The Agent Pattern Driven Business Engineering (APBDE) Approach Enabled Business-Based Systems

Faiz M. Al-Shrouf[1], Aman B.Jantan[1], Walter.L.James[2]

[1]Software Engineering and Artificial Intelligence Research Group, School Of Computer Sciences, University Science of Malaysia, USM, 11800, Penang, Malaysia

[2] Computing and Mathematics Department, Nizwa College of Technology, NTIC, 00968-431020 Sultanate of Oman

E-mail: faiz_alshrouf@hotmail.com

ABSTRACT

Agent design patterns form a new methodology used to improve the development of software agents. Agent design patterns can help by capturing solutions to common problems in agent design [Lange and Oshima, 1998]. Agent design patterns are applied in different systems such as knowledge management systems, real-time systems, and network management systems. Agent design patterns for business-based systems, aim to support different e-commerce paradigms business-to-business (B2B) and business-to-consumer (B2C). In this paper, we developed an approach for extracting agent-based design patterns for B2C e-commerce to improve business-based processes. This approach is called an Agent Pattern Driven Business Engineering (APDBE). Based on this approach, we derived two agent-based commerce design patterns namely, the De-coupler Design Pattern (DecDP), and the Dynamic Design Pattern (DynDP). These design patterns are used to support selling/buying-based processes in e-commerce domain.

1.0 INTRODUCTION

One of the most pervasive technologies used in e-commerce is software agents. Software agents, also called "Agent-Mediated e-commerce" [Guttman et al., 1998] [He et al., 2003], have started a key role in the e-commerce domain. Software agents are now used to support virtual business processes and facilitate them to enhance e-marketplaces.

Software agents have received their importance over the last decade. Increasing work has been done in which intelligent agents support e-commerce and other Internet-based transactions. Agents are developed and deployed to perform tasks such as matchmaking, monitoring, negotiation, bidding, auctioning, transfer of goods, and follow-up support. The role of agent-based commerce is to aid the comparison shopping process. These agents collect information from multiple commercial sites, filter it and provide appropriate responses for both sellers and buyers.

The agent metaphor, due to its suitability for open environments, has recently become popular with distributed, large-scale, and dynamic applications such as e-commerce and virtual enterprises. Agent design methodologies are therefore evolving to re-engineer business processes and several designs have been introduced for agent designs. The Agent-Based Role Modeling (ABRM) approach [Kendall, 1999] [Wooldridge, 2000] is started to achieve better designs in agent technology, but it has some drawbacks including limited possibilities for reuse.

Another approach has been developed by [Farhoodi and Fingar, 1997] namely, the Agent-Oriented Business Engineering (AOBE). The AOBE is an approach aimed at combining the maturing designs of agent technology and business engineering to create a new level in the research for the prize of enterprise-level computing. This approach produces the Business Object Model (BOM) based on Agent-Based Domain model to achieve user tasks analysis, and business-based requirements.

In this paper, we have developed an approach for deriving agent commerce design patterns for business based systems namely, the Agent Pattern Driven Business Engineering (APDBE) based on agent and object patterns to extract application business-based patterns. These patterns are then used to re-engineer business processes and to achieve better designs in the field of software business re-engineering.

2.0 THE AGENT UNIFIED MODELING LANGUAGE (AUML)

In order to represent agents, the Agent-Unified Modeling Language (AUML) [Odell et al., 2000] is suggested for further extensions to UML and is used to design aspects of agents. However, according to this suggestion, UML has to include role specification in sequence diagrams and agent classes. In this paper, we recommend the use of UML standards for representing agent classes' structures. This methodology gives a powerful mechanism for producing agent design patterns and it supports the transformation of agent-oriented modeling problems into object-oriented modeling problems [Yim et al., 2000]. In the transformation process, relations between agents are transformed to design pattern [Bergenti and Poggi, 2000], and these patterns are then used as relations between object classes. The result of this method is that agent designers and developers are able to use existing UML-based tools in addition to knowledge and experience from developing object-oriented systems.

3.0 ROLE OF AGENT DESIGN PATTERNS

Design patterns are widely used in modeling object abstractions [Gramma et al., 1995]. These provide descriptions of recurring themes with certain

contexts. Generally, design patterns allow software reuse, automatic code generation, and have identified schemes for refining the behavior of components within systems. Deriving design patterns is a methodology used to assist both the designers and the developers to plan good scenarios.

Agent Design Patterns (ADP) are emerging to be one of the most fruitful topics in Agent-Oriented Software Engineering (AOSE). Agent Design Patterns (ADP) can help by capturing solutions to common problems in agent design [Lange and Oshima, 1998]. Agent Design Patterns (ADP) have been extended to fill the gap between high-level agent specific languages and system-level programming languages such as Java. Agent developers can select and combine multiple patterns in a graphical environment based on a standard for these patterns.

In e-commerce environments, Agent Design Patterns (ADP) are useful to integrate business processes and business components with agent technology. This integration supports front-end and back-end business components with business actors. It is our belief that ADP for business-based systems aims to satisfy the following roles:

- Document an easy approach to assist business actors by integrating business scenarios with agent software components.
- Improve the system business designs for different potential members to support the development of system business interoperability.
- Help agent developers to establish new paradigms for agents that are similar to those in the e-business environment.
- Distinguish new relationships among agents, with common interests.
- Reduce the redundancy in system designs.
- Provide a reference template schema for agents. The reference can be documented in catalog pattern schemes.
- Assist agent developers to achieve pervasive solutions to e-business disciplines including elements of trust, security, and credentials.
- Support interoperability, flexibility, and performance for an agent's system design and its commerce-based applications. Agent developers can integrate several code designs to cope with these patterns. Errors and bugs may be more easily detected and corrected.
- Improve e-business transactions in which agents play a key role. This involves agent negotiation mechanisms, and agent-based brokering.
- Allow for the reuse of designs and automatic code generation for agents in the development of other business scenarios.

4.0 AGENT DESIGN PATTERNS (ADP)

Patterns are classified by the software community as follows: [Gramma et al., 1995][Flower, 1997] [Lange and Oshima, 1998] [Griver et al., 2000] [Schelfthout et al., 2002] [Moose, 2002], 1) design patterns. These patterns provide a scheme for refining the subsystems or components of a software system and describe a commonly recurring structure of communicating components that solves a general design problem with a particular context, 2) programming patterns. These patterns describe how to implement particular aspects of components or the relationships between them using the features of a given language, 3) analysis patterns. These patterns focus on organizational, social, and economical aspects of the system, and 4) implementation patterns. These deal with problems and generic solutions for implementing agents and multi-agent systems.

To explore agent-based commerce design patterns for business-based systems, [Papazoglou, 2001] classifies agents in multi-agent e-business environments as follows: 1) application agents. These agents collaborate as part of a distributed workflow application that implements a typical range of cross-organizational activities, 2) personal agents. These agents work directly with users to help support the management of user profiles, requests, and information collection, 3) general business activity agents. These agents address the needs of a business partner. Such are search agents, negotiation agents, billing agents, marketing agents, and business transaction agents (seller agents, buyer agents), and 4) system level support agents. These agents provide the security services required for the conduct of e-business, such as authentication agents, security agents, and authorization agents.

In our literature survey, we found a set of agent design patterns for mobile agent systems and static agents. The ongoing success of agent systems depends on the development for them of appropriate software engineering principles. Design patterns are a recognized means that should be promoted. Agent Design Patterns (ADP) are classified as given below:

- **Architecture patterns:** These patterns deal with the architecture of agents and agent-based applications [Kendall et al., 1998] [Aridor and Lange, 1998] [Lind, 2002]. Examples of agent architecture patterns are: the Layered Agent Pattern, and the InteRap Pattern.

- **Communication patterns:** These patterns deal with the way agents communicate with one another [Deugo and Wiess, 1999] [Deugo et al., 1999] [Meira et al., 2000]. Examples of communication patterns are: the Receptionist Pattern, the Secretary Pattern, the Session Pattern, the Antenna Pattern, the Direct-Coupling Pattern, the Modified Proxy Pattern, the Communication Sessions Pattern, Badges Pattern, and the Event Dispatcher Pattern.

- **Traveling Patterns:** These patterns deal with the management and movement of mobile agents, and the

quality of their service and routing [Lange and Oshima, 1998]. Examples of traveling patterns are: the Itinerary pattern, the Forwarding Pattern, and the Ticket Pattern.

- **Task Patterns:** These patterns deal with the breakdown of tasks and how these tasks are delegated to one or more agents [Lange and Oshima, 1998]. Examples of task patterns are: the Master-Slave Pattern, and the Plan Pattern.

- **Interaction Patterns:** These patterns deal with the way agents locate one another and facilitate their interactions [Lange and Oshima, 1998]. Examples of interaction patterns are: the Meeting Pattern, the Locker Pattern, the Messenger Pattern, the Finder Pattern, and the Organized-Group Pattern.

- **Coordination Patterns:** These patterns deal with managing dependencies between agent activities [Tolksdorf, 1998]. Examples of coordination patterns are: the Pull & Push Pattern, the Index Pattern, and the Traveler Pattern.

- **Commerce Patterns:** These deal with agent-based commerce for e-business processes [Silva and Delgado, 1998] [Kostiadis et al., 1999] [Wiess, 2001, 2002][Faiz and Aman, 2003] [Faiz and James, 2004]. Examples of commerce patterns are: the Agent Society Pattern, the Agent as Delegate Pattern, the Common Vocabulary Pattern, the Mediator Pattern, the De-coupler Pattern, and the Dynamic Pattern.

5.0 AGENT-BASED BUSINESS DESIGN APPROACHES

In this section, we review the current agent design methodologies used in Agent-Oriented Software Engineering (AOSE). Some drawbacks found in the Agent-Based Role Modeling approach (ABRM) in agent design have led to the development of our approach namely, the Agent Pattern Driven Business Engineering (APDBE) approach.

5.1 THE AGENT-BASED ROLE MODELING APROACH

In Multi-Agent Systems (MAS), design, the Agent-Based Role Modeling (ABRM) approach is used as a design methodology for agent development [Kendall, 1999] [Wooldridge, 2000] [Cabri, 2001] [Cabri et al., 2002]. This methodology follows the traditional topology, top-down approach which, starts by identifying the system requirements and decomposing these requirements at the analysis process into certain activities.

At the design stage, activities are tasks that assigned to a role which, performs without interacting with other roles. A role is a set of capabilities, expected behavior and knowledge that an agent exploits according to its needs. Roles mapped to individual agents. The role embeds all the information needed by the agent to interact with the system. For example, in e-business systems, customer information is kept in databases. Suppose an agent in charge of querying databases is sited on different hosts. Instead of embedding the needed knowledge in this agent such as the driver name and the query statements, a solution is to use a role. This role is embedded in a host and the agent assumes the use of it. Role embedding allows simplified development as roles allow logical separation of the different concerns of the agent. In the Agent-Based Role Modeling (ABRM) approach, the role system allows some models to make the role utilization by agents easier such models are [Wooldridge, 2000]: a service model, an acquaintance model, and a security model.

ABRM approach has been proven as a good approach for developing closed agent-systems and Internet applications [Tveit, 2001]. On the other hand, such an approach has some restrictions and possibly less value in open e-commerce environments. Another problematic issue is when the number of roles grows. This leads to an increase in the number of agents handled by the services of these roles [Cabri, et al., 2002], therefore, this results in poor reusability in code generation, inconsistency in design, and unreliable systems.

5.2 THE AGENT-ORIENTED BUSINESS ENGINEERING (AOBE) APROACH

The Agent-Oriented Business Engineering (AOBE) approach [Farhoodi and Fingar, 1997]is a an approach that aims to combine the maturing design of agent technology with business engineering to build a new area of research for the prize of enterprise-level computing. The AOBE approach has two models:

The agent-oriented lifecycle model.
This model addresses domain modeling for agent-orientation by providing an active modeling metaphor and better analysis models that enable re-use. Figure 1 shows the agent-oriented lifecycle model in AOBE.

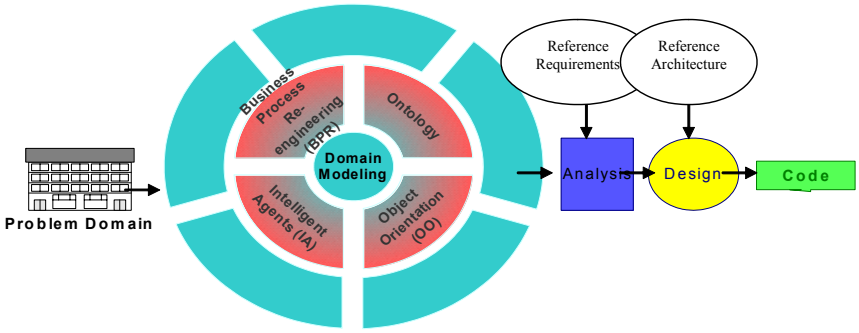

Figure 1: The Agent-Oriented Lifecycle

The Ontology-based domain models.
Ontology defines the basic concepts and entities that are assumed to exist in some area of interest and the relationships that hold among them. This is a critical initial step in producing business-based systems. There are three distinct approaches to business-domain modeling: Business Process Re-engineering (BPR), Object Oriented Technology (OO), and Intelligent Agents (IA). All these approaches are model-based and offer different techniques for describing problem domains.

The BPR methods involve process, organization, events, business rules, entities, and relationships.

The Object-Oriented Technology (OO) methods involve classes, objects, attributes, associations, operations, events, inheritance, polymorphism, and categories. These are well suited for software engineering modeling and have potential for reuse. However, they are not inherently business oriented, and provide premature commitments to design and implementation strategies.

Intelligent Agent Technology (IA) can be leveraged to enhance enterprise modeling as well as offering new techniques for developing intelligent applications and smart technical infrastructure services. An agent oriented perspective allows us to develop rich and expressive models of the enterprise and provide foundation for adaptive and reusable business software.

The convergence of OO, IA, and BPR results in a significant breakthrough in building models of the enterprise that is capable of end-to-end integration of business analysis and software systems. Therefore, the AOBE is a full-lifecycle approach. Figure 2 shows that agent-based domain modeling produces the Business Object Model (BOM) in terms of ontologies, and uses ideas and techniques from traditional business analysis (Process Modeling), OO, BPR, and IA. The Business Object Model (BOM) is agent-based and is used to support user task analysis, requirements modeling, and the specification and design of the Software Object Model (SOM).

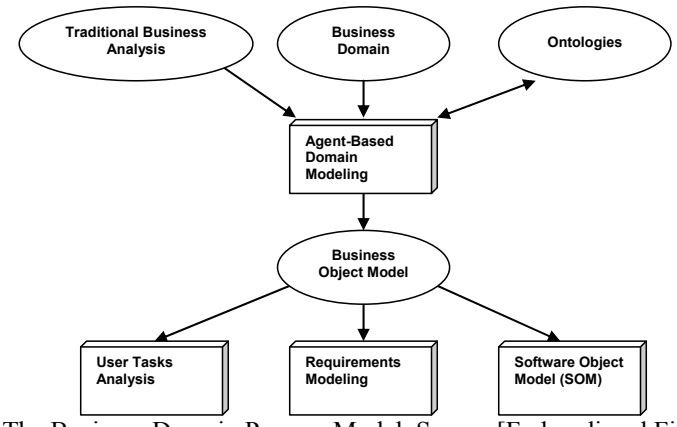

Figure 2: The Business Domain Process Model. Source [Farhoodi and Fingar, 1997]

6.0 THE AGENT PATTERN DRIVEN BUSINESS ENGINEERING (APDBE) APROACH

A more efficient approach is developed in this paper namely, the Agent Pattern Driven Business Engineering (APDBE) approach. It has its advantages over the Agent-Based Role Modeling (ABRM) approach and Agent-Oriented Business Engineering (AOBE) approaches. The APDBE approach focuses mainly on building application-based business patterns by combining agent patterns, object patterns and business patterns to re-engineer business processes.

The main objective of agent-based commerce design patterns for enterprises is to enhance communications and facilitating organizational needs along with partners. This can be achieved by establishing new designs so that integration can be used to re-engineer business processes and business components to work with agent technology and software components within the existing infrastructure.

Agent-based commerce design patterns for business-based systems are being used to help developers and designers to automate business transactions with agent designs and facilitating processes related to front-end-components. They involve interaction with the customer and delegation of tasks, plus integration of the back-end-components, that don't involve direct interaction with the customer. They instead provide mechanisms for mediating between the agents that represent customers.

Despite using software agents as supporters of in the e-commerce domain, there are still drawbacks in the developing agents. Currently, agent software developers construct their agents on one-to-one basis, and according to business transactional requirements.

The APDBE architecture as shown in Figure 3 uses new designs could be composed of several well-documented design patterns. The APDBE approach is relatively straightforward and easily develops new design patterns for agent-based commerce in execution environment such as e-business. The first step of APDBE is the analysis stage which involves identifying a business process in the business domain that represents a business scenario plus the specifying business components. Business actors then engage in the business process. At the end of analysis stage, a business pattern is established with user, services and information domains. [Dodani, 2003] identified a business pattern that involves four business dimensions. These dimensions are (1) self service: which facilitates users to access business process at any time, (2) business actor: which uses the self service and shares information with other business partners, (3) information aggregation: which facilitates data from multiple sources to be aggregated and presented across multiple channels, and (4) business components: which facilitate the integration of data and processes across enterprise and their partners.

At the design stage, agent and object design patterns and business patterns are integrated to form new design pattern along with a set of agent activities. The agent developer should address the corresponding business pattern along with the agent design pattern to form application-based patterns. These application patterns include a set of tasks that transform to achieve the user requirements in a business-based system.

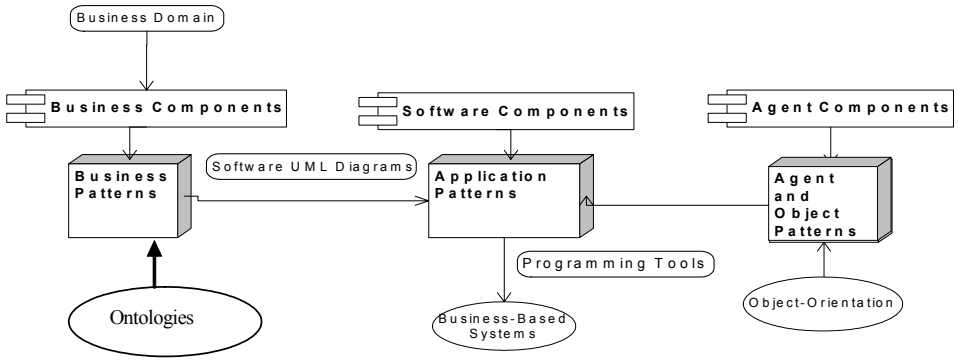

Figure 3: The Agent Pattern Driven Business Engineering (APDBE) Development Environment

7.0 EXAMPLES OF BUSINESS-BASED APPLICATION PATTERNS

In the following section, we present business-based application patterns that we have developed based on the Agent Pattern Driven Business Engineering (APDBE) approach. These application patterns are used to re-engineer business processes in B2C e-Commerce and C2B e-commerce. We believe that processes in these paradigms provide new steps towards the automation of other business processes. However, because agent commerce design patterns for business-based systems are mature, further improvements may be required to achieve scalable and reliable applications.

7.1 THE DE-COUPLER DESIGN PATTERN (DecDP)

The De-coupler design pattern was mentioned in the work of [Faiz and Aman, 2003]. The main objective of the DecDP is to decouple views of business-based selling / buying process in an e-marketplace. The DecDP applied for B2C e-commerce and uses the Model-View-Controller (MVC) design pattern to

construct a new design when an agent acts as a controller to separate views for sellers and buyers. The structure of the DecDP design pattern is given in Figure 4 using AUML notations.

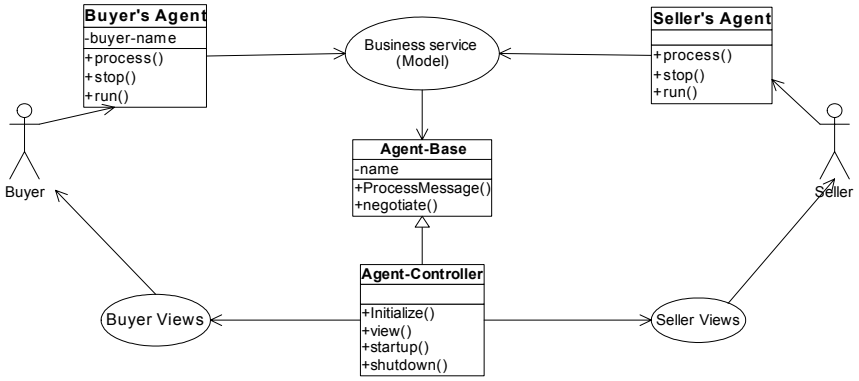

Figure 4: Structure of the De-coupler Design Pattern

The De-coupler design pattern (DecDP) is to facilitate the sellers and buyers processes, which are represented by the buyer and the seller agent. The design increases the cohesion and reduces the coupling between business components.

7.2 THE DYNAMIC DESIGN PATTERN (DynDP)

The Dynamic design pattern was mentioned in the work of [Faiz and James, 2004]. The DynDP uses the Master-Slave design pattern [Lange and Oshima, 1998]. The master-slave pattern defines a scheme where a master agent can delegate a task to a slave agent. The DynDP also developed a Facilitator agent to manage the negotiation process between the buyer agent and the sellers agents through the facilitator. Figure 5 shows the structure of the DynDP.

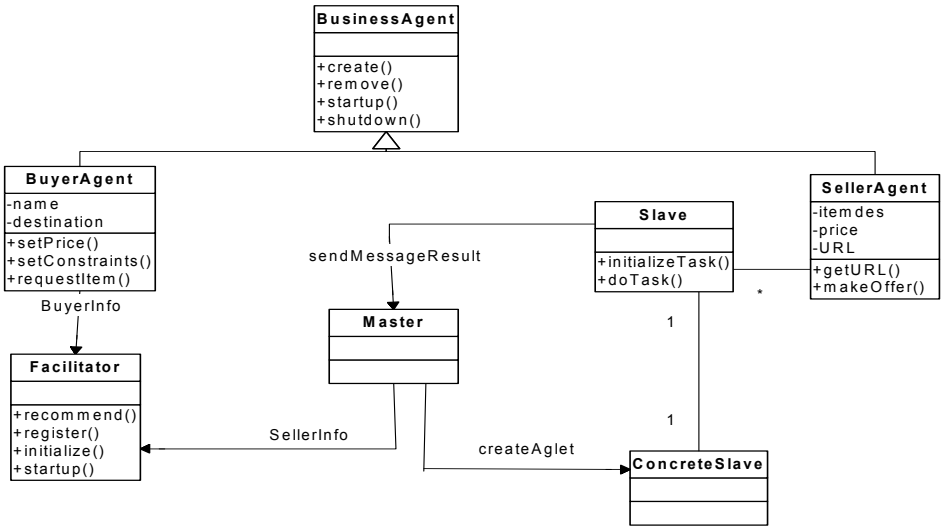

Figure 5: Structure Diagram of the Dynamic Design Pattern (DynDP)

The main objective of the DynDP is to find a mechanism for coordinating agents in the selling/buying-based business process. In a virtual marketplace, both sellers and buyers need to interact permanently to achieve the best deal. This process is a repetitive process. Several sellers need multiple interactions to achieve the final process to accept or reject the offer of a particular buyer. Direct communication with the seller would result in big network traffic. The use of the DynDP allows slave agents to carry out this communication locally. This will reduce the network traffic. Thus, the interaction takes place locally at the seller's location and the master agent directs those perspective sellers engaged in the negotiation process.

8.0 CONCLUSIONS AND FUTURE DIRECTIONS

In this paper, we have presented a methodology for deriving agent-based business engineering based on a pattern scheme. The agent Pattern Driven Business Engineering (APDBE) is an approach for identifying, specifying, and designing agent patterns for business-based systems. We believe that the ongoing success of pattern-oriented engineering methodology is

leading to the development of better approaches to agent system engineering and better designs in large environments like e-business. The APDBE allows the re-engineering of business-based processes and greater benefits to come from existing agent and object design patterns. The approach also provides a methodology for the re-use of these designs and integrates them with business and software components. We have used this approach to develop two agent-based commerce design patterns for business-based systems namely, the De-coupler design pattern (DecDP) and the Dynamic design pattern (DynDP). These design patterns support selling/buying-based business processes.

The development of new agent-based design patterns is still in the infancy stage. Future directions may require the development of new designs in B2C, and C2B e-commerce. Processes among these paradigms will need to be classified and re-engineered based on the agent-pattern scheme. Furthermore, an implementation platform will have to be supported to capture agent-based design pattern solutions in accordance with FIPA standards.

REFERENCES

Aridor Y., Lange D. Agent Design Patterns. IBM Tokyo Research Laboratory. Proc of the 2nd International Conference on Autonomous Agents. ACM Press. (1998).

Bergenti F., Poggi A. Exploiting UML in the Design of Multi-Agent Systems. Proc. of the ECOOP-Workshop on Engineering Societies in the Agents' world 2000. (ESAW'00). Pp 96-103(2000).

Cabri G. Role-based infrastructure for Agents. 8th IEEE Workshop on Future Trends of Distributed Computing Systems, Bologna (I), (2001).

Cabri G., Leonardi L., Zambonelli F. Modeling Role-Based Interactions for Agents. Workshop on Agent-Oriented Methodologies at OOPSLA, USA. (2002).

Deugo D., Oppacher F., Kuester J., Vontte I. Patterns as Means for Intelligent Software Engineering. Carleton University, KIS5B6, Ottaw, Canda. (1999).

Deugo D., Weiss M. A case for Mobile Agent Patterns. Technical Report. Business Communication Systems. Canada.(1999).

Dodani M. Pattern Driven Solution Engineering. Journal Of Object Technology JOT. Vol. 2, no.2, pp.27-33, ETH Zurich. (2003).

Faiz A., Aman. J. An Agent-Based Design Pattern For Decoupling Views Of e-Business Systems. 3rd International Conference on Information Technology in Asia. CITA'03. pp. 93-100. Sarawak, Malaysia. (2003).

Faiz A., James W. Agent-Based Dynamic Design Pattern For E-Business Systems. Proceedings of the International Conference on Applied Computing. IADIS'04. Lisbon, Portugal. (2004).

Farhoodi F., Fingar P. Developing Enterprise Systems with Intelligent Agent Technology. [online][Accessed 22,11,2003]. Available from World Wide Web: http://home1.gte.net/pfingar/docmag_part2.htm. 1997.

Flower M. Analysis Patterns: Reusable Object Models. Addison Wesley. USA. (1997).

Gramma E., Helm R., Johnson R., Vlissides J. Design Patterns Elements Of Reusable Object-Oriented Software. Person Education Singapore, 10th edition. (1995).

Griver Y., Arnheiter M., Gellis M. Visual Basic Developer's Guide To UML and Design Patterns. SYBEX Publisher, USA. (2000).

Guttman R., Moukas A., Maes P. (1998). Agent-Mediated Electronic Commerce: A survey. Knowledge Engineering Review Journal, 42(3). Vol. 13, no.2. (1998).

He M., Jennings N., Leung H. (2003). On Agent-Mediated Electronic Commerce. IEEE Transactions on knowledge and data engineering, vol. 15, no. 4. pp. 985-1003. (2003).

Kendall E. Role Modeling for Agent System Analysis, Design, and Implementation. (ASA/MA-99), ACM. (1999).

Kostiadis K., Hunter M., Hu H. The Use Of Design Patterns For The Development Of Multi-Agent Systems. Department Of Computer Science, University Of ESSEX, UK. (1999).

Lange D., Oshima M. Programming and Deploying Java Mobile Agents With Aglets. Addison Wesley, USA. (1998).

Lind J. Patterns in Agent-Oriented Software Engineering. GmbH. D-82008, agentLab, Germany. (2002).

Miera N., Silva I., Silva . A set Of A gent Patterns For More Expressive Approach. INESC-ID&IST. RUA Alves, vol., 12, no., 9, pp. 44-50. (2000).

Mosse F. Modeling Roles. A Practical Series Of Analysis Patterns. Journal Of Object Technology JOT, vol., 1, no., 4, pp/. 27-37. (2002).

Odell J., Van Dyke Paruna H., Fleisher M., Brueckner S. Modeling Agents and their Environment: The Physical Environment. Journal Of Object Technology JOT, vol., 2, no., 2, pp. 43-51. (2003).

Papazoglou M. Agent-Oriented Technology in Support Of E-Business, Communications of the ACM, vol., 44,no., 44, pp.71-77. (2001).

Schelfthout K., Coninx T., Hellboogh A. Agent Implementation Patterns. Kuleuven, 200A, Belgium. (2002).

Silva A., Delgado J. The Agent Patterns. A perspective from the Mobile Agent System Point Of View. EuroPLoP'98. (1998).

Tolksdorf R. Coordination Patterns of Mobile Information Agents. Proceedings of Cooperative Information Agents II, second International Workshop, CIA'98, Springer, pp.246-261. (1998).

Tveit A. A Survey of Agent-Oriented Software Engineering. NTNU, Norway, (2001).

Weiss M. Pattern-Driven Design of Agent Systems: Approach and Case Study. Proceedings of the Conference Pattern Languages of Programming (PloP'02). (2002).

Weiss M. Patterns for e-commerce Agent Architectures: Using Agents as Delegates. Proc. Of the PLoP'01 Conference. (2001).

Wooldridge M., Jennings N., Kinny D. The Gaia Methodology for Agent-Oriented Analysis and design. Autonomous Agents and Multi-Agent Systems, 3, 285-312. Kluwer Academic Publishers. Netherlands. (2000).

Yim H., Cho K., Jongwoo K., Park S. Architecture-Centric Object-Oriented Design Method for Multi-Agent Systems. Proc. Of the fourth International Conference on MultiAgent Systems (ICMAS-2000). (2000)