][NL]
         [plan*][NL]
         [actionPrePostCondition*][NL][EX]
   ')
   |[entityClass*]
%an agent class is transformed into an OO class
end define
define goal      % Goal is an attribute
     [goalLeaf]
   | [goalComposite]
end define
define goalLeaf
   'GOAL '( [attrBody] [goalPlans+] ')
   ...
end define
define goalComposite
   'GOAL '( [attrBody] [goalPlans*] 'SUBGOAL '( [goal+] ')')
   ...
end define
define goalPlans
   'RELATED 'TO 'PLAN [stringlit]
   ...
end define
define belief    % Beleif is an attribute
   'BELIEF '( [attrBody] ')
   ...
end define
define attrBody
   [varType] ': [varName] '= [varValue]
end define
define plan
   'PLAN '( [planName] [planActions*] [planGoal]')
   |[entityClass*]
   ...
end define
define planActions
```

```
   'COMPOSED 'OF 'ACTION [stringlit]
   ...
end define
define planGoal
   'RELATED 'TO 'GOAL [varName]
   ...
end define
define actionPrePostCondition
   [action][NL][preCondition*][postCondition*]
   |[entityClass*]
end define
define action
   'ACTION [actionAssignment]
   ...
end define
define preCondition
   'PRECONDITION '( [attrBody] ')
   ...
end define
define postCondition
   'POSTCONDITION '( [attrBody] ')
   ...
end define
```

Fig. 3 - MAS-ML grammar describing an Agent

```
AGENT
( User_Agent
         GOAL ( "boolean" :  "review" = "true")
         ...
         BELIEF ("Paper" : "paper" = null)
         ...
         PLAN ( Review_a_paper
                         COMPOSED OF ACTION "to_getPaper"
                         COMPOSED OF ACTION "to_evaluatePaper"
                         COMPOSED OF ACTION "to_acceptPaper"
                         COMPOSED OF ACTION "to_rejectPaper"
                         RELATED TO GOAL "review"
         )
         ...
         ACTION to_getPaper
                  PRECONDITION ( "Paper" : "paper" = null,
                                 "review" = "false")
                  POSTCONDITION ( "Paper" : "paper" = paper_value)
         ACTION to_evaluatePaper
         ...
)
```

Fig. 4 - Fragment of example using MAS-ML

PIM models are transformed into PSM models by making use of a specific agent platform. Agent platforms published in the literature, such as JADE and RETSINA, are implemented using object-oriented languages since no agent programming language adopted by agent researchers exists. Therefore, to transform the output of T1, described in an MAS modeling language, into code using an agent platform, implemented with an object-oriented programming language, it is important to refine the output of T1 by transforming it into UML. The refinement is accomplished by using another transformation, called T2.

The transformation T2 receives as input a text file that represents the design model in MAS-ML and generates the corresponding parse tree using the MAS-ML grammar. Transformation rules are defined in order to transform the parse tree based on the MAS-ML grammar into a UML model. The output of T2 is an XMI [42] file describing UML models. XMI is an extended markup language for interchanging UML diagrams. By using XMI, it is possible to describe the information available in any UML diagram.

The transformation rules define an abstract architecture that sketches the output of T2. Fig. 7 shows the abstract architecture for the element Agent. The T2 output is an object-oriented model that can be considered platform independent and that can be translated to different platforms, such as JADE and RETSINA.

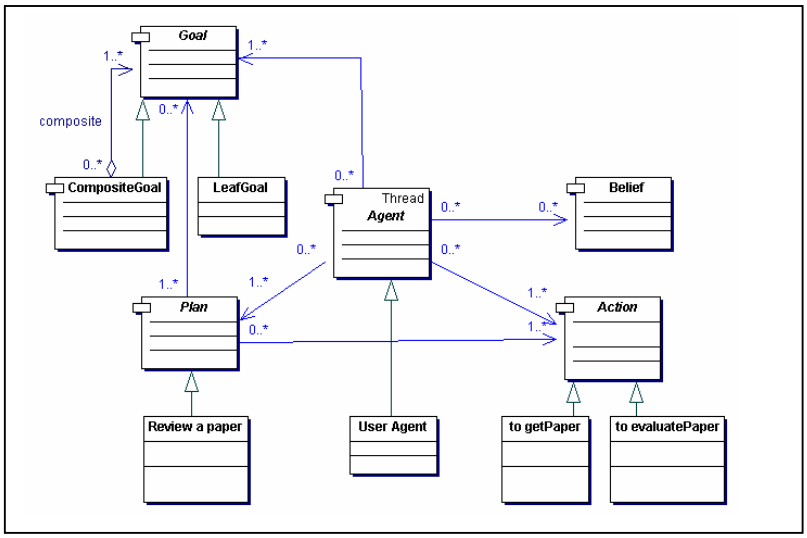

Fig. 5 - PIM abstract architecture plus UserAgent definition classes

5 PSM level

As stated before, the T2 output is an object-oriented model that describes a PIM and that can be translated to different platforms. Thus, having chosen a specific platform type, the task of transformation T3 is to combine the output of T2 with details that specify how the application must make use of a specific type of platform. For instance, in this work we have decided to use the JADE platform for T3 definition.

JADE (Java DEvelopment platform) is a middleware for the development of distributed multi-agent applications based on peer-to-peer communication architecture [2]. It provides standard agent technologies to developers, since its goal is simplifying the development process of MASs. These technologies include services such as a runtime environment, represented by containers (main container and "normal" containers). The main container provides a naming service through the Agent Management System (AMS) and a yellow pages service through the Directory Facilitator (DF) [2, 3, 20]. Communication among agents is provided by the Agent Communication Channel [20], which states that communication is asynchronous and can be initiated for every JADE agent running in the JADE runtime environment. Moreover, JADE defines a library of classes that must be used to develop JADE agents and to specify the agent tasks (Behaviour class). The Behaviour class is the root class of a hierarchy whose children are SimpleBehaviour class and CompositeBehaviour class.

The T3 transformation must map the output from T2 to a JADE model. For instance, the agent, plans and actions must be translated to the Behaviour class, since the CompositeBehaviour class and the SimpleBehaviour class naturally represent the agent plans and actions, respectively. The goals and beliefs are mapped to classes as proposed in [4].

6 Related Work

Cunha and colleagues [8] propose the use of ontologies as computation independent models. They suggest that (application domain) ontologies are very similar to CIMs

and propose some questions. Our approach defines a domain ontology for the multi-agent systems domain and proposes the use of instances of this ontology, which are application domain models, as a CIM.

Thiefaine and colleagues [34] introduced a multi-agent system development approach based on MDA. Their work presents a high level of dependency on the agent platform DIMA and this characteristic prevents all the models from being platform independent, even those that are already considered to be platform independent. As our proposal is based on a conceptual framework for agents and objects, it does not present these characteristics, letting the developer choose from different agent platforms.

Kazakov and colleagues [14] developed a model driven approach for the design of highly distributed mobile agent systems. It is a methodology for a very specific application domain, since it is addressed to support concurrent engineering processes using the mobile agent paradigm. Our approach is not addressed to a specific MAS application domain.

Kulesza et al [16] developed a generative approach to agent architectures using aspect-oriented technologies. Although their work is addressed to automatic generation of models, they are concerned about agency properties such as autonomy, adaptation, interaction, etc, which is not the focus of our work.

7 Conclusions and Future Work

In this work, we describe a model driven approach to develop multi-agent systems that begins with an ontology based on the TAO conceptual framework. The combination of the model driven approach and multi-agent systems seems to be promising because agent-oriented software engineering is a novel research and development area and this novelty brings with it an increasing amount of associated (new) technologies that may affect the way MAS implementations are carried out.

This paper presented an end-to-end development of an agent-oriented system based on the model driven approach. The approach consists of the use of an ontology based on TAO conceptual framework as a CIM. Then, we apply a transformation to (semi)automatically generate a PIM, which is further refined. The former PIM is described in the MAS-ML modeling language and the latter in UML. Finally, we sketch a transformation from the refined PIM to a PSM (in our example, specified to JADE platform).

By using an ontology as a base model for a model driven development, we intend to take advantage of its formalism, since it must be possible to check the consistency of models generated from an ontology.

References

BAUER, B. MÜLLER, J.P. and ODELL, J. Agent UML: A Formalism for Specifying Multiagent Software Systems In: Ciancarini and Wooldridge (Eds) **Agent-Oriented Software Engineering**, Springer-Verlag, LNCS vol 1957, 2001.

BELLIFEMINE, F., CAIRE, G., POGGI, A. and RIMASSA, G JADE, a White Paper, Special issue on JADE of the TILAB Journal "EXP - in search of innovation", September 2003, pp 6-19. Also available at http://exp.telecomitalialab.com/upload/issues/v3n3.pdf , visited at 04/12/2004.

BELLIFEMINE, POGGI, A. and RIMASSA, G. JADE – A FIPA-compliant agent framework, PROCEEDINGS OF PAAM'99, London, April 1999, pp 97-108.

BRAUBACH, L. LAMESDORF, W., POKAHR, A. Jadex: Implementing a BDI-Infrastructure for JADE Agents, Special issue on JADE of the TILAB Journal "EXP - in search of innovation", September 2003, pp 76-85. Also available at http://exp.telecomitalialab.com/upload/issues/v3n3.pdf , visited at 04/12/2004

BRESCIANI, P., GIORGINI, P., GIUNCHIGLIA, F., MYLOPOULOS, J. and Perini, **A. TROPOS: An Agent-Oriented Software Development Methodology**, University of Trento, Italy, Technical Report # DIT-02-0015, 2002

CAIRE, G; CHAINHO, F.; EVANS, R. Agent-oriented analysis using Message/UML. In: Wooldridge, M.; Weiss, G.; Ciancarini, P.(Eds) **Agent-Oriented Software Engineering**,. Springer, Canada, LNCS 2222, 2002, p. 119-135.

CARLEY, K. Computational organizational theory. In: **Multiagent Systems – A Modern Approach to Distributed Artificial Intelligence.** MIT Press. 1999

CUNHA, L. M.; BARBOSA, S. D. J.; LUCENA, C. J. P. de. Leveraging the Construction of Semantic Web Applications Using the Model Driven Architecture. In Ashish, N; Globe, C.(eds). SEMANTIC WEB TECHNOLOGIES FOR SEARCHING AND RETRIEVING SCIENTIFIC DATA 2003. PROCEEDINGS OF THE WORKSHOP AT THE 2ND INTERNATIONAL SEMANTIC WEB CONFERENCE - ISWC2003, CEUR Workshop Proceedings, ISSN 1613-0073, v. 83, Sanibel Island, Florida, USA, October 20, 2003. <Available at www.CEUR-WS.org>. visited at 04/12/2004

DIESTE, O., JURISTO, N., MORENO, Ana M., PAZOS, J., SIERA, A. Conceptual Modeling In Chang, S.K. (ed) **Software Engineering and Knowledge Engineering: Concepts, Techniques and Trends, Handbook of Software Engineering & Knowledge Engineering Fundamentals**, vol. 1, 2001.

ELAMMARI, M.; LALONDE, W. An Agent-Oriented Methodology: High-level and Intermediate Models. In: Wagner, G.; Yu, E. (Eds), PROCEEDINGS OF AGENT ORIENTED INFORMATION SYSTEMS, AGENT-ORIENTED INFORMATION SYSTEM (AOIS99), Washington. 1999.

FIPA – Foundation for Intelligent Physical Agents, http://www.fipa.org visited at 04/12/2004

FRANKEL, D.S. **Model Driven Architecture- Applying MDA to Enterprise Computing**, Wiley, 2003.

GRAHAM, J.R, DECKER, K.S., MERSIC, M. DECAF – A Flexible Multi-Agent System Architecture, Journal of Autonomous Agents and Multi-Agent Systems, Kluwer Academic Publishers, v.7, pp. 7-27, 2003.

KAZAKOV, M., ABDULRAB, H. and DEBARBOUILLE, G. **A model driven approach for design of mobile agent systems for concurrent engineering: MAD4CE project**, Rapport Interne 01-002, Université et INSA de Rouen, 2002.

KINNY, D. The # Calculus: An Algebraic Agent Language. In: **Intelligent Agents VIII**, Springer, LNAI v.2333, 2002, p. 32-50.

KULESZA, U., GARCIA, A., LUCENA, C. Generating Aspect-Oriented Agent Architectures. PROCEEDINGS OF THE 3RD WORKSHOP ON EARLY ASPECTS - ASPECT-ORIENTED REQUIREMENTS ENGINEERING AND ARCHITECTURE DESIGN, 3RD INTERNATIONAL CONFERENCE ON ASPECT-ORIENTED SOFTWARE DEVELOPMENT, March 2004, Lancaster, UK.

LIND, J. MASSIVE: **Software Engineering for Multi-agent Systems**. PhD Dissertation, Universität des Saarlandes, Saarbrücken, Germany, 2000.

MDA – Model Driven Architecture, http://www.omg.org/mda/ visited at 04/12/2004

MDA Guide version 1.0.1, 2003, available at http://www.omg.org/docs/omg/03-06-01.pdf visited at 04/12/2004

MORAÏTIS, P., PETRAKI, E. SPANOUDAKIS, N.I. Engineering JADE Agents with the Gaia Methodology, **Agent Technologies, Infrastructures, Tools, and Applications for e-services**, Springer, LNCS vol 2592, 2003, pp 77-91.

NWANA, H.S., NDUMU, D.T., LEE, L.C.: ZEUS: An Advanced Tool-Kit for Engineering Distributed Multi-Agent Systems. Applied AI v. 13, n.1, 29-185, 1998.

ODM, Ontology Definition Metamodel – request for proposal – OMG document: ad/2003-03-40 available at http://www.omg.org visited at 04/12/2004

OilEd, available at http://oiled.man.ac.uk/index.shtml visited at 04/12/2004

OMG - The Object Management Group, http://www.omg.org/ visited at 04/12/2004

OWL – Ontology Web Language, http://www.w3.org/TR/2003/CR-owl-features-20030818/ visited at 04/12/2004

PAIGE, R. and RADJENOVIC, A. Towards Model Transformation with TXL, PROCEEDINGS OF METAMODELLING FOR MDA, FIRST INTENATIONAL WORKSHOP, York, UK, 2003, pp. 162-177.

PETRIE, C. Agent-Based Software Engineering. In: Ciancarini, P.; Wooldridge, M. (Eds.) **Agent-Oriented Software Engineering**, Berlin: Springer, LNCS 1957, 2001, p.59-76.

Protégé 2000, available at http://protege.stanford.edu/ visited at 04/12/2004

SHOHAM, Y. Agent0: A Simple Agent Language and its Interpreter. In: PROCEEDINGS OF THE NINTH NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE, 1991, p.704–709.

SHOHAM, Y. Agent-Oriented Programming. Artificial Intelligence, v.60, 1993

SILVA, V. , GARCIA, A., BRANDÃO, A., CHAVEZ, C., LUCENA, C., ALENCAR, P. Taming Agents and Objects in Software Engineering, In Garcia et al (Eds) **Software Engineering for Large Scale Multi-Agent Systems**, LNCS, vol 2603, 2003.

SILVA, V. AND LUCENA, C. From a Conceptual Framework for Agents and Objects to a Multi-Agent System Modeling Language, Journal of Autonomous Agents and Multi-Agent Systems, Kluwer Academic Publishers, v.9, pp. 145-189, 2004.

SYCARA, K., PAOLUCCI, M., VELSEN, M. Van, GIAMPAPA, J. The RETSINA MAS Infrastructure, Journal of Autonomous Agents and Multi-Agent Systems, v. 7, p. 29-48, 2003.