Contribution to AL3 AOSE TFG:

INGENIAS methodology and meta-models

Jorge J. Gómez-Sanz, Juan Pavón

{jjgomez,jpavon}@sip.ucm.es

Grasia! research group

Facultad de Informática

Universidad Complutense de Madrid

http://grasia.fdi.ucm.es

Introduction to INGENIAS

The purpose of INGENIAS is the definition of a methodology for the development of MAS, by integrating results from research in the area of agent technology with a well-established software development process, which in this case is the Rational Unified Process (RUP). This methodology is based on the definition of a set of meta-models that describe the elements that form a MAS from several viewpoints, and that allow to define a specification language for MAS. The viewpoints are five: agent (definition, control and management of agent mental state), interactions, organization, environment, and goals/tasks.

The integration of the INGENIAS MAS specification language with software engineering practices is achieved by defining a set of activities that guide the analysis and design phases, with the statement of the results that have to be produced by each activity. This process is supported by a set of tools, which are generated from the meta-models specification. MAS modelling is facilitated by a graphical editor, automatic code generation and validation tools. The usability of this language and associated tools and its integration with software engineering practices has been validated with several examples from different domains, such as PC management, e-business, personal assistants, and collaborative filtering.

META-MODELLING

Though there may be previous interpretations of what meta-modeling is, in this document we attend to the definition provided in the Meta Object Facilities (MOF) [OMG 2000] specification of UML. This definition states that there are several levels in the definition of a language. In fact, it defines four levels where different language grammars are defined and each level defines the grammar to be used in the next level. This process could be understood as a backwards stepwise abstraction from the information level. The process ends at the M1 level, which so far has proven to be enough to UML.

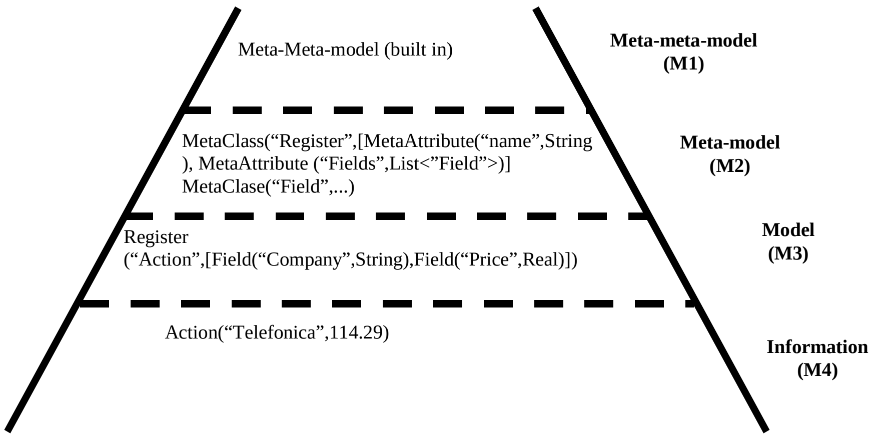

Meta-Meta-model (built in)
Meta-meta-model (M1)
MetaClass("Register",[MetaAttribute("name",String), MetaAttribute ("Fields",List<"Field">)] MetaClase("Field",...)
Meta-model (M2)
Register ("Action",[Field("Company",String),Field("Price",Real)])
Model (M3)
Action("Telefonica",114.29)
Information (M4)

Figure 1. Meta-modelling levels according to [OMG 2000]

In INGENIAS we use the schema of Figure 1 to structure de definition of the diagrams. However, we change the base M1 meta-meta-model and use a different one from MOF, GOPRR [Lyytinen 1999], as we consider its concepts simpler than MOF and initially because the availability of tool support. In INGENIAS, after different experiences with MOF, we realized that most of the diagrams that we needed did not use most of the primitives of MOF, mainly because we were not defining an object oriented language, but an agent modelling language. In this sense, we have experienced that using entity-relationship diagrams is enough for defining INGENIAS diagrams. And a suitable language to define this kind of diagrams is GOPRR. GOPRR stands for Graph Object Property Relationship and Role, since these are the elements used to define any entity-relationship diagram. GOPRR seems to be enough to define UML diagrams. As a proof of that, METAEDIT+, a meta-case tool distributed by METACASE, implements all UML diagrams, except UML sequence diagrams (however, INGENIAS supports AUML sequence diagrams).

INGENIAS meta-models are defined in the M2 level. IDK implements M2 meta-models and is used to generate M3 models. Therefore, instances of these meta-models are the concrete diagrams that the developer defines (level M3) with the IDK. There is an extra level, the M4, that is supposed to hold instances of M3 models. In INGENIAS we leave this instantiation to the developer, but provides a partial support for it.

Figure 2 shows an example of a meta-model M2 which is part of the agent meta-model. It is represented using a UML class diagrams and stereotypes. GOPRR primitives appear as stereotypes of the different elements of the diagram. Basically, the diagram says that an *agent* is an *autonomous entity* that *pursues goals*. *Goals* are *mental entities* that form part of the *mental state* of the *agent*. An *agent* plays *roles* and, in this way, assume responsibilities. An agent uses *tasks* to modify its mental state and the environment. These tasks are assigned to agents directly or through roles played. Changes in the Mental state are controlled using the *mental state manager*. This entity takes care of the consistency of the *mental state* and provides the primitives to change it. Decision procedures of the agent are built in the *mental state processor*.

INGENIAS Meta-models

INGENIAS meta-models define five kinds of elements in order to define a MAS. So INGENIAS uses five meta-models that describes the corresponding types of diagrams. Entities of these meta-models, i.e. meta-entities, are not unique in the sense that anyone could be used in any of them. As a result, an entity, instance of a meta-entity, could appear in different diagrams.

- **Organization meta-model.** It defines organization diagrams. The organization is the equivalent of the MAS architecture. An organization has structure and functionality. The structure is similar to the one stated in AALAADIN framework [Ferber 1998]. As a developer, the organization is defined taking into account how agents should be grouped. Functionality is determined when defining the goals of the organization and the workflows it should execute.
- **Environment meta-model.** It defines environment diagrams. The environment is what surrounds the MAS and what originates agent perception and action, mainly. As a developer, one of the first tasks is to identify system resources, applications, and agents. System resources are represented using TAEMS [Wagner 2001] notation. Applications are wrappers of whatever is not an agent or a resource, and could be understood as the equivalent of objects in INGENIAS. Using these elements, a developer should be able to define how the MAS interact with the system.
- **Tasks/Goals meta-model.** It describes how the mental state of agents change over the time, what is the consequence of executing a task with respect the mental state of an agent, how to achieve goals, and what happens when a goal cannot be achieved. It also gathers dependencies among different system or agent goals.
- **Agent meta-model.** It defines primitives to describe a single agent. It can be used to define the capabilities of an agent or its mental state. The mental state is an aggregate of mental entities that satisfy certain conditions. The initial or intermediate mental state is expressed in terms of mental entities such as those of AOP [Shoham 1993] and BDI [Kinny 1997].
- **Interaction meta-model.** It describes two or more agents interacting. The interaction behavior is described using different languages, such as UML collaboration diagrams,

GRASIA interaction diagrams, or AUML protocol diagrams. An interaction has a purpose that has to be shared or partially pursued by interaction participants. Usually it is related with some organizational goal.

An extensive detailed list of the INGEINAS diagrams and entities, as well as relationships, can be found in the ingenias web site: http://grasia.fdi.ucm.es/ingenias/metamodel/.

Meta-model example: agent meta-model

The agent meta-model defines an isolated agent. The agent concept underlying this meta-model is the one defined by Newell [Newell 1982]. An agent is a program that exists at the *knowledge level*. It has a physical body with which it can act in the environment, a knowledge body which contains whatever the agent knows at some time, and a set of goals. Also, an agent behaves according to the principle of rationality which says *if an agent has a knowledge that one of its actions will lead to one of its goals, then the agent will select that action*. Following this definition, an agent has goals and there should be some association type between agent tasks and goals.

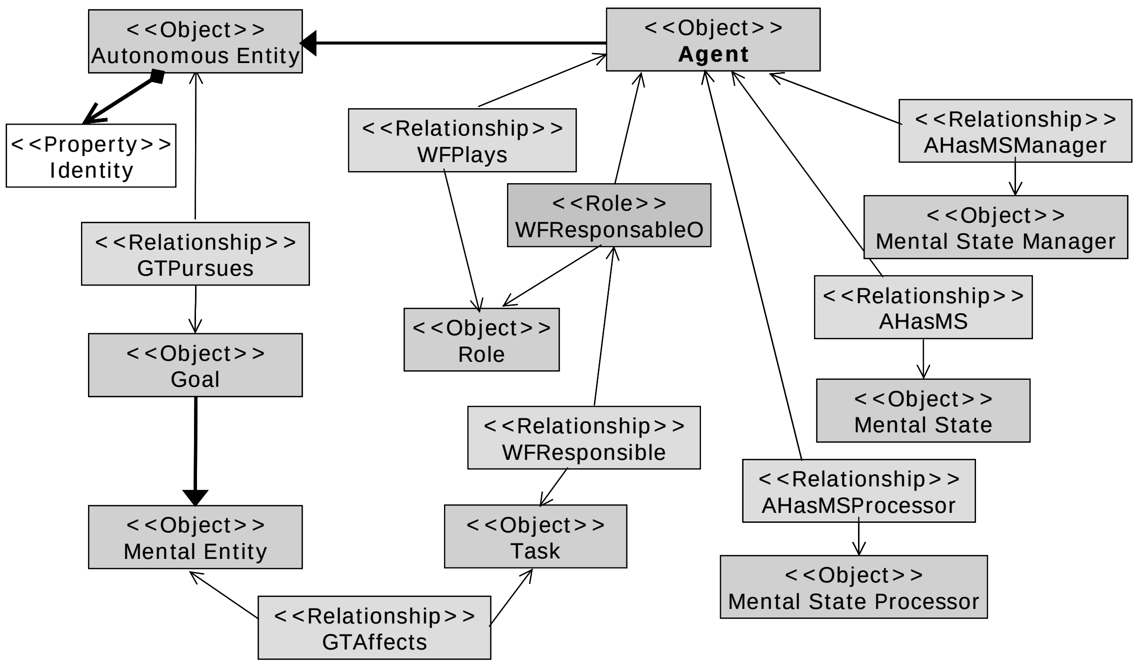

<<Object>> Autonomous Entity
<<Object>> Agent
<<Property>> Identity
<<Relationship>> WFPlays
<<Relationship>> AHasMSManager
<<Role>> WFResponsableO
<<Object>> Mental State Manager
<<Relationship>> GTPursues
<<Relationship>> AHasMS
<<Object>> Role
<<Object>> Goal
<<Object>> Mental State
<<Relationship>> WFResponsible
<<Relationship>> AHasMSProcessor
<<Object>> Mental Entity
<<Object>> Task
<<Object>> Mental State Processor
<<Relationship>> GTAffects

Figure 2. Meta-model for the agent. Stereotypes denote the GOPRR primitive

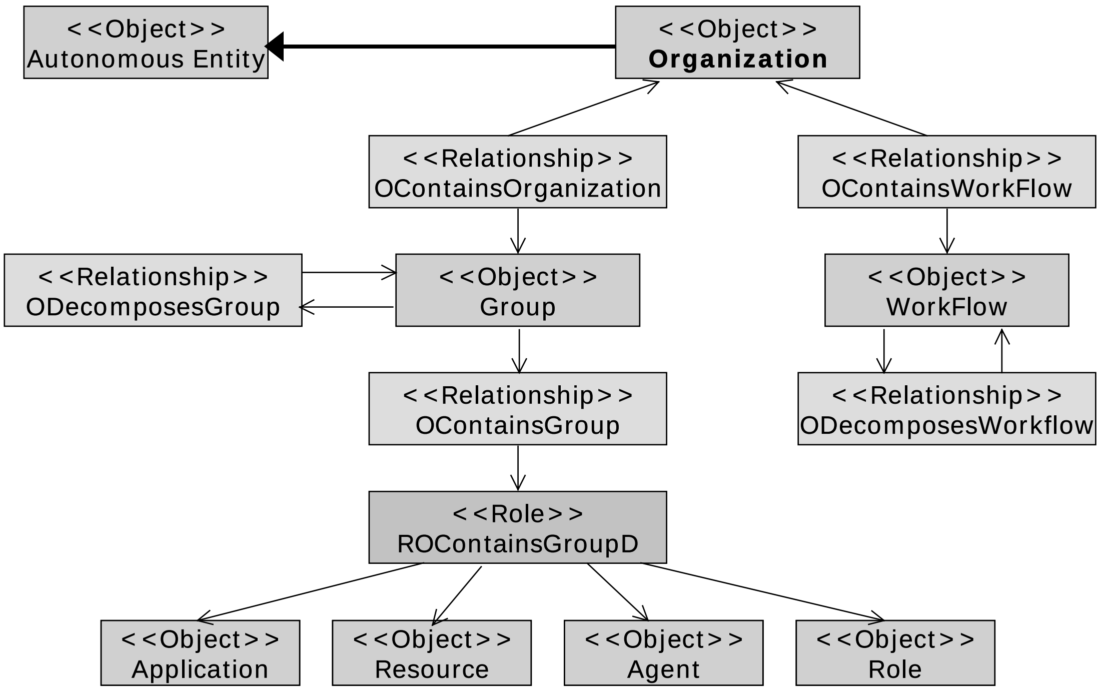
Figure 2 shows the meta-model for defining agents. In the definition is fundamental to identify the tasks that the agent has to execute (relationship *WFResponsible*) and the result of these tasks affects existing mental entities (relationship *GTAffects*). There is a taxonomy to differentiate the ways in which the result of one task may affect to the agent's mental state. The relationship *GTSatisfies* means that the result of one task implies that a goal has been reached, and the relationship *GTFails* means that the result of one task implies that a goal is considered as unreachable. On the other side, the agent plays roles in several workflows in the system. The association of an agent to a role (*WFPlays*) means that the agent acquires all the properties and responsibilities assigned to the role (goals and interactions in which the role participates).

The agent has a mental state which is used to decide what to do next (*MentalStateProcessor)*. This mental state is managed by the *MentalStateManager*. This entity is in charge of adding/removing knowledge as well as consistence maintenance. Agent's mental state consists of control entities and information entities. Control entities specify what is expected from the agent, whilst information entities describe the state of the world as seen by the agent. From these entities, the most fundamental is the *objective*, which represents a goal for the agent. The objective is an entity with a state that specifies whether it has been satisfied, will never be satisfied, or is in process of being satisfied (solving).

Meta-model example: organization meta-model

An *organization* in MAS characterizes a group of agents that work together towards a common goal (*purpose*). The organization may consist of only one agent or several groups of co-operating agents, which form part of *organizational structures* that establish relationships among them.

The organization structure aspects of the organization meta-model, shown in Figure 3, intends to structure agents in the system and reflect the goals of the system, the ways to achieve them (resources and tasks), which agents have responsibilities and their role in the global process.



<<Object>> Autonomous Entity
<<Object>> Organization
<<Relationship>> OContainsOrganization
<<Relationship>> OContainsWorkFlow
<<Relationship>> ODecomposesGroup
<<Object>> Group
<<Object>> WorkFlow
<<Relationship>> OContainsGroup
<<Relationship>> ODecomposesWorkflow
<<Role>> ROContainsGroupD
<<Object>> Application
<<Object>> Resource
<<Object>> Agent
<<Object>> Role

Figure 3. Elements in the Organization meta-model that describes Organization Structure. Stereotypes denote the GOPRR primitive

Organization structures support another level of structuring in the organization. The idea is similar to departmental organization (department as organization structure) in enterprises (organizations). In this description, an organization is an autonomous entity that has purposes and is composed of groups and workflows. Workflows express the functionality of the organization. On the other hand, Organization groups represent actors that participate in the workflows structured in functional departments.

Development Process

INGENIAS also defines a development process, the INGENIAS Development Process (IDP). By following IDP, a developer generates the full specification of a MAS. To specify the IDP we use activity diagrams that describe the kind of results to obtain.

Building each meta-model can be achieved by performing a set of activities in the software development process that leads to the final MAS specification. Initially, activities are organised and represented with UML activity diagrams showing dependencies between them. Instead of showing these activities here, Figure 4 summarises the results required in each phase of the Unified Software Development Process. Meta-models are used as specification language of the MAS the same way as UML does for object oriented applications.

		PHASES		
		Inception	**Elaboration**	**Construction**
WORKFLOWS	**Analysis**	o Generate use cases and identify actions of these use cases with interaction models. o Sketch a system architecture with an organization model. o Generate enviroment models to represent results from requirement gathering stage	o Refined use cases o Agent models that detail elements of the system architecture. o Workflows and tasks in organization models o Models of tasks and goals to highlight control constraints (main goals, goal decomposition) o Refinements of environment model to include new environment elements	o Refinements on existing models to cover use cases
	Design	o Generate prototypes perhaps with rapid application development tool such as ZEUS o Agent Tool.	o Refinements in workflows o Interaction models that show how tasks are executed. o Models of tasks and goals that reflect dependencies and needs identified in workflows and how system goals are achieved o Agent models to show required mental state patterns	o Generate new models o Social relationships that perfect organization behaviour.

Figure 4. Results to be obtained in each phase of the development process

Tools and references

INGENIAS support tools, the INGENIAS Development Kit, is distributed from http://ingenias.sourceforge.net

INGENIAS methodology can be reviewed in our official web site, http://grasia.fdi.ucm.es, http://ingenias.sourceforge.net

INGENIAS meta-models are accesible in http://grasia.fdi.ucm.es/ingenias/metamodel

Published papers about INGENIAS are [Pavón 2003], where the methodology is described, [Fuentes 2003; Fuentes 2004] present validation and verification approaches based on social theories, [Gomez-Sanz 2002] presented meta-models of INGENIAS, and [Gomez-Sanz 2002] presented an example of modelling of INGENIAS and its development process.

Case studies

INGENIAS Development Kit comes with several case studies that show how to use the methodology:

- **Juul Bookseller.** It describes with agents a e-business real situation of a bookseller that has special agreements with university students and professors.
- **Quake.** It models a prototype developed in the GRASIA group of bots coordinating themselves using natural language in the QUAKE game. This specification was contributed by Guillermo Jimenez as research work in our Ph.D. courses.
- **Robocode.** It models a robot in the robocode game from IBM. This example has two files: robocode-inception.xml is the result obtained in the inception stage following the INGENIAS Development Process (IDP); and robocode-elaboration.xml is the result obtained in the inception stage of the IDP.
- **Collaborative Filtering.** This example is not fully documented, though it is the largest. It models a community of agents that filter information for their users.

References

[Ferber 1998] Ferber, J. and O. Gutknecht (1998). *A Meta-Model for the Analysis and Design of Organizations in Multi-Agent Systems*, Proceedings of the Third International Conference on Multi-Agent Systems (ICMAS98), IEEE CS Press.

[Fuentes 2003] Fuentes, R., J. J. Gómez-Sanz and J. Pavón (2003). *Activity Theory for the Analysis and Design of Multi-Agent Systems*. Proceedings of the Fourth International Workshop on Agent Oriented Software Engineering (AOSE 2003), . Springer Verlag.

[Fuentes 2004] Fuentes, R., J. J. Gómez-Sanz and J. Pavón (2004). *Towards Requirements Elicitation in Multi-Agent Systems*. In Proceedings of the 4th International Symposium From Agent Theory to Agent Implementation.

[Gomez-Sanz 2002] Gomez-Sanz, J. J. and J. Pavon (2002). *Meta-modelling in Agent Oriented Software Engineering*. **LNAI 2527:** 606-615.

[Gomez-Sanz 2002] Gomez-Sanz, J. J., J. Pavon and F. Garijo (2002). *Meta-modelling of Multi-Agent Systems*, ACM.

[Kinny 1997] Kinny, D., M. Georgeff and A. Rao (1997). *A Methodology and Modelling Technique for Systems of BDI Agents*. Australian Artificial Intelligence Institute.

[Lyytinen 1999] Lyytinen, K. S. and M. Rossi (1999). *METAEDIT+ --- A Fully Configurable Multi-User and Multi-Tool CASE and CAME Environment*, Springer-Verlag.

[Newell 1982] Newell, A. (1982). "*The knowledge level*." Artificial Intelligence **18**: 87-127.

[OMG 2000] OMG (2000). *MOF. Meta Object Facility (specification)*.

[Pavón 2003] Pavón, J. and J. Gómez-Sanz (2003). *Agent oriented software engineering with INGENIAS.* International Central and Eastern European Conference on Multi-Agent Systems, Springer Verlag.

[Shoham 1993] Shoham, Y. (1993). "*Agent Oriented Programming*." Artificial Intelligence **60**: 51-92.

[Wagner 2001] Wagner, T. and B. Horling (2001). *The Struggle for Reuse and Domain Independence: Research with TAEMS, DTC and JAF*, Proceedings of the 2nd Workshop on Infrastructure for Agents, MAS, and Scalable MAS.