Integrating a Tropos Modeling Tool with a MDA Methodology for Engineering Multi-agent Systems

João Victor Guinelli[1], Carlos Eduardo Pantoja[1], Ricardo Choren[2]

[1]CEFET/RJ – UnED Nova Friburgo – Av. Gov. Roberto da Silveira, 1900 – Nova Friburgo – RJ – Brasil

[2]Instituto Militar de Engenharia – IME/RJ – Pça Gen Tibúrcio 80 – Rio de Janeiro – 22270-290 – RJ – Brasil

jvguinelli@gmail.com, pantoja@cefet-rj.br, choren@ime.eb.br

Abstract. *This paper presents an integration between a Model-Driven Architecture (MDA) methodology for Multi-Agent System (MAS) development and the TAOM4E, which is a tool for graphical modeling that gives support to the Tropos methodology. Additionally, the MDA methodology uses the FAML, which is a metamodel that includes concepts of several agent-oriented methodologies (including Tropos) into a single model as Platform Independent Model and the JaCaMo metamodel, composed of Jason, Moise+ and CArtAgO, as Platform Specific Model. The methodology is also able to transform FAML concepts to JaCaMo concepts and generate code to the Jason/Moise+ from JaCaMo metamodel. The objective of this is paper is to allow the MAS designer to generate Jason/Moise+ code directly from a Tropos model using the proposed integration. The transformation set between the graphical modeling tool and the FAML metamodel was specified using the Query-View-Transformation language. The paper also presents an example using the integrated solution.*

1. Introduction

Engineering Multi-Agent Systems (MAS) is an effort to construct intelligent systems in distributed environment, capable of reasoning and communicating with other systems or intelligent agents. In last decade, several methodologies and approaches to design and develop MAS were presented in order to support the MAS engineering. Among these approaches, the Model-Driven Architecture (MDA) took an important role trying to fill the gap between several methodologies and modeling languages for MAS [Nunes et al., 2011].

The MDA provides a software construction divided in different and independent abstraction levels that are supported by metamodels. Applying metamodels in Model-Driven Development (MDD) allows a point of connection between the developed solution and existent solutions, since it is provided standardized specifications for metamodels such as Query-View-Transformation (QVT) [OMG, 2011]. There are several methodologies for engineering a MAS that are supported by MDD tools such as Tool for Agent Oriented Modeling for Eclipse platform (TAOM4E) [Morandini, 2008], for Tropos; and Prometheus Design Tool (PDT) [Sun et al., 2010] for Prometheus. Although these methodologies use a MDA approach, they constraint the developer in either modeling language or language specification. Since they do not use a generic metamodel, the real potential of the MDA approach is minimized, once both are bound to their specific methodologies.

There is a MDA approach for MAS development [Pantoja; Choren, 2013] which uses a generic metamodel for MAS specification named FAML [Beydoun et al., 2009]. This approach allows the developer to choose a preferred design technology among the methodologies, modeling languages and notations adherent to FAML. After that, a set of QVT specifications to transform the FAML concepts into JaCaMo [Boissier et al., 2011] concepts in order to generate Jason [Bordini et al., 2007] and Moise+ [Hubner et al., 2002] codification. The methodology is not bound to JaCaMo because it can be extended to any specific programming language, since a cartridge with a new set of transformations (from FAML to this specific technology) can be developed. The methodology is also supported by a MDD tool. However, this tool only provides semi-automatic code generation (since it needs manual interventions in its metamodels) because there is no graphical environment associated with it in order to automatically instantiate these metamodels.

The objective of this paper is to integrate the MDA Tool TAOM4E with the MDA methodology [Pantoja; Choren, 2013] to provide a fully automatic code generation environment for Jason/Moise+ frameworks. For this, a new set of QVT transformations (between TAOM4E metamodel to FAML) is specified. An example using the integrated solution for eclipse is shown as proof for the QVT transformations. This paper is structured as it follows: in section 2 a background for the MDA methodology for engineering a MAS is presented; section 3 confers the integration between both MDD tools and the new set of QVT transformations; in section 4 an example for a conference submission system is shown; section 5 discusses some related work; section 6 presents some concluding remarks.

2. The MDA Methodology for Engineering MAS

The MDA methodology for MAS development presented in this section was firstly proposed by Pantoja and Choren (2013). In this methodology the MAS development is divided in different levels of abstraction and it uses Model-To-Model and Model-To-Text transformations to make possible the generation of MAS code for a Platform Specific model from the MAS specification. The proposed methodology is shown in the Figure 1.

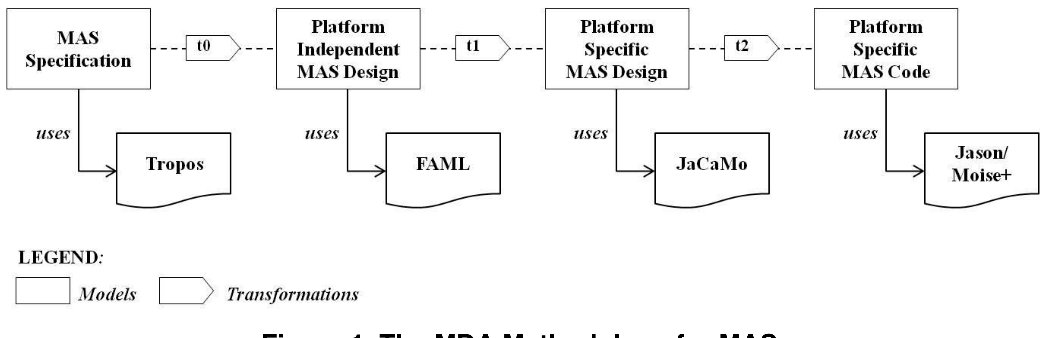


Figure 1. The MDA Methodology for MAS.

The core of this methodology is the platform independent MAS design level, which describes the MAS application in both perspectives internal level (agent) and external level (organization). One of the main purposes of this level is to provide a set of generic concepts that can be useful for any agent-oriented modeling language and methodology like Tropos, Gaia, INGENIAS and Adelfe. To achieve this purpose it was created a metamodel that implements concepts present in the FAML that is a generic metamodel for SMA specification and development. The FAML unifies, inside the same software engineering domain for MAS development, different agent-oriented modeling languages.

Even though the FAML gives more flexibility to the development and specification of SMA, the concepts of model, system, environment and agents must be present in the modeling in order to the methodology works correctly. In the figure 2 it is possible to see how the internal level of an agent is structured in FAML, where the Agent, Mental State and Plan concepts are main roles.

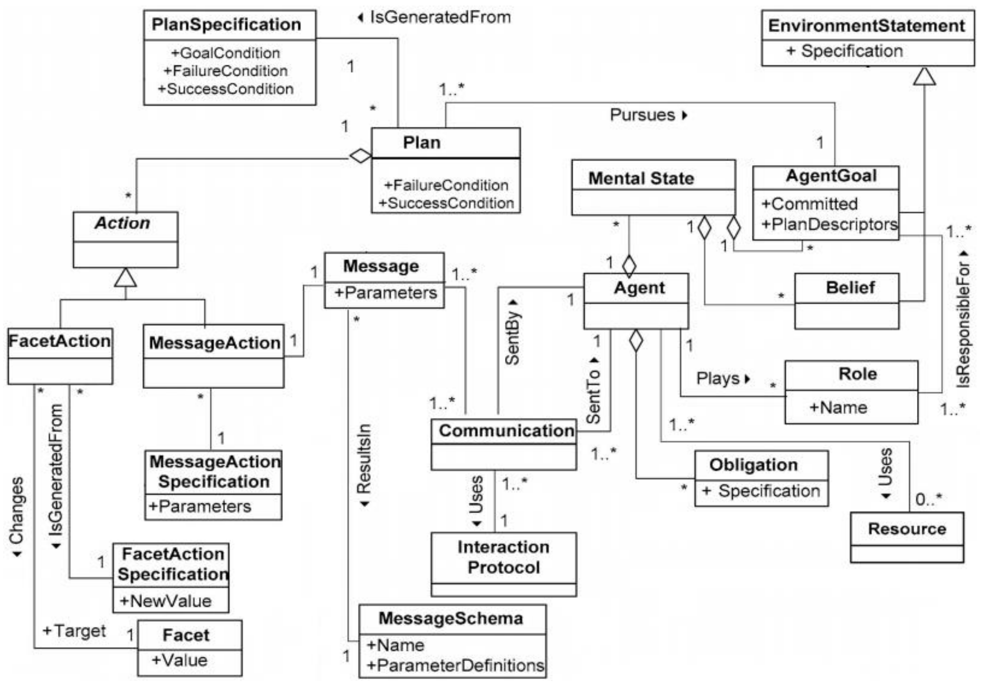


Figure 2. The FAML internal level metamodel.

The methodology starts from a MAS specification using any adherent FAML methodology (e.g. Tropos) which will create the Platform Independent MAS design after *t0* transformation. To transform the models created in the Platform Independent MAS design to the Platform Specific MAS there are the QVT transformations *t1*, which perform a Model-To-Model transformation that receive an instance of FAML model and return an instance of JaCaMo model. The JaCaMo metamodel is composed of concepts from both the agent programming and organizational programming. It uses the Jason and the Moise+ and can be edited before the *t2* transformation.

Some modifications were performed in the JaCaMo metamodel to become it more adaptable to the methodology. These modifications consist in a simplification of the environmental dimension and in a extension of the organizational dimension; they were necessary because the methodology uses the organizational model Moise+ for the organizational dimension, the agent-oriented language Jason for the dimension of the agent and, in the environmental dimension, the standard Java from Jason to implement the environment where the agents are inserted.

As the methodology do not use artifacts to implement the agent environment, it uses only Java classes for it, it is not necessary to use the CArtAgO , so the simplification of the environmental dimension consisted in to keep the *Environment* class to represent the environment, the *Percept* class to represent the perceptions and in to not use the CArtAgO.

Besides, the extension in the organizational dimension was performed adding the class *Link* in the metamodel and creating some self-relationship like *supertype*, in the class *Role*; *monitoringScheme*, in the class *Scheme*; and *subGoal*, in the class *Goal*. All these modification intent to structure the organizational Moise+ coding file considering the functional, structural and normative specifications.

The class *Link* was added to represent the relationship between groups and roles of the group specification from structural model. Moreover the *monitoringScheme* was created to permit the identification of monitoring scheme from each existing scheme, whereas the self-relationships *supertype* and *subGoal* were created to represent the role hierarchy and the goals hierarchy from the functional specification, respectively. The figure 3 shows the JaCaMo metamodel adapted to the methodology.

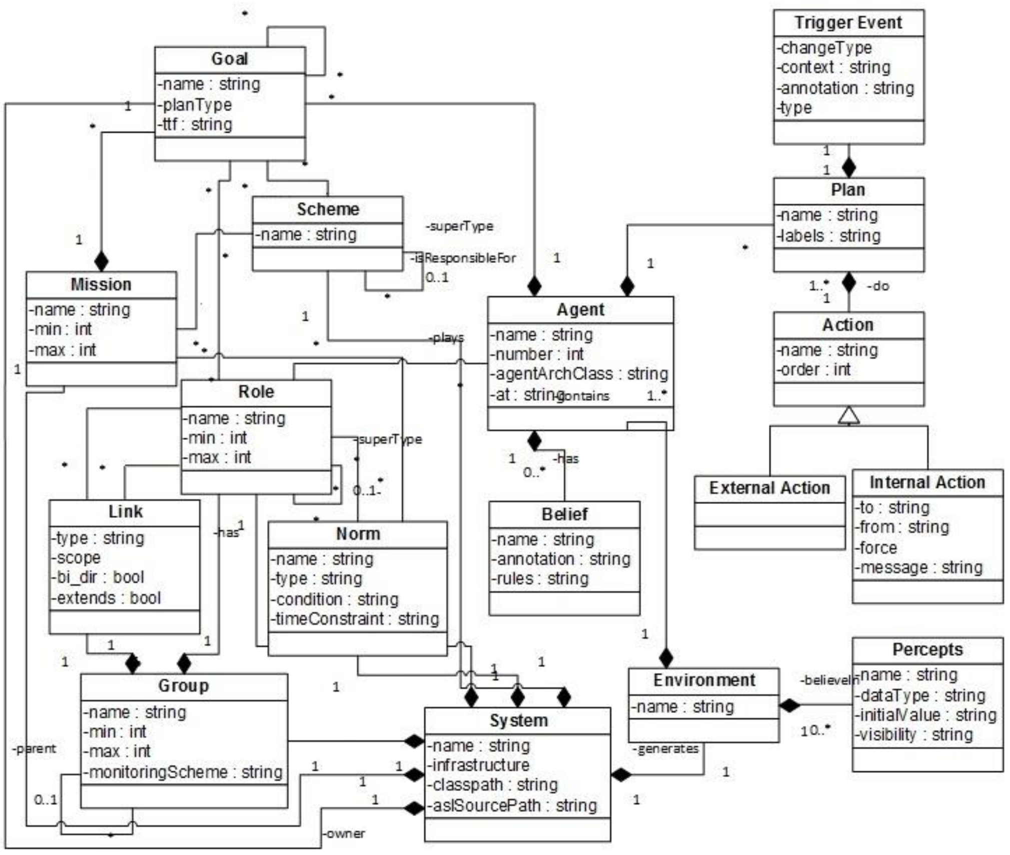


Figure 3. The Jacamo metamodel.

Finally, the transformations represented by *t2* are responsible to generate the execution code for the platform Jason and Moise+. These transformations are implemented in the M2T language, which is an OMT standard that uses templates to generate text artifacts from models, and receive as input an instantiated JaCaMo model. In addition, these transformations also contain an extension that enable the code generation for Unmanned Aerial Vehicle using AgentSpeak (UAVAS) [Hama et al., 2011], which is a Jason extension.

The UAVAS uses the Jason to program its agents, but it uses a different form to send and receive messages. Whereas in the standard Jason is used the internal actions *.send* and *.broadcast* for communicating, the Jason from UAVAS uses the external actions *request*, *inform*, *ask* and *ack*. So the transformation will generate code in two different ways, it depends if the modeled system is a Jason or UAVAS system.

3. The TAOM4E Integration to FAML Metamodel

In this section, a new set of transformations between the metamodel (*t0* transformation) used by TAOM4E and the FAML metamodel is presented (the hierarchical set can be seen in figure 4). The TAOM4E metamodel is divided in two models: the *Core*, which holds the Tropos modeling concepts; and the *View*, which defines the graphical elements of the diagram. The *Core* metamodel consists of all elements instantiated in a graphical modeling project (e.g. actors, hard goal, soft goals, resources, etc.). The transformation rules are based on mapping all those instanced concepts in *Core* metamodel to a relative concept on FAML metamodel (observing the Tropos-to-FAML adherence [Beydoun et al., 2009]).

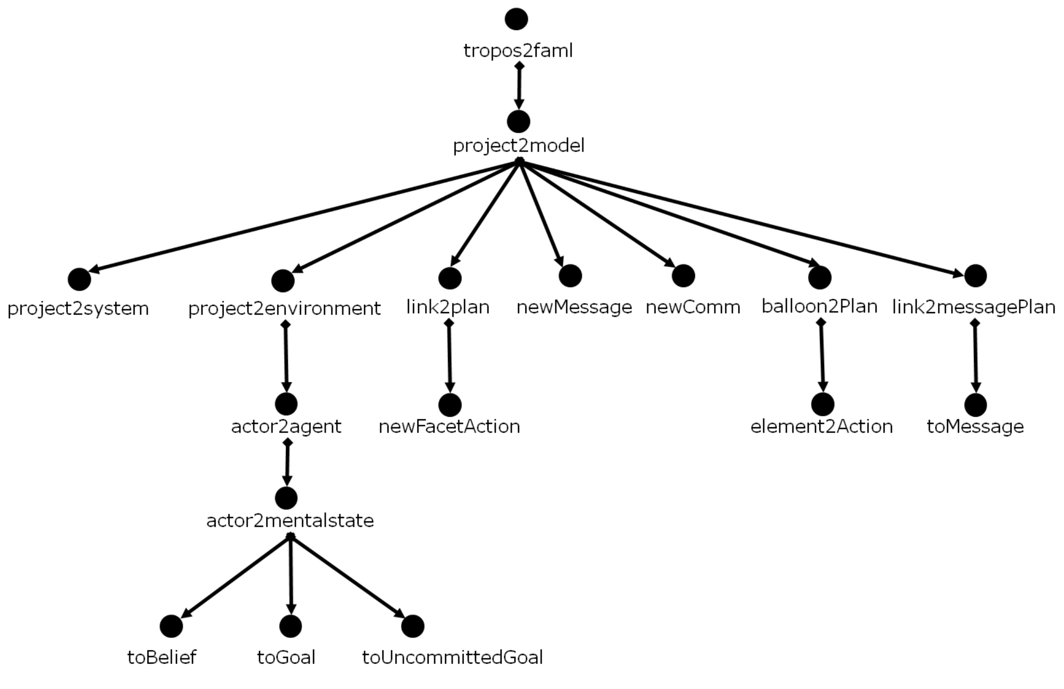
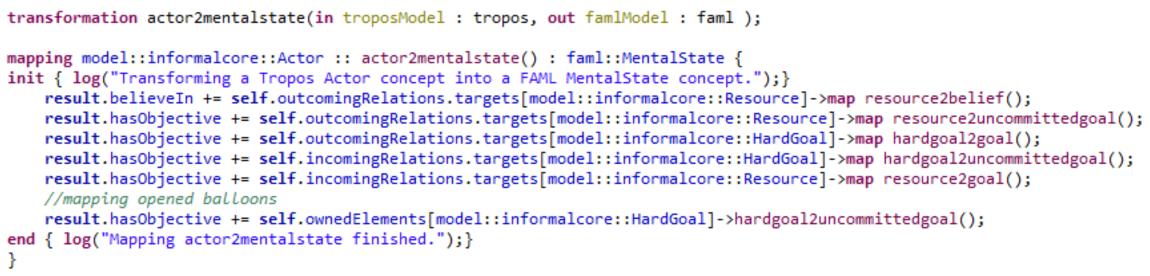


Figure 4. The transformation set from Tropos Tool to FAML metamodel.

So the transformation starts from a TAOM4E modeling project, which is mapped into a FAML project. Likewise, the Tropos model (which gathers Tropos concepts) is mapped to a FAML model. This first step of the transformation is just to create the base to hold the main concepts of the modeling. After that, the graphical components begin to be mapped. The Tropos project generates the FAML system (including references to the MAS environment). This same project also generates the environment, which maps all actors and its mental states. In FAML, a mental state is composed of beliefs and goals (that an agent can be committed or not). So, a Tropos *Resource* concept is mapped to a FAML *Belief* concept and the Tropos *Hard Goal* concept is mapped to a FAML *Agent Goal* concept. To decide if an agent is committed or not to a goal, it is observed the hard goals that an agent holds at modeling. The *Soft Goals* concepts are not materialized into code, so it was decided to not map it (but the FAML provides concepts for this kind of goals). The *actor2mentastate* transformation can be seen in figure 5.

The next steps map all plans and actions from the modeling project. Basically, the FAML has two types of actions that guide the plans mapping: a *Message Action*, which is an action of exchanging messages between agents; and a *Facet Action*, which gathers all other kind of actions (e.g. environment and reasoning actions). In the modeling project any *Hard Goal* or *Resource* that is sent to another agent becomes a message plan (specific plans containing only messages from an agent to another), otherwise, they are mapped as simple plans (plans containing the other type of actions). When a message plan is being mapped, it is necessary to create a communication concept because in FAML messages concepts only exists into a communication concept (which holds the message and all agents involved).



```
transformation actor2mentalstate(in troposModel : tropos, out famlModel : faml );

mapping model::informalcore::Actor :: actor2mentalstate() : faml::MentalState {
init { log("Transforming a Tropos Actor concept into a FAML MentalState concept.");}
    result.believeIn += self.outcomingRelations.targets[model::informalcore::Resource]->map resource2belief();
    result.hasObjective += self.outcomingRelations.targets[model::informalcore::Resource]->map resource2uncommittedgoal();
    result.hasObjective += self.outcomingRelations.targets[model::informalcore::HardGoal]->map hardgoal2goal();
    result.hasObjective += self.incomingRelations.targets[model::informalcore::HardGoal]->map hardgoal2uncommittedgoal();
    result.hasObjective += self.incomingRelations.targets[model::informalcore::Resource]->map resource2goal();
    //mapping opened balloons
    result.hasObjective += self.ownedElements[model::informalcore::HardGoal]->hardgoal2uncommittedgoal();
end { log("Mapping actor2mentalstate finished.");}
}
```

Figure 5. The *actor2mentalstate* QVT transformation.

The exploded balloons of the modeling project are mapped to simple plans, where every hard goal generates a FAML plan. In this case, a *Hard Goal* can have a link decomposition. Each decomposition, then, is mapped as a FAML *Facet Action* (the difference between a *AND* and a *OR* link is done in the platform specific model). Finally, the activities' diagrams complete the action definitions for a simple plan. After all, a complete and instantiated FAML model is provided, and the methodology can go on with subsequent transformations *t1* (from FAML to JaCaMo) and *t2* (from JaCaMo to Code).

4. A Simple Modeling Example

In this section it is presented a simple example for a Conference Management System (CMS) [DeLoach, 2002]. First it is used the TAOM4E for modeling the CMS (for Early Requirements phase) which can be seen in figure 6.

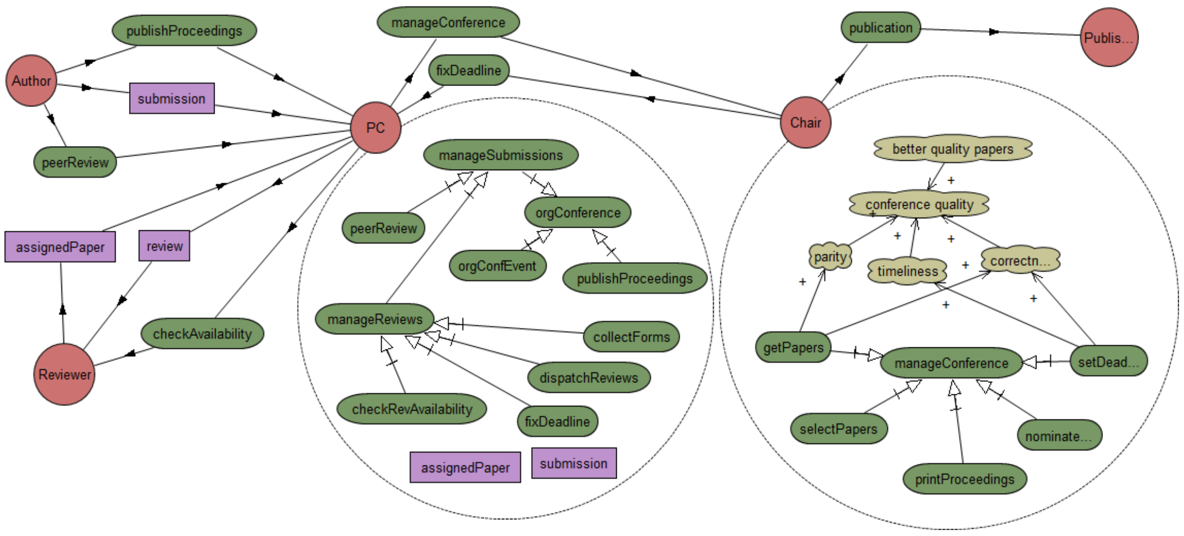
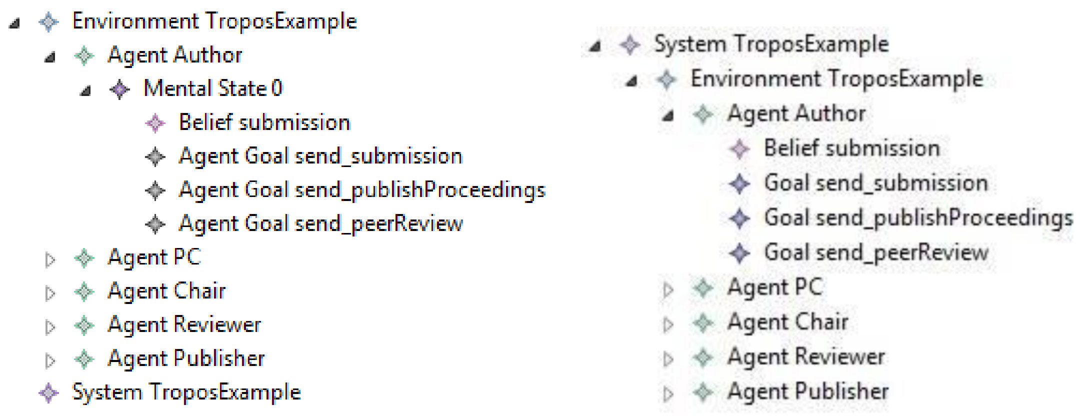


Figure 6. The TAOM4E modeling example of a Conference Management System [Morandini et al., 2008].

Second, it is necessary to perform the set of QVT transformations from Tropos to FAML (*t0* transformation) which will generate a model with the mapped concepts present in CMS modeling. These transformations were implemented using the Model-to-Model (M2M) project for eclipse. Afterwards, it is necessary to select the specific platform (executing the *t1* transformation from FAML to JaCaMo). It will generate a JaCaMo model instantiated from equivalent FAML concepts. The FAML and JaCaMo instances is shown in figure 7.


```
◢ Environment TroposExample
    ◢ Agent Author
        ◢ Mental State 0
            Belief submission
            Agent Goal send_submission
            Agent Goal send_publishProceedings
            Agent Goal send_peerReview
    ▷ Agent PC
    ▷ Agent Chair
    ▷ Agent Reviewer
    ▷ Agent Publisher
  System TroposExample
```

```
◢ System TroposExample
    ◢ Environment TroposExample
        ◢ Agent Author
            Belief submission
            Goal send_submission
            Goal send_publishProceedings
            Goal send_peerReview
        ▷ Agent PC
        ▷ Agent Chair
        ▷ Agent Reviewer
        ▷ Agent Publisher
```

Figure 7. The instances of FAML (left) and JaCaMo (right) for the environment.

Finally, the *t2* transformation generates the code for Jason/Moise+ after a set of Model-To-Text (M2T) from the JaCaMo model. The generation provides files for the system (mas2j), environment (Java), agents (asl) and organization (xml). The agent PC code can be seen in figure 8.

```
review.

!send_checkAvailability.
!send_manageConference.

+!orgConfEvent: true <-
    orgConfEvent.

+!send_review: true <-
    .send(Reviewer,achieve,review).

+!manageSubmissions: true <-
    !peerReview;
    !manageReviews.

+!send_checkAvailability: true <-
    .send(Reviewer,achieve,checkAvailability).

+!dispatchReviews: true <-
    dispatchReviews.

+submission: true <-
    submission.

+!checkRevAvailability: true <-
    checkRevAvailability.

+!publishProceedings: true <-
    publishProceedings.
```

```
+assignedPaper: true <-
    assignedPaper.

+!collectForms: true <-
    collectForms.

+!fixDeadline: true <-
    fixDeadline.

+!manageReviews: true <-
    !checkRevAvailability;
    !fixDeadline;
    !dispatchReviews;
    !collectForms.

+!orgConference: true <-
    !manageSubmissions;
    !publishProceedings;
    !orgConfEvent.

+!send_manageConference: true <-
    .send(Chair,achieve,manageConference).

+!peerReview: true <-
    peerReview.
```

Figure 8. The PC code in Jason.

5. Related Works

In this section it is discussed two related works that use MDA for designing MAS: the PDT and TAOM4E. Both solutions use the Eclipse Modeling Framework (EMF) to provide a graphical environment for MAS design. The EMF provides a software construction centered in metamodels (named Ecore), which can be interconnected with other eclipse solutions such as Object Constraint Language, Model-to-Model and Model-to-Text transformation. The proposed integration uses the EMF as part of the tool develop, so allowing interconnection with any other solution.

The PDT is a tool for the Prometheus methodology that generates code for JACK agent-oriented language. Although the PDT can be integrated to the Islander methodology, the tool still depends on Prometheus Ecore Metamodel (PEMM) and any methodology that needs to be integrated has to be compliant with it. The MDA methodology used in this work uses the FAML, which is a generic metamodel for several methodologies (including Prometheus and Islander) enabling the methodology integration with any tool adherent to FAML. It is assumed the possibility to integrate the PDT to this MDA methodology (using a new set of QVT transformation) taking advantage of both solutions.

The TAOM4E is tool based on Tropos methodology and generates code for both JACK and JADE/JADEX. In the same way, the tool is tied to a specific metamodel, limiting the MDA possibilities. Integrating the TAOM4E with this MDA methodology extends the amplitude of both solutions, allowing code generation for Jason/Moise+ (for TAOM4E), and using a graphical modeling environment to design and engineering MAS (for this MDA methodology).

6. Conclusion

This paper presented an integration between a MDD tool for modeling MAS using the Tropos methodology and a MDA methodology for engineering MAS which uses a generic metamodel for several extant methodologies. Yet, this methodology generates code for Jason/Moise+. This integration allows automatic code generation directly from a graphical modeling environment using Tropos methodology, which is a well-known approach for MAS design. This process is supported by a MDA methodology that transforms the initial modeling concepts until the MAS skeleton be generated (ensuring that all initial concepts will be presented in the final software system). Moreover, the use of metamodel enables several point of connections for existent solutions (since FAML is generic for several methodologies). The MDA methodology integrated to the TAOM4E do not aims to provide another solution for design and engineering MAS, instead, it allows to truly use the real power promised by the MDA, that is to integrate solutions and transform different abstraction models until get a software implementation.

For future work it is necessary to refine the QVT transformations applying some OCL rules in order to generate constrained models; besides of the use of JaCaMo metamodel, the solution just generates code for Jason/Moise+. So it is also necessary to extend the code genera-tion to CArtAgO, and integrate the MDA methodology to the PDT providing a methodology selection using a centered and generic metamodel.

7. References

Beydoun, G., Low, G., Henderson-Sellers, B., Mouratidis, H., Gomezsanz, J. J., Pavon, J. e Gonzalez-Perez, C. (2009) “FAML: a generic metamodel for MAS development.” IEEE Trans. Softw. Eng.

Boissier, O., Bordini, R. H., Hubner, J. F., Ricci, A. e Santi, A. (2011) “Multi-agent oriented programming with jacamo” Science of Computer Programming.

Bordini, R. H., Hubner, J. F. e Wooldridge, W. (2007) “Programming Multi-Agent Systems in AgentSpeak using Jason” Jonh Wiley and Sons, London.

DeLoach, S. A. (2002) “Modeling organizational rules in the multi-agent systems engineering methodology”. In R. Cohen and B. Spencer, editors, Canadian Conference on AI, volume 2338 of Lecture Notes in Computer Science, pages 1–15. Springer, 2002.

Hubner, J. F., Sichman, J. S. A. e Boissier, O. (2002) “A model for the structural, functional, and deontic specification of organizations in multiagent systems”, In Proceedings of the 16th Brazilian Symposium on Artificial Intelligence: Advances in Artificial Intelligence, SBIA '02, London, UK. Springer-Verlag.

Hama, M. T. ; Allgayer R. S. ; Pereira, C. E. ; Bordini, R. H. (2011) “UAVAS: An Agent Oriented Infrastructure for Unmanned Aerial Vehicles Development” In: AutoSoft@CBSoft, 2011, São Paulo. II Workshop sobre Sistemas de Software Autônomos. São Paulo: CBSoft, v. 10. p. 15-21.

Morandini, M., Nguyen, D. C., Perini, A. e Siena, A. Angelo Susi (2008) “Tool-Supported Development with Tropos: The Conference Management System Case Study”. In: Luck, M., Padgham, L. Agent-Oriented Software Engineering VIII.

Lecture Notes in Computer Science Volume 4951, 2008, pp 182-196. Springer, Germany (2008).

OMG (2011). Meta object facility (MOF) Query/View/Transfomation specification.

Nunes, I., Cirilo, E., Lucena, C. J. P., Sudeikat, J., Hahn, C. e Gomez-Sanz, J. J. "A survey on the implementation of agent oriented specifications". In: Gleizes, M., Gomez-Sanz, J. J. (eds.). Agent-Oriented Software Engineering X, pp. 169-179. Springer, Germany (2011).

Pantoja, C. E. e Choren, R. (2013) "A MDA Methodology to Support Multi-Agent System Development" In: Proceedings of 5th International Conference on Agents and Artificial Intelligence: volume 1, ICAART'13, Barcelona.

Sun, H., Thangarajah, J. e Padgham, L. (2010) "Eclipse-based prometheus design tool" In Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1 - Volume 1, AAMAS '10, Richland, SC.