On The Use of Patterns in Multiagent System Engineering (MaSE) Methodology

Radziah Mohamad*, Safaai Deris*, Hany H. Ammar‡

*Faculty of Computer Science and Information Systems
Universiti Teknologi Malaysia, Malaysia
Email: radziah@fsksm.utm.my; safaai@fsksm.utm.my

‡ Computer Science and Electrical Engineering Department, West Virginia University
Morgantown, WV 26506, USA
Email: Hany.Ammar@mail.wvu.edu

ABSTRACT
The agent-oriented approach has been successfully applied to the solution of complex problems in dynamic open environments. However, to extend its use to industrial environments, some means of engineering agent-based applications is needed. Arguably, software methodologies provide one way to engineer such applications. Although several agent-oriented methodologies have been proposed to date, none of them is mature enough to be used in industrial environments. In particular, they typically don’t include a systematic process to apply patterns that are necessary for addressing issues of reuse and speed of development. Two possible approaches to overcome such weaknesses in current agent-oriented methodologies are: to propose new methodology or to enhance the existing ones. In this paper, the latter approach is taken, offering an enhancement of the MaSE methodology to include a systematic process to use patterns. MaSE is integrated with Pattern Oriented Analysis and Design (POAD) methodology to provide a systematic development process using patterns. “Agent Identification step” is also proposed by this paper to guide the agent identification process in the analysis phase. This paper will show that the integration of the POAD methodology to the MaSE methodology in the analysis phase could leverage the reuse level of the agent-based system.

KEYWORDS
Agent-based software engineering, multi-agent systems, pattern-oriented analysis and design

1. Introduction
Agent oriented software engineering is a powerful way of approaching large scale software engineering problems and developing agent-based systems. This approach to software engineering can significantly enhance our ability to model, design and build complex (distributed) software system [1]. We consider agent oriented software engineering as a layered technology which encompasses agent oriented software process model, agent-based system architecture, agent oriented methods and agent development tools. An overview of agent oriented methods is presented in [2]. The early versions of these methods have many weaknesses. Therefore, researchers are extending and improving these methods so that they can better guide developers to develop agent-based systems. One of the most developed agent oriented methods is Gaia [3]. It has been improved to model complex open system and to model aspects of Internet applications. The MaSE [4] methodology is another one which, takes an initial system specification, and produces a set of formal design documents in a graphically based style. The primary focus of MaSE is to guide a designer through the software lifecycle from a prose specification to an implemented agent system. This methodology has been improved in order to cover some aspects of multiagent systems such as mobility [5], ontology modeling [6] and modeling organizational rules [7].

This paper focuses on the MaSE analysis phase. This paper extends and improves the MaSE analysis methodology to bridge the gaps in the methodology analysis activities and artifacts. This paper proposes a “domain model” to capture the overall view of the domain problem to be solved. This paper also proposes the integration of MaSE and POAD [8] methodology
to provide a systematic development process using patterns. The remainder of this paper will report our MaSE improvement process and the results that we have obtained in this process. Section 2 describes our methodology improvement process and the methodology evaluation criteria. In Section 3, we describe the MaSE methodology and identified weaknesses in the MaSE analysis phase during development of our case study. In Section 4, we propose our improvement models and steps in order to remove identified weaknesses in the MaSE methodology. Section 5 contains a discussion of the results. Finally, Section 6 presents our conclusions.

2. Methodology Improvement Process

In order to improve the MaSE methodology, we propose a methodology improvement process based on survey on practices in methodology improvements by [5], [7] and [9]. The improvement process is shown in Figure 1. By using this process, first we defined a case study named Water Treatment Plant Report Management System (WTPRMS) [10] which could be implemented as a multiagent system. Then, we have evaluated methodology based on the resulted software artifacts that produced in the methodology phases and the multiagent Water Treatment Plant application which implemented based on these artifacts. During this process, we had some problems in many steps of the methodology and we needed some information to implement our case study which methodology did not address them. In our evaluation process, we considered some evaluation criteria, such as reusability, preciseness, expressiveness and smooth transition between the analysis processes. In the next sections, we report the results and our contributions that we have reached to them by following the improvement process.

3. MaSE Methodology Evaluation

The MaSE methodology has been researched at the Air force Institute of Technology for the last few years. Research focuses on developing a robust methodology for constructing multiagent systems. MaSE divides the development of multiagent systems into analysis and design phases [4]. The first phase, Analysis, includes three steps: Capturing Goals, Applying Use Cases and Refining Roles. The first step, Capturing Goals, takes user requirements and turns them to top-level system goals. Using system-level use cases and defining sequence charts in the Applying Use Cases step, an initial set of system roles and communications paths are defined. In the Refining Roles step, using the system goals and roles identified in the use cases, an initial set of roles is refined and tasks to accomplish each goal are defined. The design phase has four steps: Creating Agent Classes, Constructing Conversations, Assembling Agent Classes and System Design. In the first step, Creating Agent Classes, specific agent classes are defined to fill the roles defined in the analysis phase. Then, after determining the number and types of agent classes in the system, conversations between those agent classes are constructed and the internal components that comprise the agent classes are defined. At the end, the designer defines the number of individual agents, their locations and other system specific items. The MaSE methodology is a specialization of more traditional software engineering methodologies.
We evaluated MaSE methodology according to our proposed process. During this evaluation, our focus was on the reusing previous design experience requirements which the methodology does not address and smooth transition among different analysis steps. We also considered some criteria in our evaluation process such as maintainability and efficiency of the developed system as well. We have identified the following weaknesses during analysis of the case study.

a) Existing gap between Agent Identification process and domain modeling.

One of the most important weaknesses of the MaSE methodology analysis phase is the existing gap between the agent identification process and domain modeling. There is currently no model in the methodology to capture the domain problem. Domain modeling is needed to capture the overall view of the domain problem to be solved and to later systematically identify agents to be used in the system. Besides, there is no clear transition from the requirement steps to the agent identification process.

b) Lack of reusing previous design experiences.

Despite its benefits in multiagent systems design, MaSE does not build an agent system incrementally from well-documented design patterns, leading the developers to rediscover solutions to common design problems without benefiting from how they were resolved in the past. We propose to integrate the MaSE methodology with the POAD methodology to provide a systematic process using patterns.

4. Proposed Improvement and Extensions for the MaSE Analysis Phase

To improve and extend the MaSE methodology, we have added the domain analysis phase before the agent analysis phase. This is to give an overall view of the problem domain. The development of the domain model is following the POAD methodology. The idea is to reuse previous design experiences in developing the system. The integration of the POAD methodology to the MaSE methodology in this step could leverage the reuse level of the system. Figure 2 shows the extended steps and models for MaSE. In the figure, white boxes show the steps of the original methodology and the grey boxed show the extended steps and models. In the following subsections we will describe each extension.

![MaSE/POAD analysis steps and models](image)

Figure 2 MaSE/POAD analysis steps and models

4.1 Domain Analysis Phase

Domain modeling refers to a process on how to represent the domain where the agent resides being analyzed and how the agents being identified. We suggest that the domain analysis result is represented in the use case diagram, sequence diagram and class diagram following the object-oriented approach. This research proposed a pattern driven approach to construct the domain class diagram to reap a benefit of reusing previous design experience. The integration of MaSE analysis with Pattern-Oriented Analysis and Design (POAD) is suggested. POAD utilizes UML modeling capabilities to compose design patterns at various levels of abstractions. By integrating the POAD to the MaSE analysis phase, the analyst could reuse previous designers experience to develop the system. The introduction of an extensive pattern reuse practice can be determinant in cutting down the time and cost of developing these systems. The products of this phase are the system use case, system sequence and domain class diagram. Figure 3 illustrates an example of the class diagram for the two pattern instances of the WTPRMS.
5. Discussion

Patterns are extensively used to facilitate the development of software systems; in the agent-oriented approach they have been employed to design multiple aspects of an application. [13] presents a set of social patterns as part of the Tropos [14] methodology to describe the general architecture of a system under construction. [15] presents the design of a particular type of agent pattern immersed in the PASSI methodology. They defined a pattern as consisting of a model and an implementation code. The difference between both approaches with ours is that, they start to use pattern in the agent design phase while we start to use pattern in domain modeling phase. The integration of POAD to the MaSE analysis phase provides a systematic process to develop the domain architecture with patterns. We believe that by integrating patterns at the early stage of agent-based development could increase the reusability and maintainability of the analysis and design artifacts.

6. Conclusion and Future Works

Although several agent-oriented methodologies have been proposed recently, none of them is mature enough to develop commercial and industrial applications. One step towards achieving mature methodologies is to enhance the current ones with the inclusion of software engineering best practices. One of those best practices is the use of patterns in key parts of the analysis and design processes. This paper has presented the extension to the MaSE analysis methodology and illustrated it using the WTPRMS case study. The extension includes the integration of the POAD methodology to model the system. This paper also introduced the use of role pattern [12] to leverage reuse at the MaSE analysis phase. We have shown that the integration of patterns in the MaSE analysis phase could increase reusability of the methodology artifacts and maintainability of the developed system. Future research directions include formalization of the pattern framework for MaSE and systematic process to design agent using patterns. It is hoped that the inclusion of patterns to the methodology could help to leverage reuse of the methodology and increases the accessibility of the methodology since the inclusion of patterns makes the methodology easier to use, especially by non-expert users.
References


