Task Knowledge Patterns Reuse in Multi-Agent Systems Development

WaiShiang Cheah[1], Leon Sterling[2], and Kuldar Taveter[3]

[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 clamed 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 knowledge patterns support the expressiveness of task knowledge among non-technical people and support rapid prototyping in agent systems.

Section two presents the brief description of knowledge pattern that is used for agent oriented software development. Section three presents an example of the task knowledge patterns that we have described in our improved template structure. In this section, the task knowledge pattern of 'selection of relevant source materials' is described. Section four presents a case study to develop an adviser finder multi agent system with patterns. Section five presents our observations based on feedback from two masters students in adopting task knowledge patterns for reuse in multi agent system development.

2 Knowledge Patterns for Agent Oriented Software Development

"What is knowledge? How is knowledge represented?" The notion of knowledge is defined and modelled in CommonKADS [12], a knowledge engineering methodology. In CommonKADS, template knowledge models are introduced and are viewed as design patterns or knowledge patterns for tasks [12]. The template knowledge model is also known as an expertise model. It contains predefined knowledge that is represented in the form of reusable model sets for developers. Several template knowledge models are included by CommonKADS. They are classification, assessment, diagnosis, monitoring, prediction, configuration design, modelling, planning, scheduling, and assignment. Each of the template knowledge models consists of the following pattern elements:

- **General characteristics:** Description of the features of a task like goals, typical examples, terminology (e.g., description of the object used for the task), input and output.
- **Default method:** Description of the task knowledge by modelling the actions and control structures for the task type through inference structure and task specification, respectively.
- **Method variation:** Description of the variation of the default method when dealing with a real application. For example, adding a new method or a new object when using the method in a certain application domain.
- **Typical domain schema:** Description of domain entities that will be used for a particular task type. For example, norm, decision and case are domain entities that will be used for assessment task type.

The knowledge has been represented at the knowledge level which has been abstracted away from the symbolic level [9]. The knowledge level was proposed by Newell. Newell introduced another system level that led to a simple and satisfactory view of knowledge and representation [9]. The representation at the knowledge level has a simple structure that provides neither any notion of how the knowledge is represented nor any specification of its processing mechanisms. CommonKADS has utilized the notion of knowledge level to model the problem solving method.

In CommonKADS, the problem solving method is modelled from three different viewpoints (task layer, inference layer, and domain layer) of expertise knowledge.

The task layer models the controlling of problem solving behaviour. Accordingly, it consists of the realization of goals for a task type with control elements like sequential control, conditional branching, iteration, recursion, and so on [6]. It deals with a dynamic view of knowledge. For example, the knowledge of ‘assessment’ consists of the control flow to obtain an abstract case first. This will be followed by specifying the criteria for selection, repeating the actions to take one element of the criteria and comparing it with the abstract case until producing the final decision through matching the evaluation results.

The inference layer presents the inference structure for inference actions. The inference structure represents the actions for a task type and the coupling of action with knowledge roles. The coupling determines the domain information that is required for an inference action. The domain layer represents specific terms that are needed to perform an action. Also, it is known as static knowledge, like having the constraints and preferences for the inference step of assignment [12]. Another example such as a planning task requires domain information for planning such as planning activities, physical resources available for the planning process, and planning constraints like the possible states of the resources. The domain information consists of key elements like concepts, properties of concepts, and relations between the concepts which are represented as a Unified Modelling Language (UML) class diagram. The concepts consist of domain information or terms and the relations indicate semantic relations between terms. In the paper, the terminology of task knowledge pattern and knowledge pattern is used interchangeably.

3 Task Knowledge Pattern

Task knowledge patterns are reusable artifacts that are introduced for agent oriented software development. Three desired properties are described by Oluyomi [18] in designing an agent oriented template structure. The desired properties present the overall requirements that need to be considered by a pattern designer when designing an agent oriented template structure.

We describe two of the properties within the context of this research. They are ‘completeness’ and ‘eliminating ambiguity’. We adopt these desired properties and elaborate them in designing the template structure for the task knowledge pattern in the following description.

`Completeness` Our template structure for task knowledge is complete as it captures the levels of different ‘viewpoints’ of expertise knowledge. However, we introduce the level of different ‘viewpoints’ that is related to the aspect dimension of the behaviour category, interaction category and information category at the conceptual domain modelling level. These viewpoints are taken from the text ‘The Art of Agent-Oriented Modelling’ [1]. The knowledge that is modelled in our ‘viewpoints’ involves

- The goals that are required to be achieved for solving a problem;
- The arrangement of responsibilities in fulfilling the goals given, and
- The knowledge items that are used by goals and responsibilities.

The knowledge that is modelled at such categories and levels describes the knowledge at a higher level of abstraction. It is sensible to claim that it corresponds to the knowledge level modelling of expertise knowledge as practiced in knowledge engineering [9]. Modelling the knowledge at a high level of abstraction has advantages as follows:

- It serves to communicate knowledge to a non-technical person. Apart from the agent designer and implementer, the task knowledge pattern will support a non-technical person like a novice user. The task knowledge pattern has been shown to be useful among non-agent practitioners as discussed in section 5.
- People are not restricted by details of design such as looping, attributes of terms, detailed pre-condition and post-condition, detailed information flow or implementation constraints that will influence the generality of the task knowledge.

`Eliminate Ambiguity` The template structure must cater for explicit values and unified representation to ease the ambiguity [20, 21] of the pattern description. Explicitness of the template structure through explicit values is a common practice for designing agent patterns. The explicit value outlines a particular agent development life cycle and agent development task will create a consistent viewpoint when people intend to adopt the pattern for the task at hand [18]. It is important for having those explicit values to allow communication of the pattern with terminology that seems common in agent development.

Next, we provide the description of an example task knowledge pattern.

3.1 Example Description of Task Knowledge Patterns

The pattern elements for the task knowledge patterns are described below.

Pattern name: Represents the name of the task type or problem solving method. It indicates the name of the pattern to be modelled. The name is normally related to the problem at hand.

Intent: The purpose(s) in having this pattern. It consists of the description to elaborate on the motivation for having this pattern.

Use when/ Applicability: Descriptive of situations that lead to the usage of the pattern.

Problem: The problem that needs to be solved by this pattern.

Force: Requirement of the problem, *solution properties* in which the pattern is situated.

Solution: The knowledge level of the problem solving method in solving the problem given. It contains the description that explicitly models the knowledge in solving the task.

Dynamic: The dynamic element provides a typical scenario in describing the runtime behaviour of the pattern. In other words, it details the arrangement of the solution according to a particular situation.

An example of task knowledge pattern is as follows:
Pattern Name: Selection of Relevant Source Material

Intent:
The purpose of this pattern is to develop an agent that is able to perform a search and provide relevant results based on particular criteria. In other words, the agent will locate certain information that is supplied by the information provider from a set of keywords and then produce a relevance result.
Also known as:
Melisa [14], Amathaea [15], Sourcer [16]

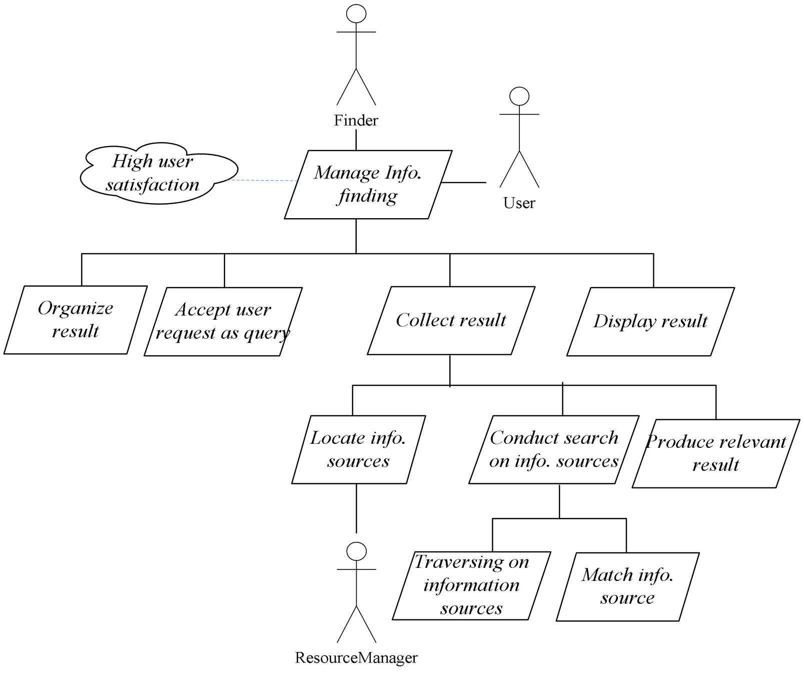
Context / Applicability:
Use this pattern when
-you want to explore within a collection of information or repository regardless of the scale of the repository.
-you want to obtain relevant documents from your search.

Problem: Deals with the finding of a set of documents in response to a user request.

Forces: Describes the solution properties in which the pattern is situated or based in the context of the problem.

Goal: The user is able to provide his/her preferences from the solution provided. Meanwhile, the returned documents may be arranged accordingly.
Quality goal: Achieving user satisfaction is needed. The solution must be able to provide a collection of relevant results which are closer to the user keyword. When performing a query, the solution may be required to provide the returned information in an efficient manner. In this case, time to search for relevant documents must be taken into consideration when designing the solution.
Role: Three roles are involved when conducting a search. They are the role played to manage the finding like handling a query, conduct search, ranking or combined result, the role played to keep the sources for finding purpose and the role played to send the search request.
Resource: The domain entities of query, criteria, relevant content, information resource and domain are basic entities that are required in conducting the task type of 'selection of relevant source materials'.

Solution:



< **Goal Model**>. Goal model for selection of relevant sources

- **Organize result** Ranking and/or combination of searched results. There is a mechanism to send the user query to multiple search engines. Each search engine will be involved in information finding and the returned documents will be combined into a final result presentation.

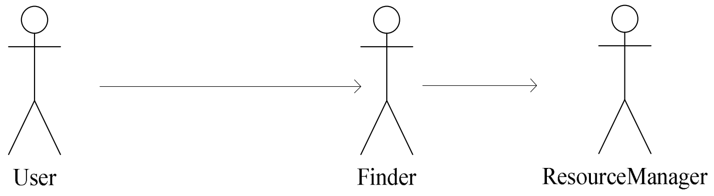
- **Accept user request as query** The purpose of this goal is awaiting the user query.
- **Collect results** The goal of 'organize result' is the core activity in the information finding. It involves activity to obtain references by checking the query against a page hyperlink description or meta-information; performs searched on the referenced entity (e.g. documents like web page) and performs matching or evaluates on a searched item.
- **Display result** The purpose of this goal is to present the finding returned like downloaded document references, name list and so on in an appropriate manner

<Role model>. Role model for selection of relevant sources

Role name	Information finder
Description	Manage information finding
Responsibilities	Receive incoming query for information finding Obtain relevant sources. -obtaining relevant references or indexing. -Traverse given documents. -Search through the content by giving the references. -Perform matching based on user request. -Create relevant sources. -Organize relevant sources -Perform ranking and combination of searched results. Display the relevant sources.
Constraints	Query must first assign prior selection. All the search may be provided with any return.

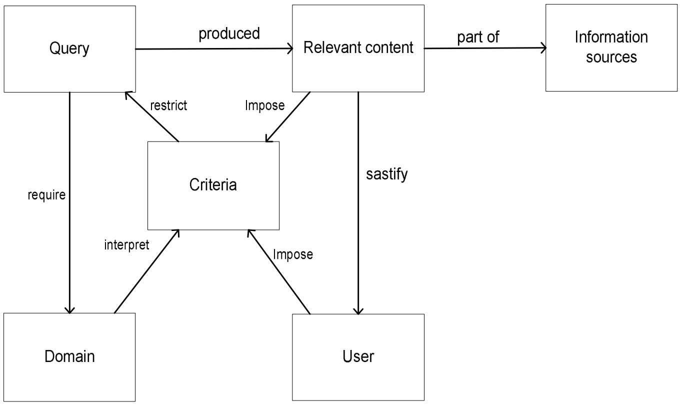
Role name	ResourceManager
Description	Manage information finding
Responsibilities	Keeping the sources for finding purpose.
Constraints	-interact with incoming request for finding the relevant. -provide information spaces for search.

Role name	User
Description	Request for search
Responsibilities	-send request for search. -receive relevant sources material.
Constraints	-



<Organizational model>. Organizational model for selection of relevant sources
Organization Structure. The members involved in dealing with information finding task type are User, Finder and ResourceManager. The Finder realizes the request from

the User. The ResourceManager provides information spaces for the search by the Finder.



The resources that will be consumed by this task are listed in the domain model. Accordingly, further elaboration of the resources is described below.

- **Query** The domain entity that will be used in conducting a search. It is a form of user request or query term.
- **Relevant content** The domain entity that represents the response from the search, which is derived from a collection of information sources.
- **Criteria** The criteria consist of user preferences that are imposed by the user as searching criteria. For example, search modifier, special filter like search through year from, year to, abstract, maximum retrieval time per page, credit status, capability available.
- **Domain** The domain consists of a topic of discussion which describes the element of criteria and is required as part of the query's domain entity. It may contain a vector of keyword for keyword search.
- **User** The domain entity of requester. The user (e.g. software agent or human agent) that will impose a query for a finding.
- **Information sources** The domain entity that represents the sources of information like documents, semi-structural data like web, images, video, medical catalog, text file, pdf file and so on.

Consequences
The key consequence of the task knowledge pattern is to help to reduce the effort to search within the information space and produce a relevant return to the user.

Related Pattern
The 'assessment' template knowledge model in the CommonKADS is related to achieving the goal of 'collect result'. On the other hand, the task knowledge pattern of 'relaxing the search term' is related to this pattern to increase the accuracy of the search. The pattern deals with the query expansion to produce a set of queries from the user request. In other words, the early user request is expanded to enable the search in a more precise manner.

We have presented an example of a task knowledge pattern. In summary, ten task knowledge patterns are proposed [8]. In the following section, we present the use of task knowledge pattern in developing an adviser finder multi agent system within the ROADMAP and AOR methodologies.

4 Case Study

Task knowledge patterns introduced reusable models sets to prevent the developer from reinventing the wheel in solving a problem at hand. In this section, we demonstrate the reusability of the model sets (e.g. goal model, role model, organization model and domain model) in the early development stages to rapidly prototyping an adviser finder multi agent system.

The background problem of the adviser finder multi agent system is described as follow. Students receive Government scholarships to study for a PhD overseas if they are able to find an adviser within a reputable university. To find an adviser, a substantial amount of knowledge is needed which includes an "advisor domain" like research areas, research experience, professional activities and so on. These are usually described differently among the academics at different institutions. To sustain the search, it is always believed that a student will browse from one page to another, collect information from several institutions, interpret and understand the information collected and short list the candidates for potential supervisors. To facilitate the adviser finder, we propose an agent oriented adviser finder multi agent system to automate the adviser finder application. The adviser finder multi agent system accepts the user request, conducts search across semi-structured data such as academics' web pages and returns a list of potential advisers based on the user request.

4.1 Task Knowledge Patterns Reuse in Developing Adviser Finder MAS

As shown in Figure 1, a combined modelling process has been introduced by Sterling and Taveter [1] to engineer a multi agent system in a more unified way to support the rapidly prototype of agent oriented system [1]. We refine the combined modelling process for ROADMAP and AOR that places the knowledge reuse within the modelling process, as shown in Figure 1.

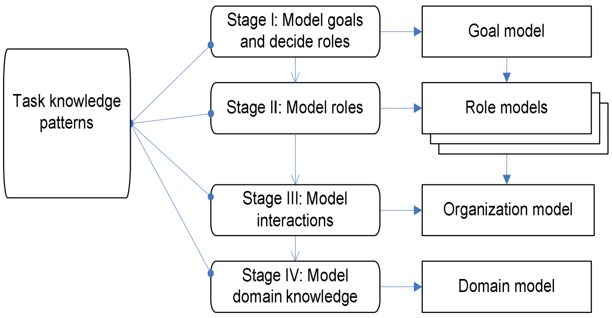

Task knowledge patterns
Stage I: Model goals and decide roles
Stage II: Model roles
Stage III: Model interactions
Stage IV: Model domain knowledge
Goal model
Role models
Organization model
Domain model

Fig. 1. Agent oriented modelling processes

Figure 1 shows how to relate the task knowledge patterns for early agent development in the combined modelling process of ROADMAP and AOR methodologies In the following description, we describe how the task knowledge patterns can be integrated in the modelling processes to rapidly prototype an adviser finder multi agent system. We present the reuse of the model sets presented in the task knowledge patterns of early stages of agent development. We demonstrate a task to manage adviser finding (e.g. a task to match information) through pattern.

It is sensible to claim that the predefined knowledge that is presented in the task knowledge patterns can provide the answer during the requirement elicitation phase for an agent system. For example, for the adviser finding problem, we can hire a position like adviserFinder to search the potential adviser which the job description can be derived from the role model and knowledge for the position will be derived from the domain model within the task knowledge pattern of 'selection of relevant source materials'. However, instead of showing how to reuse the task knowledge patterns during the requirement elicitation, we present the reuse of models set that are presented in the task knowledge patterns in modelling the goal model, role model, organization model, domain model at the early stages of agent development as shown in Figure 1.

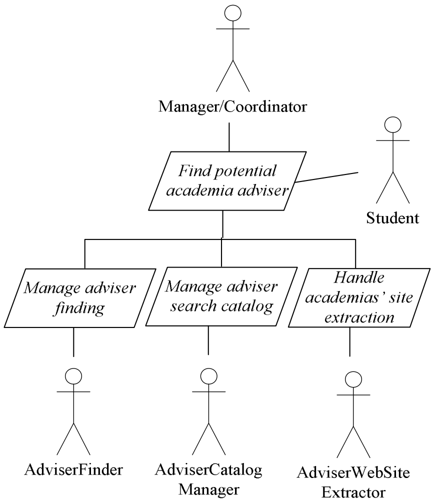


Fig. 2. Overview of goal model for the adviser finding problem

Stage I Model the goal: The early stage in agent modelling involves modelling goals and decides roles. Figure 2 shows the overview goal model for the adviser finder multi agent system. The goals and roles that have been played are derived based on the study on related kind of system [17]. We can model the requirements through an overview goal model as shown in Figure 2, the goal of 'find potential academic adviser' consists of sub-goals like 'manage adviser finding', 'manage adviser search catalogs', 'handle academic's site extraction'. We can interpret that those goals rely upon the people that played the role like adviserFinder, adviserCatalogManager, adviserWebExtractor, Student and manager for fulfilling the goals.

We can further detail the goal model of ‘manage adviser finding’ according to practice below. Instead of working from scratch, we can adopt the model sets that were introduced in the task knowledge patterns for solving our problem at hand. In working into this process, we presented guideline1 to integrate the task knowledge patterns in early development stages (e.g. Stage I, Stage II, Stage III and Stage IV). Assume the developer has accepted all the forces from the pattern description. The procedures of guideline1 are described below.

1. For each subgoal of a goal model, we can reuse task knowledge patterns for its further elaboration. Each of the subgoals may be further elaborated with subgoals from a goal model included by task knowledge patterns.
2. In deciding on roles, we can reuse the roles that are included by task knowledge patterns. The details of a role model can also be derived from task knowledge patterns. However, effort is needed to relate a derived role model to the application context. The roles involved in an organization can be further refined when creating the organization model.
3. We can reuse the organization model included by task knowledge patterns. The organization model can be further elaborated thereafter. For example, we can add a new role to control the processes involved or some roles can be combined or can enter into association or aggregation relationships with the other role(s) depending on the application context. The organization model provides a foundation for modelling the interactions between the agents playing the roles of the organization.
4. When creating the domain model, we can reuse domain entities and relationships between them from task knowledge patterns. A domain model thereafter needs to be refined according to the application context.

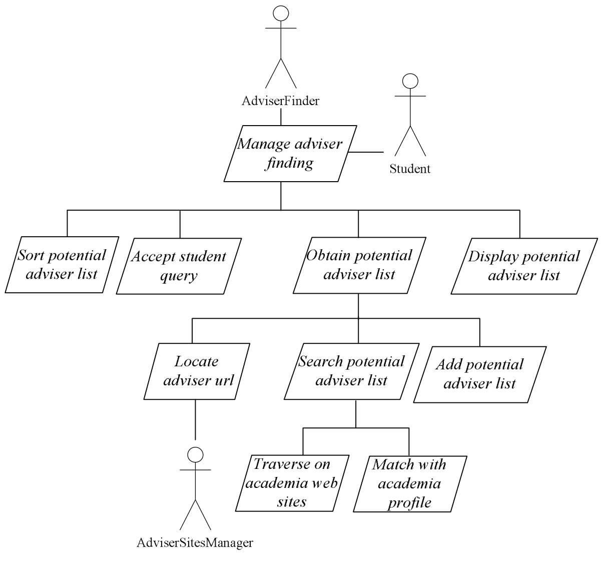

AdviserFinder
Student
Manage adviser finding
Sort potential adviser list
Accept student query
Obtain potential adviser list
Display potential adviser list
Locate adviser url
Search potential adviser list
Add potential adviser list
Traverse on academia web sites
Match with academia profile
AdviserSitesManager

Fig. 3. Goal model for selection of potential academia adviser

The task knowledge patterns of 'selection of relevant source material' record the experience for solving the finding problem. As a result, we can reuse the goal model that is described in the task knowledge pattern to further detail the goal of ‘manage adviser finding’ as shown in Figure 3. We adopt the goal model that described in Section 3.1 by relating it to our context. In other words, we relate the role of user with student, the role of finder with adviserFinder, the goal of 'accept user request' with 'accept student query' and so on to further detail the goals as shown in Figure 2.

Stage II - Model the role model: We relate the role name, responsibilities and constraints of the role schema that are presented in the task knowledge pattern as described in the Section 3.1 into our application context as shown in Table 1. For example, we relate the role of ‘Information Finder’ with the role name of the ‘AdviserFinder’; the job description of the role (e.g. responsibilities) like query with student query, traversing with academic profile and so on.

Table 1. Role model of AdviserFinder

Role name	AdviserFinder
Description	Manage potential adviser finding
Responsibilities	Receive incoming student query for adviser finding Obtain potential adviser listing -obtaining academia urls or indexing. -Traversing on academia profile. -Searching through the academia profile by giving the url -Perform matching based on student request. -Create potential adviser list -Organize potential adviser listing -Perform ranking and combination of candidate adviser. =Display the potential adviser listing
Constraints	Student query must first assign prior selection. All the search may be provided with any return.

Stage III- Model organization: The organization model models the arrangement of the roles involved for task accomplishment. The organization model for the adviser finding problem is modelled in Figure 4. The organization model is derived from the selected task knowledge patterns together with the roles that have been modelled from the overall goal model as shown in Figure 2. For example, within the task type of ‘selection of relevant source material’, the AdviserFinder relies on the request from the user. The ResourceManager provides information spaces for search by the AdviserFinder.

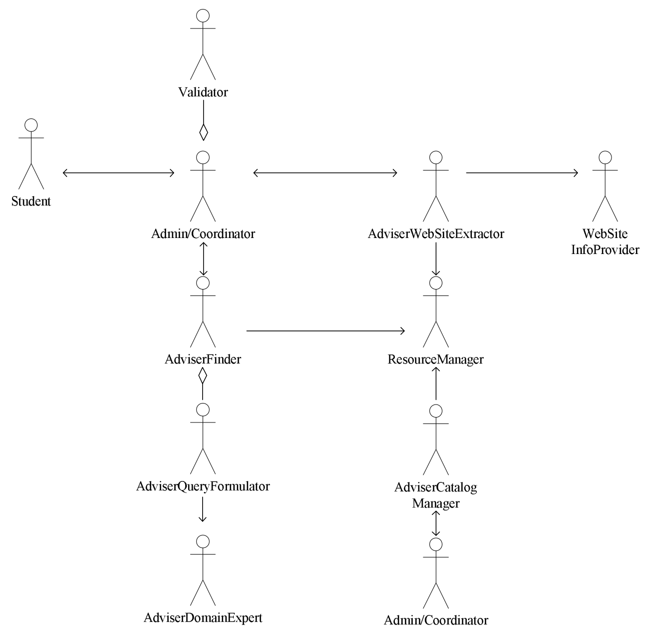


Fig. 4. Organizational model for the adviser finding MAS

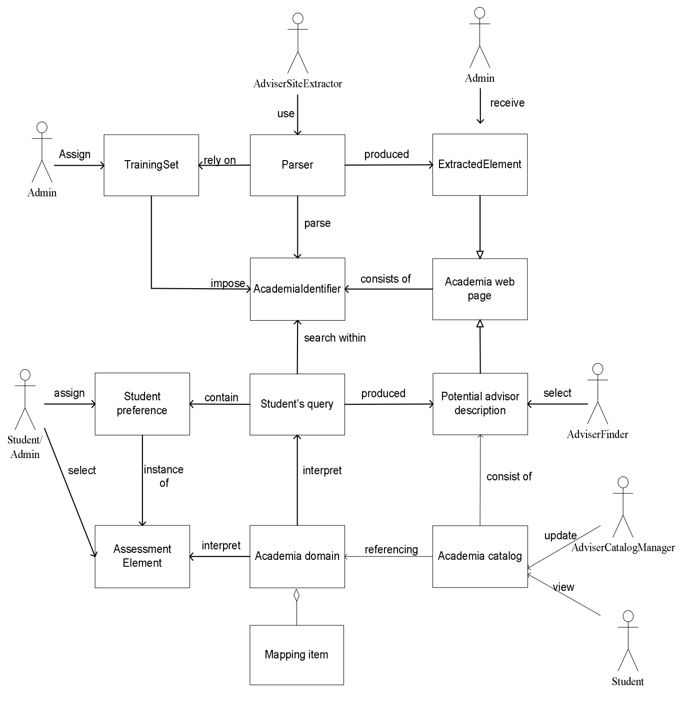


Fig. 5. Domain model for the adviser finding MAS

Stage IV- Model domain knowledge: Modelling the domain knowledge involves identifying the domain entities and the relation among the domain entities for the problem at hand. To model the domain model for the agent system, we can reuse the domain models presented in the selected task knowledge patterns during this modelling process. Figure 5 presents the domain model for adviser finding problem. We can adopt the domain models according to our application context. In this case, we can describe the domain entity of Query as StudentQuery, Domain as AcademiaDomain, Criteria as StudentPreference and AssessmentElement, RelevantContent as PontentialAdviserDescription, InformationSources as AcademiaWebPage. The domain model that presents in Figure 5 is the integration among the domain models among the text extraction pattern and categorization pattern.

We presented the modelling of goal models, role model, organization model and domain model at the early stages of multi agent system development (e.g. stage I to stage IV). We introduce the reuse of the model sets that are provided in the selected task knowledge patterns when modelling the goal models, role model, organization model and domain model. It is interesting to show that the task knowledge patterns have shared the recurring problem and solution and prevent us from reinventing the wheel for developing the adviser finder MAS. Consequently, we can put much effort to continue the modelling process for the adviser finder multi agent system at Stage V: decide agent types; Stage VI: model the knowledge of agents; Stage VII: model interactions between agents and Stage VIII: model agent behaviours as described in [8].

The screenshot for the adviser finder multi agent system is shown in Figure 7. A student posts the query through a normal search or advance search from the search by '..' menu. Figure 7 presents the screenshot of a typical search. The student can key in any search items (e.g. supervisor name, publication, research area and so on). Then the system returns with a candidate adviser list.

5 Conclusion and Discussion

Task knowledge patterns are typically reusable at the early development stages of a multiagent system and reusing them supports rapid prototyping of a multiagent system. We have proposed an improved template structure in describing the task knowledge and demonstrated how the pattern is reusable in rapidly prototype the adviser finder multi agent system. In addition to the results reported in this paper, we have conducted an evaluation of the usefulness of task knowledge patterns for agent development. Several questionnaires were prepared for conducting the evaluation. The questionnaires were designed to assess the pattern content and the learnability and usefulness of the patterns. A survey was conducted with two Masters students at Tallinn University of Technology, Estonia, with novice experience in agent-oriented software development. These students were respectively required to develop an agent-oriented recommendation system and an agent-oriented interoperability system for their Masters Thesis projects. At the beginning of their study, the students explored the ROADMAP and AOR methodologies. After that, the students were presented with task knowledge patterns for agent-oriented development. They were required to study the patterns before they started to design an agent-based system. The students had approximately two months for designing a multiagent system facilitated by task knowledge patters. Upon completion of the project, the students were provided with questionnaires to evaluate the task knowledge patterns adopted by them.

In general, novice users (e.g., students) seem to be satisfied with the usage of task knowledge patterns that have been expressed by means agent-oriented models. Both of the students surveyed agreed that the task knowledge patterns were useful when developing multi-agent systems and were easy to learn. According to the surveys, agent-oriented models were easily able to communicate ideas and concepts behind task knowledge patterns and both students preferred to adopt the patterns also for future multi-agent system development. In other words, task knowledge patterns facilitated solving the problem at hand for both students. On the other hand, the reviews also addressed the problem that the content of some patterns lack sufficient information. We have seriously considered this feedback and as a result have further refined task knowledge patterns by introducing expected runtime behaviours into the patterns. In addition, generality of patterns has been increased. For example, we removed the goal 'Content selection' from the profiling pattern because the content selection really belongs to the pattern of information finding. We also remark here that conducting a survey with just two students is naturally not sufficient for obtaining a real picture but has nevertheless provided us with useful insight and feedback about the application of task knowledge patterns. In our future research work, we plan to conduct similar surveys with more participants.

References

1. Sterling, L., Taveter, K.: The Art of Agent Oriented Modelling. MIT Press, Cambridge (2009)
2. Do, T.T., Kolp, M., Pirotte, A.: Social patterns for designing multi-agent systems. In: Proceedings of the 15th International Conference on Software Engineering and Knowledge Engineering. Citeseer (2003)
3. Cossentino, M., Sabatucci, L., Chella, A.: Patterns Reuse in the PASSI Methodology. In: Omicini, A., Petta, P., Pitt, J. (eds.) ESAW 2003. LNCS (LNAI), vol. 3071, pp. 294–310. Springer, Heidelberg (2004)
4. Luck, M., McBurney, P., et al.: Agent technology: Enabling next generation computing. In: Agent Link Community, pp. 74–75 (2003)
5. Chandrasekaran, B., Josephson, J.R., et al.: The ontology of tasks and methods. In: Proceedings of the Eleventh Workshop on Knowledge Acquisition, Modelling and Management (KAW 1998), pp. 18–23 (1997)
6. Annamalai, M.: Modelling knowledge for scientific collaboration on the semantic web, The Melbourne University. PhD (2006)
7. WaiShiang, C.: Patterns for Agent oriented software development, The Melbourne University. PhD (2010)
8. Newell, A.: The knowledge level. In: AI Magazine, Department of Computer Science, Carnegie-Mellon University (1981)
9. Koutsabasis, P., Darzentas, J.: Methodologies for agent systems development: underlying assumptions and implications for design. AI & Society 23(3), 379–407 (2009)

10. Lima, E.F.A., Machado, P.D.L., et al.: An approach to modelling and applying mobile agent design patterns. ACM SIGSOFT Software Engineering Notes 29(3), 1–8 (2004)
11. Schreiber, G.: Knowledge engineering and management: the Common KADS methodology. MIT Press (2000)
12. Henderson-Sellers, B., Giorgini, P.: Agent-oriented methodologies. Idea Group Pub. (2005)
13. Abasolo, J.M., Gómez, M.: MELISA: An ontology-based agent for information retrieval in medicine. In: Proceedings of the First International Workshop on the Semantic Web, vol. 3, pp. 73–82 (2000)
14. Tang, C., Xu, L.D., Feng, S.: An agent-based geographical information system. Knowledge-Based Systems 14, 233–242 (2001)
15. Loewus-Deitch, D., Herdrick, B.: The Sourcerer: An Expert Human Resource Agent; Nick, A., Koenemann, J., et al.: ELFI: information brokering for the domain of research funding. Computer Networks and ISDN Systems 30(16-18), 1491–1500 (1998)
16. Oluyomi, A., Karunasekera, S., et al.: Description templates for agent-oriented patterns. The Journal of Systems & Software 81(1), 20–36 (2008)
17. Yoshioka, N., Washizaki, H., et al.: A survey on security patterns. Progress in Informatics 5, 35–47 (2008)
18. Zdun, U., Avgeriou, P.: A catalog of architectural primitives for modeling architectural patterns. Information and Software Technology 50, 1003–1034 (2007)
19. Zambonelli, F., Jennings, N.R., et al.: Organisational rules as an abstraction for the analysis and design of multi-agent systems. International Journal of Software Engineering and Knowledge Engineering 11(3), 303–328 (2001)