Agent Oriented Software Engineering with MESSAGE

Jorge J. Gómez-Sanz, Juan Pavón

[1]Dep. Sistemas Informáticos y Programación,
Univ. Complutense, 28040 Madrid, Spain
{jjgomez,jpavon}@sip.ucm.es
http://grasia.fdi.ucm.es

Problems in current Agent Oriented Software Engineering

After reviewing existing AOSE methodologies, we have the impression that some more work is still needed, from a software engineering viewpoint, in order to apply them in an industrial context. For instance, the software development process is usually summarized in four or five steps for analysis and design, which is not realistic. Just taking a look at the Rational Unified Process [4] is enough to get an idea that the definition of an object-oriented methodology is not trivial. And we may agree that developing a Multi-Agent Systems (MAS) is more complex than a conventional object oriented application.

It is true that in some cases, like MaSE [2], the presence of a tool, like AgentTool [7], simplifies the development process. However, this brings the risk to loss in flexibility. ZEUS [6] facilitates the development process fixing the target multi-agent system architecture. AgentTool sacrifices the flexibility in the design assuming a fixed set of diagrams mostly based on state-machine-like specifications similar to those of SDL [3]. When the MAS is a small one, when the problem is restricted to academic domain, such tools are really useful and a methodology of just a short number of steps sounds reasonable. However, an industrial development, involving a team of several engineers requires management of activities, a more detailed software development process, and the control of the cost involved in making a MAS work under real world workload.

To deal with MAS development problems, specially in analysis and design phases, we need flexible tools and specification methods that permit researchers develop MAS as if they were developing conventional software. And this does not mean that there is a fixed target platform which has to be instantiated with some parameters. Target platform is important but should not suppose a constraint for a methodology. With this assumption, MESSAGE methodology [1] proposes to integrate well-proven object-oriented software engineering practices with agent technology results.

MESSAGE approach

MESSAGE bases the definition of the methodology in the use of meta-models as specification formalism. Meta-modeling languages make possible the extension of the core specification by using built-in generalization mechanisms. Also, instances of the meta-models (i.e., the models that describe the MAS), opposite to most formalisms, can be detailed partially and refined through successive steps (similarly to UML, where class diagrams do not need to be complete from the beginning, but can grow in details during the development). Furthermore, the use of meta-modeling tools, such as METAEDIT+ [5] in our experimentation, makes possible to perform analysis and design as in object-oriented applications using Rational Rose or TogetherJ, but instead of using UML, with MAS specific notation.

Though meta-models help as a notation for the specification of a MAS, they are not yet the complete solution, since the generation of models from meta-models is not trivial. This requires the definition of a development process, and MESSAGE has adopted the Rational Unified Process [6], and has defined a set of activities for analysis and design phases, that consider the iterative and incremental nature of the process. Furthermore, there are dependencies among the different meta-models that add difficulties to the development process. Though this was not addressed in MESSAGE, we have been working on the

identification of these dependencies to achieve a consistent MAS specification. Such consistency checking activities are grouped together with model generation activities in activity diagrams. And all activities are managed inside the unified software development process. This solution has two advantages: working with activities facilitates management of a project and also the activities act themselves as guidelines for users of the methodology. Our experience is that a realistic MAS development can easily involve around seventy activities. So many activities may be an argument to discredit the methodology, but in fact we should consider that activities are defined for each view of the system (there are five in MESSAGE: organization, agent, tasks/goals, interactions, and domain), and these views are normally developed in parallel (because of their relationships). Also, software engineers could circumvent the application of some activities when dealing with simple application domains.

Conclusions

The definition of an agent oriented software engineering methodology is not an easy task. Despite the first impression that existing MAS methodologies can provide, following a methodology is not as simple as a six or seven steps process. The MESSAGE approach starts by identifying first the elements required to build a MAS, organizing them in five views, and expressing them in five meta-models. These meta-models can be applied by integration in a well-proven software development process model, which in the case of MESSAGE is the Unified Software Development Process, but others could be also considered. Finally, a methodology should not be tied with a concrete implementation model, although, to be realistic, it should demonstrate its application in the definition of implementations over different MAS platforms.

References

[1] Caire, G., Leal, F., Chainho, P., Evans, R., Garijo, F., Gomez-Sanz, J. J., Pavon, J., Kerney, P., Stark, J., and Massonet, P. Eurescom P907: MESSAGE - Methodology for Engineering Systems of Software Agents. http://www.eurescom.de/public/projects/P900-series/p907/default.asp . 2002.

[2] DeLoach, S. Analysis and Design using MaSE and agentTool. 2001. Proceedings of the 12th Midwest Artificial Intelligence and Cognitive Science Conferece (MAICS).

[3] International Telecommunication Union. ITU 100:Formal Description Techniques (FDT)- Specification and Description Language (SDL). Z.100. 99. Series Z: Languages and General Software Aspects for Telecommunication Systems.

[4] Kruchten, P., *The Rational Unified Process* Addison-Wesley, 1999.

[5] Lyytinen, K. S. and Rossi, M. *METAEDIT+ --- A Fully Configurable Multi-User and Multi-Tool CASE and CAME Environment*. LGNS#1080. 1999. Springer-Verlag.

[6] Nwana, H. S., Ndumu, D. T., Lee, L. C., and Collis, J. C., "ZEUS: A Toolkit for Building Distributed Multi-Agent Systems," *Applied Artificial Intelligence Journal*, vol. 1, no. 13, pp. 129-185, 1999.

[7] Wood, M. and DeLoach, S. Developing Multiagent Systems with agentTool. 2000. ATAL 2000. LNAI 1986. Castelfranchi, C. and Lespérance, Y.

[8] Jacobson, I., Booch, G., Rumbaugh J., The Unified Software Development Process, Addison-Wesley, 1999.