The PRACTIONIST Development Tool

Fabio Centineo*, Angelo Marguglio* Vito Morreale* Michele Puccio*,

*R&D Laboratory - ENGINEERING Ingegneria Informatica S.p.A.

I. The PRACTIONIST suite

PRACTIONIST (PRACTIcal reasONIng sySTem) [1] is a suite of tools including (see figure 1): *(i)* a methodology, consisting of a UML-based modelling language (PAML) and an iterative and incremental development process, *(ii)* the PRACTIONIST runtime and framework (PRF), which defines and supports the execution logic and provides the built-in components according to such a logic to support the development of BDI agents in Java (using JADE[1]) with a Prolog belief base, and *(iii)* the PRACTIONIST Development Tool (PDT), a design and development environment which supports the methodology. The PRF also includes the PAIT, to monitor the intentional components of each agent and the PRACTIONIST Autonomic Manager (PAM) which enables PRACTIONIST applications to support the self-chop features [2] (self-configuring, self-healing, self-optimizing and self-protecting)

In this abstract we give an overview of the PDT, the modelling environment that is a part of the PRACTIONIST suite (figure 1), the metamodel it is built on, and a brief introduction of the PDT visual editors.

II. The PRACTIONIST Development Tool

The PRACTIONIST suite provides developers with the PRACTIONIST Development Tool, a tool to design and develop multi-agent systems according to the PRACTIONIST design methodology. Indeed, it supports such a methodology from the requirements analysis to the code generation of agents and artefacts (according to the A2A approach [2]), including a set of visual editors for each phase of the methodology. As an example, in figure 2 a snapshot of the Class editor is shown.

Some editors of the PDT are based on UML 2.0 meta-model[3], such as the class and use case editors, whereas the others are based on the PRACTIONIST Agent Modeling Language (PAML), which is a semi-formal UML-based visual modeling language for specifying, representing and documenting multi-agent systems designed with PRACTIONIST.

As the PAML aims to the definition of PRACTIONIST agents, its metamodel contains metaclasses to model intentional components of such agents, such as beliefs, goals and relations among them, plans and so forth.

The PDT has been developed by using several Eclipse[4] plug-ins, such as: UML2, Eclipse Modelling Framework (EMF), Graphical Editing Framework (GEF), Graphical Modeling Framework (GMF) and other Eclipse extensibility features. All the PDT editors share a common infrastructure, so that new editors can be added in it without any impact on the existing ones. Moreover, each editor inherits several features described below by the above infrastructure.

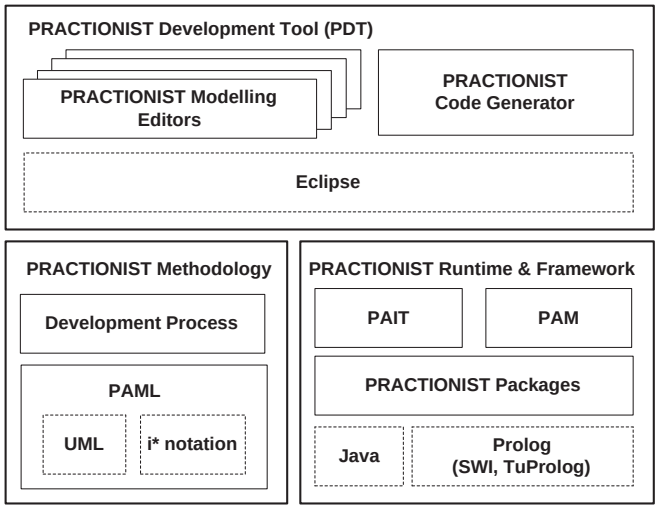

Fig. 1. The PRACTIONIST suite.

As many well-known CASE tools, the PDT editors provide all the features that support the development of complete and consistent visual models. Some of them are provided by GMF, such as the *cut and copy* and *save diagram as image* supports, the *look and feel* management, the diagram validation and so on, whereas the other ones have been built by generalizing some of the GMF project features, such as:

- *Unified model*: all diagrams created inside a PRACTIONIST project share the same model (i.e. an instance of the meta-model), whereas each generic GMF diagram file has usually its own model file. Sharing the same model file means sharing the same command stack, allowing us to execute cross-checks among elements and consequently model more complex and greater systems as a whole;
- *Drag and Drop* support: a PRACTIONIST project has its own model view, where the developed model is displayed as a tree. From this view it is possible to *drag and drop* the elements into diagrams, enabling us to use the same elements in different diagrams as well. Thus, if an element is modified in a diagram, it will be updated in all the other diagrams.

[1] http://jade.tilab.com
[2] http://www-03.ibm.com/autonomic/library.html
[3] UML 2.0 Superstructure Specification: formal/05-07-04
[4] http://www.eclipse.org/

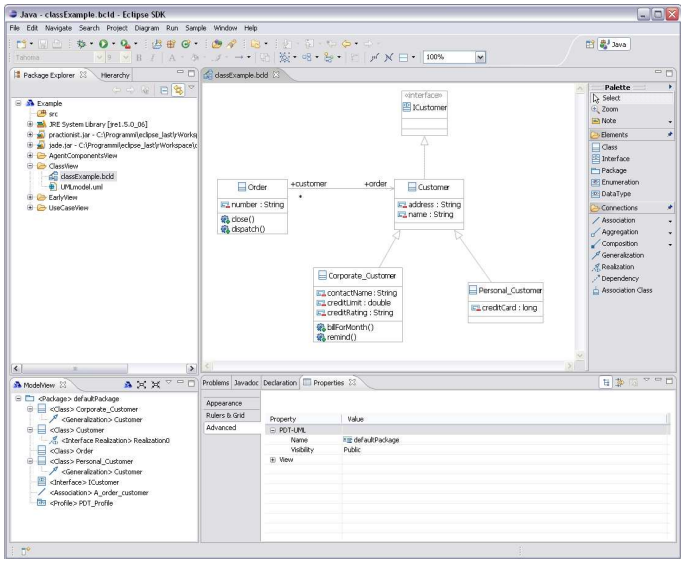

Fig. 2. A snapshot of the PRACTIONIST Development Tool.

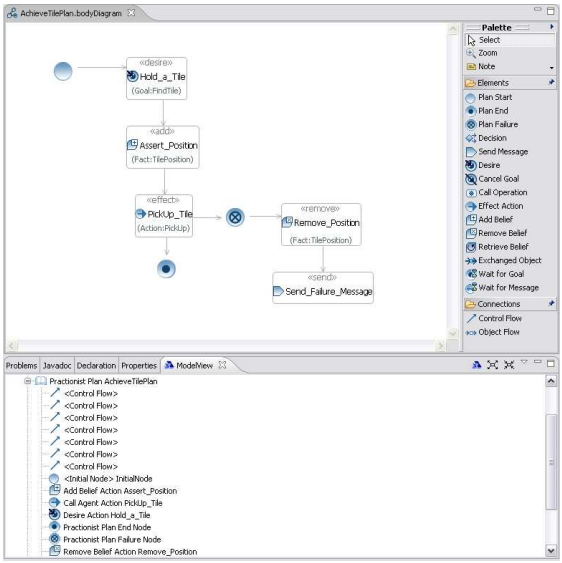

Fig. 3. A snapshot of the PDT Plan Body editor.

- *Delete from diagram* and *delete from model* actions: in a GMF diagram the *delete from model* action is enabled by default, so when an element is deleted in the diagram it is also automatically deleted from the model. Such a behaviour was modified in order to get the *delete from view* action as well, and thus have a more flexible model management.

For the development of the PDT, the support provided by the Eclipse environment has been fully exploited. As a consequence:

- a PRACTIONIST project, which is a custom Eclipse Java project, provides several sections where developers can create their own diagrams and the source folder that will contain the generated source code;
- the model view of a PRACTIONIST project is a custom Eclipse view that displays the unified model underlying the project;
- the PRACTIONIST Java code can be generated starting from diagrams in a simple way.

As mentioned, the PDT provides PRACTIONIST developers with a rich set of visual modelling editors, as follows:

- *i*-based [3] editors*:
  - *Strategic Dependency (SD) editor*: to describe the dependency relationships among various actors in an organizational context;
  - *Strategic Rational (SR) editor*: to describe stakeholder interests and concerns and how they might be addressed by various configurations of systems and environments;
- *UML2.0 based editors*:
  - *Use Case editor*: to model use cases and system funcionalities from the actor's point of view;
  - *Class editor*: to model the structure of a system or of its parts for instance (see figure 2);
- *PRACTIONIST agent editors*:
  - *Agent editor*: to model agents and specify their components;
  - *Domain editor*: to model facts about the world the agent believes true, false or has no belief about;
  - *Goal editor*: to model agent goals and the relationships among them;
  - *Effector/Action - Perceptor/Perception editor*: to model the means agents interact with their environment;
  - *Plan editor*: to model the internals PRACTIONIST plans;
  - *Plan Body editor*: to model the body of PRACTIONIST plans (see figure 3).

Finally, the PDT represents a powerful visual modeling environment that supports the representation of the concepts underlying the BDI model as well as several features present in well-known UML-based CASE tools. Moreover, it let us reduce the development time of PRACTIONIST applications thanks to the code generation of agents and artefacts. The PDT aims at bridging the gap between the increasing need of development of multi-agent systems and the availability of tools for their design.

REFERENCES

[1] V. Morreale, S. Bonura, G. Francaviglia, F. Centineo, M. Puccio, and M. Cossentino, "Developing intentional systems with the practionist framework," in *Proceedings of the 5th IEEE International Conference on Industrial Informatics (INDIN07)*, July 2007.

[2] A. Ricci, M. Viroli, and A. Omicini, "Programming MAS with artifacts." in *PROMAS*, 2005, pp. 206–221.

[3] E. S. K. Yu, "Towards modelling and reasoning support for early-phase requirements engineering," pp. 226–235. [Online]. Available: citeseer.ist.psu.edu/article/yu97towards.html