INGENIAS Development Kit: a visual Multi-Agent System development environment

(Demo Paper)

Jorge J. Gomez-Sanz
Facultad de Informática
Universidad Complutense de Madrid
28040 Madrid, Spain
jjgomez@sip.ucm.es

Rubén Fuentes
Facultad de Informática
Universidad Complutense de Madrid
28040 Madrid, Spain
ruben@sip.ucm.es

Juan Pavón
Facultad de Informática
Universidad Complutense de Madrid
28040 Madrid, Spain
jpavon@sip.ucm.es

Ivan García-Magariño
Facultad de Informática
Universidad Complutense de Madrid
28040 Madrid, Spain
ivan_gmg@fdi.ucm.es

Categories and Subject Descriptors

D.2 [**Software Engineering**]: Design Tools and Techniques;
D.2 [**Software Engineering**]: Coding Tools and Techniques;
D.2 [**Software Engineering**]: Testing and Debugging

General Terms

Design, Experimentation

Keywords

Integrated Development Environment, Agent Oriented Software Engineering, Multi-Agent System

1. INTRODUCTION

The INGENIAS Development Kit, IDK from now on, is the development support tool for the INGENIAS methodology [2]. It is distributed as GPL software and developed in Java. The distribution can be downloaded from `http://ingenias.sourceforge.net` while updated versions are always available in the subversion repositores hosted within the INGENIAS sourceforge project.

This tool was created in 2002 as part of a thesis on the modelling of Multi-Agent Systems [1]. The project is rather active and has originated around 5800 downloads since its creation. Compared to other development environments, it contributes with some original functionalities, like round-trip features (generating code and managing the changes made to it), debugging facilities (breakpoints at the task level) and openness (possibility of integrating new functionality by means of plugins).

The demo will illustrate how a development with the IDK is like. The demo will review the aspects related with the MAS specification creation (section 2), the implementation (section 3), or testing and debugging (section 4).

Cite as: INGENIAS Development Kit (Demo Paper), Jorge J. Gomez-Sanz, Rubén Fuentes, Juan Pavón, and Ivan Garcia-Magariño, *Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, Padgham, Parkes, Müller and Parsons (eds.), May,12-16.,2008,Estoril,Portugal,pp. 1675-1676.

Copyright © 2008, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

2. PRODUCING A MAS SPECIFICATION

In INGENIAS, specifications of the system are produced with a visual editor, see figure 1. This editor has the following relevant features:

Project management features. The tool uses multiple diagrams constructed according to the methodology. These diagrams can be organised into packages. The organizations can be modified at any time by drag and dropping diagram icons.

MAS Specification entities management features. The user can browse the instances created so far of INGENIAS meta-model entities. The user can as well search within this dictionary, reuse elements across diagrams, or decide to remove them from all diagrams.

It is an open tool. The source code is avaliable and anybody can create plugins that are automatically integrated in the tool. Plugin creation is rather easy, since it suffices to extend one of the classes of the application. The tool comes with the sources of different plugins to exemplify how to do this.

The notation. The notation is not hard-coded in the tool as in other existing alterantives. It can be modified by us by changing the original meta-model it bases on. This permits to change the tool as the methodology evolves. Besides, the icons used in the different diagrams be changed to be UML-like or INGENIAS at any time. This reduces the learning curve since users do not have to learn what means each icon.

Remarking the documentation aspects. It supports exporting the diagrams in different formats, including extended postcript. This is very useful for people writing papers in latex, for instance. Besides, there are plugins for generating a full html documentation of the current specification. The tool itself has a manual that introduces its basic features, and it is distributed from sourceforge as well. There are also manuals and training material that tells how to use the tool to develop complex multi-agent systems. Training material is distributed from `http://grasia.fdi.ucm.es/UK/index.php?enlace=training/index.html`.

Saving the time of the user. Just in case the tool could crash, automatic backup copies are generated periodically. Besides, the tool can simplify the creation of diagrams in several steps when the circumstances allow it. Speed is important as well. Profiling techniques have been applied to improve the performance of the tool. Right now, it can be executed with a reasonable efficiency in computers with 512MB and conventional Pentium processors.

3. IMPLEMENTING THE MAS

The tool uses extensively a model driven development approach [3], producing automatically functional multi-agent systems from the specification. The responsible of producing the implementation is the INGENIAS Agent Framework or IAF, part of this framework is integrated as a plugin in the IDK. It produces MAS working over the JADE platform.

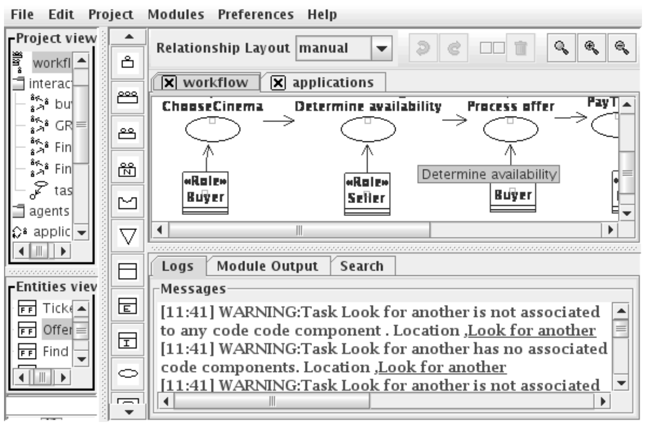
When the specification is complete enough, the developer can order the IAF plugin to transform the specification into Java code. As a result, several error or warning messages will be presented to the user, indicating missing entities or incorrect combinations of elements. After, solving all errors, source code files as well as compiling/executing scripts are stored in the project folder. Though the system is executable at this step, the functionality provided is not complete yet. It requires from the developer to tell each individual generated task file how the transformation from the input entities to the output entities occur. These changes can be introduced externally with any editor, like the Eclipse editor. Changes can be uploaded to the original specification with the aid of the *INGENIAS Code Uploader*, another plugin developed for this purpose. The plugin traverses the different task Java files looking for concrete modifications. When one is found, the change is uploaded to the specification. This way, the next time the code generation is invoked, the modifications made will not be lost. Thanks to this plugin, the specification remains synchronised with the code.

4. TESTING AND DEBUGGING

The specification can define the existence of tests and the MAS initialisation conditions these test require. Later, when the test need to be concreted, a developer uses libraries from the IDK to inspect the internal state of the agents and track each individual behaviour. Tests are defined as extensions of JUnit classes. To run them, a JUnit based system is needed, although command line execution of all tests is provided in the IAF as well.

During execution, the developer is aided with GUIs that provide useful information, such as what mental entities are stored within each agent or what interactions are being executed. The IAF GUI works into two modes: automatic and manual. In the manual mode, tasks which are going to be executed are presented to the user first. In this mode, the inputs and outputs of a task can be inspected. The user can accept to execute the task by pressing a button. The results can be immediately observed in the mental state inspector of the corresponding agent. In the automatic mode, tasks scheduled for execution are launched as they reach the top of the executing queue.

Using this automatic/manual binomy, a breakpoint mechanism has been devised. The system can be told to switch to manual mode whenever certain type of task is going to be executed. This permits inspecting the current inputs and outputs of scheduled tasks at that point and detect what is going wrong. Also, searchable log facilities are accessible from the GUI. The developer can inspect logs associated to concrete actions of the agent or filter existing logs according to typed keywords.



Relationship Layout: manual

```
[11:41] WARNING:Task Look for another is not associated
to any code code component . Location ,Look for another
[11:41] WARNING:Task Look for another has no associated
code components. Location ,Look for another
[11:41] WARNING:Task Look for another is not associated
```

Figure 1: The visual editor for producing specifications

5. CONCLUSIONS

The IDK is a powerful tool that has been applied in academy and industry developments. The list of projects in which our research group participates is avaliable at `http://grasia.fdi.ucm.es/UK/index.php?enlace=projects/index.html`.

6. ACKNOWLEDGMENTS

This work has been supported by the project *Methods and tools for agent-based modelling* supported by Spanish Council for Science and Technology with grant TIN2005-08501-C03-01.

7. REFERENCES

[1] J. J. Gomez-Sanz. *Modelado de Sistemas Multi-agente*. PhD thesis.

[2] J. Pavón, J. Gomez-Sanz, and R. Fuentes. *Agent-Oriented Methodologies*, chapter The INGENIAS Methodology and Tools, pages 236–276. Idea Group Publishing, 2005.

[3] J. Pavón, J. J. Gómez-Sanz, and R. Fuentes. Model driven development of multi-agent systems. In A. Rensink and J. Warmer, editors, *ECMDA-FA*, volume 4066 of *Lecture Notes in Computer Science*, pages 284–298. Springer, 2006.