agentTool III: From Process Definition to Code Generation

Juan C. Garcia-Ojeda Scott A. DeLoach Robby

Kansas State University
234 Nichols Hall, Manhattan
KS, 66506 - USA
+1-(785)532-6350

{jgarciao, sdeloach, robby}@ksu.edu

ABSTRACT

The agentTool III (aT^3) development environment is built on the Eclipse platform and provides traditional model creation tools to support the analysis, design, and implementation of multiagent systems following the Organization-based Multiagent Systems Engineering (O-MaSE) methodology. It also provides the ability to compose, verify, and maintain customized O-MaSE complaint processes. Additionally, aT^3 provides a verification framework which helps designers to maintain consistency between their O-MaSE models.

Categories and Subject Descriptors

D.2.2 [**Software Engineering**]: Design Tools and Techniques – Computer-aided software engineering (CASE).

General Terms

Design.

Keywords

Agent-oriented Software Engineering, Multiagent Systems

1. INTRODUCTION

The agentTool III (aT^3) development environment supports the creation of multiagent systems following the *Organization-based Multiagent Systems Engineering* (O-MaSE) Process Framework [2]. aT^3 is a successor to the original agentTool that was developed in 2000 – 2001 at the Air Force Institute of Technology and is currently a project of the Multiagent & Cooperative Robotics Laboratory[1] (MACR) at Kansas State University. aT^3 was developed in Java and built on top of the Eclipse[2] platform and the Eclipse Process Framework[3] (EPF).

The goal of aT^3 is to support and enforce O-MaSE, which allows users to create their own customized development processes. O-MaSE has three structures – a metamodel, a set of methods fragments, and a set of guidelines – that enable users to use method engineering techniques to create custom processes. The O-MaSE *metamodel* defines the key concepts needed to design and implement multiagent systems while the O-MaSE *method fragments* include the actual tasks that are performed and the work products that are produced. The O-MaSE *guidelines* define how the method fragments are related to one another and thus how customized processes can be built.

2. aT^3 COMPONENTS

aT^3 has four components that are integrated into a single tool. These components are the graphical editor, the process editor, the verification framework, and the code generation facility.

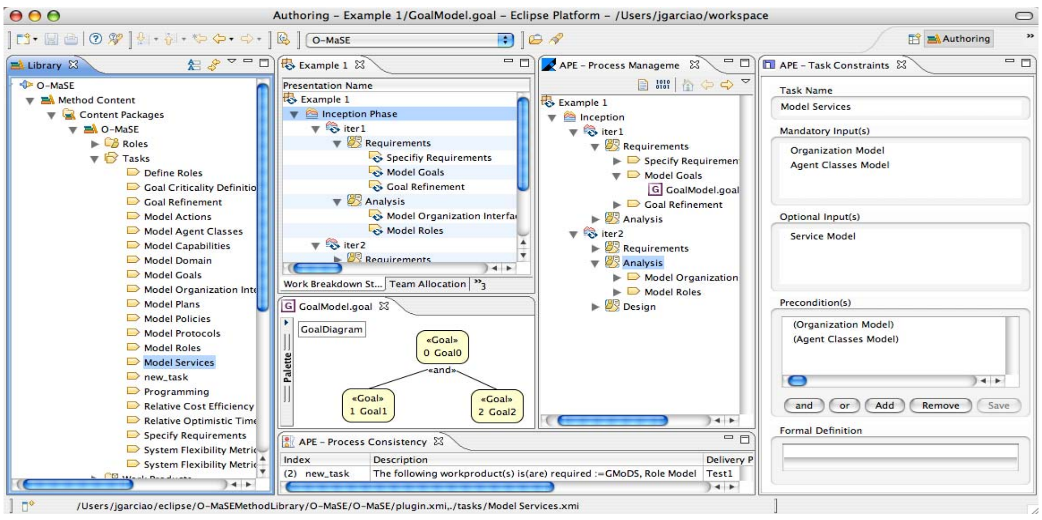
Graphical Editor. aT^3 supports the graphical editing of models that define all the concepts defined in the O-MaSE's metamodel. aT^3 supports drag-and-drop addition of icons and ensures the only appropriate connections are made between the various kinds of icons. aT^3 also provides pop-up panels for editing the internal detail of the various concepts such as parameters, etc. The Graphical Editor supports the following O-MaSE models.

The *Goal Model* allows designers to capture the purpose of the agent organization using and AND/OR goal tree structure, supplemented with relations for dynamic concepts such as goal triggering and precedence. The *Organization Model* shows the relationship of the system under development to external users. The *Role Model* defines the type of roles in the organization and the goals they are designed to achieve. Role Models also include the notion of interaction protocols. The *Domain Model* contains the definition of the environment in which the multiagent system is situated, which is defined in terms of a set of environment objects their relationships.

The *Agent Class Model* defines the agent classes and sub-organizations that populate the organization. Agent Class Models include agents, actors, organizations, roles, capabilities, and protocols and the various relationships between them. The *Protocol Model* is similar to AUML Interaction Diagrams and is used to describe sequences of messages sent between, roles, organizations, and external actors. The *Capability Model* captures the abilities of an agent, which can include algorithmic or access capabilities, or more hardware like capabilities such as sensors and effectors. Capabilities are defined in terms of actions that are used by the plan model. The *Agent Plan Model* is a Finite State Automata-based model that describes how a role/agent can achieve a goal. The Plan Model uses actions defined in the Capability Model to perform its basic actions. The *Policy Model* contains a formal definition of all the policies applicable to the system. The Policy Model Editor uses policy patterns to help user specify policies appropriately.

[1] http://macr.cis.ksu.edu/
[2] http://www.eclipse.org/
[3] http://www.eclipse.org/epf/

Cite as: AgentTool III: From Process Definition to Code Generation, Juan C. Garcia-Ojeda, Scott A. DeLoach, Robby, *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009),* Decker, Sichman, Sierra and Castelfranchi (eds.), May, 10–15, 2009, Budapest, Hungary, pp. 1393 – 1394



Task Name: Model Services

Mandatory Input(s): Organization Model; Agent Classes Model

Optional Input(s): Service Model

Precondition(s): (Organization Model) (Agent Classes Model)

Index	Description	Delivery P
(2) new_task	The following workproduct(s) is(are) required :=GMoDS, Role Model	Test1

/Users/jgarciao/eclipse/O-MaSEMethodLibrary/O-MaSE/O-MaSE/plugin.xmi,./tasks/Model Services.xmi

Figure 1. agentTool Process Editor

Process Editor. The aT^3 Process Editor (APE) allows process engineers to compose agent-based customized processes [3]. APE provides five basic structures: a Method Fragment Library, the Process Editor, a set of Task Constraints, a Process Consistency checker, and a Process Management tool as shown in Figure 1. The *Library* is a repository of O-MaSE compliant method fragments, which can be extended by APE users. The *Process Editor* allows users to create and maintain O-MaSE compliant processes. The *Task Constraints* modeler helps process engineers specify guidelines to constrain how tasks can be assembled, while the *Process Consistency* mechanism verifies the consistency of custom processes against those constraints. Finally, the *Process Management tool* provides a way to measure project progress using Earned Value Analysis. For more details see [3].

Verification Framework. The aT^3 Verification Framework gives designers a way to maintain consistency between their O-MaSE models, based on a predefined rule set. Since processes are customized, this rule set can also be customized by turning on and off certain rules. Each time a model is saved, the Verification Framework checks that document against all related documents in the current project based on the currently enabled rules. Verification problems are show to the user through the Eclipse Problems panel similar to compiler errors and warnings.

Code Generation Facility. Automatic code generation is also available in aT^3. Currently, the only platform targeted has been JADE [1]. However, we have created a framework consisting of the Organization, Operation, Social, and Environment levels. At the Organization level, agents and roles are chosen for achieving specific goals. At the Operation level, agents achieve goals by performing actions based on their available capabilities. At the Social level, agent's interactions are captured via messaging, while at the Environment level, the knowledge of object types and relationships are generated. Due to the detail of the O-MaSE models, the aT^3 JADE generator is capable of generating 100% of the code necessary to create functional JADE systems.

3. Summary

This paper has given a quick overview of aT^3. The motivation behind our work is the fact that despite the documented advantages of the agent paradigm in the development of complex system, agent-based processes and CASE tools are still key factors in the successful application of agent-oriented techniques in software development. As discussed and when compared with other similar tools (i.e., IDK[4],PDT[5],TAOM4E[6], and APTK[7]) , aT^3 provides the ability to create custom agent-oriented processes (i.e., O-MaSE compliant), track and manage those processes, as well to generate 100% percent of the code necessary (following a consistent layered approach) to create functional JADE-based applications. Also, aT^3 provides the ability to graphically edit O-MaSE models; and, verify those models for consistency.

The aT^3 web site (http://agenttool.cis.ksu.edu/) contains current information on agentTool and installation instructions, user documentation, a complete set of tutorials, and related publications.

4. ACKNOWLEDGMENTS

Our thanks to Matthew J. Miller, Carol Fries, Stephen A. Odell, Kyle Hill, Patrick Gallagher, Binti Sepaha, and Deepti Gupta. This work was supported by grants of the US National Science Foundations (0347545) and the US Air Force Office of Scientific Research (FA9550-06-1-0058).

5. REFERENCES

[1] Bellifemine, F.L., Caire, G., & Greenwood, D. Developing Multi-Agent Systems with JADE. Wiley & Sons, England, 2007.

[2] Garcia-Ojeda, J.C., DeLoach, S.A., Robby, Oyenan, W.H., and Valenzuela, J.L. O-MaSE: A Customizable Approach to Developing Multiagent Development Processes. In AOSE VIII. LNCS 4951, 1-15. Springer. 2008.

[3] Garcia-Ojeda, J.C., DeLoach, S.A., and Robby. agentTool Process Editor: Supporting the Design of Tailored Agent-based Processes. To appear in ACM SAC. 2009.

[4] Robby, DeLoach S. A., and, Kolesnikov, V. A. Using Design Metrics for Predicting System Flexibility. In FASE'06. LNCS 3922, 184-198. 2006.

[4] http://grasia.fdi.ucm.es/main
[5] http://www.cs.rmit.edu.au/agents/
[6] http://sra.itc.it/tools/taom4e/
[7] http://www2.pa.icar.cnr.it/icar2/