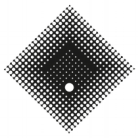
VICTORIA UNIVERSITY
MELBOURNE AUSTRALIA

Application of Multi-Agent Technology to Information Systems: An Agent-Based Design Architecture for Decision Support Systems

This is the Published version of the following publication

Zhang, Hao Lan, Raikundalia, Gitesh K, Zhang, Yanchun and Yu, Xinghuo (2009) Application of Multi-Agent Technology to Information Systems: An Agent-Based Design Architecture for Decision Support Systems. Australasian Journal of Information Systems, 15 (2). pp. 91-107. ISSN 1449-8618

The publisher's official version can be found at
https://journal.acs.org.au/index.php/ajis/article/view/24
Note that access to this version may require subscription.

Downloaded from VU Research Repository https://vuir.vu.edu.au/4082/

APPLICATION OF MULTI-AGENT TECHNOLOGY TO INFORMATION SYSTEMS: AN AGENT-BASED DESIGN ARCHITECTURE FOR DECISION SUPPORT SYSTEMS

Hao Lan Zhang
School of Computer and Electrical Engineering,
RMIT University, Melbourne, Australia

Gitesh K. Raikundalia
School of Engineering and Science
Victoria University

Yanchun Zhang
School of Engineering and Science
Victoria University

Xinghuo Yu
School of Computer and Electrical Engineering,
RMIT University, Melbourne, Australia

ABSTRACT

One of the most difficult issues in building efficient Information Systems (IS) is the integration of these systems with the organization's other systems. This issue is particularly acute for Decision Support Systems (DSSs). To become more effective and efficient, a DSS must have an open structure to adapt to the dynamic environment. However, current IS, especially DSSs, tend to rely excessively on traditional System Development Life Cycle (SDLC) and this places limitations on current systems' infrastructures. The emergence of multi-agent technology addresses this issue and its applications to IS are becoming highly efficient. In this paper, we introduce a Matrix-Agent connection design, called Agent based Open Connectivity for Decision Support Systems (AOCD), which balances the manageability and flexibility in a system and maximizes system performance.

INTRODUCTION

During the past decade, the successful applications of IT-based systems in business processes have declined because of obstacles in the development of effective IS for business processes. A survey provided by The Standish Group (The Standish Group, 2001) shows that in U.S.A, 23% of IS failed outright and 49% systems were late and or went over budget. Apart from the U.S., similar cases

were also found in the U.K and other countries. The integration difficulty is one of the most crucial reasons that causes the failure of IS application (Turban, et al, 2005).

Problems in Existing DSS

Among the various obstacles, Steven Alter (Alter, 1999) summarizes five major obstacles when applying information technology in the real world, which include:

(1) Unrealistic expectations and techno-hype.

(2) Difficulty building and modifying IT-Based Systems.

(3) Difficulty integrating IT-based systems.

(4) Organisational inertia and problems of change.

(5) Genuine difficulty anticipating what will happen.

Most of these obstacles are caused by the obturation of current information system structures. Figure 1 shows a schematic view of a traditional Decision Support System. From Figure 1, we find the system structure of traditional DSS is closed and that causes many problems, such as lack of compatibility, lack of adaptability, lack of expandability, and cost ineffectiveness in SDLC. To transform an IT-based IS into an open system, multi-agent technology can be deployed. Multi-agent technology is a superior solution that can enhance a system's connectivity, extensibility, reusability, and manageability.

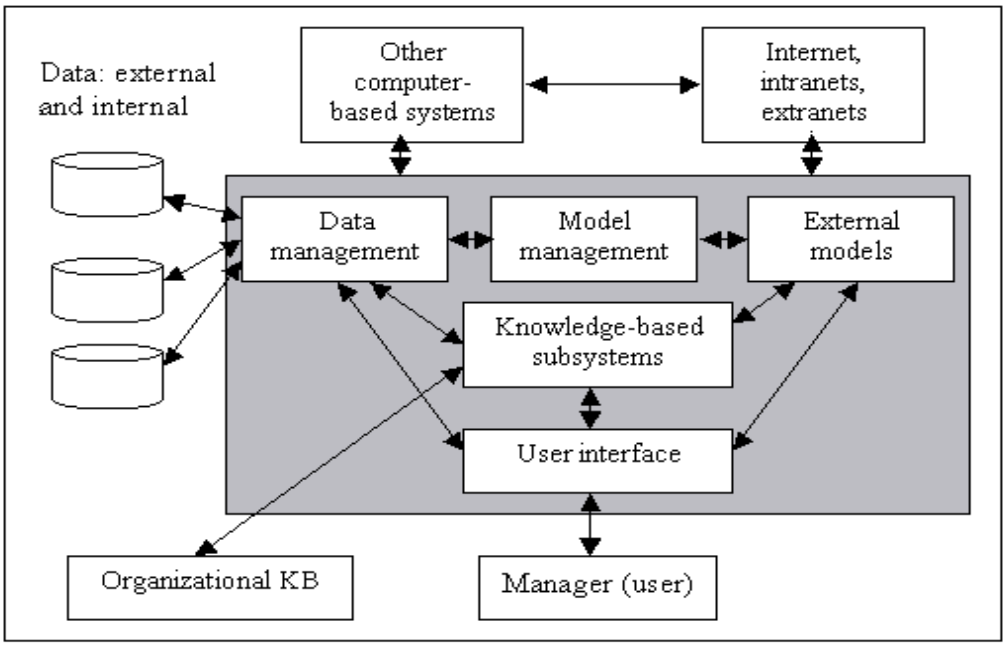

Figure 1: Schematic view of Traditional DSS. (*Source from Turban, et al, 2005)*

Intelligent agents are goal-oriented, collaborative, flexible, self-starting, intelligent, mobile and interactive (Turban, et al, 2005). These characteristics of agents ensure the applications of multi-

agents in DSS to be successful. Rapid development of intelligent agent technology changes SDLC. Traditional information systems involve seven development phases, which include: requirement phase, specification phase, design phase, implementation phase, integration phase, maintenance phase, and retirement. Although an Object-Oriented SDLC emphasises individual object design, it still based on these seven phases with conceptual changes. Multi-agent-based systems completely shift current system development methodologies. Such systems offer "Plug-and-Play", which allows external components (agents) to be integrated into an existing system without disturbing the system structure and working process. We believe multi-agent-based systems will eventually replace current information system model, including object-oriented system model. Objects or components in current information systems will be replaced by intelligent agents, which are more powerful, more intelligent, more flexible, and proactive. Based on previous studies (Turban, et al, 2005; Wagner, 2003; Zhang et al, 2005), we draw a comparison between traditional IS and AOCD-based IS (agent-based) as shown in Table 1.

Characteristics	Traditional Information Systems	AOCD based Information Systems
Compatibility	Poor	Good
Consistency	Good	Modest
Cost-effectiveness	Modest	Good
Expandability	Poor	Good
Flexibility	Poor	Good
Stability	Good	Uncertain
Reusability	Poor	Good

Table 1: Comparison of Traditional and AOCD-based Information Systems

Multi-agents Applications in DSS

Back in 1975, Hewitt *et al*. (1975) described an actor concept, which has many similarities to today's multi-agent-based DSS. Current applications of multi-agent technology in DSS have been mainly focusing on decentralized frameworks. For instance, Ossowski *et al*. (2004) propose and implement the multi-agent decision support system in transportation management; Vahidov (2006) suggests the use of intelligent agents in situated systems, and Padgham and Winikoff (2002) introduce the Prometheus methodology for building Belief-Desire-Intention (BDI) based distributed agent systems. However, these systems mainly rely on the efficient concurrent control and synchronous communication technologies. Unfortunately, current technologies in these areas are not efficient enough, particularly, when they deal with large volumes of transactions. Many decentralized multi-agent systems suffer from inefficient manageability (Minar, 2002), particularly, with respect to the difficulties of handling concurrent control and synchronous communication problems.

Here, we propose a novel multi-agent-based architecture for DSS, called *Agent-based Open Connectivity for DSS* (*AOCD*). AOCD architecture adopts a Matrix-agent connection design (see Figure 2), which overcomes the limitations of decentralized multi-agent systems by using a hybrid framework. The Matrix-Agent connection design involves three major components: agents, a Matrix center, and databases. The Matrix-Agent connection design adopts a two-phase communication mechanism, which involves communication between two agents and communication between the Matrix and the agents. There are two stages in AOCD framework design. The first stage involves the Matrix-Agent connection design, which forms the backbone of AOCD architecture. The second stage involves the individual agent design, which focuses on solving specific problems. Generally, once the Matrix-Agent connection design is made, it will not be changeable unless the overall system is replaced. On the contrary, the agents are dynamic components, which may be updated constantly. Each agent has the ability to solve problems in its domain. An agent is able to seek cooperation from other agents when it has difficulties in solving a problem, and this makes the AOCD architecture more powerful than other non-cooperative DSS architectures. The AOCD architecture is more powerful in that it mirrors the way human groups make decisions: by using a set of interacting agents, more creative and innovative decisions can be produced as a result of such cooperation and interaction.

The aim of AOCD architecture design is to provide an open, efficient, and flexible architecture for DSS. It has been focusing on the architectural design of AOCD system, which provides a foundation for future AOCD research. The previous research involves:

(1) Introducing the concept of Matrix-Agent connection, which has been proven efficient and effective compared with other decentralized or centralized frameworks (Wagner, 2003),
(2) Experimental design of agent network topologies in AOCD-based systems (Zhang, et al, 2005; 2006a).

This paper is structured as follows; in the next section, we provide an example scenario to clarify the implementation of AOCD. In the Matrix-Agent connection design section, we briefly introduce the design of the Matrix-Agent connection, which mainly focuses on *(i)* the Matrix interface design, *(ii)* the Matrix Register design, and (*iii*) the agent architectural design. The final section concludes our research work and considers future issues that need to be addressed.

A CASE STUDY OF AOCD APPLICATION IN IS

Most organisational managers often find it is very difficult to decide whether their organisational information systems need an upgrade or not (Alter, 1999). There are many factors that plague decision makers about whether or not to upgrade their systems, which, of course, are traditional SDLC based systems.

The first factor is the enormous time consumption for the development of a traditional information system. For traditional SDLC based systems, the upgrade period not only includes the time for designing specific components but also includes the time for re-evaluating user requirements, modifying existing system structure (possible re-design when upgrade components affect the overall system processes), integrating a new system to maintain system consistency and extra outsourcing manpower is often used for communication in every upgrade phase. Meanwhile, the training processes for staff to understand new systems also take time.

The second factor is the high possibility of upgrade failure. As we mentioned in the introduction section, the success rate of developing IT-based systems for business process is disappointing. Managers often face the risk of system failure when developing or upgrading a new system.

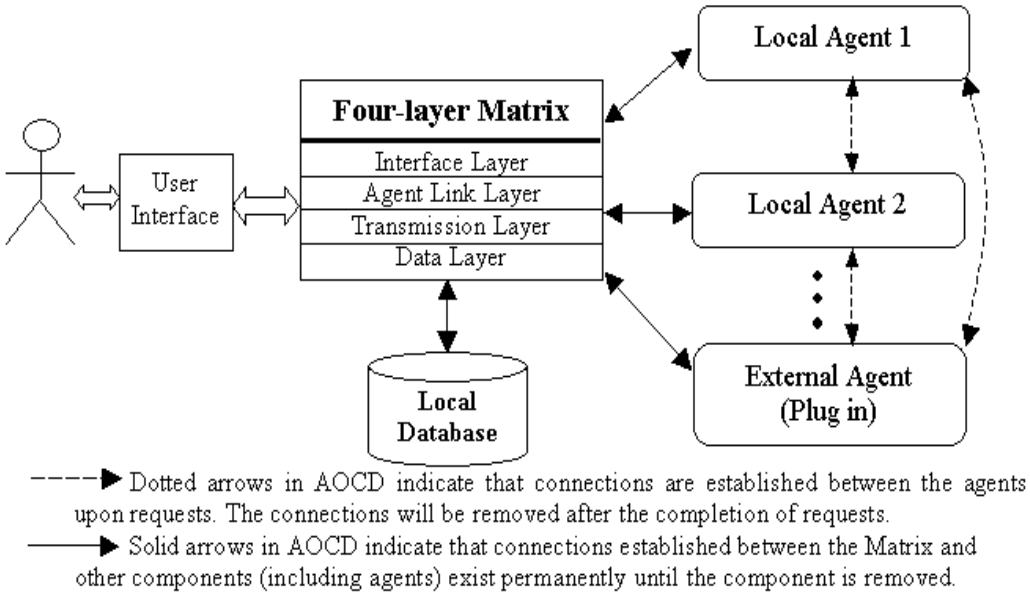

Figure 2: Schematic view of AOCD-based DSS. (*Source from Zhang, et al, 2005*)

Thirdly, the dynamic environment might cause their upgrade useless. People are living in a so-called information explosion age, which makes our day-to-day environment very difficult to predict. New technologies, new business opportunities, and new inventions are emerging everyday. In this circumstance, any system upgrade cannot last for long. Therefore, the constant upgrade processes confuse organisational mangers. Certainly, there are many other reasons for the problems of upgrading or re-designing, such as the technology gap between existing and new system, system structural inefficiency that results in some unknown information loss between the new and existing system, human errors that might occur when constantly shifting their familiar interfaces to some unfamiliar interfaces, etc.

The use of multi-agent technology is able to solve the above problems. AOCD is a multi-agent-based architecture, which is open and flexible. To illustrate the effectiveness and efficiency of the architecture we use the following case as an example.

Nowadays, mergers of international corporations happen frequently as the globalisation phenomenon has extended to today's economic section. Hidden behind the encouraging benefits of merger, most merging corporations have encountered difficulties in integrating various information systems from different mergees. In many traditional information system development frameworks, the process of combining two different information systems, especially DSSs, from two different organizations into one system is time-consuming, costly and inefficient. The traditional

development process may involve re-analysing the user requirements, reconstructing the database, redeveloping the code, etc. For instance, during the merger process of the German automaker Daimler Benz with the U.S. automaker Chrysler (Alter, 1999; Zhang, et al, 2008), difficulties were found when integrating Daimler's CAD systems with Chrysler's CAD systems because the two companies' CAD systems were designed based their own company's business and manufacturing processes and their CAD systems were incompatible with each other. Many similar cases can be found when merging two corporations. Using the AOCD architecture can easily solve the integration obstacles that Chrysler and Daimler faced. To simplify the case, we use two fictitious companies, Company A and Company B, to explain the merging process in AOCD-based systems.

Company A and Company B are the manufacturer of vehicles. Both companies have developed two different information systems based on AOCD architecture. In Company A, there are five major intelligent agents that include: sales agent, human resource agent, accounting agent, marketing agent, and manufactory agent (including CAD systems). Each agent is responsible for the information process in its business domain. The human resource agent manages current staff information and offers advice to the manager if it is necessary to reduce labour of the company. The sales agent provides information such as current sales performance, sales geographic distribution, etc., to decision makers. The marketing agent provides the information about the company's current marketing status, rivals' marketing status, and the market trends, and so on. The accounting agent mainly deals with staff salaries, reimbursement, supply, income payments, etc. The manufactory agent is responsible for controlling auto-manufacturing systems, CAD systems, warehouse management, etc.

Company B's information system is different from Company A's as its business strength is auto-motor design. There are four major intelligent agents in Company B's system including: human resource agent, financial agent (including accounting, sales and marketing functions), manufactory agent (including CAD systems), and design agent. The human resource agent and manufactory agent of Company B are similar to Company A's. The financial agent deals with all the financial issues, including current sales performance, sales geographic distribution, the company's current marketing status, rivals' marketing status, market trends, staff salary, reimbursement, supply, income payments, and so on. The design agent provides the latest auto-motor design information, existing auto-motor design pattern database, new auto-motor design software information, and so on.

As the strategic alignment would benefit both companies, the executive boards of the two companies decide to merge two companies. There would be many problems to merge the companies' information systems if they were using traditional information architectures because the headquarters of the two companies are located at two different cities and their CAD systems embedded in Manufactory Agent are incompatible. Now, most of the difficulties can be solved within AOCD framework. Figure 3 shows the two companies' information systems before the merger. There are some similarities in the two companies' information systems, the two human resource agents are very similar and Company B's financial agent can deal with all the financial issues in Company A's framework.

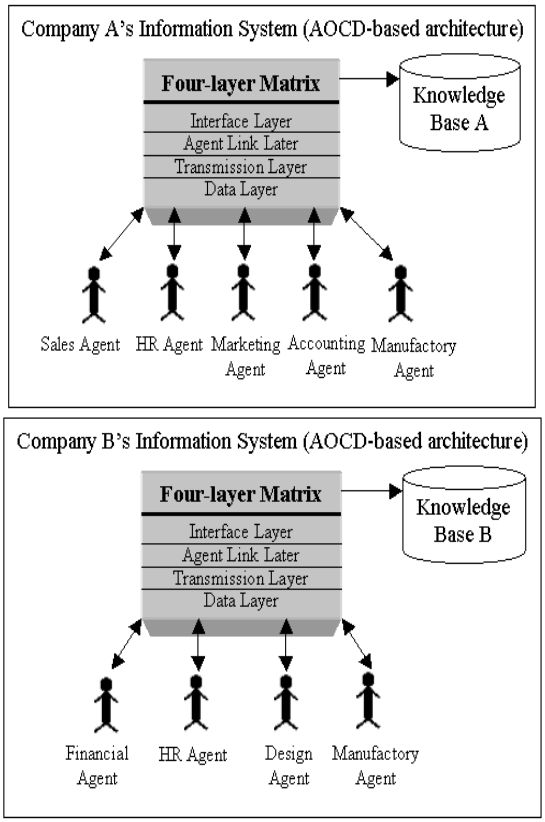

Figure 3: Two Companies' AOCD-based Information Systems (before merger)

Matrix in AOCD is standardized; therefore, the integration process can use one Matrix. Because the human resource agents (HR agents) in two companies' information systems are very similar, the integration process can use one HR agent instead of using both of them. Company B's financial agent can replace the sales agent, marketing agent and accounting agent of Company A as the financial agent is able to deal with all the financial issues in Company A. The integration process is simple and easy because AOCD design supports plug-and-play. The integrated information system after merger (see Figure 4) can support both companies' business processes. The new company's information system retains two manufactory agents because the former two companies' CAD systems were incompatible. The new information system also retains two knowledge bases, which are located at the two headquarters, respectively. Matrix can be located at one of the headquarters and the agents are connected to the Matrix through local area network, wide area network and the Internet.

As we can see from this case, the difficulties in the integration process, which plague traditional information systems, are solved in AOCD architecture. In AOCD architecture, there are no complicated re-design procedures involved in integration or upgrade process and the risk of upgrading failure is minimized or even completely eliminated. The advantages of using AOCD architecture in information systems, especially DSS, can be summarised as follows (Bill, 1998): (*i*)

Speed-up and efficiency, (*ii*) robustness and reliability, (*iii*) enhancing scalability and flexibility, (*iv*) cost effectiveness and (*v*) reusability.

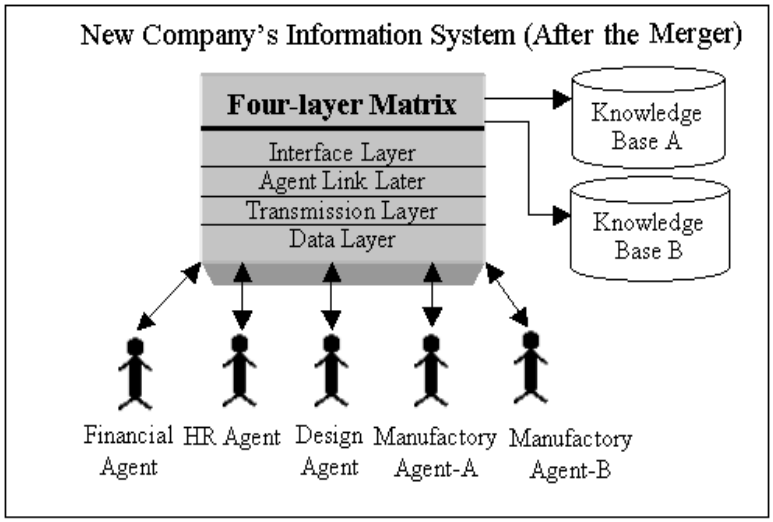

Figure 4: New Company's information system (after merger)

MATRIX-AGENT CONNECTION DESIGN IN AOCD

The *Matrix* concept is one of the key ideas in the AOCD architecture. The basic framework of AOCD architecture is formed by *Matrix-Agent* connections, which are done through the connection interfaces provided by Matrix. A connection interface will be allocated to an agent when the Matrix receives a request of connection from the agent. An agent can connect to the Matrix at any time without disturbing the current system. Matrix offers a living platform for agents. The use of the Matrix component distinguishes AOCD architecture from other multi-agent based systems (Weiss, 1999). The Matrix consists of three major parts include (*i*) *agent society* that provides a virtual space to agents, (*ii*) *information acquisition section* that acquires knowledge from external environment, and (*iii*) *Matrix control panel* is the core part that mainly handles agent matching, requests processing, and resource allocation.

The Matrix consists of four layers: interface layer, agent link layer, transmission layer, and data layer. The interface layer provides interfaces for establishing connection between agents and the Matrix. The agent link layer provides two operation mechanisms. The first mechanism is a manual linking process, which enables a user to link different agents, in order to solve a problem through agent-cooperation. The other mechanism is an auto-match process, which allows the Matrix to match the corresponding agents by using the information provided by the Matrix register. The transmission layer provides users with agent language communication functions and message receiving and sending functions. The data communication layer enables Matrix to establish a connection with the local database in the AOCD framework.

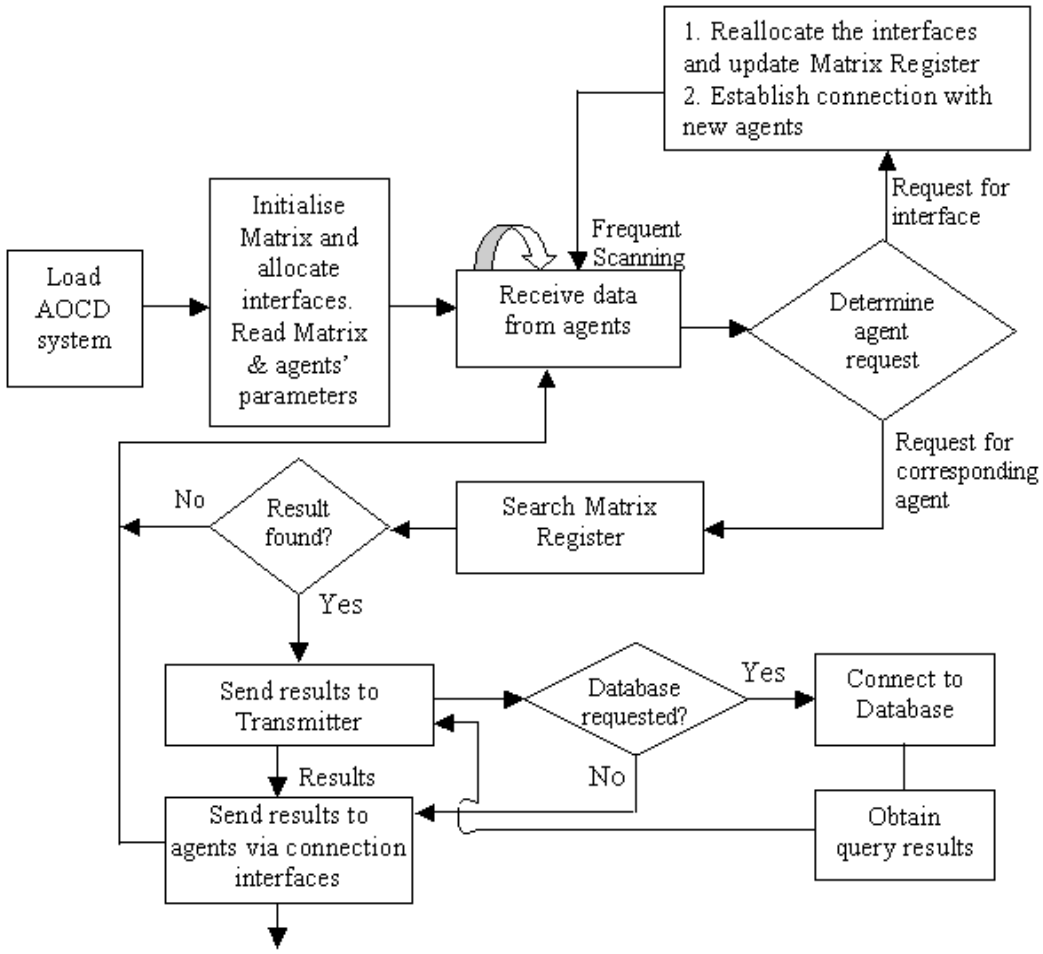

Figure 5: Operation of AOCD System (*Source from Zhang, et al 2006c*)

Figure 5 shows the operation process of the AOCD architecture. An AOCD system starts with loading the Matrix control panel, which contains the information of previously allocated connection interfaces, such as the number of connection interfaces allocated in the last system operation. When the connection interfaces have been established, the Matrix control panel begins to receive requests from the agents. There are two types of requests: one is the request for connection and another is the request for corresponding agents. Once a request for connection is received, the Matrix control panel will change the Matrix register and allocate a new connection interface for the new agent. In the meantime, a *Dynamic-detection* process is frequently performed to ensure the system is receiving requests. If a request for a corresponding agent is received, the Matrix will forward the request to the Matrix register to search for a corresponding agent or agents. If a corresponding agent is found, the result will be delivered to the *Transmitter* to encode/decode the request into a set of recognizable transmission protocol and operation commands. If the request agent requires information from the database, then the Matrix needs to perform the database operation according to

the decoded commands. The database sends the feedback to the Transmitter. The final step is sending results to the corresponding agent via the connection interface.

Matrix Interface and Matrix Register Design

Matrix interfaces are located at the Matrix interface layer. When a Matrix receives the connection requests from the agents, the Matrix allocates a number of interfaces (according to requests) and establishes connections between the Matrix and the agents. The number of interface(s) provided by a Matrix to connect with agents is unlimited. A *Dynamic-detection* method is deployed to identify the number of interfaces generated for an agent. In Dynamic-detection method, an *Interface Tag* is deployed to indicate whether a connection request has been received by the Matrix. Once a Matrix detects the indication of a new connection, a new interface will be allocated to the connection. The Matrix examines the interface tag and allocates interfaces to agents according to the value of the interface tag when the AOCD system starts to load the Matrix. A new connection request changes the value of the interface tag that triggers the re-allocation of the connection interface resources.

The *Matrix Register* plays a primary role in the Agent-link layer. The major functionalities of Matrix Register include: *(i)* searching for a corresponding agent based on the descriptions, and *(ii)* providing the corresponding agent address to Transmitter. The Matrix Register makes use of a three-column list, which contains the agents' details, functionality descriptions, and data structure information. The agent information column contains the information of an agent name and agent IP address. The functionality column contains the information of agent's functionalities, which is used to provide agent information when the Matrix searches for a corresponding agent. The Detail description column consists of four major items: organization/department, data table name, data attribute, and description control (if applicable).

A Unified Matrices Architecture

The Matrix-agent framework combines the centralised topology and the peer-to-peer communication (See Figure 2). The bottleneck problem rises when the framework expands to a large scale because of the limitation of the centralised topology. To tackle the bottleneck problem, a *Unified Matrices Framework* has been introduced (Zhang, et al, 2006b). In the Unified Matrices Framework, a number of Matrices are connected through the Matrix interface layer. A Matrix is allocated a number of agents according to the agents' locations and domains. Once the agents in a Matrix cannot deal with a request, the Matrix will forward the request to its nearest or most familiar Matrix for further processing. This unified Matrices framework minimizes the bottleneck problem in a large-scale agent network, such as AOCD framework.

The process of searching a service-providing Matrix involves two major methods, including the *Most familiar partner* method and *Supplemental partner* method. We use a Matrix capability description table to describe the functionalities and capabilities of a Matrix. This table contains the information of all the connected agents' functionalities and domain information, indicating the major area for which an agent is responsible. Each agent's functionality is described by several keywords and these keywords also represent the functionalities and capabilities of the Matrices. Generally, there are some keyword intersections among the Matrices (Figure 6). The number of keyword intersections of a requesting Matrix with the other service-providing Matrices are calculated to generate *intersection scores*. If a service-providing Matrix has the greatest intersection score with a requesting Matrix (compared to other Matrices), then this Matrix is regarded as the most familiar partner of the requesting Matrix. Whereas if a service-providing Matrix has the smallest intersection score with a requesting Matrix, then this Matrix is regarded as the

supplemental partner of the requesting Matrix. Both methods follow a four-step calculation procedure. The intersection score is calculated from the following equation.

$$R_{aj} = \sum_{i=1}^{\#Sa} (R_{aj} + 1/M_i) \quad (1)$$

where R_{aj} is the keyword intersection score between Matrix a and Matrix j. S_a denotes a keyword set of Matrix j that intersects with S_a and S_j. M_i denotes the total number of the Matrices (including the Matrix a) that have the same intersections with Matrix a. The detailed calculation procedure can be found in (Zhang, et al, 2006b).

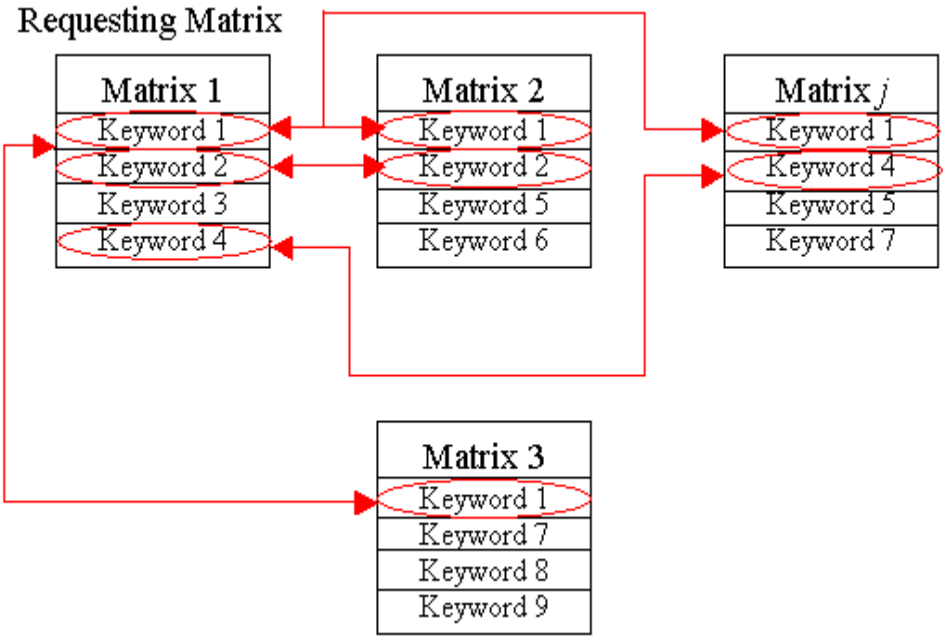

Figure 6. Intersections of Matrices' capabilities (Source from Zhang, et al, 2006d)

Agent Architectural Design in AOCD

The agent architectural design in AOCD architecture adopts BDI model (Zhang, et al, 2006d), which was first described in IRMA architecture by Bratman, Israel, and Pollack (Zhang, et al, 2006c). The BDI model has been used widely in many industrial systems and it has been found efficient in some highly successful agent architectures to date (Bratman, et al, 1988). BDI is not the only option for AOCD agent design; layered architecture for agent design also has proven useful. Layered architecture design decomposes an agent's behaviours and creates separate subsystems to deal with these different types of behaviour. Touring Machines architecture (Wooldridge, 1999) is a typical layered architecture design. The main problem with the layered architecture is it lacks the conceptual and semantic clarity of unlayered approaches. This disadvantage makes the transformation from logic-based semantics to a layered architecture a very difficult process, as

many logic-based semantics cannot be separated completely. This weakness discourages the use of a layered architecture in AOCD agent design because the AOCD architecture is a DSS-based architecture that involves many logic-based semantics. We deploy a knowledge base to represent the information that the agent has about its current environment. Figure 7 shows the AOCD agent architectural design.

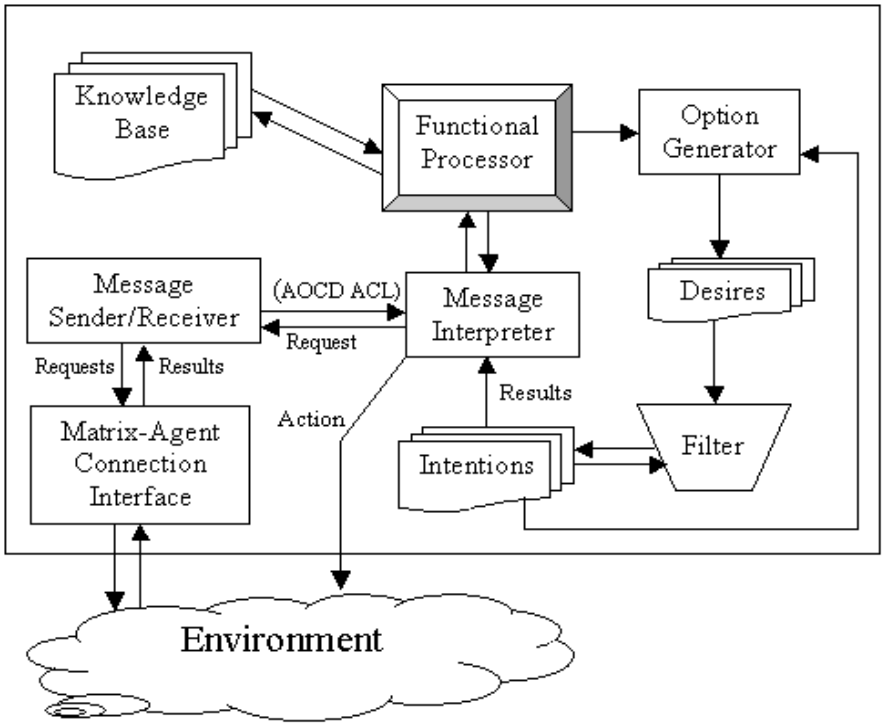

Figure 7: AOCD Agent Architecture Design (*Source from Zhang, et al, 2006c*)

An agent establishes a connection with the Matrix or other agents through the Matrix-Agent connection interface. Once a connection is established, the agent is able to send messages to or receive messages from the environment by using a Message Sender/Receiver component. The Message Sender/Receiver decomposes the message into several categories and forwards the content category to a Message Interpreter in the format of AOCD ACL. The Message Interpreter decomposes the received information and sends a set of commands to a Functional Processor. The Functional Processor provides the abilities of solving domain problems. A Knowledge Base provides the Functional Processor with a set of rules (beliefs) in the domain. The Functional Processor generates the results and delivers them to an Option Generator. The Option Generator maps a set of rules in the Knowledge Base (beliefs) and a set of intentions to a set of desires, which can be expressed logically as follows (Bratman, 1988). We adopt the following notations, if A is an arbitrary set, then $\wp(A)$ is the powerset of A, *Bel* represents the rules in Knowledge Base, *Int*

represents the Intention sets, and *Des* represents the desires sets. The options can be represented as a mapping.

Options: $\wp(Bel) \times \wp(Int) \rightarrow \wp(Des)$

The Filter function performs a deliberation process, which can be expressed as follow.

Filter: $\wp(Bel) \times \wp(Des) \times \wp(Int) \rightarrow \wp(Int)$

After the filtering process, an agent is able to generate one or more intentions, which will be sent back to the Matrix Interpreter. Matrix Interpreter has the responsibility to determine whether to forward results to the Matrix-Agent (or agent-agent) connection or take actions immediately. The Matrix Interpreter is designed with this ability because some messages from the environment do not require feedback, e.g., a manager sends a message to the sales agent demanding a printout of today's timetable. This message does not contain any complicated decision making process and is one-way. In this case, it is not necessary to send any feedback to the manager. Thus the Matrix Interpreter would simply take the action, which is producing a printout of timetable, instead of performing any further activities.

RESULTS OF THE AOCD TOPOLOGICAL EXPERIMENTS

AOCD framework is a hybrid multi-agent network. We have conducted a set of experiments to evaluate the topological performance of AOCD-based (hybrid) system (Zhang et al, 2006a). Figure 8 and Figure 9 show two snapshots of the AOCD topological experiments.

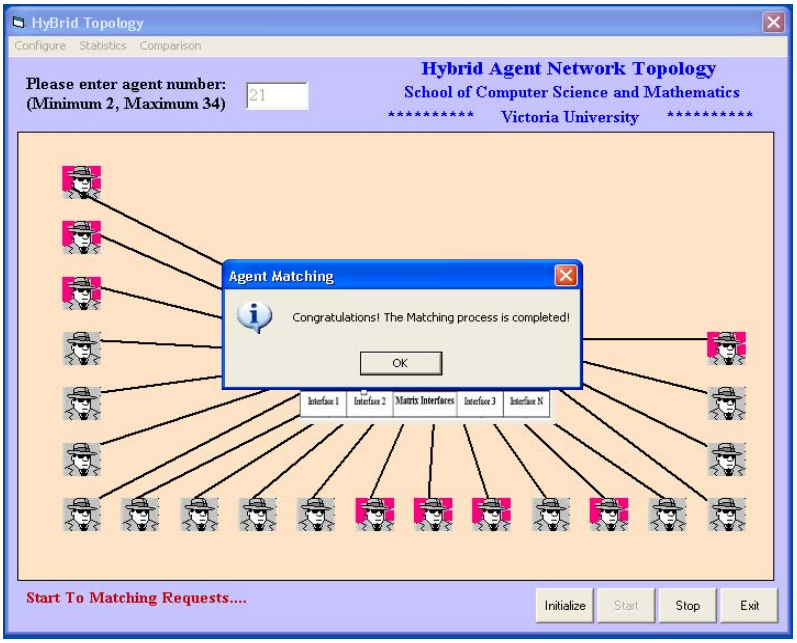

Figure 8: Agent Matching Process of the Experiment (*Source from Zhang, et al, 2006d*)

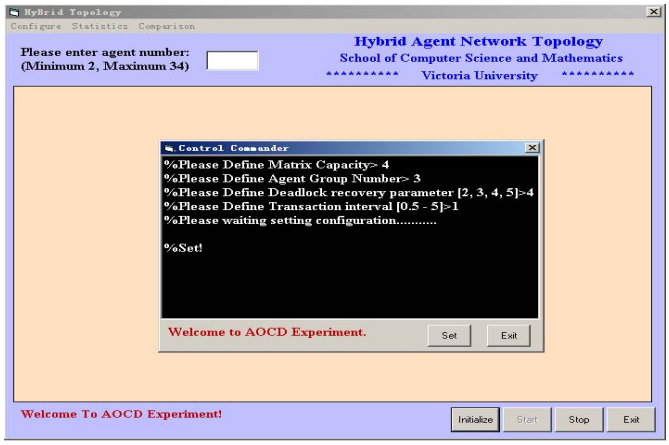

Figure 9: Configuration Interface of the Experiment

Our study shows the hybrid topology presents a superior performance compared to the other two topologies, especially the centralised topology. In the centralised topological experiments, the total time consumption for processing 2917 requests is 28477 seconds. The average time consumption for one request in the centralised system is 9.76 seconds, whereas the average time is only 3.41 seconds in the hybrid topology (Figure 10). In addition, the average success rates of the two topologies are 86.74% (centralised) and 95.45% (hybrid), respectively (Figure 11). The experimental design and more detailed experimental analysis can be found in our previous work (Zhang, et al , 2006a).

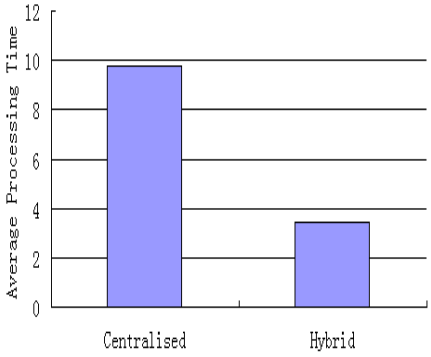

Figure 10: Average processing time comparison

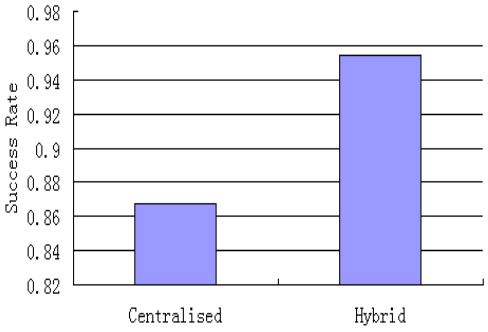

Figure 11: Success rate comparison

We noticed that for both centralised and hybrid topologies when the produce of the request-volume/participated-agents increases, the success rate of processing the requests appears to decrease (Figure 12 and 13).

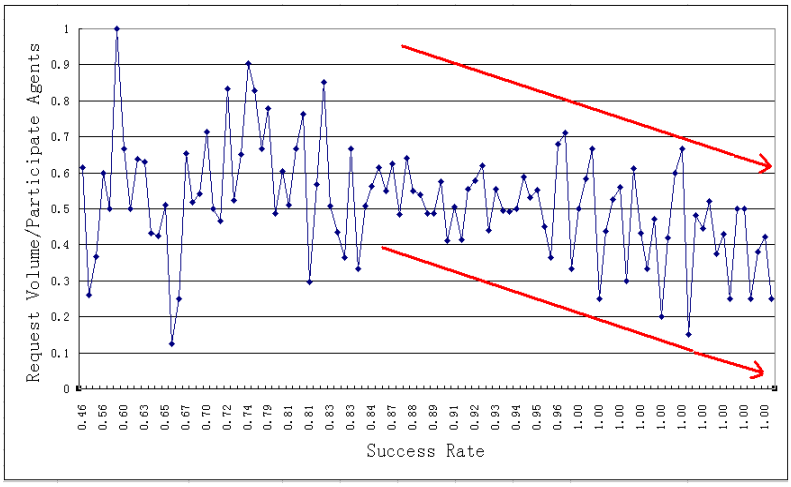

Figure 12: Decreasing trends of success rate in centralised topology (*Source Zhang, et al, 2006d*)

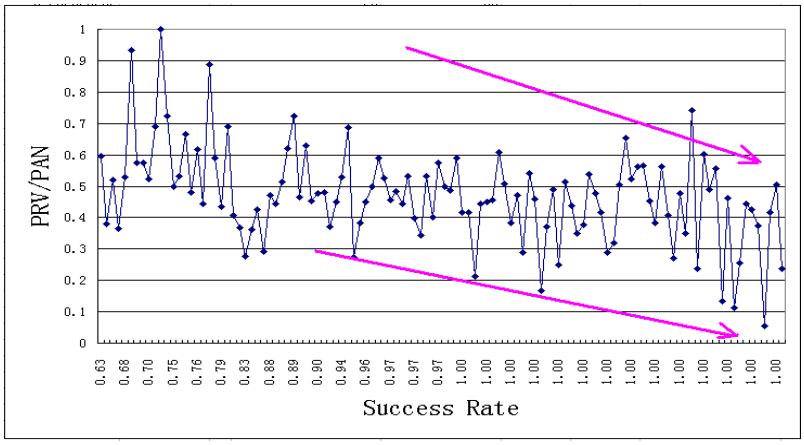

Figure 13: Decreasing Trends of Success Rate in Hybrid Topology (*Source Zhang, et al, 2006d*)

These trends suggest that to increase the success rate of processing the requests in AOCD-based systems, we need to balance the request volume and the number of participating agents. In other words, in order to process a number of requests, the number of participating agents should not be too small. The more agents that are involved in dealing with the requests, the probability of increasing the success rate is higher.

CONCLUSIONS AND FUTURE WORK

The significance of the AOCD design is that it:

(1) Provides a flexible plug-and-play architecture, which eliminates the difficulties of integrating different information systems or upgrading information systems,

(2) Combines centralized communication and decentralized communication in the same framework, which allows the agents to communicate in a more effective way. By imitating the way human communities make decisions, more creative and innovative decision outcomes can be produced as a result. A concrete performance analysis also has been performed (Zhang, et al, 2008), which indicates that such architecture is able to deliver good performance.

Future work on AOCD will mainly focus on three aspects:

- The use of agent design patterns (Ferguson, 1992; Aridor, 1998) in AOCD is considered to be helpful because of their flexibility, reusability, and understandability for agent application design. Future work can conduct a survey on current methodologies in the agent design pattern area.
- Developing an efficient agent-matching algorithm. Agent matching in AOCD is a crucial issue, which could enhance the efficiency of agent cooperation.
- Analysing the possible negative impacts of using multi-agent technology in information systems.

REFERENCES

Alter, S. (1999) Information Systems – a management perspective (3rd edition), Addison-Wesley Educational Publisher Inc., pp. 23-27.

Aridor, Y., & Lange, D. B. (1998) 'Agent Design Patterns: Elements of Agent Application Design', *Proceedings of the Autonomous Agents 1998 Conference*, Minneapolis, USA.

Bill, V. (1998) "The First Global Car Colossus", *Business Week*, pp. 40-43.

Bratman, M.E., Israel, D.J., & Pollack, M.E. (1988) 'Plan and resource-bounded practical reasoning', *Computational Intelligence*, pp. 349-355.

Ferguson, I., A. (1992) 'Touring Machines: An Architecture for Dynamic, Rational, Mobile Agents', *PhD thesis*, Clare Hall, University of Cambridge, UK.

Hewitt, C., Bishop, P., & Steiger, T. (1975) 'A Universal Mosular Actor Formalism for Artificial Intelligence', *Proceedings of the Fourth International Joint Conference on Artificial Intelligence*, pp. 235-245.

Minar, N. (2002) 'Distributed system topologies'. http://www.openp2p.com/lpt/a/1461 accessed 20 July 2005.

Ossowski, S., Fernandez, A., Serrano, J.M., Perez-de-la-Cruz, J.L., Belmonte, M.V., Hernandez, J.Z., Garcia-Serrano, A.M., & Maseda, J.M. (2004) 'Designing Multiagent Decision Support System-The Case of Transportation Management', *Proceedings of 3rd AAMS*, Vol. 3, New York, U.S.A.

Padgham, L., & Winikoff, M. (2002) 'Prometheus: A Methodology for Developing Intelligent Agents', *Proceedings of AAMAS 2002*. Bologna, Italy.

The Standish Group International, Inc. (2001) 'Extreme Chaos', Publication of CHAOS Report 2001 Standish Group International Inc.

Turban, E., Aronson, J. E., & Liang, T. (2005) 'Decision Support Systems and Intelligent Systems' (7th edition), Prentice Hall Publication, pp. 223 and pp. 109, 707.

Vahidov, R. (2006) 'Design Researcher's IS Artifact: a Representational Framework', Proc. of First International Conference on Design Science in Information Systems and Technology, Claremont, U.S.A.

Wagner, G. (2003) 'The Agent-Object-Relationship Metamodel: Towards a Unified View of State and Behavior', Information Systems, 28:5.

Weiss, G. (ed.) (1999) 'Multiagent Systems – A modern approach to distributed artificial intelligence', The MIT Press, pp. 8-9.

Wooldridge, M. (1999) 'Intelligent Agent', In Weiss, G. (Ed.) *Multiagent systems-A modern Approach to Distributed Artificial Intelligence*, The MIT Press, pp. 27-77.

Zhang, H. L., Leung, C.H.C., and Raikundalia, G. K. (2005), "AOCD: A Multi-agent Based Open Architecture for Decision Support Systems", *Proceedings of International Conference on Computational Intelligence for Modelling, Control & Automation*, Vienna, Austria, 2, pp. 295-300.

Zhang, H. L., Leung, C.H.C., and Raikundalia, G. K. (2006a), "Performance Analysis of Network Topology in Agent-based Open Connectivity Architecture for DSS", *Proceedings of 20th International Conference on Advanced Information Networking and Applications*, Vienna, Austria, vol. 2, pp. 257-261.

Zhang, H. L., Leung, C.H.C., and Raikundalia, G. K. (2006b), 'Towards the Matrix Concept: A Virtual Platform for Intelligent Agent Application in Decision Support Systems', *Proceedings of The 7th International Symposium on Knowledge and Systems Science (KSS2006)*, Beijing, China, pp. 115 – 122, Lecture Notes in Decision Sciences 8, Global-Link Publisher.

Zhang, H. L., Leung, C.H.C., and Raikundalia, G. K. (2006c), 'A Novel Matrix-Agent Connection Design in Multi-agent Based Architecture for Decision Support Systems', In Technical Reports of the IFIP TC8/WG 8.3 Open Conference, The London School of Economics, London, UK, 28 June – 1 July 2006.

Zhang, H. L., Leung, C.H.C., and Raikundalia, G. K. (2006d), 'Matrix-Agent Framework: A Virtual Platform for Multi-agents', *Journal of System Sciences and Systems Engineering*, Vol. 15, No. 4, pp. 436 – 456, Springer-Verlag Press.

Zhang, H. L., Leung, C.H.C., and Raikundalia, G. K. (2008), 'Topological analysis of AOCD-based agent networks and experimental results', *Journal of Computer and System Sciences*, Vol. 74, No. 2, pp. 255 – 278, Elsevier Publication

ACKNOWLEDGEMENTS

The experimental design of this research was supported by Victoria University. Thanks also go to Dr Frederic Adam of University College Cork for his comments on the previous version of the Matrix design.