International Journal of Software Engineering & Computer Sciences (IJSECS)
ISSN: 2289-8522 ,Volume 1, pp. 55-76, February 2015
©Universiti Malaysia Pahang
DOI: http://dx.doi.org/10.15282/ijsecs.1.2015.5.0005

IMPROVED TWO-WAYS CLASSIFICATION FOR AGENT PATTERNS

Wai Shiang Cheah, Azman Bujang, Masli, Edwin Mit, and Alfian Abdul Halin

Faculty of Computer Science and Information Technology,
Universiti Malaysia Sarawak,
94300 Kota Samarahan, Sarawak, Malaysia
Email: wscheah@fit.unimas.my

ABSTRACT

Agent technology has been used in building various domain specific applications. The agent methodologies are proposed to aid the agent developer with the introduction of techniques, terminology, notation and guidelines during the development of the agent system. Alternatively, agent patterns have been introduced by sharing the experience of engineering the agent system and allow the novices to solve the problem in a more systematic and structured way. To ease the accessibility of agent patterns, various catalogs or pattern classifications have been introduced. The pattern classification allows the user to find the patterns by organizing the patterns within a particular catalog. To date, various styles of pattern classification have been introduced. We argue that those styles are still insufficient to classify the current pools of agent patterns. This paper presents a classification scheme for agent patterns. It is an improvement of the existing pattern classifications. The improved classification was able to classify 204 agent patterns, which indirectly will ease of pattern selection in multi agent system development. In addition, we show the usage of pattern classification in determine the quality of the agent patterns. In fact, it is the first report that shows the quality of the agent patterns to date.

Keywords: Word agent patterns; classification; design quality.

INTRODUCTION

Agent technology has been used in building various domain specific applications. The agent paradigm introduces a software entity (e.g. agent) that is autonomous, proactive and able to interact with other agents for task accomplishment (Gonzalez and Luck, 2004; Henderson and Giorgini, 2005). This kind of software supports sophisticated applications like ambient intelligence, e-business, peer-to-peer, bioinformatics which demand the software to be robust (Ren and Anumba, 2004), effective (Marik and McFarlane, 2005), co-operative to wide environments (González et al., 2009), customizable to support user needs (Sánchez et al., 1998; Pavlıcek et al., 2007), secure, and to evolve over time to cope with changing requirements (Henderson and Giorgini, 2005; Munroe et al., 2006).

Lot of agent methodologies is proposed to aid the agent developer with the introduction of technique, terminology, notation and guideline during the development of the agent system (Koutsabasis and Darzentas, 2009). For example, in ROADMAP methodology (Juan, 2002), we first model roles and goals for an agent system. This is followed by producing interaction models, environment models, protocol models and agent models and service models prior to implementation. It is an iterative process to refine the models until sufficient information has been captured for the system. The roles and goals detail high level abstraction of information which caters to the job position and the tasks of accomplishing from the problem given. The protocol and interaction models support the social interaction for an agent system. The

environment models and knowledge models capture the sources that are required for the system. The agent model and service model consist of a design decision on agent behaviour, classes to implement the software agent.

Alternatively, one idea to help people start the agent development pragmatically is through patterns (Wai Shiang, 2010). Pattern is a platform for sharing the development experience and allow the developer to reuse the development experience that occurs again and again. It allows the novices to rely on expert knowledge and solves the problem in a more systematic and structured way. Patterns have shared the recurring problem and solution to prevent the developer from reinventing the wheel for application development. The use of patterns in agent development can reduce the development cost and time, promote reuse and reduce the complexity when developing applications.

To support the adoption of agent patterns, researchers are working on pattern classification. The pattern classification supports the ease of accessibility of agent patterns by arranging the collection of agent patterns in a structured manner.

Patterns are organized and synthesized into a particular collection, classes or catalogues. The pattern classification is used to support the ease of accessible on patterns by users and software developers (Tichy, 1997; Coninx, 2002; Lind, 2003; Schumacher, 2003; Yan, 2004; Sauvage, 2004). The pattern classification allows the user to search for the agent patterns for their problem at hand and it allows the pattern designer or writer to place the pattern in a particular class. In addition, the pattern classification supports the understanding on the domain and application of each pattern, to identify new patterns as well as differentiating among the patterns (Aridor, 1998).

To date, various styles of pattern classification have been introduced. We argue that those styles are still insufficient to classify the current pools of agent patterns. This paper conducts a survey on agent pattern classification. Based on our knowledge, it is the first paper that presents the survey of agent pattern classification. From the survey, we present the inadequacy on current classification and present an improved pattern classification for agent patterns. The improved classification was able to classify 204 agent patterns. In addition, we demonstrate how the use of the improved classification in determining the design quality of agent patterns.

Section 2 presents the classification of agent patterns. It consists of the description on pattern classification which we ranged from three styles. They are cataloging, layered and classification scheme. Section 3 presents the issues on current pattern classifications. We first present the collection of agent patterns from 1992 until year 2010. It is a pool of patterns that allow the software developer to adopt agent technology. Then, we conduct a qualitative study on reusing those patterns and describe the gaps on recent pattern classification. From the gaps that are identified in Section 3, we present an improved pattern classification in Section 4. In section 4, an improved pattern classification is presented. It consists of new dimensions, attributes to analysis and classifies agent patterns. In addition, we present an example on how to analysis and classify a pattern and the result of the classification for 204 agent patterns. It is interesting to report that the result of the classification able to conclude the current practices on agent patterns as described in Section 5. Also, it drives the discussion especially on the design quality of agent patterns based on the conclusion given. This paper is concluded in Section 6.

CLASSIFICATION OF AGENT PATTERNS

Works have been done to propose the classification on agent patterns. They are Y. Aridor (1998), Elizaberth A. Kendall (1997), Hayden et al (1999), Tahara et al (1999), Shu et al (1999),

Schelfthout et al (2002), Lind (2003), Sauvage (2004), M. Weiss (2004), Modak (2004), De Wolf et al (2006), and Oluyomi (2007).

We grouped the existing works on agent pattern classifications into three styles based on the nature of the discussion in the pattern classification literature. They are simple catalogs, layered and classification scheme as further described below.

CATALOGING

Y. Aridor (1998) presents three classes for agent design patterns. They are travelling patterns, task patterns and interaction patterns. The travelling patterns consist of patterns that share the knowledge of mobility management. The task patterns consist of patterns that share the knowledge of tasks delegated to an agent system and the interaction patterns share the knowledge of agent interaction. Hayden et al (1999) group the co-ordination patterns into four basic architecture styles- hierarchical, federated, peer to peer and agent peer. Shu et al (1999) classify the agent design patterns into two types. They are architecture patterns and method or component patterns. Schelfthout et al (2002) classify the agent patterns into two dimensions. They are context and realization. The context dimension consists of categories like agent, environment and multi agent. The realization dimension consists of categories like system and aspect. Lind (2003) proposes seven views for agent pattern catalog. They are interaction, role, architecture, society, system, task and environment. Sauvage (2004) classifies the patterns into four categories. They are metapatterns, metaphoric pattern, architecture pattern and antipatterns. The metapatterns consist of patterns at a higher conceptual level; metaphoric consists of behaviour patterns like agent communication; architecture patterns consists of common agent architecture like BDI and antipatterns consist of patterns that handle issues like redesign. A mobile agent pattern catalog is presented in (Modak, 2004) and co-ordination catalogs are presented in (Wolfm, 2006).

THE LAYER APPROACH

Elizaberth A. Kendall (1997) propose a layered agent architecture pattern for agent systems as shown in Figure 1. An agent has been decomposed and structured into seven layers. They are sensory, belief, reasoning, action, collaboration, translation and mobility, which demonstrate agent system behaviour. Each layer handles the incoming and outgoing elements from the layer above and below it and populate with corresponding agent patterns. For example, the intention pattern falls within the action layer and the proxy pattern falls within the translation layer.

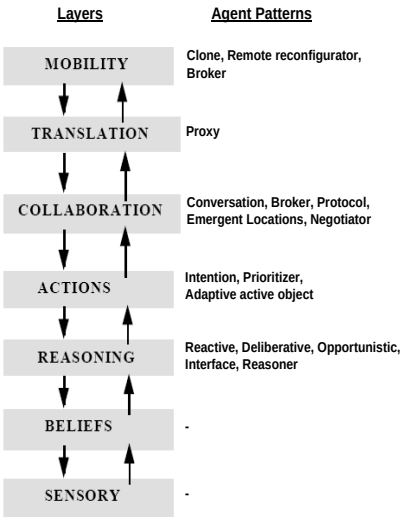

Figure 1. The layered agent architecture patterns

Tahara et al (1999) propose three layers of architectural level to classify patterns based on high level design and low level detailed design, as shown in Table 1. The macro-architecture level presents the outline of the system and agent behaviour. It is generic and independent on any specific agent platforms. The micro-architecture level and object oriented level present the detail of the system and agent behaviour; the former is focused on a specified agent platform and the latter is focused on program code.

Table 1. Layered agent patterns

Macroarchitecture patterns	Microarchitecture patterns	OO design patterns
Basic mobility---------- ---No continuation ---Shallow Continuation ---Deep Continuation	Migration ----------	--Mediator, strategy, state
Itinerary and Star-Shaped-	Migration----------	--Mediator, strategy, state --Proxy
Branching---------- ---Synchronous Fusion ---Selective Fusion	--Migration --Copying	--Mediator, Strategy, State --Builder, Factory Method, Template Method

CLASSIFICATION SCHEME

Although it is not described in detail, M. Weiss (2004) suggests a classification of agent patterns in two dimensions. The horizontal dimension focuses on the domain involved for the agent application. The vertical dimension focuses on agent design activity like design agent internal structure, design agent system and various design process.

In year 2007, a comprehensive pattern classification has been presented by Oluyomi et al (Oluyomi et al., 2007). Oluyomi proposes a comprehensive framework to analyse, classify and describe the agent pattern. A platform, attributes and process to classify and analyse agent patterns are introduced to produce the comprehensive view of agent patterns. A platform known as Two-ways classification scheme is introduced as shown in Table 1.

Table 1. Two ways classification scheme

Agent oriented analysis (Level 1)	Organizational	Interactional	
Multiagent System architecture (Level 2)	Definitional	Structural	Interactional
Agent Internal architecture (Level 3)	Structural	Interactional	Strategic
Multiagent System realization (Level 4)	Definitional	Others	

The Two-ways classification scheme consists of a vertical dimension and a horizontal dimension. The vertical dimension presents the stages of agent oriented software development. The stages of the agent oriented software development are agent oriented analysis level, multiagent system architecture level, agent internal architecture level and multi-agent system realization level. The agent oriented analysis level reflects the requirement and analysis of the agent system. Patterns that classify at this level consist of approaches to goal analysis, role analysis, role to role relationship analysis or role to interaction relationship. The multiagent system architecture level reflects the specification of the multi agent system design view. Patterns that classify at this level consist of approaches to organize the structure of multi agent system or handling interfacing with the agent interactions and environment. The agent internal architecture level reflects the specification of the individual agent's design view. The patterns that classify at this level consist of approaches to identify agent components, determine agent interaction strategy or behaviour strategy. The multi agent system (MAS) realization level reflects the MAS detail design and implementation, however, no detailed description is provided from the Oluyomi work.

The horizontal dimension presents the agent development tasks at each of these stages. The tasks are further grouped into categories. At the agent oriented analysis level, the organization category contains patterns that describe the organization of MAS into logical units. The interactional category contains patterns that describe the decision to determine the interaction of the members within an organization. At the multiagent system architecture level, the category of definition contains patterns that describe approaches for mapping the role to agents. The category of structural contains patterns that describe approaches in designing MAS internal structure like the agent communication component.

The category of interactional contains of patterns that describe approaches for the interaction within the logical unit modelled in the MAS system. At the agent internal architecture level, the category of structural contains patterns that define the design of internal structure for an individual agent. The category of interactional contains patterns that handle the agent interaction as well as interaction of the components. The category of strategy contains patterns that describe approach in designing agent types.

To classify an agent pattern through the Two-ways classification scheme, each dimension is assigned with attributes. The attributes are derived from the literature on agent oriented systems. The attributes are used in attribute-based analysis. The attribute-based analysis is used to determine into which dimension (e.g. level and category of Two-ways classification scheme) a pattern is belonging to based on the identification of pattern's level attributes and identification of pattern's category attributes.

Analysis processes are proposed for attribute-based analysis. It involves processes to examine the pattern (e.g. read on the pattern description); identification of the pattern's level attributes; determine in which level the pattern belong to based on the result from the identification of pattern's level attributes; identification of the pattern's category attributes;

determine in which category the pattern belong to based on the result from the identification of pattern's category attributes. During the analysis processes, identification of the attributes within the pattern description is based on human intuition. The determination of the level and category to which a pattern belongs to is based on the maximum number of appearances for the attributes that have been matched within the Two-ways classification scheme.

ISSUES ON THE CLASSIFICATION SCHEME

A collection of agent patterns are compiled based on our survey of existing agent patterns through various digital libraries as well as the World Wide Web (Wai Shiang, 2010). To ease of adoption of those patterns, we are trying to classify them into the existing pattern classification prior reuse and use the patterns in multi agent system development. We think it is a straightforward process, but it is not at all. We have identified some insufficient as reporting below.

• Some catalogs are too specific. Modak (2004)'s work is focused on mobile agent patterns where as Dewolf work is focused on coordination patterns.

• Insufficient in Oluyomi classification scheme (Oluyomi et al., 2007):- 1) The inconsistency of horizontal dimensions that are presented in the Two-ways classification scheme. 2) The missing attributes of the 'Others' category have caused uncertainty. When conducting the analysis, one always tends to think and guess about the 'missing' categories. 3) Redundancy of the attributes that have been introduced. 4) Confusion about the intended meanings of the attributes and dimensions.

• Some catalogs are too general. For example, does the matchmaker pattern, P11, is considered as method patterns?

This leads to the ambiguities on adopting the patterns in multi agent system development. The ambiguities presented above have led us to explore new dimensions and attributes for the classification scheme. We argue that there is a need for new dimensions that can introduce a more unified and consistent classification scheme. The dimensions must be precise in indicating what has been done and they must introduce cross cutting concerns with other dimensions. In other words, it should be easy to differentiate one dimension from the other to allow for easy accessibility of the patterns.

IMPROVED CLASSIFICATION SCHEME FOR AGENT PATTERNS

Table 6. Two-ways classification scheme for pattern classification

		Aspect Dimension		
Software Process Dimension	Conceptual independent modelling	Information	Interaction	Behaviour
	Platform independent design & modelling	Information	Interaction	Behaviour
	Platform specific design & modelling	Information	Interaction	Behaviour

The improved Two-way classification scheme for pattern classification is shown in Table 6. The classification scheme is structured along two dimensions analogously to Oluyomi's Two-

ways classification scheme. Thanks to the viewpoint framework that has been proposed in (Sterling, 2009), we can adopt the viewpoint abstraction layers and viewpoint aspects of the framework into our context. We believe that the elements from the framework are well defined and the researchers have put much effort into producing the framework. The viewpoint framework consists of viewpoint abstraction layers and viewpoint aspects and presents a unified and consistent view for engineering a socio-technical system (e.g. agent oriented software system).

We can classify the patterns according to the software process dimension and aspect dimension as shown in the Table 6. The software process dimension consists of three levels. They are the levels of conceptual domain modelling or conceptual domain modelling, platform independent computational design, and platform specific design and implementation. This is consistent with Model Driven Architecture (MDA) and involves modelling a socio-technical system (e.g., agent system) from a motivation layer, system design layer and deployment layer. It includes the owner's, designer's and builder's perspectives on systems engineering. The aspect dimension of the classification scheme consists of three categories: information, interaction and behaviour. The aspect dimension presents a crosscutting of the tasks at each level of MAS development.

In the following discussion, we provide a description of the dimensions within the Two-ways classification scheme. We further elaborate on the new set of dimensions for the classification scheme and provide an informal definition for each dimension. The definition is derived based on our study of agent oriented literature together with our effort to classify and analyse a collection of 204 agent patterns that is presented in this paper.

SOFTWARE PROCESS DIMENSION OF THE CLASSIFICATION SCHEME

The software process dimension for the new classification scheme contains three levels. They are the levels of conceptual domain modelling (CIM), platform independent computation design (PIM) and platform specific design and implementation (PSM). Accordingly, the levels correspond to how the system is motivated, how the system is designed and how the system is designed for a specific platform and situated in an environment. The corresponding patterns are those that record the experience in analysing the problem domain of a MAS, patterns that record the experience in designing a MAS, and patterns that record the experience in implementing a MAS. The definition of each of the levels is presented next. The definition further elaborates each of the levels.

A. CONCEPTUAL DOMAIN MODELLING LEVEL

The conceptual domain modelling level reflects the early requirements and analysis for an agent oriented system. Patterns at this level address the approaches to elicit the requirements of and analyse the problem domain of an agent oriented system. Requirements elicitation and analysis is a common stage among various agent oriented methodologies to model an agent system at a higher level of abstraction and to understand and analyse the requirements for developing an agent system. It is intended to present an overview of the system and determine its functionalities. Ignoring it can lead to misunderstanding about the system in design. Also, analysis normally involves the activities that provide the context in which the system is to be designed (Bresciani, 2004).

B. PLATFORM INDEPENDENT COMPUTATIONAL DESIGN LEVEL

The platform independent computational design level (PIM) reflects the design of an agent system. The patterns at this level record the experience in designing an agent oriented software system. This level is also known as a system design layer. The pattern descriptions given within this dimension are closer to the designer perspective. The pattern descriptions given do not consider any specific tools, platforms or software architectures. The "*PIM shows part of the system design specification that does not change from one platform to the other*" (Sterling, 2009).

C. PLATFORM SPECIFIC DESIGN AND IMPLEMENTATION LEVEL

The platform specific design and implementation level (PSM) reflects the detailed design of an agent system using a specific technology, architecture or agent platform. This level can be related to the micro-architecture level that was introduced by Tahara et al (1999) and the micro-level by Bresciani et al (2004).

Patterns at this level record the experience in platform specific design or implementation of an agent oriented software system. The pattern descriptions given are platform dependent in which a platform denotes a set of subsystems and technologies that provide a coherent set of functionality through interfaces and specific usage patterns (Sterling, 2009). The platform specific design and implementation level provides design details that "*specify how the system is to be implemented in a specific platform, architecture and tool or programming language"* (Sterling, 2009).

In the previous description, we presented the software process dimension and the definition of each level of the software process dimension. Generally, the conceptual domain modelling level models how the system is motivated; the platform independent computation design level models how the system is designed, and the platform specific design and implementation level models how the system is implemented and situated in an environment.

Apart from knowing MAS development stages as shown within the software process dimension, each abstraction level needs to be elaborated for MAS development. The aspect dimension serves this purpose. The aspect dimension provides explicitness and captures various abstractions that are introduced at each level of MAS development. Such abstractions are reflected by agent development tasks at each of the levels in the software process dimension as described next.

ASPECT DIMENSION OF THE CLASSIFICATION SCHEME

In the Two-ways classification scheme, the aspect dimension represents detailed tasks on each of the levels of developing a socio-technical system (e.g., agent system). We can interpret that the tasks are what people intend to do when modelling and designing a socio-technical system. For example, tasks like describing the roles required in the system, the responsibilities that are played by each role, the relationship between the roles being played in solving a problem, and so on. Accordingly, the tasks can be grouped into three categories. They are the categories of information, interaction and behaviour.

From the agent perspective, the information category corresponds to modelling the domain knowledge that is required for an agent system; the interaction category is related to modelling collaboration/interactions between agents of the system, and the behaviour category is related to modelling internal behaviours of agents.

In the following section, we list the categories at each of the three levels. In addition, we provide an informal definition of each category. The informal definition is derived based on our study of agent concepts that is described within the viewpoint framework and agent concepts that are described in various agent oriented methodologies.

A. CONCEPTUAL DOMAIN MODELLING LEVEL

The aspect dimension at this level consists of categories that classify the patterns into conceptual domain modelling level- information patterns, conceptual domain modelling level- interaction patterns and conceptual domain modelling level- behaviour. The categorization reflects tasks to conceptualize an agent system. The information category at the CIM level reflects the patterns that record experience to analyse the domain entities for an agent system; the interaction category at the CIM level reflects the patterns that record experience to analyse the roles played in the organization to solve a problem and the behaviour category at the CIM level reflects the patterns that record experience to analyse agent behaviours. In the following description, we provide an informal definition of each of the categories mentioned.

Conceptual Domain Modelling- Information Patterns

The patterns in this category record experience to analyse the domain entities for an agent system. The patterns describe the approaches to identify and structure domain entities or resources that are required for an agent oriented system. A resource, also known as asset or task specific terms or use-specific knowledge type in CommonKADS, is needed by the worker to perform a specific task or process within an organization. Resources can range through different types such as an information system, equipment, materials, technology, patents, and rights. For example, components, parameters, constraints and calculation values are needed for each configuration design problem. Identification of the resources or domain entities occurs to determine the purpose and scope of the ontology and analyse the information for use in the ontology (Dileo, 2002).

Conceptual Domain Modelling- Interaction Patterns

The patterns in this category record experience to analyse the roles played in the organization. The patterns describe the determination and arrangement of role(s) and the responsibilities and organizational structure for handling an application domain. People playing role within an organization work together in groups and grouping may happen in various structures like hierarchical, flat, and so on. This involves study of the relationships among the people that play roles within an organization.

Conceptual Domain Modelling- Behaviour Patterns

The patterns in this category record experience to analyse agent behaviours. The patterns describe the identification and appliance of high level goals for solving a particular problem. For example, in brokering, achieving the highest level goal involves activities like selecting, broadcasting, comprising, ranking, and presenting. Those activities can be linked to sub goals that further determine what needs to be achieved by an agent system.

B. PLATFORM INDEPENDENT COMPUTATIONAL DESIGN LEVEL

The aspect dimension at this level consists of categories that classify the patterns into platform independent computational design level- information patterns, platform independent computational design level- interaction patterns and platform independent computational design level- behaviour patterns. It reflects tasks to produce the detailed computational design of an agent system. Within the context of pattern, the information category at the PIM level reflects the patterns that record experience in designing knowledge entities to be used by an agent system; the interaction category at the PIM level reflects the patterns that record experience in designing agent interaction and the behaviour category at the PIM level reflects the patterns that record experience in designing agent behaviour. In the following description, we provide an informal definition of each of the categories at this level.

Platform Independent Computational Design- Information Patterns

The patterns in this category describe approaches to handling the conceptualization of the knowledge model or producing an information domain specification for the agent system. Hence, this involves dealing with designing the ontology. The ontology design involves conceptualization, which consists of steps like modelling of concept hierarchies, aggregation and generalization; modelling of concept roles, class, property, modelling of class and property hierarchies; extension of the conceptual property schema (Husemann, 2005).

Platform Independent Computational Design- Interaction Patterns

The patterns in this category describe approaches in dealing with designing agent interactions. Issues such as handling agent collaboration, the arrangement of agents, deciding message exchanges, managing agent interaction components, and determining interaction protocols are taken into consideration in these patterns. Interaction protocol is a set of rules that govern the communication or conversation between software entities (e.g. agents) (FIPA 2000). It determines communication patterns and message sequences during agent conversation. Agents interact through exchanging agent messages expressed in an agent communication language such as FIPA ACL (FIPA 2000).

Platform Independent Computational Design- Behaviour Patterns

The patterns in this category describe approaches to handling the internal structure of agents which includes the issue of “intelligence”: an agent designed to be “intelligent” or should “intelligence” rather emerge from agent interactions. It involves the arrangement of agent internal components, the inference steps and strategy that is used by agent. This involves activities to decide the temporal order of and control over the actions, triggering events and decision making points.

C. PLATFORM SPECIFIC DESIGN AND IMPLEMENTATION LEVEL

The aspect dimension at this level consists of categories that classify the patterns into platform specific design and implementation level- information patterns, platform specific design and implementation level- interaction patterns and platform specific design and implementation level- behaviour patterns. It reflects tasks to produce lowest detailed design of an agent system. These tasks involve the realization of knowledge entities, interaction components, actions and control behaviour for an agent system. Within the context of patterns, the information category at

the PSM level reflects the patterns that record the experience in implementing agent interactions and communication; and the behaviour category at the PSM level reflects the patterns that record experience in implementing agent behaviours. In the following description, we provide an informal definition of each of the categories at this level.

Platform Specific Design and Implementation- Information Patterns

The patterns in this category record experience in implementing the information required by an agent system. The patterns record experience to conceptualize the concepts and properties of concepts in a specific semantic language. The patterns that belong to this aspect are also known as syntactic patterns. Syntactic patterns consist of an arrangement of representation symbols like concept, axiom and relation according to semantic languages like OIF/FaCT language system and F-Logic/SiLRi language system (Blomqvist, 2005).

Platform Specific Design and Implementation- Interaction Patterns

The patterns in this category provide the lowest detail design of agent interaction. For example, agent interaction and communication are designed through method invocation, class inheritance, and classes and attributes of object oriented technology. Another example is the pattern that records experience to describe the type of messages exchanged (e.g., properties of the message) in a particular message encoding language like FIPA-SL (FIPA 2000) and the use of proxy for message delivery.

Platform Specific Design and Implementation- Behaviour Patterns

The patterns in this category address approaches to constructing agent behaviours for an agent system. For example, behaviour patterns can record experience in designing code (e.g. Prolog programming) that implements the control structure of inference methods. Other examples are the realization of agent behaviours on the JADE platform through the behaviours provided by JADE like OneShotBehaviour and SequentialBehaviour [Bellifemine'08], and the realization of agent behaviours through Belief, Desire, and Intention (BDI) architecture.

In summary, we presented the improved Two-ways classification scheme for pattern classification. The dimensions, software process and aspect dimension of the Two-ways classification scheme were described together with an informal definition of each dimension. To classify and analyse agent patterns through the Two-ways classification scheme, the attribute-based analysis that was introduced by Oluyomi needs to be refined. The attribute-based analysis is used by Oluyomi for classifying agent patterns according to the dimensions of the Two-ways classification scheme. The attribute-based analysis is elaborated in the next section.

ATTRIBUTES FOR ATTRIBUTE-BASED ANALYSIS

Attribute-based analysis was proposed by Oluyomi in order to facilitate the classification and analyse agent patterns through Two-ways classification scheme. Attribute-based analysis is introduced to characterize and categorize the patterns through the Two-ways classification scheme (Oluyomi, 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 attributes represent the features or common knowledge elements that fall within a particular software process dimension and aspect dimension in the Two-ways classification

scheme. They represent the agent concepts that people use during the engineering of an agent-oriented system. Activities to identify the attributes for the Two-ways classification scheme are described in this section. We conducted several studies to identify the attributes to be used to characterize the software process and aspect dimensions for the new classification scheme. As mentioned before, the attributes reflect the agent concepts that feature in the practice of developing agent systems. As a result, it is sensible for us to identify the attributes by studying through conceptual space for socio-technical systems, various agent-oriented methodologies and agent oriented software development projects.

Table 7 shows the attributes that we identified for the improved Two-ways classification scheme. Since we had introduced new sets of dimensions for the Two-ways classification scheme, a refinement of attribute-based analysis was required. Each dimension covers a set of attributes that is used to classify and analyse agent patterns through attribute-based analysis.

In our work (Wai Shiang, 2010), we provide an informal definition for the attributes that can be populated within the software process dimension and aspect dimension of the improved Two-ways classification scheme. Reader can refer to (Wai Shiang, 2010) for the description of each of the attributes for attribute-based analysis.

Table 7. Attributes for improved Two-ways classification scheme

Software Process Dimension	***Aspect Dimension***		
	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 • Social order/ communication strategy • Message	• Agent type • Information type/ Service • 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

To analyse the patterns using attribute-based analysis, the processes of attribute-based analysis proposed by Oluyomi (2006) are refined. To classify and analyse the patterns, we first examine the shared experience towards discovery of its attributes. We review and observe the shared experience (e.g. pattern) in engineering the system, to review what might be shareable from the description (e.g. pattern description) given. This activity makes up *step1*. From the

observation, we proceed to the level attributes table analysis and category attributes table analysis. This involves the activities of placing the patterns at the appropriate level and category and indicating which of the level attributes and aspect attributes listed in the analysis table are included by the pattern, based on the examination of the pattern. These activities make up *step2* and *step3*. The attribute analysis tables are tools in tablet forms that are populated with attributes within the Two-ways classification scheme. Altogether, four sets of attribute-analysis tables are introduced. Each of the analysis tables consists of three sections. They are the *knowledge source* that records the input for analysis; the *level and category dimension* and the *attributes* under the dimension and the *analysis outcome*. The knowledge source consists of a pattern description that is used for evaluation. The dimension and attributes cater for the elements that are used for identification of the pattern's level and category attributes.

Finally, we conclude the analysis results based on the frequent occurrence of attributes within the dimensions of the Two-ways classification scheme. For example, from the analysis of the pattern*X* at the CIM level, we conclude that the attributes for the pattern*X* include *domain entities*, *resource* and *quality goal*. Hence, the pattern*X* includes one out of one possible attributes in the information category and two out five possible attributes in the behaviour category. Based on the frequency of attributes` occurrence for the pattern*X* (e.g., one out of one in the information category), we can conclude that the pattern*X* belongs to the CIM level-information category.

EXAMPLE OF CLASSIFICATION ON STRUCTURE IN 5 PATTERN

The example in this section demonstrates the classification and analysis of a pattern through the Two-ways classification scheme using attribute-based analysis.

Name: Structure in 5 Pattern

Source: (Kolp et al., 2003)

Intent: The purpose of this pattern is to present an organizational structure for an agent oriented software system.

Solution: The Structure in 5 Pattern is shown in Figure 2. According to the pattern, the organizational structure comprises five actors. They are the following ones: Apex, which acts on strategy management to provide strategic objectives and direction of the organization; Middle Agency, which performs supervision; Operational Core, which carries out basic tasks and operations required to achieve the overall purpose of the system; Support, which works on the logistics and co-ordination role and generate detailed plans for achieving strategic objectives.

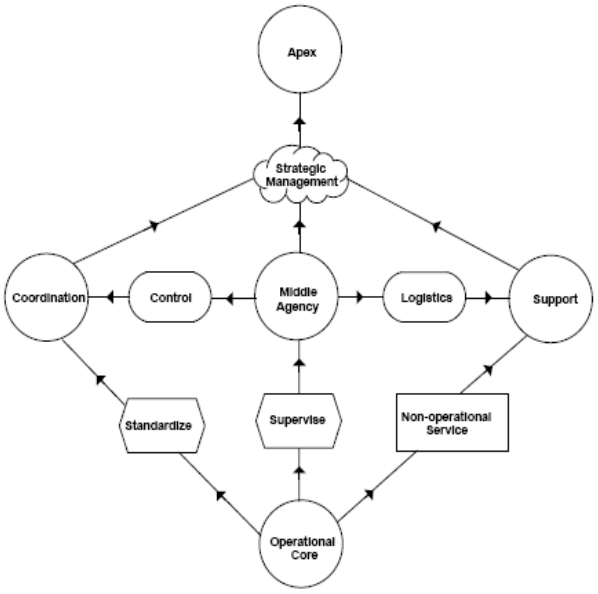

Figure 2. Structure in 5 for organization architecture pattern

Level attributes analysis: The Structure in 5 Pattern is analysed using the level attributes table in order to determine the level that the pattern belongs to. As shown in Figure 2, five roles have been included by the Structure in 5 and each role is organized according to its specific responsibilities within the organization. As a result, we can conclude based on the attributes 'identify role' and 'organization of roles' that this pattern belongs to the CIM level of the software process dimension.
Category attributes analysis: The Structure in 5 Pattern is analysed using the category attributes table in order to determine the category that the pattern belongs to. From the analysis, the pattern is identified with attributes like 'roles' (the five roles included by the pattern), 'organizational structure' (the dependencies between the roles), 'responsibility' (the responsibilities included by the roles) and 'social policy' (the dependencies between the roles that determine the authority relationships between them). Hence, the pattern belongs to interaction category.

We conducted the analysis and classification of the Structure in 5 Pattern. From the level attributes analysis and category attributes analysis, we can conclude that the *Structure in 5 Pattern* belongs to the conceptual domain modelling level- interaction category. In other words, this pattern records the experience of analysing the roles to be played in an agent system and the relationships between them. Altogether the result of the classification for 204 agent patterns through the Two-ways classification scheme is described in the next section.

CLASSIFICATION OF 204 PATTERNS THROUGH TWO-WAYS CLASSIFICATION SCHEME

In this section, we present the result of the classification of 204 agent patterns through the Two-ways classification scheme as shown in Table 8. The result shows to which dimensions a particular pattern belongs. In other words, the table expresses that a pattern belongs either to the analysis level, platform independent design level, or platform specific design and implementation level, and that a pattern belongs to the information, interaction, or behaviour category.

From the result of the classification, we can use and reuse the design of the general pedagogical agent and co-learner, P62, when we want to develop agent oriented education system; we can design the agent negotiation through agent negotiation architecture, P51, and so on. This is the common practice among the agent developer when adopting patterns for MAS development. On the other hand, we can identify the design quality of the agent patterns based on the classification result as described in the following section.

Table 8. Result of the classification of 204 agent patterns through Two-ways classification scheme

	Aspect Dimension		
	Information	***Interaction***	***Behaviour***
Conceptual domain modelling	P70. General use case.	P7. Hierarchical organization, holarchical organization, coalition-based organization, congregations of agents, agent federation, markets, matrix organization, compound organization, agent society. P26. Agency guard, sandbox, agent authentication, access controller. P33. Pipeline. P44. Flat structure, structure in 5, pyramid, join venture, bidding, take over, arm-length agreement, hierarchical contracting, vertical integration, co-optation, booking pattern(social diagram), monitor, matchmaker, mediator. P49. Ontology perception (responsibility), social merging pattern.	P28. Proxy bidding, dispute resolution, payment, fraud detection. P51. Implicit organization.

	Aspect Dimension		
	Information	***Interaction***	***Behaviour***
Platform-independent computational design	P15. Content reference model. P17. Timeline hierarchy. P58. Locally inverse relation. P59. Pheromone, gradient coordination, market based, tag based, token based coordination.	P8. One shot query, event driven subscription, temporal analysis interaction pattern. P10. SCADA control sequence, optimization. P12. Distributor mediator, conversation observer, delegate mediator. P14. Blackboard, market maker, master slave, negotiating agent pattern, intraplace communication pattern, mobile place communication pattern. P19. Timeout, refuse, cancel, failure. P23. Monitor, personal assistance, global assistance. P34. Contract net protocol. P37. Agent communication protocol. P42. Request pattern P44. Booking pattern (communication diagram), meeting, monitor, matchmaker, mediator. P45. Trust facilitator pattern, total disclosure, incremental disclosure. P46. Agent as delegate, common vocabulary, agent as mediator. P47. Assert, do action with Ack, assert with ack, request & answer, request to anybody. P48. Receptionist, secretary pattern, communication session pattern, mobile session, antenna, private session. P49. Ontology perception (interaction mechanism), society merging patter. P54. Ontology negotiation protocol. P55. Broker, embassy, mediator, monitor, wrapper. P65. FIPA request IP. P66. Explanation interaction protocol	P3. Direct effect, influence for no reason, manipulation, signalling, revealing/denying. P5. Websiffer Agent system architecture. P6. Generic agent structure. P9. Q-learning. P13. Sentinel agent behaviour. P21. Agent interaction architecture. P22. Standard, silent, deadend, isolated. P24. Agent architecture. P30. Cycle based style, view based style, volcano architecture. P31. Event based blackboard architecture. P39. Sensible agent architecture. P41. Collective sort, evaporation, aggregation, diffusion. P42. Sequential share resource, generic agent pattern. P46. Cataloging, user agent, buyer agent, seller agent, user profile, notification. P49. Ontology perception (task diagram). P50. Timeout protocol. P51. Agent negotiation architecture. P53. Pull, push, index, traveller. P56. Conversation agent. P57. Mark. P60. Query agent pattern. P61. Lifecycle pattern. P62. General pedagogical agent, co-learner. P63. Agent architecture for large scale environment. P64. Interface agent, arbitrary agent, matchmaker, broker, task agent. P69. Basic mobility.

	Aspect Dimension		
	Information	***Interaction***	***Behaviour***
Platform specific design & implementation	P58. Locally inverse relation described in language system.	P14. Meeting P16. Master slave, meeting. P18. Decentralized agent collaboration, broker. P20. Meeting pattern, meeting pattern in aglet platform, grasshopper platform, jade platform. P25. Facilitator agent design pattern. P40. Environment mediated communication pattern. P68. Master slave, meeting.	P1. Behaviour helper. P2. Mobility aspect pattern. P4. BDI architecture. P10. Agent interface, agent "SCADA" pattern, agent optimization pattern. P11. Matchmaker. P14. Message based communication pattern. P16. Itinerary, branching, notifier, messenger, star-shaped, finder. P18. Active object pattern, automated reasoning, access DB pattern. P27. Learning pattern, event-message handling, travelling. P29. Service client pattern, service representative. P30. INTERRAP(P.35) architecture, Desire GAM(P.38 DECAF), subsumption architecture. P32. EBDG, single minded communication pattern, relativised commitment goal, open minded commitment pattern. P36. Learning design pattern. P40. Synchronous, shared state, behaviour based decision. P43. JAM BDI architecture P45. INFFRA. P67. Market organizer, agent side comparison shopping agent, server side comparison shopping agent, itinerary balancing pattern. P68. Itinerary pattern.

DISCUSSION

The aim of the Two-ways classification scheme is to allow people to identify from the categorization what might be shareable. On the other hand, the results of the classification enable us to express more clearly what might be shared from the patterns. Within the Melbourne University AgentLab seminar, the result of the classification has been used as a tool enhancing discussions.

The results of the classification enable us to identify the design quality of agent patterns. Our results confirm the claim that a pattern classification allows a pattern designer to identify the overlapping among the patterns (Hafiz, 2006) and improve the new patterns which are being written. In addition, a classification reports on the density of agent patterns by indicating whether there is a sufficient amount of patterns in a particular category (Yoshioka, 2008).

Table 9. Density of agent patterns

Software Process Dimension	Aspect Dimension		
	Information	Interaction	Behavior
CIM	1	30	5
PIM	8	56	47
PSM	1	12	43

Table 9 shows the density of agent patterns by accumulating the number of patterns in each dimension. From the Table 9, it reports the density of the agent patterns by indicating whether there is a large amount of patterns in a particular category, a smaller amount of patterns and more need to improve the sufficiency of a particular category. For example, there are 99 interaction patterns have been proposed across various levels of software process dimension and it can be further categorized into 30 interaction patterns that fall under the analytical aspect of the agent system; 56 interaction patterns that described the design view of the agent interaction and 13 patterns that described the implementation view of the agent interaction.

It is shown that there is sufficient number of patterns that described the design level of multi agent system development. Among them, people focus on sharing the development experience of multi agent systems in designing agent interactions and agent behaviour. For example, people have shared a myriad of agent interaction types like the designing of agent co-ordination, designing an agent request message, designing agent learning behaviour, and so on.

On the other hand, we have discovered that only a few works have proposed patterns for describing the analysis, design, and detailed design of the information aspect. Since the information aspect focuses on ontology development as has been indicated in the MaSE (Dileo, 2002) and MESSAGE methodologies, we believe that borrowing ontology patterns can fill this gap because ontology patterns can provide solutions for recurring modelling or conceptualization problems. In addition, patterns at the computation independent model-behaviour aspect are also not much described in the existing works.

In the collection of agent patterns explored, there exist several agent patterns with similar names. For example, there are four negotiating patterns, two learning patterns, four market patterns, three matchmaker patterns, and four meeting patterns. Although similar names have been used, the analysis results explicitly show the differences between those patterns. For example, Table 10 shows the result of the classification of market patterns that were introduced by the four groups of pattern researchers identified by P7, P14, P59 and P67, respectively. As shown in Table 10, researchers have recorded experience in solving the agent co-ordination

problem by showing how to model the structure of a market organization, how to design market-based agent interactions, how to conceptualize market coordination mechanisms and how to design an Aglet based market organizer.

This finding has driven our interest towards discussing the need of having a standard for agent patterns. We propose that the agent pattern community should start thinking about standardizing agent patterns according to the categorization dimensions of the Two-ways classification scheme.

Table 10. Classification for market pattern

Software process dimension	*Aspect dimension*		
	Information	*Interaction*	*Behaviour*
CIM		P7.	
PIM	P59.	P14.	
PSM			P67.

CONCLUSION

A pattern classification allows the user to find patterns by organizing the patterns within a particular catalog. From the classification, the user will identify what might be shareable from the patterns that are located within a particular catalog. In this paper, we review three different styles of agent pattern classification. We conduct qualitative study and identify some inadequacy to date.

Hence, an improved agent pattern classification, Two-ways classification scheme, is introduced. We argue that any pattern classification must advocate various abstraction levels of an agent system to allow the categorization of patterns in a unified and comprehensive manner. The Two-ways classification scheme is able to support the accessibility of various patterns by improving the comprehension of the patterns. Having a comprehensive view of agent patterns shows the differences, similarity, and variability among the agent patterns which we believe is difficult to achieve within the current pattern classifications. Our classification allows software practitioners not belonging to the agent community to recognize what stage of agent-oriented software engineering the patterns relate to and relate the patterns from one dimension to the patterns in the other dimensions. The results of the analysis enabled us to produce a qualitative report of the 204 agent patterns.

To sum, much works are needed to ensure the quality of the agent patterns. Reusability in software development has brought so much benefit in term of time to develop a system, cost and effort. Furthermore, it able to help novice designers handles unfamiliar domains by grounding the designer with early knowledge when designing a system. Continuously from this research, we are currently working on evaluating the design quality of agent patterns. This involves the study of agent patterns in various case studies as well as testing the adoption of agent patterns among wider audience through workshop.

REFERENCES

Aridor, Y. and D. B. Lange (1998). Agent design patterns: Elements of agent application design. Proceedings of the second international conference on Autonomous agents International Conference on Autonomous Agents archive, ACM New York, NY, USA: 108-115.

Bellifemine, F., G. Caire, et al. (2008). Developing multi-agent systems with JADE, Springer.

Bresciani, P., A. Perini, et al. (2004). "Tropos: An agent-oriented software development methodology." Autonomous Agents and Multi-Agent Systems 8(3): 203-236.

Blomqvist, E. and K. Sandkuhl (2005). Patterns in ontology engineering: Classification of ontology patterns. International Conference on Enterprise Information Systems**:** 413–416.

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.

DiLeo, J., T. Jacobs, et al. (2002). Integrating ontologies into multiagent systems engineering. 4th International Bi-Conference Workshop on Agent Oriented Information Systems: 15-16.

Gonzalez-Palacios, J. and M. Luck (2004). A Framework for Patterns in Gaia: A case-study with Organisations. 5th International Workshop of Agent oriented Software Engineering NewYork, NY, USA: 174-188.

González-Vélez, H., M. Mier, et al. (2009). "HealthAgents: distributed multi-agent brain tumor diagnosis and prognosis." Applied Intelligence 30(3): 191-202.

Hafiz, M. and R. E. Johnson (2006). Security patterns and their classification schemes, Technical report, 2006.

Hayden, R. C., C. Carrick, et al. (1999). Architectural design patterns for multiagent coordination. Proceeding of the 3rd International Conference on Autonomous Agents. Seattle, WA: 230–237.

Henderson-Sellers, B. and P. Giorgini (2005). Agent-oriented methodologies, Idea Group Pub.

Husemann, B. and G. Vossen (2005). Ontology engineering from a database perspective. Proceedings 10th Asian Computing Science Conference Kunming, China. 3818: 49-63.

Juan, T. (2008). ROADMAP : agent oriented software engineering of intelligent systems, The Melbourne University. PhD. Juan, T., A. Pearce, et al. (2002). ROADMAP: extending the Gaia methodology for complex open systems. 1st ACM Joint Conference on Autonomous Agents and Multi-Agent Systems, ACM New York, NY, USA.

Kingston, J., N. Shadbolt, et al. (1996). CommonKADS models for knowledge based planning. Technical report University of Edinburgh Artificial Intelligence Applications Institute.

Kolp M., P. Giorgini, and J. Mylopoulos, 2003. "Organizational patterns for early requirements analysis," *Lecture Notes in Computer Science*, pp. 617-632, 2003.

Kotis, K. and G. A. Vouros (2006). "Human-centered ontology engineering: The HCOME methodology." Knowledge and Information Systems 10(1): 109-131.

Koutsabasis, P. and J. Darzentas (2009). "Methodologies for agent systems development: underlying assumptions and implications for design." AI & Society 23(3): 379-407.

Lind, J. (2003). Patterns in agent-oriented software engineering. Third International Workshop of Agent-Oriented Software Engineering. Bologna, Italy. 2585: 47-58.

Marik, V. and D. McFarlane (2005). "Industrial adoption of agent-based technologies." IEEE Intelligent Systems: 27-35.

Modak, V. D. (2004). A model to simplify the deployment of Mobile Agent, University of South Alabama. Master of Science.

Munroe, S., T. Miller, et al. (2006). "Crossing the agent technology chasm: Lessons, experiences and challenges in commercial applications of agents." The Knowledge Engineering Review 21(04): 345-392.

Oluyomi, A. (2006). Patterns and Protocols for Agent-Oriented Software Development. Department of Computer Science and Software Engineering, The Melbourne University. PhD.

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

Pavlıcek, D., M. Pechoucek, et al. (2007). Multi-agent-Based Diagnostics of Automotive Electronic Systems. Proceedings of the 3rd international conference on Industrial Applications of Holonic and Multi-Agent Systems: Holonic and Multi-Agent Systems for Manufacturing, Regensburg, Germany Springer-Verlag Berlin, Heidelberg.

Ren, Z. and C. J. Anumba (2004). "Multi-agent systems in construction–state of the art and prospects." Automation in Construction 13(3): 421-434.

Sauvage, S. (2004). Design patterns for multiagent systems design. Third Mexican International Conference on Artificial Intelligence. Mexico City, Mexico. 2972**:**352-