COMPONENT MODEL FOR AGENT PATTERN-ORIENTED DESIGN (AGENTPOD) FRAMEWORK

MAIZATUL AKMAM BINTI ISMAIL

A project report submitted in partial fulfillment of the
requirements for the award of the degree of
Master of Science (Computer Science)

Faculty of Computer Science and Information Systems
Universiti Teknologi Malaysia

APRIL 2010

ABSTRACT

The explosive growth of agent system areas such as JACK (Rao and Georgeff, 1995), JADE (Bellifemine et al., 1999), Jadex (Braubach et al., 2006), and multi-agent system demand suitable agent component model that is can operate within a wide range of environments, and can evolve over time to cope with changing requirements and design. AgentPOD approach is involved in analysis and designs phase but not in implementation phase. Instead, there is a gap between design and implementation phase. An approach to easily deploy the AgentPOD into agent system is unavailable. In this project, the main focus is to identify and study an existing agent component models. The purpose of this project is to solve the limitation of AgentPOD framework with reduce the gap between design and implementation phase by using existing agent component model. An overview and meta-model of the existing agent component model such as Jadex, Cougaar, and JMX are identified. Meta-model provides the core language that is used in determine the suitable agent component model and in mapping process. The mapping process aims to motivate the use of patterns and component model at software development process that subject to deploy pattern-oriented agent system. Jadex is a FIPA specifications compliant agent development environment that gives several facilities for an easy and fast implementation. The case study “Automated Negotiation System for On-line Book Shops” is used to prove the suitability of Jadex and AgentPOD.

ABSTRAK

Perkembangan agen sistem yang pesat seperti JACK (Rao and Georgeff, 1995), JADE (Bellifemine et al., 1999), Jadex (Braubach et al., 2006), dan agen multi sistem memperuntukkan komponen model yang sesuai yang boleh beroperasi dalam persekitaran yang meluas, dan boleh berkembang mengikut masa bagi mengatasi perubahan keperluan dan rekabentuk. Pendekatan AgentPOD hanya melibatkan fasa analisis dan fasa rekabentuk tetapi tidak melibatkan fasa pelaksanaan. Malahan, terdapat ruang pemisah antara fasa rekabentuk dan fasa pelaksanaan. Tiada lagi pendekatan yang digunakan untuk membangunkan AgentPOD kepada agen sistem. Dalam projek ini, fokus utama adalah untuk mengenalpasti agen komponen model yang telah sedia ada. Tujuan projek ini ialah untuk menyelesaikan masalah AgentPOD dengan mengurangkan ruang pemisah antara fasa rekabentuk dengan fasa pelaksanaan dengan menggunakan pendekatan agen komponen model. Pengenalan dan meta-model kepada agent komponen model yang telah wujud seperti Jadex, Cougaar, dan JMX telah dikenalpasti. Meta-model ini menyediakan teras bahasa yang akan digunakan untuk menentukan agen komponen yang sesuai dan proses pemetaan. Tujuan pemetaan ini adalah untuk motivasi penggunaan pola dan agen komponen model pada proses pembangunan perisian bagi membangunkan agen sistem berorientasikan pola. Jadex merupakan agen pembangunan persekitaran FIPA compliant spesifikasi yang memberi pelbagai kemudahan untuk menyenang dan mempercepatkan pelaksanaan. Kajian kes "*Automated Negotiation System for On-line Book Shops*" digunakan untuk membuktikan kesesuaian penggunaan Jadex dengan AgentPOD.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ACL	Agent Communication Language
ADF	Agent Definition File
ADL	Advanced Distributed Learning
AgentPOD	Agent Pattern-Oriented Design
API	Application Program Interface
ASP	Active Server Page
BDI	Beliefs Desire Intention
BSD	Berkeley Software Distribution
CBSE	Component Based Software Engineering
CCM	CORBA Component Model
COM	Component Object Model
CORBA	Common Object Request Broker Architecture
Cougaar	Cognitive Agent Architecture
DARPA	Defense Advanced Research Projects Agency
DMK	Dynamic Management Kit
EJB	Enterprise Java Beans
FCS	Federation of Communication Services
FIPA	Foundations for Intelligent/Smart Physical Agents
HTML	HyperText Modeling Language
IDEs	Integrated Development Environments
IEEE	Institute of Electrical and Electronics Engineers
IIOP	Internet Inter-Orb (object request broker) Protocol
I/O	Input Output
ISECOM	Institute for Security and Open Methodologies

JADE	Java Agent Development
Jadex	Java Agent Development Extension Framework
JCC	Jadex Control Center
JCP	Java Community Process
JDBC	Java Database Connectivity
JDK	Java Development Kit
JMS	Java Message Service
JMX	Java Management Extension
JRE	Java Runtime Environment
JSR	Java Specification Request
JVM	Java Virtual Machine
JXTA	Juxtapose
J2EE	Java 2 Platform, Enterprise Edition
J2SE	Java 2 Platform, Standard Edition
LAN	Local Area Network
MAS	Multi-Agent System
MBeans	Managed Beans
NNTP	Network News Transfer Protocol
OKBC	Open Knowledge Base Connectivity
OMG	Object Management Group
OQL	Object Query Language
OO	Object Oriented
OSSTMM	Open Source Security Testing Methodology Manual
PECOS	Pervasive Component System
POAD	Pattern-Oriented Analysis and Design
POD	Pattern-Oriented Design
PRS	Procedural Reasoning System
P2P	Peer-to-Peer
RI	Reference Implementation
RM	Ringgit Malaysia
RMA	Remote Monitoring Agent
RMI	Remote Method Invocation
SLP	Service Location Protocol
SMTP	Simple Mail Transfer Protocol

SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
UDP	Unit Data Packet
UIs	User Interfaces
UML	Unified Modeling Language
UPnP	Universal Plug and Play
URL	Uniform Resource Locator
US	United State
XML	Extension Modeling Language

LIST OF APPENDICES

CHAPTER 1

PROJECT OVERVIEW

1.1 Introduction

The explosive growth of agent system areas such as JACK (Rao and Georgeff, 1995), PRS (Myers, 1997), JADE (Bellifemine *et al.*, 1999), AgentBuilder (Acronumics Inc., 2004), Jadex (Braubach *et al.*, 2006), and multi-agent system demand suitable component model that is can operate within a wide range of environments. In addition, it also can evolve over time to cope with changing requirements and design. Autonomous agents and agent systems are being used to cope with ever increasing complexity in such system requirements and design. Nevertheless, most of these agent systems are developed with a specific technological focus such as cognitive or infrastructural architecture.

According to Thome *et al.* (2004), component model is a framework for loading modular software. Component model are reusable approach that have proven useful in the implementation of software development. The component model illustrates the software components are used to build the system. Nowadays, many researchers study about component model for agent system in software engineering. For instance, OMG CORBA Component Model called CCM, Jadex, Cougaar, and so on.

Design pattern is a design solution to a frequently recurring design problem in a given application domain (Yacoub *et al.*, 2000). According to Schmidt (1995), definition of design pattern is to capture the static and dynamic structures of solution that occur repeatedly when producing applications in a particular context. While, patterns means reusable solutions to recur design problems and also provide a vocabulary for communicating these solutions to others. Yacoub and Ammar (2004) define a pattern as a problem that frequently occur in software development and then define the explicit solution that it can be reused. While Fernandez and Yuan (2000) define a pattern is a recurring combination of meaningful units that occur in some context.

An agent is an independent person or entity that autonomously accomplishes tasks for another person entity (Dana Moore *et al.*, 2002). In Todd Wright (2007), agents are autonomous software entities which communicate with other agents or external services to achieve domain-specific functionality. An agent component is a component as well as an agent. It synthesizes the advantages of the agent technique and overcomes the pitfalls of the method of the single agent or component development. Agent Pattern-Oriented Design (AgentPOD) main concept is to define and utilize design patterns as building components of agent-based system designs (Radziah, 2007).

Pattern-oriented agent system is an agent system that follows pattern-oriented approaches (has static and dynamic structures of AgentPOD framework) which integrates with existing component model. This agent system uses the same principles of Component Based Software Engineering (CBSE) where building the systems from the existing and selecting components within the suitable software architecture (AgentPOD).

In this project, the meta-model of existing component model for an agent such as Jadex, Cougaar, and JMX is identified and then evaluated by using the criterias of AgentPOD framework. The determination of the suitable agent component model is briefly discussed based on evaluation result. Lastly, it will be applied to the case study (Automated Negotiation System for On-line Book Shops).

1.2 Problem Background

Agent Pattern-Oriented Design (AgentPOD) which is introduced by (Radziah, 2007) is a component-based framework that utilizes the design patterns used as building blocks for the agent-based application design. Since an AgentPOD that has not involved in implementation phase, an existing agent component model is used as a solution to bridge the gap between design and implementation phase. Furthermore, component model illustrates the software components that will be used to build the system. Software component offers a predefined service, and able to communicate with other components. Whereas, components are high level aggregations of smaller software pieces and provide a “black box” building block approach to software construction. A component may be something like an ActiveX control (either a user interface control or a business rules server). A component was designed to solve particular problems and resembles like a pattern that forced the developers to use predefined process or design to be used in software development process.

In Thome *et al.* (2004), components are a mixture of custom built which will be assembled to provide the required functionality. For instance, the Cougaar component model is implemented to Ultralog Security Service as a case study. Cougaar component model is a component model that separates containment from service. This component model is a dynamic component lifecycle and description-based specification. The inter-component relationship of Cougaar was defined by position in component hierarchy and services offered and requested. The binder approach is used in this component model to active insulation surrounding each component, enforces strong encapsulation, and transforms all (component lifecycle event, service hookups, and service invocations as inter-component method calls). The Cougaar infrastructure is fully “componentized”. The UltraLog Security Services is a security services functional and CCM view. The UltraLog gives benefits of the security services componentization. The security binders usage patterns and examples. UltraLog used wiring of security components to security binders. The security binders usage patterns are divided into access control and service proxy. In this work, all components and agents run in their own security context. Security context is used to enforce per-agent and per-component security policies. This approach was implemented as binders between the Agent Manager and the Agent.

Another work (Ghitescu, 2007), a component may implement another model element or a component may be implemented by another element. In JMX agent’s integration models, daemon model, component model, and driver model were integrated together to evaluate the Java Management eXtension (JMX) Models performance. Java Management eXtension (JMX) is a standard for managing Java based applications that starting with J2SE 5.0 (a part of the Java platform). JMX framework follows the agent-manager paradigm. Daemon model was applied in daemon agent process and application process. While component model applying the application process that is integrated of daemon agent process and application process in daemon model. For driver model, it applies driver agent process that is enhanced form of application process in component model with applying Bootstrap.

The limitations of AgentPOD process are based on the deployment of the design patterns for agent world and based on the Pattern-Oriented Analysis and Design (POAD) (Yacoub and Ammar, 2004) software development approach. POAD is a methodology that constraint on the uses of components and patterns from the analysis stages until the design (Yacoub and Ammar, 2004). Thus, POAD is a structural approach of software development rather than behavioral approach (Yacoub and Ammar, 2004). AgentPOD is a general reusable solution to a commonly occurring problem in software design. AgentPOD is not completed design that can be transformed directly into code. It is a description or template for how to solve the problem that can be used in many different situations. AgentPOD also used design pattern as a first- class modeling concept to design application.

AgentPOD approach is involved in analysis and designs phase but does not in implementation phase. The implementation phase is a crucial phase of agent world software development lifecycle. This phase is very important to develop the agent system after complete analysis and design phase. In this phase, code template based on analysis and design phase can be decoded using agent implementation platforms such as Jadex (Braubach *et al.*, 2006), JADE (Bellifemine *et al.*, 1999) and AgentBuilder (Acronumics Inc., 2004). From this phase, the performance of agent system can be measured either it is better or bad.

AgentPOD is just to discovering and documenting patterns as design component. An approach to easily deploy the AgentPOD into agent system is unavailable. In order to overcome those problems, the main focus in this project is to identify and study the existing agent component models that can be mapped to AgentPOD framework. Component model can present the implementation of the model and a solid foundation for agent development. So, this project proposes an existing agent component model to bridge the gap between the design and implementation phase to deploy pattern-oriented agent system. The uses of pattern and component together could reduce the complexity of the software process. This is because of the use of pattern for the whole process glued together with the component could reduce the time and cost of software development.

1.3 Problem Statement

An approach to easily deploy the AgentPOD into agent system is unavailable. Usefulness of reusable component model approach has been proven in the implementation of software development. Hence, the primary research question in this study that remains unexplored is:
"Which component model is suitable to be applied to AgentPOD framework to produce a pattern-oriented agent system?"

The secondary research questions that need to consider fulfilling the primary research question are:

1. What are the features of agent component model that are needed by AgentPOD framework?
2. How to map an existing meta-model of agent component model with AgentPOD meta-model?
3. In what ways an agent component model can be applied to AgentPOD?

1.4 Project Aim

The project aim is to study, mapping and apply the suitable existing agent component model to AgentPOD framework. The component model is used in order to discover the limitations of the AgentPOD framework. Indeed, AgentPOD approach is involved in analysis and designs phase but not in implementation phase. Hence, the mapping between existing agent component model with AgentPOD is proposed. To show the suitability of the proposed mapping, it will be applied into the selected case study of Automated Negotiation System for On-line Book Shops.

1.5 Objectives

This project seeks to accomplish the project aims, these objectives are identified as listed below:

i. To study the existing agent component model for AgentPOD framework.

ii. To enhance meta-model of component model to work with AgentPOD meta-model.

iii. To apply suitable component model to AgentPOD framework (in the selected case study).

1.6 Project Scope

No new agent component model is developed. The study is based on an existing agent component model and AgentPOD framework. This study focuses on implementation phase only where an agent existing component model is used to bridge the gap between design and implementation phase. The implementation is applied for a small application to show the suitability of the selected agent component model.

1.7 Significance of the Project

This project significantly applies a suitable existing agent component model to AgentPOD framework to develop pattern-oriented agent system. The study concentrates on a deep understanding of component model and AgentPOD framework. AgentPOD is involved analysis and design phase but does not have implementation phase. Based on that knowledge, the mapping between the selected component model, meta-model and AgentPOD meta-model will be done. The advantages that can be derived from this study use an existing agent component model to bridge the gap between design phase and implementation phase.

1.8 Organization of the Report

This report is consisting of six chapters that organized as follows. The CHAPTER 1, project overview provides an overview of the project with more details. It presents an introduction, problem background, statement background, and aim of the project. It also gives the objectives and project scope as well as the significance of the project. CHAPTER 2, literature review that concerned with presenting a survey of background relevant to the area of investigation and leading to an evaluation of pre-existing implementations or designs and of candidate re-usable components. It will be discussed reviews on the previous work and relevant literature reviews to this project such as component model, agent, AgentPOD, and so on. CHAPTER 3, research methodology will be discussed on methodology that will be used in this project which conducted in achieving the project objectives and scopes. One case study is used. CHAPTER 4, meta-model for each existing agent component model is constructed. Then, mapping process between agent component model meta-model and AgentPOD meta-model is done. The final task is to construct component diagram for the selected agent component model. CHAPTER 5 presents the code template that transforms from component diagram. The implementation is done by applying the selected agent component to case study to prove the suitability between the selected agent component model with AgentPOD framework. CHAPTER 6, concludes the project summary, discusses on the achievement, constraints and challenges which is involved in this project, recommendations and future work are conducted.

1.9 Conclusion

To sum up, component model is an important methodology that must be study in this project. Even though there are many existing component model that have been used in the previous work, but it is so difficult to find out the suitable component model that will be match with AgentPOD framework. Hence, this approach will be applied on the selected case study. This study focuses on suitable existing component model for AgentPOD framework to deploy a pattern-oriented agent system. The suitable component model will be useful to bridge the gap between design phase and implementation phase. After the mapping process is done, it will be applied to case study to proof its suitability.

REFERENCES

Acronymics Inc. (2004). *AgentBuilder*, http://www.agentbuilder.com.

Aridor, Y. and Lange, D.B. (1998). *Agent Design Patterns: Elements of Agent Application Design*. In Sycara, P. and Wooldrigde, M. (editors), Agents ’98. ACM Press.

Braubach, L. Pokahr, A., Lamersdorf, W., (2004). *Jadex: A Short Overview*.

Braubach, L. Pokahr, A., Lamersdorf, W., (2006). *Jadex: A BDI-Agent System Combining Middleware and Reasoning*, in Unland, R., Calisti, M., Klusch, M., (Hrsg.): Software Agent-Based Applications, Platforms and Development Kits, Birkhauser-Verlag, Basel-Boston-Berlin, pp. 143-168.

Braubach, L. and Pokahr, A. (2007). *Jadex Tutorial*.

Braubach, L. and Pokahr, A. (2007). *Jadex Tool Guide*.

Braubach, L. and Pokahr, A. (2007). *Jadex User Guide*.

Bellifemine, F., Rimassa, G. and Poggi, A. (1999). *JADE – A FIPA-compliant agent framework. In 4th International Conference on the Practical Applications of Agents and Multi-Agent Systems (PAAM-99)*, pages 97–108, London, UK, December 1999.

Brown, S. (2008). *Scalability Principles*.

Boloni, L. and Marinescu, C. (1999). *A Component Agent Model – from Theory to Implementation*.

Boshoff, W. H. and Ehlers, E. M. (2006). *Reusable Component Oriented Agents: A New Architecture*.

Busetta, P., Ronnquist, R., Hodgson, A. and Lucas, A. (January 1999). JACK *Intelligent Agents – Components for Intelligent Agents in Java*, AgentLink Newsletter 2.

Clements, P. (1996). *A Survey of Architecture Description Languages*. In Proceedings of the Eighth International Workshop on Software Specification and Design (pp. 16-25). Paderborn, Germany: ACM.

Cossentino, M., Sabatucci, L., Sorace, S., and Chella, A., (2003). *Patterns reuse in the PASSI methodology*. Fourth International Workshop Engineering Societies in the Agents World (EASW '03) – 29-31 October 2003, Imperial College London.

Dana Moore and Bill Wright (2002). *Java^{TM} Technology-based Internet Applications Using Software Agents and P2P*. Sun's 2002 Worldwide Java Developer Conference.

Demicheil, L., Yalcinalp, L., & Krishnan, S. (2001). *Enterprise JavaBeans Specification Version 2.0*.

Deugo, D., Weiss, M., and Kendall, E. (2001). *Reusable patterns for agent Coordination*. In Omicini, A., Zambonelli, F., Klusch, M., and Tolksdorf, R., (eds.) Coordination of Internet Agents: Models, Technologies and Applications, Springer-Verlag.

Fernandez, E. B., & Yuan, X. (2000). *Semantic Analysis Patterns*.

Fisher, M., Muller, J., Schroeder, M., Staniford, G. and Wagner, G. (1997). *Methodological Foundations for Agent-Based Systems*. The Knowledge Engineering Review, 12(3): 323-329.

Gamma, E., Helm, R., Johnson, R. and Vlissides, J. (1995). *Design Pattern: Elements of Reusable Object-Oriented Software*. Addison-Wesley, Reading.

Garlan, D., & Shaw, M. (1996). *Software Architecture: Perspectives on an Emerging Discipline*. Prentice Hall.

Ghitescu, A. (2007). *Performance Evaluation of Java Management eXtension (JMX) Models*.

Helsinger, A., (2007). *Cougaar: A Scalable, Distributed Multi-Agent Architecture*.

http://disi.unitn.it/~dnguyen/ejade/download.html

http://docs.cougaar.org

http://jade.tilab.com/

http://sourceforge.net/projects/jadex/

http://vsis-www.informatik.uni-hamburg.de/

http://vsis-www.informatik.uni-hamburg.de/projects/jadex/download.php

http://www.cougaar.org

http://www.eclipse.org/

http://www.sparxsystems.com.au

www.wikipedia.org

Jawawi, D., Mamat, R. and Deris, S. (2007). *A Component-Oriented Programming for Embedded Mobile Robot Software*. Proceedings of the International Journal of Advanced Robotic Systems, Vol. 4, No. 3 (2007), ISSN 1729-8806, pp. 371-380.

Jenning, N. R. (2000). *On Agent-Based Software Engineering*. Artificial Intelligence, 117 (2000), 277-296.

Jennings, N. R., and Wooldridge, M. J. (1998). *Applications of Intelligent Agents, in Agent Technology: Foundations, Applications, and Markets, eds*. Jennings, N.R. and Wooldridge, M., pp. 3-28.

Jennings, N. R., Faratin, P., Lomuscio, R., Parsons, S., Sierra, C. and Wooldrigde, M. J. (2001). *Automated negotiation: prospects, methods and challenges*. International Journal of Group Decision and Negotiation, 10(2):119.215.

Kendall, E. A., Murali Krishna, P.V., Pathak, C.V. and Suresh, C.V. (1998). *Patterns of Intelligent and Mobile Agents*. Proceedings of the Second International Conference on Autonomous Agents, May 9-13, pp. 92-99, Minneapolis, Minnesota, USA.

Konolige, K., Myers, K.L., Ruspini, E.H., and Safflotti, A., (1997). *The saphira architecture: A design for autonomy*. Journal of experiment & theoretical artificial intelligence (JETAI), Vol 9(1), pp. 215 – 235.

Lau, K. K., & Wang, Z. (2005). *A Taxonomy of Software Component Models*. Proceeding of the 2005 31st EUROMICRO Conference on Software Engineering and Advanced Applications .

L. Braubach, W. Lamersdorf, and A. Pokahr (2003). *Jadex: Implementing a BDI-Infrastructure for JADE Agents*. Pages 76-85, exp - Volume 3 – n.3 – September 2003.

M. Bratman (1987). *Intention, Plans, and Practical Reason*. Harvard University Press, Cambridge, Massachusetts.

Meyer, B. (2003). *The grand Challenge of Trusted Components*. Proceedings ICSE 2003 IEEE, 660-667.

Meyer, B. (1997). *Object-Oriented Software Construction*. Prentice-Hall.

Michele Piunti, Alessandro Ricci, Lars Braubach, and Alexander Pokahr (2008). *Goal-Directed Interactions in Artifact-Based MAS: Jadex Agents playing in CARTAGO Environments*. 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology.

Musavi, M., Dolcet, X., and Tejedor, E. (2008). *Modeling the Poker Game with Jadex Agents*.

Mohamad, R., Deris, S., and Ammar, H.H. (2006). *Pattern-Oriented Design for Multi-Agent Systems: a Process Framework*. Proceedings of the International Conference on Software Engineering Research and Practice (SERP'06), June 26-29, Las Vegas, Nevada, USA, pp. 350-356.

Mohamad, R., Deris, S., and Ammar, H.H. (2006). *Agent Design Patterns Framework for MaSE/POAD Methodology*. IEEE International Conference on Computer Systems and Applications, March 8, 2006, pp. 521 – 528.

Mohamad, R., Deris, S., and Ammar, H.H. (2007). *Pattern-Oriented Design for Multi-agent System: A Conceptual Model*. Proceedings of the Journal of Object Technology, May-June 2007, Vol. 6, No.4, pp. 55-75.

Mountzia, M. A. (1996). *An Intelligent-Agent Based Framework for Distributed System Management*. In Proceeding of 5th European Conference on Patterns Languages of Programs (EuroPLoP 2000), 5-9 July, Irsee, Germany.

Myers, K. L. (1997). *User Guide for the Procedural Reasoning System*. Technical Report, Artificial Intelligence Center, California, USA.

Nor Azizah Sa'adon (2009). *Pattern Oriented Design Framework For Agent Based Smart Wheelchair Using Agentpod Framework*. Master of Science (Computer Science) Thesis, Universiti Teknologi Malaysia.

Nor Hazilawati Awang, W M N Wan Kadir, and Shamsul Shahibuddin, 2009, *Comparative Evaluation of the State-of-the-art on Approaches to Software Adaptation*.

Nierstrasz, O., Grenssler, T., & Schonhage, B. (2002). *Components for Embedded Software*. The PECOS Approach CASES . OMG. (2007). UML Superstructure Specification. http://www.omg.org.

Oluyomi, A., Karunasekera, S., and Sterling L. (2008). *Description templates for agent-oriented patterns Source*. Journal of Systems and Software, Vol. 81, Issue 1, pp 20- 36.

Parra, C. and L. Duchien. *Model-Driven Adaptation of Ubiquitous Applications*. In Proceedings of the First International DisCoTec Workshop on Context-aware Adaptation Mechanisms for Pervasive and Ubiquitous Services (CAMPUS 2008). 2008: EASST.

P. Busetta, N. Howden, R. R¨onnquist, and A. Hodgson. *Structuring BDI Agents in Functional Clusters*. In N. R. Jennings and Y. Lesp´erance, editors, Intelligent Agents VI, Proceedings of the 6th International Workshop, Agent Theories, Architectures, and Languages (ATAL) '99, pages 277–289. Springer, 2000.

Qusay H.Mahmoud (2004). *Getting Started with Java Management Extensions (JMX): Developing Management and Monitoring Solutions*.

Radziah Mohamad (2007). *Pattern-Oriented Design Approach for Developing Multi-Agent Systems*. Doctor of Philosophy (PhD. Computer Science) Thesis, Universiti Teknologi Malaysia.

Rao, A. S. and Georgeff, M. P. (1995). B*DI Agents: From Theory to Practice*. Technical Note 56. Australian Artificial Intelligence Institute. Victoria, Australia.

Schmid, H. (1996). *Creating Applications from Components: A Manufacturing Framework Design*. IEEE Software, 13(6):67-75, November.

Schmidt, D.C. (1995). *Using Design Patterns to Develop Reusable Object-Oriented Communication Software*. Communications of the ACM.

Seyler, F. and Aniorte, P. (2002). *A Component Meta Model for Reused-Based System Engineering*.

Sparx System UML Tutorials, (2004). *The Component Model. Enterprise Architect*.

Szyperski, C. (2002). *Component Software: Beyond Object Oriented Programming*. Addison Wesley Longman, Reading, Mass, 2002.

Todd, W. (2007). *Cougaar Overview*.

Thome, M. and Wright, T., (2004). *The Cougaar Component Model*.

Winikoff, M. (2005). *JACK TM INTELLIGENT AGENTS: AN INDUSTRIAL STRENGTH PLATFORM*.

Wooldridge, M. (1999). *Intelligent Agents, in Multi Agent Systems: A Modern Approach to Distributed Artificial Intelligence*, ed. Weiss, G., 27-77, London: The MIT Press.

Wooldridge, M. J. (2002). *An Introduction to MultiAgent Systems. Chichester, England*. John Wiley and Sons.

Wooldrigde, M. and Ciancarini, P. (2001). *Agent-oriented software engineering: The state of art*, Agent-Oriented Software Engineering, Lecture Notes in Artificial Intelligence, vol. 1957, Ciancarini P, Wooldrigde M (eds.). Springer: Berlin.

Yacoub, S. M., Xue, H., & Ammar, H. H. (2000). *Automating the Development of Pattern Oriented Designs for Application Specific Software Systems*. Proceeding IEEE.

Yacoub, S.and Ammar, H.H, (2004). *Pattern-Oriented Analysis and Design: Composing Patterns to Design Software System*. Addison Wesley.

Yau, S., & Dong, N. (2000). *Integration in Component based Software Development Using Design Pattern*. Proceeding IEEE.