Task Knowledge Patterns Reuse in Multi-Agent Systems Development

WaiShiang Cheah¹, Leon Sterling², and Kuldar Taveter³

¹ Faculty of Computer Science & IT, UNIMAS 94300 Kota Samarahan
Sarawak, Malaysia
c.waishiang@gmail.com
² Faculty of ICT, Swinburne University of Technology, Australia
lsterling@swin.edu.au
³ Department of Informatics, Faculty of IT, Tallinn University of Technology
kuldar.taveter@ttu.ee

Abstract. Template-based knowledge models can be viewed as design patterns for specifying a task [12]. The models can serve as reusable artifacts during the development of a multi agent system using the MAS-CommonKADS methodology. However, based on our observation of existing patterns, we note limitations of reusing those patterns in agent development. This paper presents task knowledge patterns that are described through our improved agent oriented template structure. The improved template structure presented in this paper provides an alternative approach to defining task knowledge patterns by incorporating a two dimensional view of agent oriented models. The task knowledge patterns introduced in this paper describe task knowledge in an agent context, while explicitly providing a description designed to encourage use and reuse in agent oriented software development. A demonstration of the reuse of task knowledge patterns in agent oriented modelling is presented in this paper. Specifically we show how a particular task knowledge pattern, selection of relevant source materials, can be used to rapidly prototype an adviser finder multi-agent system.

Keywords: agent-oriented modeling, task knowledge patterns, advisor finder.

1 Introduction

Agent technology has been used in building various domain specific applications. However, agent technology has not been widely adopted by the software community. Factors in the lack of adoption is the lack of an agreed standard among the diversity of agent oriented software engineering methodologies, and the lack of maturity in some of the methodologies [5].

The agent methodologies have been proposed to aid the agent developer with the introduction of techniques, terminology, notation and guidelines during the development of the agent system. To date, about 30 agent oriented methodologies have been designed [10]. It has been reported that some agent methodologies lack generality and are focused on specific systems and agent architectures [21]. In addition, some of the methodologies do not contain sufficient detail to be of real use.
Alternatively, one idea to help people start agent development pragmatically [3,8] is through patterns. Patterns are a means for sharing development experience to allow a developer to reuse development experience repeatedly. Patterns can allow novices to adapt expert knowledge and help develop software in a systematic and structured way. Patterns are targeted at shared recurring problems, and solution patterns can prevent the developer from reinventing the wheel during application development. The use of patterns in agent development can reduce development cost and time, promote reuse and reduce complexity [11].

The notion of reuse has played an important role during the agent development process in MAS-CommonKADS [2], Skwyrl [3] and PASSI [4]. In the MAS-commonKADS methodology, knowledge patterns are used as a reusable artefact during the development of a multi-agent system. The knowledge patterns contain predefined knowledge that represent how experts solve a specific problem; an expert’s problem solving capabilities [6]; and the knowledge people have of the task they perform [7].

Expertise models of CommonKADS or knowledge patterns are reused during the analysis phase of MAS-CommonKADS. For example, the task of coordinating a meeting has been described in a template task model [13]. Instead of working iteratively to detail the template task model, an assessment template knowledge model is selected to further detail it. In other words, the assessment template knowledge model is used to guide the task modelling.

Based on our observations, current knowledge patterns are found to be lacking in terms of standardization, expressiveness and characterization capabilities. We can summarize our observations as follows:

- The template knowledge model or task knowledge pattern does not feature the concept of agent technology. It has been reported that since the patterns realize their potential in the development of an agent system, it is required to develop the pattern that is tailored to the development of agent system and use agent oriented concepts.

- Task knowledge patterns lack explicitness in expressing certain knowledge elements like control structure.

- The issue of generalization and universality of the CommonKADS template knowledge model. The template knowledge models have been used in MAS-CommonKADS for agent oriented software development. However, it is difficult to enforce the use of a particular term to mean the same thing in all domains and situations.

From the observations, we introduce several task knowledge patterns together with an improved agent oriented template structure for describing task knowledge. It has been claimed that explicitness and comprehensiveness of patterns are two of the important design properties for agent oriented pattern templates [18]. The pattern template acts as a communication medium among developers. If the pattern description is explicitly described, it will improve the communication and comprehension of the patterns for software practitioners [18]. Indirectly, this will improve the representation and delivery of the potential of patterns for agent development.

This paper introduces an improved template structure for task knowledge patterns. The improved template structure that is presented in this paper provides an alternative design for task knowledge patterns with the introduction a two dimensional view of agent oriented models. The task knowledge patterns introduced in this paper describe the task knowledge in an agent context and explicitly describe the useful description for use and reuse in agent oriented software development. Furthermore, the task