ANALYSIS AND CLASSIFICATION OF TASK KNOWLEDGE PATTERNS

Cheah WaiShiang, Leon Sterling
Department of Computer Science and Software Engineering
University of Melbourne, Australia
E-mail: w.cheah@pgrad.unimelb.edu.au, leonss@unimelb.edu.au

KEYWORDS
Classification scheme, attribute-based analysis, patterns

ABSTRACT

Patterns have recorded the experience of engineering software systems. Various patterns have been introduced and described in several domains. A particular class of patterns, *task knowledge patterns* that are important for agent oriented software development, is investigated in this paper. It has been reported that descriptions of task knowledge given in the pattern literature is not clearly structured, and some of the useful description is left implicit to the developer. This paper reports our investigation on task knowledge that has been shared across a wider spectrum of articles to showcase such scenario. We present a new way to analyse and classify task knowledge patterns. Furthermore, we demonstrate how the experience of different dvelopment groups is described very differently based on the classification result. The investigation shows the need to present task descriptions in a structured manner. The analysis and classification techniques demonstrated are applicable for agent-based simulation.

INTRODUCTION

Over the past several years, various agent patterns have been introduced, for example as listed in (Oluyomi, Karunasekera et al. 2007) and (WaiShiang 2010). Patterns record experience of engineering software system. While various patterns have been introduced and patterns have been described in several domains by Oluyomi, a particular pattern, *task knowledge patterns*, that is important but paid less attention to by Oluyomi, is investigated in this paper.

In (WaiShiang 2010), we report on task knowledge patterns known as CommonKADS template knowledge models (Schreiber 2000) that describe the problem solving method or task knowledge within task template description. Task knowledge is being shared by describing the experience of dealing with particular task types across a wider spectrum of articles (De Wolf and Holvoet 2006). For example, various articles have been presented in dealing with mediating user requests and services through engineering a brokering system.

It has been reported that the description for the task knowledge given in the articles is not clearly structured, has indirect descriptions and some of the useful description seems implicit to the developer (De Wolf and Holvoet 2006). The investigation to showcase such scenario is presented in this paper. The investigation raises the need to present task descriptions that are spread across various articles in a structured manner. We present a new way to analyse and classify task knowledge patterns. Furthermore, we demonstrate how the experience of different dvelopment groups is described very differently based on the classification result. The investigation raises the need to present task descriptions that are spread across various articles in a structured manner. The analysis and classification techniques are applicable for agent-based simulation.

This paper consists of five sections. Section 2 introduces a Two-ways classification scheme that will be used to analyse and classify task knowledge patterns. An analysis method, attribute-based analysis, is presented in Section 3. An example of classification, brokering systems, is presented in Section 4. Altogether, we present the result of classifying 58 articles that describe the experience of engineering a brokering. The sample collection of 58 articles is first described, followed by the result of the classification and observations. The paper's conclusions are given in Section 5.

TWO-WAYS CLASSIFICATION SCHEME

Viewpoint abstraction layers	Viewpoint aspects			
Conceptual independent modelling level	Information	Interaction	Behaviour
Platform independent design & modelling level	Definitional	Structural	Interaction
Platform specific design & modelling level	Structural	Interaction	Strategy

Figure 1: Two-ways classification scheme

We can describe the task knowledge that has been shared using a Two-ways classification scheme adapted from (Sterling and Taveter, 2009) as shown in Figure 1. The Two-ways classification scheme is used to classify knowledge according to viewpoint abstraction layers and viewpoint aspects. The viewpoint abstraction layers presents the stages of developing sociotechnical systems. The viewpoint aspects presents aspects for each stage of the development.

The viewpoint abstraction layers constitute three levels. They are conceptual domain modelling (CIM) level, platform independent design and computation design (PIM) level and the platform specific design and

Proceedings 24th European Conference on Modelling and Simulation ©ECMS Andrzej Bargiela, Sayed Azam Ali David Crowley, Eugène J.H. Kerckhoffs (Editors)
ISBN: 978-0-9564944-0-5 / ISBN: 978-0-9564944-1-2 (CD)

implementation (PSM) level. The conceptual domain modelling level records the experience in analyzing the system. The description given is presented at a high level of abstraction. For example, the listing of players and tasks related to them within the insurance brokerage; listing of information that is required to be collected by the broker and such description does not describe the detail elements like association among the information, attributes; listing of brokering activities. The PIM level records experience to provide a detailed design of the system. The PIM shows part of the system design specification that does not change from one platform to the other. The design activities will range from conceptualization, arrangement of external functions that rely upon, arrangement of inferences steps or internal structure and method control. The PSM level will complement the PIM level with the lowest detail design that specifies how the system is to be implemented in a specific platform, tool or programming language. For example, the functionality of the mediator is designed at code level, the service description for matchmaking is designed through MONET description (Ludwig, Rana et al. 2006).

The viewpoint aspects represent the software development task that is split into three categories. They are information category, interaction category and behaviour category. The information category reflects the terms used. The interaction category reflects who are involved in solving the problem. It is related to the modelling of transfer function with the external world when solving a problem with CommonKADS. The behaviour category reflects the inference steps or actions, and the sequences of actions together with the control behaviour of the actions. For example, an online housing system will obtain information from the database system. The system will use certain resources like user identification, price list and so on for task execution. It performs tasks like assessment of user application, and monitoring housing applications. Typically it will rely on an external application like a banking system for validating user financial information.

To classify and analyse the shared experience through the Two-ways classification scheme, attribute-based analysis is introduced. Attribute-based analysis is introduced to characterize and categorize the shared experience through the Two-ways classification scheme.

ATTRIBUTE-BASED ANALYSIS

Attribute-based analysis is introduced to characterize and categorize the patterns through the Two-ways classification scheme (Oluyomi, Karunasekera et al. 2007). In other words, the attribute-based analysis is used for classifying agent patterns according to the dimensions of the Two-Ways classification scheme. The attribute-based analysis will determine into which dimension (e.g. level and category of Two-ways classification scheme) a pattern belongs to based on the identification of pattern level attributes and identification of pattern category attributes. For example, a pattern is classified at a conceptual independent modelling level and behaviour category, if the pattern has been described at a high level of abstraction like presenting a collection of goals with the notion of the goal reflecting the behaviour aspect within the Two-Ways classification scheme.

The attributes represent the features or common knowledge elements that fall within a particular level and category in the Two-Ways classification scheme. They represent the agent concepts that people use during the engineering of software system (e.g. agent oriented software system or non-agent oriented software system).

Figure 2 shows the attributes that we identified for Two-ways classification scheme. Each dimensions (e.g. level and category of Two-ways classification scheme) cover sets of attributes that will be used to classify and analyse the agent patterns through attribute-based analysis. The attributes are identified by studying the conceptual space for social technical system, various agent oriented methodologies and agent oriented software development projects (Sterling and Taveter 2008). Due to space limit, we only present the informal definition for the attributes at viewpoint abstraction layers for the following description. This will be followed by the description of attribute analysis tools and process.

Two-ways classification scheme level attributes

This section presents the informal definition for the attributes at viewpoint abstraction layers.

Conceptual Domain Modelling level.

Identify Role/Multiple roles Identification of roles have been played for solving a particular problem and opportunity within a real life organization. This attribute looks at the activity to analyse the roles that have been played and the responsibilities of the role in solving a problem.

Assigning Goal Assigning goals and sub-goals appears in the analysis phase of agent oriented software development. In addition, the attribute looks at the activity to identify the goal(s) achieved by a particular role and resource usage.

Identify domain entities Identifying the resources or sources that are required for the problem and opportunity given is the activity that is described by this attribute. Activities like understanding the scope and the aim of the domain entities and identifying the glossary are the focus of this attribute.

Viewpoint Abstraction layer	***Viewpoint aspect***		
	Information	***Interaction***	***Behaviour***
Conceptual domain modelling • Identify roles • Assigning goals • Identify domain entities • Organization of roles	• Domain entities	• Roles • Organization structure • Social policies/authority power • Responsibility	• Goal • Quality goal • Goal dependency • Roles/actors • Resource
Platform-independent computational design • Ontology/ domain conceptualization • Arrangement of agents • Agents message exchange • Agent or module internal activity	• Concepts • Slot • Relation	• Agents/ multiple agents • Communication state/ social order/communication strategy • Message exchanged	• Agent type • Information type • Reasoning/strategy • Proactive • Reactive
Platform-specific design and implementation • Plan specification • Interaction specification • Agent instantiation, aggregation, inheritance • Information specification	• Concrete object (i.e. Belief) • Syntax of Knowledge representation in language specific	• Message scheme • Communication support • Concrete organization structure	• Behaviour construct (i.e. Plan, event) • Concrete agents/ agent instances • Perception • Resource Utilization

Figures 2: Attributes of Two-ways classification scheme

Organization of role Organization of role describes the arrangement of role in dealing with a problem and opportunity in an organization. This attribute looks at describing an organization structure such as peer to peer, flat, hierarchical, etc.

Platform Independent Computational Design level.
Ontology/ domain conceptualization The attribute captures activities for modelling concepts of roles, class, property and concept hierarchies.
Arrangement of agents Agent acquisition is the design activity that is captured in this attribute. The agent in this context can refer to artifact like humans or people, functional roles, software components, modules, information systems or software agents.
Agent interaction Activity in determining the flow of information in exchanging the information among external software components or designing the interaction protocol is captured in this attribute.
Agent internal activity The agent internal activity captures the designing of agent internal state(s) corresponding to decision making, responding to changes that occur and goal achievement. Some examples of designing an agent internal structure are determining post condition, pre-condition or execution control (e.g. looping, recursive) of agent behaviour, assigning actions and inference steps for agent reasoning.

Platform Specific Design and Implementation level.
Behaviour specification Behaviour specification describes the lowest level detailed design of the event and control behaviour of an event that is used in various platforms like Jason, JACK, .Net and so on.
Interaction specification Interaction specification describes the lowest level message exchanged among agents. In addition, the designing of a message transport protocol as well as a communication support like bandwidth, communication hardware determination will be captured in this attribute.
Agent instantiation, aggregation, inheritance An agent in this level involves the designing of a low level detail of concrete agent like designing the agent through an object oriented class diagram, with OO class representation as agent type, methods, attributes of the class and inheritance, association among the classes as agent behaviour.
Information specification In ontology engineering (Uschold and Gruninger 1996), this is known as the process of ontology coding.

Attribute-based analysis: Tool and processes

Tool and processes are introduced for attribute-based analysis. The attribute-based analysis tools are analysis tables used for identification of pattern level attributes to determine the level that the pattern belongs to and identification of pattern category attributes to determine the category that the pattern belongs to.
The tool is a tablet form that is populated with attributes from within the Two-Ways classification scheme. Altogether, four sets of attribute-analysis tables are introduced as shown in Table 1 to Table 4. Each of the analysis tables outlines three sections. They are the *knowledge source* that records the input for analysis purpose; the *viewpoint abstraction layer or dimension* and the *attributes* underneath the dimension and the *analysis outcome* of the analysis.
The knowledge source consists of a pattern description used for evaluation. The dimension and attributes cater the elements used for identification of pattern level attributes and pattern category attributes.

Table 1: Identification of viewpoint abstraction layer attributes for patterns

Knowledge source	Information	Interaction				Behaviour					
	Domain entities	Roles	Responsibility	Organization Structure	Social policy	Goal	Quality goal	Role/ actors	Goal dependency	Resource	Analysis conclusion

Knowledge source	Conceptual domain modelling				Platform independent modelling		Platform specific modelling						
	Identify role	Assigning goal(s)	Identify domain entities	Organi-zation of role	Ontology / Conceptu alization	Arrangement of agents	Agent internal activity	Agent message exchange	Plan/ event spec.	Inter-action spec.	Agent In-stantiation	Info spec	Viewpoint abstraction layer

Table 2: Identification of category attributes for conceptual domain modelling level

Know. source	Information			Interaction			Behaviour					Analysis Conclusion
	Concept	Slot	Relation	Agents/ multiple agents	Social order	Message	Agent	Resource/ Service	Reasoning	Proactive	Reactive	

Table 3: Identification of category attributes for platform independent modelling level

Know. source	Information		Interaction			Behaviour				
	Concrete object	Syntax of KR	Message content	Communication Support	Concrete organization structure	Behaviour construct	Concrete Agent	Perception	Resource utilization	Analysis Conclusion

Table 4: Identification of category attributes for platform dependent and implementation modelling level

Table 1 is designed to classify and analyse the patterns according to a particular vertical dimension. It has been populated with attributes that can be identified within the vertical dimension. For example, analysis table of Table 1 is outlined with levels of viewpoint abstraction layers and the attributes in each of the levels (e.g. *identify roles, assigning roles, identify domain entities, organization of role* at conceptual independent modelling level). Other analysis tables are designed to classify and analyse the patterns according to a particular viewpoint aspect. Table 2 is the analysis table that is used to identify which viewpoint aspect the pattern belongs at the conceptual domain modelling level. Table 3 is the analysis table that is used to identify which viewpoint aspect the pattern belongs at the platform independent design level. Table 4 is the analysis table that is used to identify which viewpoint aspect the pattern belongs to at the platform specific design and implementation level.

To analyse the pattern using attribute-based analysis, the attribute-based analysis processes are described as follows. We first examine the shared experience towards discovery of attributes. We review and observe the shared experience in engineering the system, to review what might be shareable from the description given, *step1*. From the observation, we proceed to viewpoint abstraction layer attributes analysis and viewpoint aspect attributes analysis. This involves activity in placing the patterns on the attribute analysis table (e.g. Table 1 to Table 4) and to indicate which of the viewpoint abstraction layer attributes or viewpoint aspect attributes listed in the analysis table are presented in the pattern, based on the examination of the pattern. Finally, we conclude the analysis result.

Concluding the analysis result is based on the frequent occurrence of attributes within dimensions of Two-ways classification scheme. In other words, we conclude that a pattern falls in a particular dimension if the number of occurrence attributes for the shared experience in one dimension is higher than the number of occurrence of attributes in other dimension.

In the previous description, we presented the dimensions of Two-ways classification scheme and the informal description on those dimensions. This is followed by our description on attributes for attribute-based analysis as well as informal definition on the attributes at viewpoint abstraction layers. In addition, we presented the tool and processes for attribute-based analysis. To demonstrate the feasibility of the classification scheme, an example is presented in the following sections. We conduct an analysis and classification of 58 ‘brokering patterns’ and present our observation from the result of the classification.

ATTRIBUTE-BASED ANALYSIS FOR ‘BROKERING PATTERNS’

In this section, we demonstrate the analysis and classification of ‘brokering pattern’ through Two-Ways classification scheme using attribute-based analysis. In other words, the pattern description is analysed using the attribute analysis table of Table 1 to Table 4. A sample to demonstrate the analysis and classification of brokering pattern through the Two-ways classification scheme using attribute-based analysis is presented first. This is followed by the result of the analysis of 58 ‘brokering patterns’. At the end of this section, we present observations on the result of the classification.

In the example, we present the source of the 'brokering pattern', overview of the pattern, justification of attribute-based analysis and result of the classification. In addition, the example presents the experience that has been shared in the pattern documentation.

Source: (Lau and Goh 2002)

Overview: The system presents the brokering of the e-marketplace through the broker model. The broker model consists of formulization of the brokering problem by treating it as a set packing problem and solved through a proposed iterative greedy approximation algorithm.

Solution:

```
FUNCTION LOCAL_SEARCH(set S)
    TempSBest ← S;
    FOR EACH COLUMN j in Candidates DO
        TempS ← RemoveConflict(S, j);
        TempS ← TempS + {j};
        TempS ← GREEDY(TempS);
    IF (Value(TempS) ≥ Value(TempSBest)) THEN
            TempSBest ← TempS;
    ENDIF
    ENDFOR
    RETURN TempSBest;
```

Figure 3: Functions that present the algorithm in handling the brokering problem as described.

Viewpoint abstraction layer attributes analysis: The pattern presented in Figure 3 was analyzed using the level attribute table (Table 1) in order to determine the level that the pattern belongs to. Figure 3 records the detail description to identify best fit for a particular agent. From the analysis, we can identify the attribute for this pattern include 'agent internal activity'. Although it does not indicate as a software agent, the function component that presented a detail design on part of the system internal activity can relate to the attribute of agent. As a result, we conclude that the pattern belongs to the PIM-level.

Viewpoint aspect attributes analysis: The pattern presented in Figure 3 was analyzed using the category attribute table (Table 2, Table 3 or Table 4) in order to determine the viewpoint aspect that the pattern belongs to. At the PIM-level, the set of attributes for the pattern in Figure ks17c include agent, resource (e.g. input on the function), reasoning (e.g. influence step, control structure of for loop, if condition) and proactive (internal influence, if value). Hence, it belongs to the behaviour category.

From the analysis and classify result, the knowledge and experience that have been shared in this 'brokering pattern' belongs to the PIM level behaviour category as it contain a detail description of broker agent activity

Table 5: Result of the classification of 'brokering pattern'

Viewpoint abstraction layer	Viewpoint aspect		
	Information	Interaction	Behaviour
CIM			
PIM			Figure 3
PSM			

Continuing the above practice, we conduct a further study to analyse and classify 58 'brokering patterns'. The reader can request the sample collection of the 'brokering patterns' by contacting the authors. The document contain the references for the 'brokering patterns'.

Result of the analysis of 'brokering patterns' and observation

Table 6 shows the comprehensive view of the 'brokering patterns'. From the table, each figure represents the identification of a brokering pattern that has been shared within a particular article or pattern documentation. Each experience that has been shared will label as 'X', in which each figure contains the pattern description that has been structured in the article accordingly. In addition, each figure will capture different representation (e.g. algorithm, architecture etc.) that has been used by pattern designer to describe the shared knowledge. For example, from the Table 6, we compiled the shared experience with figures 1a and 1b. Both figures shows the experience in engineering the brokering system in different sections of the article accordingly. In addition, both figures show different representations that have been used by pattern designer in sharing the development experience of brokering system.

From the result of the analysis, it presents what might be shareable from the 'brokering patterns'. On the other hand, it is interesting to observe that the knowledge and experience in engineering the brokering system that are shared across the various articles do not present in clear structure, indirect description and some of the useful description is seem implicit to the developer. Our observation can further describe with the following description.

- Within a particular 'brokering pattern', the pattern designer will share the analysis of the brokering system, followed by the design view and continue to share the analysis view and so on. To link from one representation to the others and switch the description from one view to the others within the pattern description has created some confusion and difficulty. The patterns that happen in these kinds are knowledge source of ks2, ks3, ks4, ks5, ks8, ks9, ks11, ks12, ks14, ks15, ks16, ks17, ks22, ks27, ks30, ks33, ks37, ks38, ks39, ks40, ks42, ks43, ks45, ks47, ks51, ks52, ks53 and ks57.

Table 6: A Comprehensive view of 58 ‘brokering patterns’ in Two-ways classification scheme

Vertical dimension/	**Horizontal dimension/ Viewpoint aspect**																								
	Information								***Interaction***								***Behaviour***							
Conceptual domain modelling			3c	4c						2a						8a								
					13d		15c										9b		11e	12b	13a	14b		16a
											11d	20a		14a		16c			19a					
														30a			25a		27a					
										34a				38a		40c								
	41a																							
																						54a		
																	57a							
Platform-independent computational design						6b	7a		1a		3a		5b			8c	1b	2b	3c	4a, b	5a, c	6a	7b	8a, b
	9c				13d					10a	11b				15a		9a, d	10b ,c	11a	12a	13c	14b ,c,d	15b	16b
							23a		17a				21b	22a	23c		17b ,c	18a	19b	20b	21a		23b	24a
	25c			28c			31b			26a		28b				32a	25b	26b	27c	28a	29a	30b ,d	31a	32b
	33b	34c							33a, b					38c	39a		33c ,d	34b		36a	37a ,b	38b ,d	39c	40b
										41b				46b		48a		42b ,c	43b	44a	45a ,b,c	46a	47a ,b	
						54b			49a					54e			49b		51a ,b	52a ,b	53a	54 d		56a
	57b																57a ,c	58a						
Platform-specific design and implementation																8d								
											11c														
														22b, c						20c				
								32b						30e					27b ,d,e			30c	31c	
														38e		40a			35a				39b	
	43 a															48b		42a						
					53b		55a										49c	50a			53b ,c		55b	56 b
						54c																		

- Some knowledge is shared explicitly and some is kept implicitly. This can look to the distribution of the knowledge has been shared across the dimensions of the Two-ways classification scheme as shown in Table 6. For example, some patterns explicitly shared the information that was required for the system; internal structure or behaviour of the system; external component or person that it relies upon for task accomplishment. As a result, we face difficulty in understanding the implicit knowledge that has been described in the patterns.
- A similar aspect of the system is shared in different representations in a pattern. For example, pattern of knowledge source, ks10 has described the behaviour of the system in the architecture view and pseudo-code. Other patterns are ks8, ks10, ks14, ks17, ks22, ks27, ks33, ks37, ks38, ks42, ks45, ks47, ks51, ks52, ks53 and ks54. This will lead to some confusion and extra effort is needed to learn the representations that are used for describing the system.

DISCUSSION AND RELATED WORKS

Various pattern classifications have been introduced for patterns. Gamma et al introduce several pattern catalogs for design patterns (Gamma, Helm et al. 1995). A comprehensive pattern classification scheme for agent patterns was introduced by Oluyomi in the University of Melbourne AgentLab (Oluyomi, Karunasekera et al. 2007). We extended Oluyomi's pattern classification to introduce a more comprehensive and simpler classification scheme for patterns. The pattern classification allows the ease of accessibility of patterns by user and pattern designer (Oluyomi, Karunasekera et al. 2007). From the proposed classification scheme, we demonstrate how the software practitioners are able to recognize what state of software engineering the patterns relate to and are able to relate the patterns from one dimension to the others. In addition, the result of the classification enable us to justify that the experience of different development groups for the case of brokering mechanism is described very differently.

We believe that it is hard to evaluate each of the classification scheme. The best we can offer is to demonstrate how the classification scheme is used to analyse and classify 58 brokering patterns.

CONCLUSIONS

An investigation of task knowledge patterns that have been shared across various articles is conducted in this paper. It is interesting to observe that the experience in engineering the brokering system given in the 'brokering patterns' documentations is not presented with clear structure, has indirect description and some of the useful description seems implicit to the developer.

We presented a comprehensive view of the 'brokering patterns' based on the analysis and classification of the 'brokering patterns' through the Two-ways classification scheme using attribute-based analysis. We demonstrated how the Two-ways classification scheme can be adopted to analyse and classify the 'brokering patterns' that ranged from an agent oriented software system to a non-agent related system. Hence, it can turn into a simulation platform to visualize the knowledge and experience that have been shared in a wider spectrum of articles. Continuing studies are being conducted to demonstrate the Two-ways classification scheme to analyse and classify agent patterns; to demonstrate the usage of the result's classification in guiding the pattern writing (WaiShiang 2010).

REFERENCES

De Wolf, T. and T. Holvoet (2006). A catalogue of decentralised coordination mechanisms for designing self-organising emergent applications. Technical Report, Department of Computer Science, K.U. Leuven.

Gamma, E., R. Helm, et al. (1995). Design patterns: elements of reusable object-oriented software, Addison-Wesley.

Lau, H. C. and Y. G. Goh (2002). An intelligent brokering system to support multiagent web-based 4th-party logistics. Proceedings 14th IEEE International Conference on Tools with Artificial Intelligence.

Oluyomi, A., S. Karunasekera, et al. (2007). "A comprehensive view of agent-oriented patterns." Autonomous Agents and Multi-Agent Systems **15**(3): 337-377.

Schreiber, G. (2000). Knowledge engineering and management: the CommonKADS methodology, MIT press.

Sterling, L. and K. Taveter (2008). The Art of Agent Oriented Modelling, MIT Press, Cambridge.

Uschold, M. and M. Gruninger (1996). "Ontologies: Principles, methods and applications." The Knowledge Engineering Review **11**(2): 93-136.

WaiShiang, C. (2010). Patterns in agent oriented software development. Department of Computer Science and Software Engineering, The University of Melbourne PhD thesis.

AUTHORS' BIOGRAPHIES

Leon Sterling was born in Melbourne, Australia. He completed a B.Sc.(Hons.) at the University of Melbourne, and a Ph.D. in computational group theory at the Australian National University. He has worked at universities in the UK, Israel and the US. In 1995, he returned to the University of Melbourne, and founded the University of Melbourne Intelligent Agent Lab. He moved in 2010 to become Dean of the Faculty of Information and Communication Technologies at Swinburne University of Technology.

Cheah WaiShiang is currently a PhD student at the University of Melbourne, Australia.