agentTool Process Editor: Supporting the Design of Tailored Agent-based Processes

Juan C. Garcia-Ojeda
Department of Computing and Information Sciences
234 Nichols Hall, Manhattan, KS. 66506 – USA.
+1-(785)532-6350
jgarciao@ksu.edu

Scott A. DeLoach
Department of Computing and Information Sciences
234 Nichols Hall, Manhattan, KS. 66506 – USA.
+1-(785)532-6350
sdeloach@ksu.edu

Robby
Department of Computing and Information Sciences
234 Nichols Hall, Manhattan, KS. 66506 – USA.
+1-(785)532-6350
robby@ksu.edu

ABSTRACT
This paper describes the agentTool Process Editor (APE), an Eclipse plug-in based on the Eclipse Process Framework. The aim of APE is to facilitate the design, verification, and management of custom agent-oriented software development processes. APE provides five basic structures. The *Library* is a repository of agent-oriented method fragments. A *Process Editor* allows the management of tailored processes. *Task Constraints* help process engineers specify guidelines to constrain how tasks can be assembled, while a *Process Consistency* mechanism verifies the consistency of tailored processes against those constraints. Finally, the *Process Management* integrates APE with the agentTool III development environment and provides a way to measure project progress using Earned Value Analysis.

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

General Terms
Documentation, Design, Standardization, Verification.

Keywords
agentTool Process Editor, O-MaSE, Method Engineering, agentTool III, CASE tool

1. INTRODUCTION
Processes (i.e., methodologies) for developing agent-based systems and their supporting tools are a key factor to enabling success in multiagent system development projects. Currently, while there are many different agent-based processes [15], their applicability is limited. One reason for this limited applicability is that many of these processes are rigidly defined and not flexible or adaptable. They tend to focus on specific types of multiagent systems, or try to cover everything. Therefore, they are either too specific or too general and often they do not fit the projects requirements. In practice, we need to adapt the processes or build the new processes for each new project or project family.

Method Engineering holds promise for creating such tailored processes [4]. In essence, method engineering allows process engineers to construct processes (i.e., methodologies) from a set of method fragments, which are stored in a repository. Thus, to create a new process, process engineers select the most appropriated method fragments from the repository and assemble them into a complete process based on project requirements.

Nonetheless, most of the tools that support process assembly and customization do not offer support for agent-based processes. For instance, a set of Computer Aided Method Engineering (CAME) tools such as Decamerone [14], MENTOR [30], MetaEdit+ [16], and MethodBase [29] have been developed to support information systems development processes in terms of process administration, process assembly, and process generation. Other examples include Xome and the Eclipse Process Framework[1] (EPF). Xome provides a non-specific modeling environment; and, a collection of tools for creating, editing, and interchanging models at any abstraction slevel [12]; and, EPF provides more general tools for software process engineering: method and process authoring, library management, process configuration, maintenance, implementation, and publication.

In this paper, we present the agentTool Process Editor (APE), a tool for creating and customizing processes for multi-agent system development. APE is an Eclipse-based[2] plug-in which uses EPF to facilitate the management of tailored agent-based processes based upon the O-MaSE Process Framework (hereafter, simply referred to as O-MaSE) [11]. O-MaSE provides the concepts, rules, methods fragments, and guidelines needed by process engineers to assemble O-MaSE compliant processes, while APE provides the infrastructure to develop, verify, and maintain O-MaSE compliant processes. The rest of the paper is organized as follows. Section 2 discusses the basis

SAC'09, March 8-12, 2009, Honolulu, Hawaii, U.S.A.

[1] http://www.eclipse.org/epf/
[2] http://www.eclipse.org/

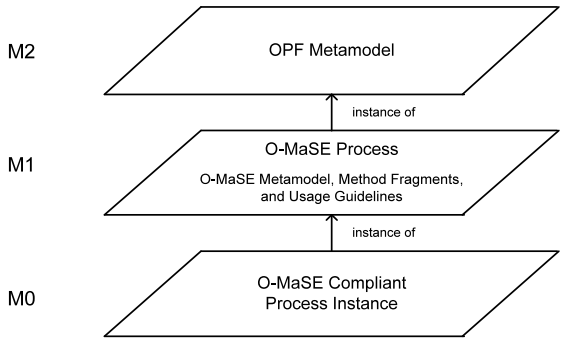

Figure 1. O-MaSE Process Framework

of APE. Section 3 describes APE functionality in detail. Section 4 describes related work while Section 5 presents conclusions and areas for future work.

2. OVERVIEW

2.1 O-MaSE

O-MaSE is a new approach in the analysis and design of agent-based systems. It is adapted from the OPEN Process Framework (OPF) [9] and takes some useful concepts from the MaSE methodology [8]. The aim of O-MaSE is to allow designers to create customized agent-based software development process based on project-specific requirements. O-MaSE uses an integrated meta-model based framework that allows designers to select method fragments from a repository and construct custom processes using construction and tailoring guidelines [9]. Such a meta-model based framework is supported by a three-layer schema as shown in Figure 1. The M2 layer uses the OPF meta-model, which contains seven meta-elements: stages, work units, producers, work products, and languages. A *stage* is defined as a "formally identified and managed duration within the process or a point in time which some achievement is recognized" [9, pp. 55]. Stages are used to organize *work units*, which are defined as operations that are carried out by a *producer*. There are three kinds of work units in OPF: activities, tasks, and techniques. *Activities* are a collection of tasks. *Tasks* are small jobs performed by one or more producers. *Techniques* are detailed approaches to carrying out various tasks. Producers use techniques to create, evaluate, iterate, and maintain work products. *Work Products* are pieces of information or physical entities produced (i.e., application, document, model, diagram, or code) and serve as the inputs and the outputs of work units. Work products are documented in appropriate *languages*.

The M1 layer serves as repository of method fragments instantiated from the M2 meta-model. A set of rules governing the relationship between the process-specific meta-model and the various reusable method fragments is also defined in M1. Process engineers use these guidelines to extend, instantiate, and tailor the method fragments for creating custom processes in the M1 layer. These custom processes are then instantiated at the M0 level on specific projects; the actual custom process as enacted on a specific project is termed a process instance. Next, we explain each component depicted in the M1 layer.

2.1.1 O-MaSE Processes

As mentioned before, O-MaSE processes are constructed by selecting methods fragments from a repository and assembling them into a complete process based on project requirements and following the O-MaSE guidelines. In O-MaSE these methods fragments are stored in the M1 layer. In the remainder of this section, we describe the three O-MaSE's components contained in the M1 layer: the meta-model, the set of method fragments, and the set of guidelines.

2.1.1.1 Metamodel

The meta-model defines the key concepts needed to design and implement multiagent systems following an organization-based approach. That is, an organization encapsulates a set of goals, roles, agents, capabilities, policies, within a domain (refer to [11]).

A *Goal* defines the overall function of the organization. That is, it could be seen as a desirable situation [28] or the objective of a computational process [31]. On the other hand, a *Role* defines a position within the organization whose behavior is expected to achieve a particular goal or set of goals. An *Agent* can be a human or an artificial (hardware or software) entity that perceives its real world and can perform actions on it. Those actions can be either perceived or carried out by means of capabilities. Thus, a *Capability* defines the percepts/actions the agent has at its disposal. These capabilities can be seen as an aggregation of two concepts: plans (i.e., soft capabilities) and actions (i.e., hard capabilities). *Plans* represent algorithms used by agents to carry out specific computational tasks, while *Actions* allow agents to sense and effect objects in the environment.

On the other hand, the real world where the agent perceives/acts is captured by means of the Domain Model. The *Domain Model* defines the type of objects in the environment and the relations between them. The domain model is also used to define organization *policies*, which describe how an organization may or may not behave in a particular situation. Finally, the meta-model also introduces the concepts of *Organization Agent* to represent organizations that functions as agents in a higher-level organization and *Protocols* that capture the interaction between the organization and *External Actors* or between agents in the organization.

2.1.1.2 Method Fragments

As described earlier, method fragments are operations or tasks carried out by a producer (i.e., software developments team member) to produce a set of work products, which may include either models, documents, or code. The method fragments are defined in terms of stages, work units (i.e., activities, tasks, and techniques), work products, producers and languages. The major method fragments in O-MaSE are shown in Table 1. Due to the page limit, we cannot discuss each of them separately; however, to illustrate our basic approach, we describe the details of the Requirements Engineering activity.

In the *Requirement Engineering* activity, we seek to translate systems requirements into system level goals. Thus, the *Model Goal* task focuses on transforming system requirements into a system level goal tree. Also, the *Goal Modeler* must be able to: (1) use the AND/OR Decomposition technique, (2) understand

Table 1. O-MaSE Method Fragments [11]

Work Units		
Activity	**Task**	**Work Products**
Requirement Engineering	Model Goals	Goal Model
	Goal Refinement	Goal Model for Dynamic Systems (GMoDS)
Analysis	Model Organizational Interfaces	Organization Model
	Model Roles	Role Model
	Model Domain	Domain Model
Design	Model Agent Classes	Agent Classes Model
	Model Protocols	Protocol Model
	Model Plans	Agent Plan Model
	Model Policies	Policy Model
	Model Capabilities	Capability Model
	Model Actions	Action Model

the *System Description (SD)* or *System Requirement Specification (SRS)*, and (3) interact with domain experts and customers. As result, a *Goal Model* is obtained.

2.1.1.3 Guidelines

The guidelines define how the method fragments are related to each other. In O-MaSE, guidelines are specified in terms of constraints (captured by means of preconditions and post-conditions) related to work units and work products. The guidelines are formally specified as a tuple <Input, Output, Precondition, and Post-condition>. *Input* is the set of work products that may be used in performing a work unit while *Output* is the set of work products that may be produced from a work unit. The *Precondition* specifies the work unit/producer states and the *Post-condition* specifies the work product state that guaranteed to be true after successfully performing a work unit if the precondition was satisfied. For instance, we can specify the guideline for the task Goal Refinement as follows:

<GM, RG, (completed (<GM, n, m>) ∧ available (GMP)), exist (<RG, n, m>)>

This guideline formally states that the mandatory input for the Goal Refinement task is a completed Goal Model (GM) and the output is the GMoDS (RG) work product. In addition, the Goal Modeler Producer (GMP) must be available at the time where the task is started.

2.2 Eclipse Process Framework

The Eclipse Process Framework (EPF) is an open source project supported by the Eclipse Foundation. The goals of the EPF project are to provide (1) a complete set of tools for process software engineering and (2) a complete and extensible process content for a wide range of software development and management processes such as the OpenUP/Basic software process.

Using the *EPF Composer*, process designers can construct their own software development processes by structuring them according to a predefined schema. In EPF, such schema is referred as the Unified Method Architecture (UMA). UMA is an evolution of the Software and Systems Process Engineering Meta-model 2.0 (SPEM 2.0) and supports the organization of wide variety of different types of method fragments and processes beyond traditional software engineering fragment types and processes.

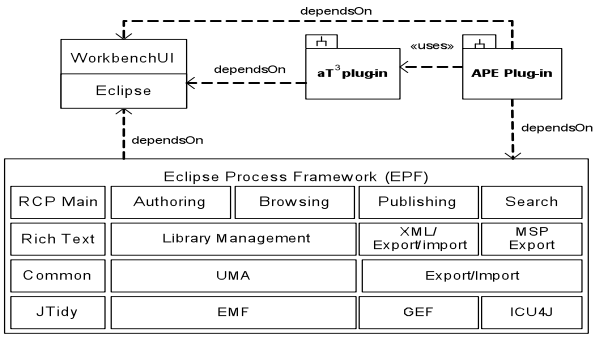

Figure 2. APE Architecture

3. agentTool Process Editor (APE)

The overall goal of APE is to facilitate the design, verification, and management of customized agent-based O-MaSE compliant processes. APE is an Eclipse-based plug-in built on top of the Eclipse Process Framework, which in turn, relies on the Eclipse Modeling Framework (EMF), the Graphical Editing Framework (GEF), the Web Standard Tools (WST), and the Common Properties User Interface (UI). It also uses the open source components JTidy and Lucene along with specific EPF tool components such as UMA services[3] (see Figure 2).

Figure 3 shows a screenshot of APE. APE can be seen as an aggregation of five basic views: a Library view (Window 1), a Process Editor (Window 2), a Task Constraints view (Window 3), a Process Consistency view (Window 4), and a Process Management view (Window 5). It is important to note that *Window 1* and *Window 2* use base EPF functionality while the rest of the windows were developed specifically for APE as an Eclipse plug-in. APE is also loosely integrated with the agentTool III[4] (aT[3]) –development environment (Window 6). The integration of APE and aT[3] allows designers to analyze, design, and implement multiagent systems while following the process as defined in APE. Next, we introduce the main features of APE.

3.1 Features of the Tool

Using APE, we seek to enable a more effective way for designing, verifying, and managing agent-based O-MaSE compliant processes, by focusing on the following features.

3.1.1 Customizing O-MaSE Content

As mentioned before, APE directly uses EPF components such as EPF Authoring. EPF Authoring provides the basic tools required for process engineers to manage method content elements such as roles, tasks, work products, guidance, content categories and processes.

[3] http://www.eclipse.org/epf/composer_architecture/
[4] http://agenttool.cis.ksu.edu/

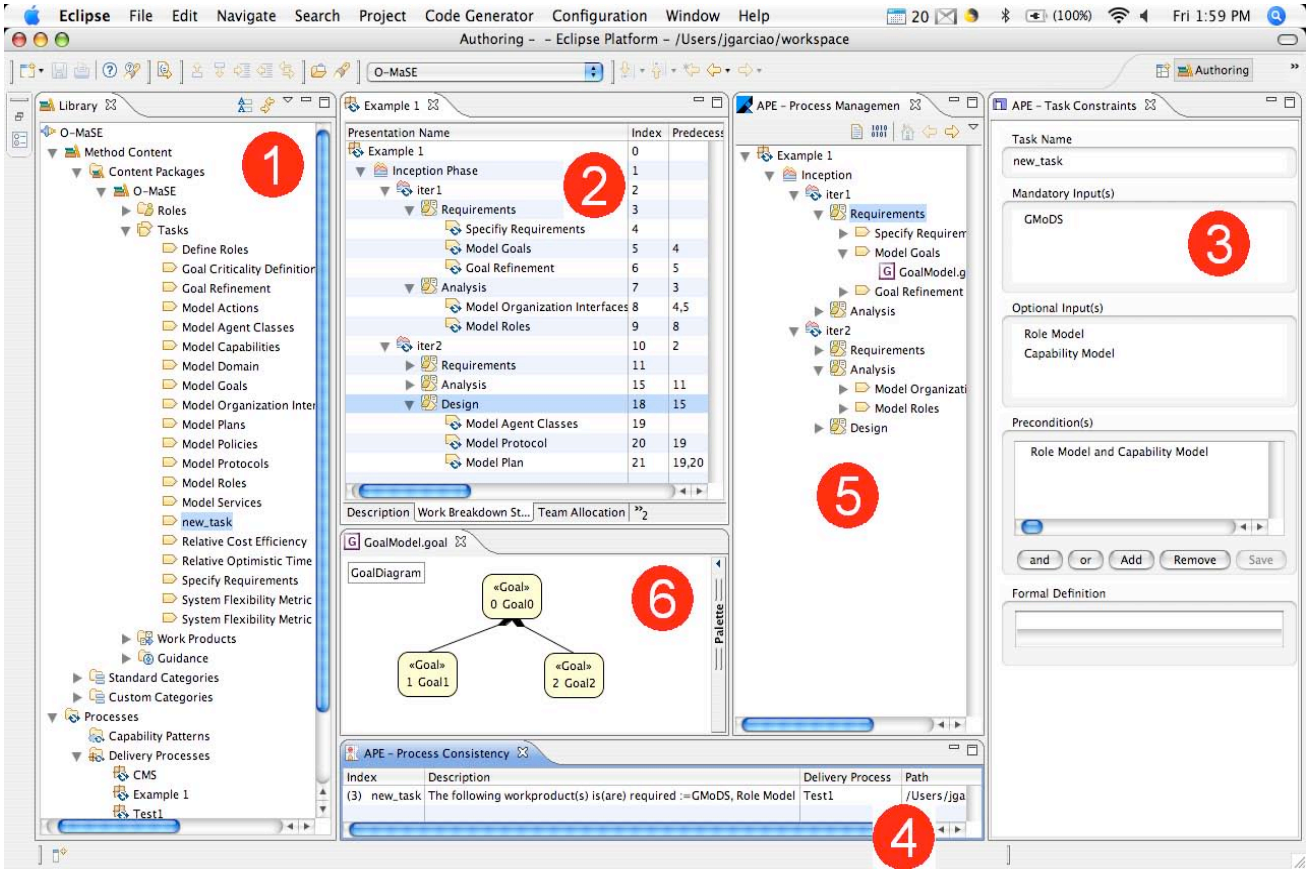

Figure 3. Screenshot of APE

For instance, if a development team may need to perform tasks that are not defined in the current O-MaSE method fragment repository (see Window 1), process engineers simply right-click on the label "Tasks" and then specify the main attributes of the new tasks (i.e., general description, steps, roles, work products, and guidance). For example, in Window 1, we show the result of adding the *new_task* task in the repository.

Because processes should be assembled logically, process engineers can also specify how method fragments are related to one another by adding/modifying task pre- and post-conditions. To modify these conditions, the process engineer selects a task from Window 1 and modifies an existing condition or specifies new conditions in Window 3. Preconditions are specified by selecting work products from the *Optional Inputs(s)* list in Window 3, and including them in the logical formula (e.g., (*work_product_1* ∧ *work_product_2)* ∨ *work_product_3*). APE automatically verifies the precondition before it is added into the *Precondition(s)* list. This verification was developed using Another Tool for Language Recognition[5] (ANTRL). Window 3 shows the precondition (i.e., *Role Model and Capability Model*) specified for the *new_task* task.

The purpose of pre- and post-conditions is to help process engineers in the process of verifying the consistency of their customized processes. APE also provides a mechanism for automatically verifying customized processes (Window 2). Any problems found during this verification steps are displayed as processing inconsistencies in Window 4. In this case, adding the new_task task to the Test1 process (not shown) results in an error since the *new_task* task was inserted before the *Goal Refinement* task, resulting in the inconsistency shown in Window 4.

3.1.2 Engineering Custom Processes

In Window 2, the software development team – perhaps supervised by a process engineer and/or a project manager – has the challenge of creating a "tailored" or "customized" process. Such processes define sequences of tasks, which are performed by roles that produce specified work products. These processes are used to describe the desired lifecycle for a specific project (e.g., waterfall, incremental, iterative, etc.).

To create a new custom process in APE, process engineers must select the "Process" label, then right-click on the "Delivery Processes" label (see Window 1), and finally, specify a name for the process and choose O-MaSE as default method library. Once the new process has been created, the process designer can add children (phases, iterations, activities, tasks, or milestones) to it in Window 2. Children are added by right clicking on the process (or other child) name in Window 2 and then selecting a phase, iteration, or task from the method library.

[5] http://antlr.org/

3.1.3 Managing Custom Processes

To help manage custom processes, APE provides three basic functionalities via the Process Management view shown in Window 5. First, it provides the function for uploading documentation regarding the requirements for the given project. Second, it integrates aT^3, so managers and developers can directly view and edit the work products expected for each task. Finally, APE also provides support for entering and tracking budget and scheduling information using Earned Value Analysis (EVA) [10]. The core concept of EVA is *earned value*, which refers to the cost of work performed at a given point according to a process development plan [1]. Thus, the foundation of the EVA is a good work breakdown structure with clearly defined tasks (see Window 2), each of which has been assigned a cost and a task completion date.

3.2 Running Example

To demonstrate APE, we walk through the development of a cooperative multiagent system. We assume that a development team has been hired to develop a typical agent-based system where several agents playing different roles must cooperate in order to achieve some specific goals.

Window 2 in Figure 3 shows the process "Example 1" for developing the system. The process is structured as follows. The inception phase helps analyst to clarify the scope, project objectives, and feasibility of the intended solution. The inception phase would help an analyst to: (i) understand what to build, (ii) identify the key system functionality, (iii) determine at least one possible solution, and (iv) understand the risk associated with the project (i.e., cost and schedule) [16]. In our example, the inception phase has two iterations. In iteration 1 (iter 1) we specify two activities: Requirements and Analysis. These two activities help analyst to determine the scope and key functionalities of the system. In the Requirements activity we include three tasks: Specify Requirements, Model Goals, and Goal Refinement.

The Specify Requirements task allows analysts to capture the essence of the system by describing high-level requirements and design constraints. The Model Goals task allows goal modelers to transform system requirements into a goal tree. In this task, the goal modeler identifies the goals of the system from the initial specification, decomposes the goals into lower-level goals, and logically arranges those goals using AND-OR decomposition [31]. Finally, the Goal Refinement task helps goal modelers capture the dynamism of the system by identifying precedence and triggering relations between goals and adding parameters to goals.

The Analysis activity includes two tasks: Model Organizational Interfaces and Model Roles. In the *Model Organization Interfaces* task, modelers identify external actors, capture interactions between external actor and the system, and identify the goal to be achieved by the system. In the *Model Roles* task, role modelers identify the roles required to achieve the system goals, the interactions between roles, and the capabilities required to play each role.

Finally, in Iteration 2 (iter 2), the Requirements and Analysis activities are repeated with the aim of clarifying the results obtained from Iteration 1 with stakeholders and end-users, and making necessary adjustments. In addition, in Iteration 2 we include the Design activity whose aim is to determine at least one possible solution and the feasibility of such solution. The Design activity includes three different tasks: Model Agent Classes, Model Protocols, and Model Plans. The aim of *Model Agent Classes* is to map roles to agent classes, identify the capabilities possessed by agents, and define the protocols between agents. In the *Model Protocol* task, the protocol modeler identifies interactions between agents/roles and documents the details of the messages in the protocol. Finally, in the *Model Plan* task, plan modelers define the actions necessary to achieve the goals, and the messages sent and received by an agent carrying out the plan.

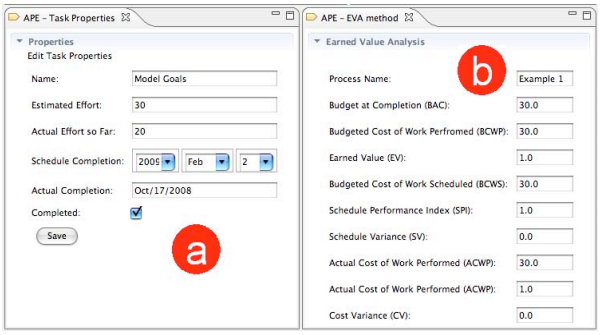

Figure 4. Earned Value Analysis Support in APE

After customizing the process, the process engineer delivers the process to the development team. The project manager can manage the custom process by means of the Process Management view (as explained in Section 3.1.3). Window 5 shows the structure of the process *Example 1*, which includes the Goal Model work product (i.e., GoalModel.goal). Managers and developers can invoke the associated aT^3 Diagram Editor directly by clicking on the GoalModel.goal file name as shown in Window 6. Currently, aT^3 supports all work products depicted in Table 1.

By interacting with Window 5, project managers can assign cost and task completion data to compute EVA numbers. Every time the project manager selects one of the tasks displayed on the Window 5, the *Task Property View* is shown (see Figure 4, part a). In this view, project managers can update the data related to the associated cost in the accomplishment of a given task (i.e., *Estimated Effort to*), the money spent in its accomplishment (i.e., Actual Effort so far), the expected date of its completion (i.e., *Schedule Completion*), and the current task completion date (i.e., *Actual Completion*). Finally the project manager can save the information and produce the *EVA Metrics View* as shown in Figure 4, part b. The EVA Metrics View shows the current project indicators, which include the *Budgeted Cost of Work Scheduled* (BCWS), the *Actual Cost of Work Performed* (ACWP), and the *Budgeted Cost of Work Performed* (BCWP). For a detailed description of the indicators see [10].

Thus, APE facilitates the design, verification, and management of customized agent-based O-MaSE compliant processes. First APE allows process engineers assembling logical and organized tool for creating custom processes. This is provided by means of the method fragment repository (Window 1) and the process editor (Window 2). Notice that in our example, only the 40% (8 tasks out of 20) of the tasks stored in the repository were instantiated to create the custom process for our example. This

clearly demonstrates how customization can be used to eliminate unnecessary work products, which in turn reduces schedule and cost. APE also provides a mechanism to allow process engineers to validate the consistency of the process via the *Process Consistency* view (see Window 4). The Process Consistency view automatically displays errors based on the formal guideline constraints defined for each method fragment. Finally, both project managers and developers can manage the custom process by means of the *Process Management* view (Window 5).

4. RELATED WORK

As mentioned before, there are several tools that support the design of non agent-based customized processes. For instance, MethodBase aims to facilitate method customization [29]. The MethodBase's repository consists of method fragments that can be selected and customized to fit specific project requirements. A commercial tool, MetaEdit+ provides three basic components for process customization: a repository management system, a method assembly system, and an environment generation system [16]. The first two systems are similar to functionality provided by APE, however, APE does not provide a mechanism for customizing associated CASE tools. In APE, if new tasks are defined and stored in the O-MaSE method fragment repository, additional functionality must be incorporated into aT^3. However, since aT^3 is built on the Eclipse development environment and aT^3's source code is available, this task is quite possible. Likewise, Decamerone supports the design of customized processes by maintaining a Method Base repository containing method fragments and their relationships. Additionally, Decameron, by means of Maestro II, provides project specific CASE tools support [14]. Also, MENTOR provides method fragment repository support and CASE tool support (similar to APE) [30].

Also, there are at least four groups working on tools that support the development of agent-based software applications: GRASIA (Grupo de Agentes Software: Ingenieria y Aplicaciones)[6], the RMIT Intelligent Agents Group[7], the Center for Scientific and Technological Research[8] (ITC-irst), and the Institute of High Performance Computing and Networking (ICAR)[9]. The GRASIA group has focused its effort in the development of the INGENIAS IDE (IDK). IDK provides a tool to support the INGENIAS methodology, notations and implementation techniques [23]. Basically, IDK has two main components: the Editor, and the Tools and Code generators. The Editor allows designers to generate specifications of multiagent systems using generic agent concepts. The Tools and Code generators facilitate rapid application development starting from the system specification through the implementation of the solution. On the other hand, the RMIT Intelligent Agents Group has developed what is called the Prometheus Design Tool (PDT). PDT supports the analysis and design of agent-based software systems following the Prometheus methodology [22]. PDT provides design diagram support, consistency checking, scoping, and entity propagation, hierarchical views, naming lookup assist, report generation, and diagram printouts [22]. Finally, PDT can be used in Eclipse by means of the Agent Development Tool Plug-in (ADPT). Likewise, ITC-irst developed TAOM4E (Tool for Agent Oriented visual Modeling for the Eclipse platform) [3] to support the analysis and design of agent-based software systems using the Tropos methodology. TAOM4E supports early requirements, late requirements, architectural design, and detailed design. Implementation and Test Cases is supported by means of JADE [2] and Jadex [24].

Although the agent-based tools discussed above provide useful functionality (e.g., consistency checking, entity propagation, implementation and test-cases support), none directly support the customization of their processes in order to fit specific needs of the organization, product and/or project. One exception is the PASSI project at ICAR, which has been actively involved in developing customized agent-oriented processes based upon the method engineering approach. For instance, the Agile PASSI Toolkit (APTK) [6] – an add-in based on the MetaEdit+ design tool [16] – supports the construction of customized agent-oriented processes based upon the modular representation of the PASSI methodology [5] and the method fragments extracted from it. Although APTK offers several features to designers (e.g., automatic compilation of diagrams, support of changes, consistency check, project documentation, patterns reuse, and automatic code generation), APTK has to be improved in order to fully support Agile PASSI-based processes (i.e., more flexibility and effective tool support).

Based on previous experiences (i.e., APTK and the PASSI Toolkit [5]), ICAR is working on the MetaMeth tool [7]. MetaMeth is a Computer-Aided Process Engineering (CAPE) tool which aim is to allow the design and instantiation of customized agent-oriented processes. MetaMeth supports two basic steps, Process Definition and Process Execution. In *Process Definition*, process engineers assemble customized processes as workflows. This is supported by the Enhydra JaWE[10] (Java Workflow Editor), a graphical workflow editor that allows process engineers to design tailored processes. Also, JaWE allows processes to be exported using the XML Process Definition Language (XPDL) format, which has been adopted by the Workflow Management Coalition (WfMC) for describing workflow processes. Also, in this step, process engineers define the rules on how to supervise the design activities. Such rules are specified by using JESS[11] (Java Expert System Shell). Finally, process engineers use Protégé[12] to specify and edit the multiagent system meta-model required by tailored processes. In *Process Execution*, customized processes are verified against its rules. This verification is done using Enhydra Shark[13], which ensures the design process instantiation and the execution of the different activities based on XPDL specifications. Also, JESS was used to build an expert system that provides services to the activity agents in order to assist designers in their task. These activity agents are developed on the JADE platform and are responsible for interacting with external editors used by designers to model the agent-based system. Finally, the Eclipse IDE is used to support the development of their UML editors.

While MetaMeth and APE provide much of the same type of support for the design of tailored agent-based processes, there

[6] http://grasia.fdi.ucm.es/main
[7] http://www.cs.rmit.edu.au/agents/
[8] http://sra.itc.it/
[9] http://www2.pa.icar.cnr.it/icar2/
[10] http://www.enhydra.org/workflow/jawe/index.html
[11] http://www.jessrules.com/jess/index.shtml
[12] http://protege.stanford.edu/
[13] http://www.enhydra.org/workflow/shark/index.html

are a few differences as well. Both MetaMeth and APE support a multiagent system-based fragment repository; MetaMeth supports the PASSI/Agile-PASSI based fragment repository [5][6] while APE supports the O-MaSE based fragment repository [11]. Processes are defined following a common breakdown structure or workflow. Both tools support iterative and incremental design processes. Processes are verified using logically specified semantic composition rules. One difference between MetaMeth and APE is the fact that MetaMeth supports reuse based on design patterns and code generation using the Model Driven Architecture approach [17], which transforms specific design models in templates that are then coded. On the other hand, APE allows process engineers to easily publish processes to websites, export the O-MaSE method library, and import other method libraries (e.g., OpenUP/Basic). Perhaps the most significant difference between APE and MetaMeth is the fact that APE uses the same infrastructure as its associated development environment, aT^3. This common framework allows the two tools to be integrated and share data easily. The use of a common infrastructure is significant since there is a need in the agent community to use common infrastructures rather than creating each new tool from scratch, thus boosting agent tool development and reducing implementation costs [18].

5. CONCLUSIONS AND FUTURE WORK

This paper has introduced APE. The main motivation behind this work is the fact that software development processes must be customizable to the needs of specific organizations, products or projects. APE provides the ability to

(a) create custom agent-based processes based on O-MaSE,
(b) maintain and update the O-MaSE method library,
(c) verify custom processes to ensure O-MaSE compliance,
(d) maintain and add new tasks and guidelines,
(e) manage processes in terms of the work products expected for each task,
(f) project cost and schedule performance; and,
(g) be fully integrated into the Eclipse and the aT^3 development environments.

Although we believe APE is headed in the right direction by supporting the construction of custom agent-based processes, there is considerable work to do before APE can be an industrial strength tool. First, although APE supports the construction of custom O-MaSE-based process, the addition of new user-defined tasks into the O-MaSE method fragment repository would likely require additional tool support. Thus, APE requires a robust approach for creating, maintaining, and transforming non aT^3-based work products. While Eclipse provides approaches for using external tools within Eclipse, this approach needs to be extended to allow APE to recognized and manage these work products. Second, while APE provides some metrics-based project management via EVA, we are currently working on introducing design-level software metrics that will predict system performance before implementation [26]. Third, we are continuing to formalize our process guidelines in order to avoid ambiguities between the metamodel and the method fragments used to assembly the agent-oriented applications. Finally, we are developing a code generation functionality for a variety of platforms as well as integrating the Bogor model-checking framework [27] for model verification.

6. Obtaining APE and aT^3

APE is packaged with aT^3 as an Eclipse plugin and is available for download from the aT^3 website[14]. The website includes download and installation procedures, as well as a complete set of online tutorials for most aT^3 and APE functions.

7. ACKNOWLEDGMENTS

This work was supported by grants of the US National Science Foundation (0347545) and the US Air Force Office of Scientific Research (FA9550-06-1-0058). Also, the authors express gratitude to Walamitien H. Oyenan for his valuable comments and suggestions.

8. REFERENCES

[1] American National Standards Institute. *Earned Value Management Systems*. Standard ANSI/EIA-748-A-1998, American National Standards Institute (ANSI), New York, NY, 1998.

[2] Bellifemine, F. L., Caire, G., and Greenwood, D. *Developing Multi-Agent Systems with JADE*. John Wiley & Sons Ltd, England, 2007.

[3] Bertolini, D., Novikau, A., Susi, A., and Perini, A. Technical Report, ITC-irst, 2006. TAOM4E: An Eclipse ready tool for Agent-based Modeling. Issue on the development process. Technical Report, ITC-irst, Trento, Italy, 2006.

[4] Brinkkemper, S., 1996, Method engineering: engineering of information systems development methods and tools, *Inf. Software Technol.*, 38, 4 (Apr. 1996), 275–280.

[5] Cossentino, M., and Potts, C. A CASE Tool Supported Methodology for the Design of Multi-agent Systems. In *Proc. of the International Conference on Software Engineering Research and Practice (SERP'02).* Las Vegas, USA, 2002.

[6] Cossentino, M., and Seidita, V. Composition of a New Process to Meet Agile Needs Using Method Engineering. In *Software Engineering for Multi-Agent Systems III*. Springer Verlag / Heidelberg, 2005, 36 – 51.

[7] Cossentino, M., Sabatucci, L., Seidita, V., and Gaglio, S. An Agent Oriented Tool for New Design Processes. In *Proc. of the Fourth European Workshop on Multi-Agent Systems.* Lisbon, Portugal, 2006.

[8] DeLoach, S. A. The MaSE Methodology. In *Methodologies and Software Engineering for Agent Systems. The Agent-based Software Engineering Handbook Series: Multiagent Systems, Artificial Societies, and Simulated Organizations*. Kluwer Academic Publishing, 2004, 107 – 124.

[9] Firesmith, D. G., and Henderson-Sellers, B. *The OPEN Process Framework: An Introduction.* Addison-Wesley, Harlow,England, 2002.

[10] Fleming, Q. W. and Koppelman, J. M. 2006 *Earned Value Project Management*. Project Management Institute.

[14] http://agenttool.cis.ksu.edu/

[11] 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 *Agent-Oriented Software Engineering VIII*. Springer Berlin / Heidelberg, 2008, 1–15.

[12] Gonzalez-Perez, C. Tools for an Extended Object Modelling Environment. In *Proceedings of the 10th IEEE international Conference on Engineering of Complex Computer Systems* (June 16 - 20, 2005). ICECCS. IEEE Computer Society, Washington, DC, 20-23.

[13] Gustafson, D. *Shaum's Outline of Software Engineering*. McGraw-Hill Professional, New York, NY, 2002.

[14] Harmsen, F. *Situational Method Engineering*. Moret Ernst & Young Management Consultants, 1997.

[15] Henderson-Sellers, B, and Giorgini, P. *Agent-Oriented Methodologies*. Idea group Inc. Hershey, PA, 2005.

[16] Kelly, S., Lyytinen, K., and Rossi, M. MetaEdit+ : A Fully Configurable Multi-User and Multi-Tool CASE and CAME Environment. In *Advanced Information System Engineering*. Springer Berlin / Heidelberg, 1996, 1–21.

[17] Kleppe, A., Warmer, J., and Bast, W. MDA Explained: The Model Driven Architecture – Practice and Promise. Adisson-Wesley. 2003.

[18] Kroll, P. *"The RUP: An industry-wide platform for best practices"*, The Rational Edge, http://www.ibm.com/developerworks/rational/library/873.html. Accessed on June 28th, 2008.

[19] Luck, M., McBurney, P. Shehory, O., and Willmott, S. Agent Technology: Computing as Interaction (A Roadmap for Agent Based Computing). Agent Link, 2005.

[20] Object Management Group. *"Software & Systems Process Engineering Meta-Model Specification – Version 2.0."* http://www.omg.org/docs/formal/08-04-01.pdf. Accessed on June 28th, 2008.

[21] Odell, J., Van Dyke Parunak, H., and Bauer, B. 2001. Representing agent interaction protocols in UML. In *First international Workshop, AOSE 2000 on Agent-Oriented Software Engineering* (Limerick, Ireland). Springer-Verlag New York, Secaucus, NJ, 2001, 121-140.

[22] Padgham, L., Thangarajah, J., and Winikoff, M. Tool Support for Agent Development using the Prometheus Methodology. *In Fifth International Conference on Quality Software (QSIC 2005)*. IEEE, 2005, 383–388.

[23] Pavon, J., Gomez-Sanz, J., and Fuentes, R. The INGENIAS Methodology and Tools. In *Agent-Oriented Methodologies*. Idea Group Inc., 2005, 236–276.

[24] Pokahr, A., Braubach, L, and Lamersdorf. Jadex: A BDI Reasoning Engine. In *Multi-Agent Programming*. Springer Science+Business Media Inc., USA, 2005, 149–174.

[25] Result Centre of LogicaCMG. *"OpenUP"* http://epf.eclipse.org/wikis/openup/openup/guidances/concepts/inception_phase_C4456871.html. Accessed on June 28th, 2008.

[26] Robby, DeLoach S. A., and, Kolesnikov, V. A. Using Design Metrics for Predicting System Flexibility. In *Fundamental Approaches to Software Engineering*. Springer Berlin / Heidelberg, 2006, 184–198.

[27] Robby, Dwyer, M.. B., and Hatcliff J. Bogor: A Flexible Framework for CreatingSoftware Model Checkers. In *Proceedings of the Testing: Academic & industrialConference on Practice and Research Techniques*. IEEE Comp Society, Washington, DC, 3–22.

[28] Russell, S., and Norvig, P. *Artificial Intelligence: A Modern Approach. 2nd ed*. Prentice Hall. Englewood Cliffs, NJ, 2003.

[29] Saeki, M., Iguchi, K., Wen-yin, K., and Shinohara, M. A Meta-Model for Representing Software Specification & Design Methods. In *Proceedings of the IFIP Wg8.1 Working Conference on information System Development Process*. North-Holland Publishing Co., Amsterdam, the Netherlands, 1993, 149–166. North-Holland, 1993.

[30] Si-Said, S., Rolland, C., and Grosz, G. MENTOR: A Computer Aided Requirements Engineering Environment. In *Advanced Information Systems Engineering*. Springer Verlag / Hielderberg, 1996, 22–43.

[31] van Lamsweerde, A., Darimont, R., and Letier, E. Managing Conflicts in Goal-Driven Requirements Engineering. *IEEE Transactions on Software Engineering, special Issue on Managing Inconsistency in Software Development 24, 11* (Nov. 1998), 908 – 92.