AGENT-BASED AUTOMATED NEGOTIATION SYSTEM FOR E-MARKETPLACES

LIAO XUAN

NATIONAL UNIVERSITY OF SINGAPORE

2003

AGENT-BASED AUTOMATED NEGOTIATION SYSTEM FOR E-MARKETPLACES

LIAO XUAN
(B.Eng, Huazhong University of Sci & Tech)

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF ENGINEERING

DEPARTMENT OF ELECTRONIC & COMPUTER ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2003

Acknowledgements

I would like to express my sincere gratitude and appreciation to my supervisor Dr. Bharadwaj Veeravalli for his invaluable support, encouragement and guidance during the whole process of my master’s study.

Many thanks to all friends in Open Source Software Laboratory, who are so warm, kind and give me much help in the past two years. I am also grateful to those people whose works has been quoted in the thesis because I have obtained a lot of ideas from them. The National University of Singapore is also acknowledged for providing the graduate scholarship.

Last, but not least, millions of thanks to my parents, boyfriend, brother, sister-in-law and many other relatives and friends, for their tremendous love, support and encouragement.

Table of Contents

Summary

Commerce is all about the interaction between buyers and sellers at all stages, no matter whether the interaction is in traditional commerce activities or in modern e-commerce activities. The most common mechanism of commerce is *negotiation*. Software agent technologies could play a critical role in it regarding the potential of eventually delivering unprecedented levels of autonomy, customization and general sophistication in the way negotiation is conducted in online trading.

In this thesis, we propose two models, namely a Local Service Model (LSM) and a Network Cluster Model (NCM), for an Agent-Based Automated Negotiation System (AANS) in e-marketplace web applications that are suitable for local and distributed computing environments.

After demonstrating the two models, we discuss the common design issues for our proposals, including agent communication protocols, system openness, infrastructure services, negotiation strategies, negotiation algorithms and some other issues. We extend Flat Organization to Complex Flat Organization for the open Multi-Agent System (MAS), which makes a tradeoff between the openness and dynamism of Flat Organization and the low communication cost of Hierarchical Organization; We mix Symmetry Communication Model and Asymmetry Communication Model to a Symmetry & Asymmetry Communication Model, using connection-oriented peer-to-peer and multicast communication approaches; We propose the negotiation MAS as a Half-Dynamic Openness System, in which Management Agents get ready in advance, but Shopping Agents are unpredictable and enter or leave the MAS dymatically; We choose Madkit (Multi-Agent Development Kit), a Java multi-agent platform, to provide the infrastructure services to the AANS; We discuss the specific mechanisms to implement automated negotiation, including modularization architecture of negotiable Shopping Agents, Half-Bilateral Negotiation Model for the negotiation protocol, negotiation strategies and algorithms, the negotiation estimation algorithm and the reputation evaluation mechanism.

To demonstrate the real-life potential of AANS, we design, implement and test a Second-Hand Computer E-Marketplace (SHCM) for the proposed models. We analyze the business requirements, demonstrate system structure diagrams, use case diagrams, sequence diagrams and class diagrams in Agent Unified Modeling Language (AUML). Finally, we summarize the implementation and testing results.

Generally, unlike other existing negotiation systems that either focus on the design of negotiation algorithms and protocols for agents or focus on an integrated agent-mediated e-trading solution, our proposed models for AANS are shown to be completely adaptable in terms of modularized system design, dynamic negotiation organization and efficient communication mechanisms. Our approach seems to be more compatible with the emerging Web Services Architecture (WSA) and will make sense for adoption and acceptance of a wide range of agent-based intelligent services.

Abbreviations

AANS	Agent-Based Automated Negotiation System
AANSC	Agent-Based Automated Negotiation Service Center
AANWS	Agent-Based Automated Negotiation Web Service
ADS	Agent Delegation Sheet
AIM	Agent Interaction Model
AGR	Agent-Group-Role model
ANS	Automated Negotiation Service
AOB	Agent Office Branch
AP	Agent Platform
ASDK	Aglets Software Development Kit
CFO	Complex Flat Organization
COMAS	Co-Ordination in Multi-Agent Systems
CPS	Customer Particular Sheet
DB	Database
HBNM	Half- Bilateral Negotiation Model
HDO	Half-Dynamic Openness
HSC	Half-Symmetry Communication
IRS	Item Requirement Sheet
LSM	Local Service Model
Madkit	Multi-Agent Development Kit
MAS	Multi-Agent System
NCM	Network Cluster Model
OOD	Object Oriented Design
SHCM	Second-Hand Computer E-Marketplace

AUML	Agent Unified Modeling Language
WSA	Web Services Architecture
XML	Extensible Markup Language

Figures

Tables

Chapter 1 Introduction

1.1 Software Agent Technology in E-Commerce

As the Internet continues to spawn new markets, electronic commerce is changing many market rules. Old commercial rules are being adapted to the new conditions of the global network. Software agent technology could play a critical role regarding the potential of eventually delivering unprecedented levels of autonomy, customization and general sophistication in the way electronic commerce is conducted. At the same time a number of theoretical, technological, economic, sociological and legal issues will need to be addressed before such opportunities become a reality [1].

A large number of techniques have been used to build the relation between buyers and sellers. However, most development made up to now focuses on a passive web query type of interaction. Richer and more flexible ways of interaction are what we will witness soon with the help of agents mediating in many of the steps of commercial transactions. Mediation is the corner stone of commerce. Software agents can similarly be of extreme utility for electronic commerce due to several reasons:

- **Autonomy** Agents may work proactively, reactively and independently of human intervention. They can wait for good deals without diverting our attention.
- **Personalization** Agents can be equipped with a personal profile to reflect their users' preferences.
- **Social Ability** The communication ability of agents permits them to negotiate over prices, services and transactions.

- **Intelligence** Agents can learn and hence perform better over time. In e-commerce scenarios this may equate to making more money.

1.2 Agent-Based Negotiation

As known, commerce is all about interaction between buyers and sellers at all stages. The two basic mechanisms currently used are Auction [2] and Negotiation [3, 4]. Between these two, the most fundamental, powerful and widely used mechanism for managing inter-agent dependencies at run time is negotiation. This is so because negotiation can be used to coordinate or share limited resources between two or more autonomous self-interested entities. Agents have to engage in a negotiation process by which a joint decision is made by two or more parties. In other words, an acquaintance needs to be convinced to act in a particular way. Since the agents have no direct control over one another, they must persuade their acquaintances to act in particular ways.

Negotiation, understood in its more wide form as a process by which groups of agents communicate with one another to reach mutually acceptable agreements on some matter, is still to be witnessed in real applications. The parties first verbalize contradictory demands and then move towards an agreement by process of concession making or search for new alternatives. We can envisage that agents will be initially used to explore the set of possible agreements, referred to as Informal Contracts, but will not be authorized to sign the contract. Agents will then pass the information of Informal Contracts to the users who will then make the final decision.

Negotiation problems can be categorized in many ways. One way to classify these problems is according to the underlying incentive and information structures:

- The actual size of the limited resource to be shared is unknown.
- The set of agents is known, but their characteristics are unknown. For instance, agents are barging with deadlines which are private information, or several agents are competing for buying an object the worth of which is private information.
- Everything is known but the strategy. For example, two agents with known time preferences negotiate a contract with different strategies.

We can envisage that agent-based electronic commerce applications with fully automated negotiation will have some of the following characteristics.

First of all, interactions are very fast. For instance, in bandwidth trades, there is no time to go back to the user between trading rounds. Secondly, agent learning is effective. Either the interaction is repeated with a high communication overhead, or the interaction domain is limited. Thus, learning by the agent about the user behavior is effective. Thirdly, each trade is of relatively small value. If each transaction is of relatively small value, it is possible to monitor the process and stop the automatic trade after some time without significant loss. It is important to stress the importance of relativity here. A small value for a company is completely different from a small value for a private end consumer. Also, the process is repeated over long times. The first reason is there must be a significant value over in order to justify investments in software, hardware, and training. The second one is automatic agent learning of customer preferences is highly desirable and is a very time consuming business. Finally, the product is relatively easy to specify. A number of traditional difficult computer science problems pose major difficulties to negotiations over complex objects. Such problems are mainly related to semantics of the communications. This is also related to preference elicitation. It is simply too time consuming to tune huge numbers of difference parameters.

The challenges in the near future for agent-based negotiation are trust, protocol engineering, preference models and argumentation. The following is the concrete description regarding these four challenges.

(1) Trust

Many aspects that will require the attention of researchers for the next generation systems refer to a reliable communication channel between participants in trade, mainly including confidentiality, integrity, authentication and non-repudiation. Also, safe payment and delivery will be focus of attention.

The first generation negotiation of agent-based electronic commerce systems covering the negotiation phase will be most probably a brokered service, where the contact between buyers and sellers will be supervised, following a very strict protocol and with a total control by the institution acting as the broker. Later on, as robust trust mechanisms become available, direct contracts between buyers and sellers, with less structured transactions, will be possible.

Reputation could also be achieved via external reviewers or generated by peer rating. However, these systems are very vulnerablep Reviews must be substantiated in some way to avoid false reviews. In open markets, agents must know as much about trust issues as about actual buying and selling.

(2) Protocol Engineering

Scientists and industries must agree on conventions of communications at the strategic level. For example, agents will have to signal each other the negotiation problems they think they are facing and agree on the most appropriate negotiation protocol to follow.

(3) Preference Models

The modeling of preference is essential to give an agent the capability of acting according to a user model when negotiating. The challenges we are facing correspond not only to the way of modeling preferences, but also models which indicate the preferences of the opponent, as this may reveal important aspects of its strategy. Near future challenges include:

- Dynamics of Preferences Interests and preferences of consumers change over time. AI approach like learning, introduction algorithms, belief revision techniques, or model logic will become increasingly poignant in dealing with this challenge.

- Difference Ontology Providers and customers will probably use different ontology to characterize products and interests. Different reasoning techniques and models to match providers and customers are necessary to move away from the requirement of a common ontology.

- Fuzziness Preferences are by no means clear cut but fuzzy (uninteresting, interesting, very interesting). The use of fuzzy logic and statistics to model users' preferences is one of the techniques to explore.

- Learning In order to construct a model of the user preferences without direct query, it is necessary to develop appropriate techniques to observe the user behavior.

(4) Argumentation

Many problems can not be solved by a simple offer and counter offer negotiation protocol. In order to persuade an acquaintance, it is necessary to use arguments to support our proposals. Protocols permitting critiques to these arguments are hence necessary. Reasoning models generate arguments, rebut and undercut them, and models of preference for arguments adapted to the area of electronic commerce are also needed.

1.3 Our Objective and Approach

In this thesis, we propose two models, Local Service Model (LSM) and Network Cluster Model (NCM), for an Agent-Based Automated Negotiation System (AANS). We then present a complete design and implementation of a Second-Hand Computer E-Marketplace (SHCM) scenario for these two models by using a formal modeling approach for agent-based software project development [5]. To simplify the implementation, we developed the agent-based electronic commerce applications with fully automated negotiation between Buyer Agents and Seller Agents that are controlled by AANS.

For both of these two models, various intelligent agents run on the customer side and the marketplace side. On the customer side, we have Seller Agent and Buyer Agent. Similarly, on the marketplace side, we have, Agent Manager, Database Broker (DB Broker), Contract Manager and Data Packager, respectively. Our negotiation system is implemented on an Open Source multi-agent platform-Madkit [6]. Our primary goal is to explore and develop applicable e-marketplace systems with agent-based negotiation service which can effectively implement an m-to-m (multiple Seller Agents to multiple Buyer Agents) and multi-issue (price, brand, etc.) related automated negotiation.

Apart from the innovative models and organizations of the negotiation system we develop in SHCM, most frequently used components demanded by conventional e-marketplaces are also developed, including customer account management, messaging system, item (selling product and buying request) management and reputation evaluation mechanism.

The organization of the thesis is as follows. Chapter 2 presents the related works and issues on agent-driven e-marketplace schemes. Chapter 3 presents LSM and NCM for e-marketplaces supported by AANS, including system architectures, negotiation organizations, agent communication models and infrastructure services. Chapter 4 shows the implementation of these two models for the proposed SHCM scenario and summarizes the results. Finally, Chapter 5 concludes the thesis with some open ended issues.

Chapter 2 Related Work

Research on agent-mediated e-marketplace systems have been substantially discussed in the recent years. The work reported in some of the existing popular e-trading sites [7-11] essentially address some novel e-trading services, such as brokerage, affiliating, price-naming and auction. However, few sites carry out an automated negotiation approach. It indicates that the technological support to automated negotiation still stays in an experimental stage and is far from wide commercialization.

While there are numerous studies that can be referred to, the following is probably the closest and most relevant literature to the problem addressed in this thesis. These include the work by Tsvetovovatyy and Gini [12], Kasbah [13, 14, 15] and Jangter [16]. In these works, some novel experiments, in the sense of matching models, negotiation algorithms, and reputation mechanisms, are discussed.

Specifically, in [12] the study attempts to represent the real world, modeling banks, accounts, money, etc. Their approach in providing a complete solution is only applicable to new e-trading applications, but not to inherited systems. This somewhat reduces the adaptability and usability of their proposal. Conversely, we only keep our focus on the automated negotiation process and allow AANS to be easily integrated with existing e-marketplaces as a value-added service and make it compatible with Web Services Architecture (WSA).

Shopping Agents designed in [14] only considered a single negotiation factor, price, but our AANS can handle multiple negotiation issues. The agent-based negotiation service in [14] was closely coupled with other marketplace services such as user account management [13]. In contrast, our proposal attempts to loosely couple AANS to the existing e-marketplaces by only connecting them by Database (DB). Agent communication in [14] was processed in a symmetry peer-to-peer model [13], but in AANS the multi-casting/broadcasting model is also carried out besides the peer-to-peer model to facilitate communication efficiency.

The work reported in [16] on e-marketplace proposed a "Floor-and-Room" matching mechanism and a reputation evaluation algorithm for marketplaces. The weakness of the approach was that it also considered only a price-relevant negotiation system. In addition, it conducted an m-to-m direct negotiation in a "Room" which increased the complexity and computation for agent communication. When compared with the study in [16], AANS limits the negotiation activity within a single Negotiation Group to 1-to-m (One Buyer Agent to multiple Seller Agents), meeting the buyer-market in the real world and significantly reducing agent communication overload.

Other related works on modeling virtual e-marketplace also include [18], which is close to one of our proposed models, LSM. However, their model focuses on realizing an e-marketplace completed driven by software agents. Conversely, we deploy the Multi-Agent System (MAS) as a value-added support system which is loosely coupled with the existing systems in marketplaces. Our approach is more compatible with the emerging WSA and will make sense for adoption and acceptance of various agent-based intelligent services.

Generally, our MAS is similar to those marketplaces in the sense that it attempts to efficiently and optimally match buyers and sellers, conduct automated negotiation by using software agents, and pursue the optimal contract for users. However, our system focuses on the applicable models and organizations for the negotiation system which have commercialization potential, instead of sophisticated negotiation algorithms and negotiation protocols [19, 20].

Chapter 3 Proposed Solution

In this section, we shall propose two different models for AANS that are directly applicable for e-marketplace applications. The first model, referred to as Local Service Model (LSM) is applicable on a customized system while the second model, referred to as Network Cluster Model (NCM) is applicable in a networked environment.

3.1 Local Service Model (LSM)

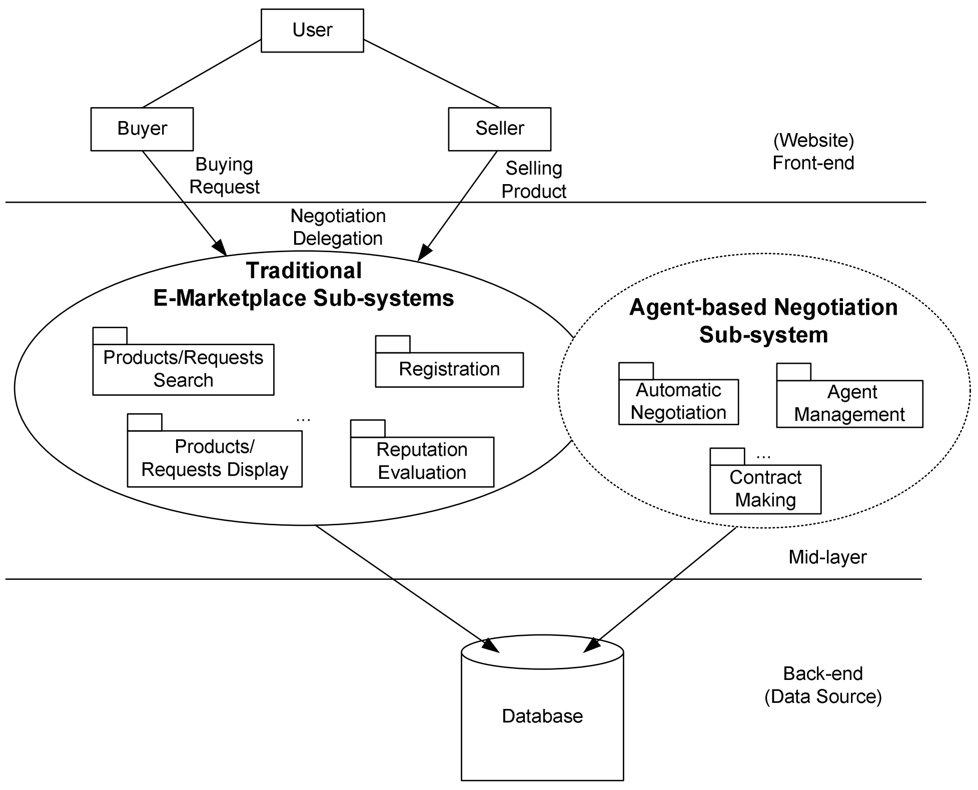


Fig.1 Architecture of Local Service Model

As mentioned above, this model is suitable for standalone systems in which AANS can be independently plugged into e-marketplaces as an optional local service component. In other words, the negotiation engine can be regarded as part of the business logic layer to comprise an integral e-marketplace. Physically, AANS is a sub-system which is running locally at Application Server with Web Server and DB together, and the negotiation sub-system is exclusively used by the e-marketplace where it is situated. Fig.1 shows the architecture of our proposed LSM.

From the architecture, we observe the front-end of this system, a normal e-marketplace interface. With the business logic support of mid-layer, users can register profiles and create personal accounts, search interested items (selling products or buying requests) and submit their own items. The system will administrate user accounts, items, as well as users' reputation.

What distinguishes this e-marketplace from the existing conventional e-marketplaces is that we have now an optional value-added service, Automated Negotiation Service (ANS), which is supported by AANS. This optional service can create self-interested Shopping Agents for users' sale or purchase demands. Shopping Agents are able to search, match and negotiate with potential trading counterparts and AANS organizes all matched Shopping Agent peers together to pursue optimal Informal Contracts.

It may be noted that our AANS is intentionally designed to be a loosely coupled entity with traditional e-marketplace systems, and therefore it is easy to be integrated with any similar e-marketplaces at a low cost. What links the traditional e-marketplace sub-systems to AANS is only a common-shared back-end DB in our design. The e-marketplace saves all data of users, items, and agents in the DB. On the other hand, AANS scans the DB periodically to fetch the relevant data on users, items, and agents for negotiation execution. After reaching a successful Informal Contract, AANS saves the contract details into the DB and informs both users to evaluate the Informal Contract. Though the Informal Contract completely meets users' requirements, users still have the right either to accept or to refuse it.

One important feature of LSM is that AANE is plugged into the traditional e-marketplace in a relatively independent and flexible pattern. For tremendous inherited e-marketplaces on the Internet, AANE can transparently deliver an advanced service to help online shoppers efficiently and profitably sell or purchase goods. For this architecture, because MAS is physically located in the same place with the Web Server and DB, the system load for communication and collaboration will be very small and consequently, the whole e-marketplace system is robust as an integrated entity.

While, as it may be noted, a slight disadvantage of this architecture is that AANE exclusively works for the e-marketplace where it situates and can only process the service requests from this e-marketplace. Therefore, if the implementation demands a customized and localized service infrastructure, then this model is directly applicable. However, for a network-based environment we have the following alternate version.

3.2 Network Cluster Model (NCM)

In this model, the negotiation system is totally established as a separate centralized Agent-Based Automated Negotiation Service Center (AANSC). AANSC specially provides ANS to all e-marketplaces in an e-marketplace network cluster, which is an open community of a cluster of e-marketplaces. By and large, e-marketplaces in this cluster have some common product categories and they do not exchange data mutually but only exchange data with AANSC using standard XML documents defined in advance. If any e-marketplace in the cluster needs ANS, it should add certain web interfaces for its users to input negotiation constraints and create certain tables in the DB to store agent-related information. The architecture of the proposed NCM is shown in Fig.2.

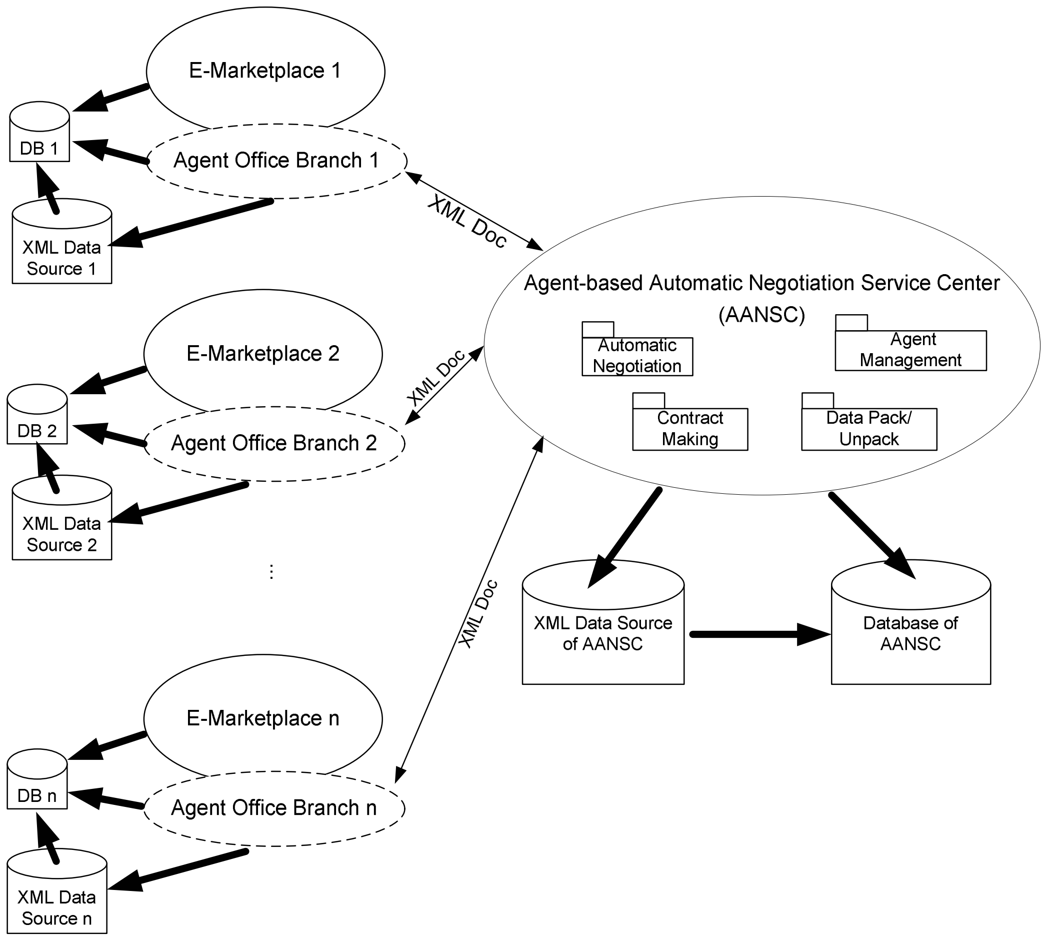


Fig.2 Architecture of Network Cluster Model

In this e-marketplace network cluster, each distributed e-marketplace comprises two parts, the traditional e-marketplace sub-system and the Agent Office Branch (AOB) sub-system. The former one is a normal e-marketplace providing platform to trade goods between sellers and buyers. Usual functions of these traditional e-marketplaces include user account management, reputation management, item search engine, item display, customer notification, etc. Besides these, the e-marketplaces of this cluster also can supply other value-added services, such as ANS for our case. To support ANS, AOB sub-system is additionally plugged in each e-marketplace to process agent-related information. The primary task of AOB before negotiation is to get agent-related data from the e-marketplace DB, pack them into standard XML documents and transmit XML documents to AANSC. AOB is also responsible for receiving, unpacking and parsing XML documents of Informal Contracts achieved in AANSC, and storing data of Informal Contracts into the local e-marketplace DB.

The AANS of NCM resembles the AANS of LSM, except for the additional XML package processing mechanism in NCM, which include packing, unpacking and parsing XML documents. Negotiation-related data is stored in the centralized AANSC DB, which provides all necessary data for executing negotiation in AANSC. Once a successful Informal Contract is achieved in AANSC, the contract details will be stored in the AANSC DB, packed into a standard XML package, and then sent back to the original e-marketplaces.

In the following, we summarize some significant advantages of NCM.

Large Customer Base*:* NCM calls on all potential distributed e-marketplaces together to chase optimal contracts for their users. The huge data sources linked with AANSC create tremendous opportunities for any user of any e-marketplace to be in touch with all potential peers in and out of the e-marketplace the user registers. This means that, once a user publishes an item in any e-marketplace within the cluster, all matched items within the cluster consequently become available to this user.

Web Service Compatibility For the emerging WSA, AANSC can be easily packed as a Web service and can be made available in World Wide Web. Therefore, any web applications need to carry out the particular function of automated negotiation can search a global directory and recognize this Agent-Based Automated Negotiation Web Service (AANWS) and use it dynamically without any human intervention.

Data Format Transparency The data formats of every distributed e-marketplace can be totally different, but by transforming these data into a standardized XML format, AANSC can transparently make use of the unified data which may originally be heterogeneous. In this sense, the isolated e-marketplace islands can be connected into a broad virtual e-marketplace.

Cost-efficiency Because no Seller Agents or Buyer Agents will be running at e-marketplaces in this model, as no negotiation activities will be carried out locally, the overall load that each e-marketplace is expected to handle in providing ANS is largely minimized. Thus, the service infrastructure becomes a cost-effective solution and hence creates a huge market potential.

On the other hand, the following is some inherent disadvantages of NCM which are also obvious when compared with LSM.

Firstly, frequent XML documents transmission between distributed e-marketplaces and the centralized AANSC may increase the communication consumption of individual servers and enhance posses messaging difficulties among distributed agents.

Also, Management Agents in e-marketplaces and AANSC need to collaborate and communicate remotely within the cluster, which may demand an increased requirement for agents' interoperability and synchronization.

Finally, as the performance of a distributed cluster is affected by the network traffic conditions, network loading conditions and resource availabilities may influence the performance of AANS.

3.3 Design Issues in the Proposed Models

3.3.1 Negotiation Organization

By and large, MAS are organized in the following ways: hierarchy, flat, subsumption, and modular organization.

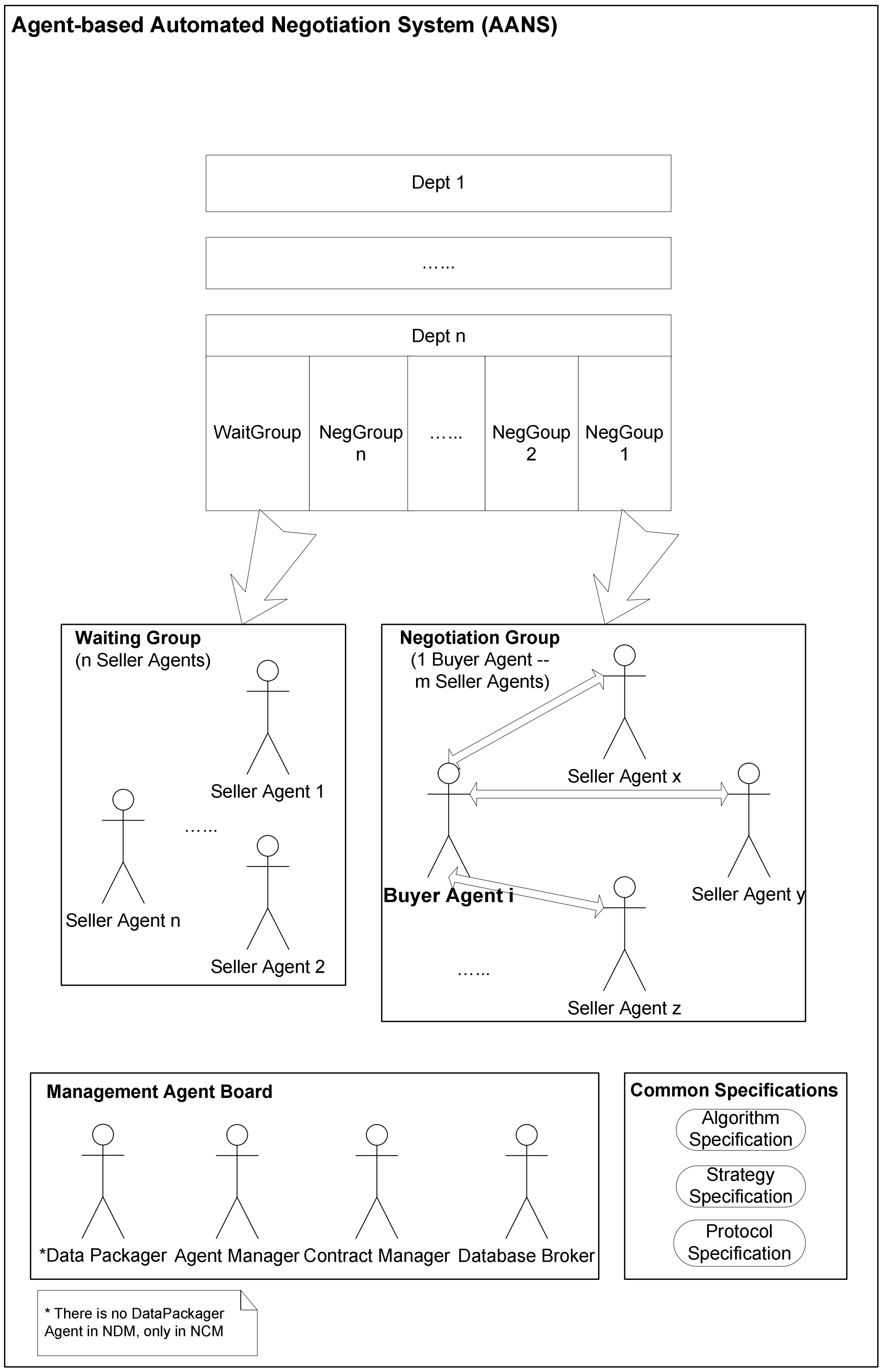


Fig.3 Organization of Agent-Based Automated Negotiation System

In Hierarchy Organization, agents can only communicate subject to the hierarchical structure. The advantage of this restriction is that there is no need for a mechanism for agent location and the communication is significantly reduced in the system. But hierarchy implies that the lower levels depend on the higher levels, and higher levels are in control of the lower levels. Also, the strict structure of hierarchy organization prohibits dynamic re-organization to best fit the specific tasks.

In Flat Organization [21], each agent can directly contact another agent. There is no fixed structure applied to the system, however agents may dynamically form structures to perform specific tasks. In addition, no control of one agent by another is assumed. Such an organization can be a closed one, members of which are fixed in advance, or an open one, members of which is dynamically changed during the running time. The advantage of Flat Organization is that it fully supports autonomy and self-interest of agents as well as distribution and openness of MAS. These openness and dynamism, however, result in communication overheads, the need for agent location mechanisms as well as mechanisms for dynamic re-organization. The amount of reasoning and computation an agent performs with regards to other agents increase significantly in a Flat Organization.

In Subsumption Organization, some agents are components of other agents. These agents are subsumed by the container agents, which in turn may be components of larger container agents. This organization style is the extreme case of the Hierarchy Organization, requiring that the subsumed agents completely surrender to the control of the container agent. From a software architectural viewpoint, such architecture resembles an inclusion of objects within a larger object, except for the important difference in the control methods. While objects are usually controlled and activated by (possibly remote) procedure call or event invocation, agents are activated by high-level communication. The strict control relationship in the Subsumption Organization results in efficient tasks execution and low communication overhead, however restricts the system to address a well defined set of tasks, with virtually no flexibility and adaptability. It is also not simple to modify the MAS of Flat Organization in the face of long-term changes in tasks and of the system environment.

In Modular Organization, the MAS is comprised from several models, where each is virtually a stand-alone MAS. Typically, the partition of the system into modules is done along dimensions such as geographical vicinity or a need for intense interaction among agents and services within the same module. Often, the systems are comprised of such parts as a result of its development process, during which new modules were gradually added to an already existing system. Modularity increases efficiency of task execution and reduces communication overhead. Similar to Flat Organization, high flexibility is usually enabled within each module in Modular Organization. On the other hand, re-organization across modules is rather complex, thus flexibility is limited. In addition, the given modularity implies constrains on inter-module communication.

In our MAS, we design a Complex Flat Organization (CFO), as shown in Fig.3, which basically extends the Jangter Model, Flat Organization and Hierarchy Organization, by clearly striking a trade-off among them.

CFO is designed as open MAS, which is supported by an agent location mechanism provided by the infrastructure. Agents running in it are divided into two categories, Management Agents and Shopping Agents. Shopping Agents are controlled by Management Agents that include DB Broker, Agent Manager, Contract Manager, and Data Packager, which only exists in NCM, but not in LSM.

Agent Manager is responsible for building various negotiation departments, negotiation groups and waiting groups in departments based on product categories and attributes. Also, it specify the right department, negotiation groups or waiting groups for new Shopping Agents, as shown in Fig.4. It also search for any matched Shopping Agent peers and activate them, as shown in Fig.4. Finally, it terminates Shopping Agents after they achieve successful Informal Contracts, when they are deleted by users or when the deadlines specified by users expire.

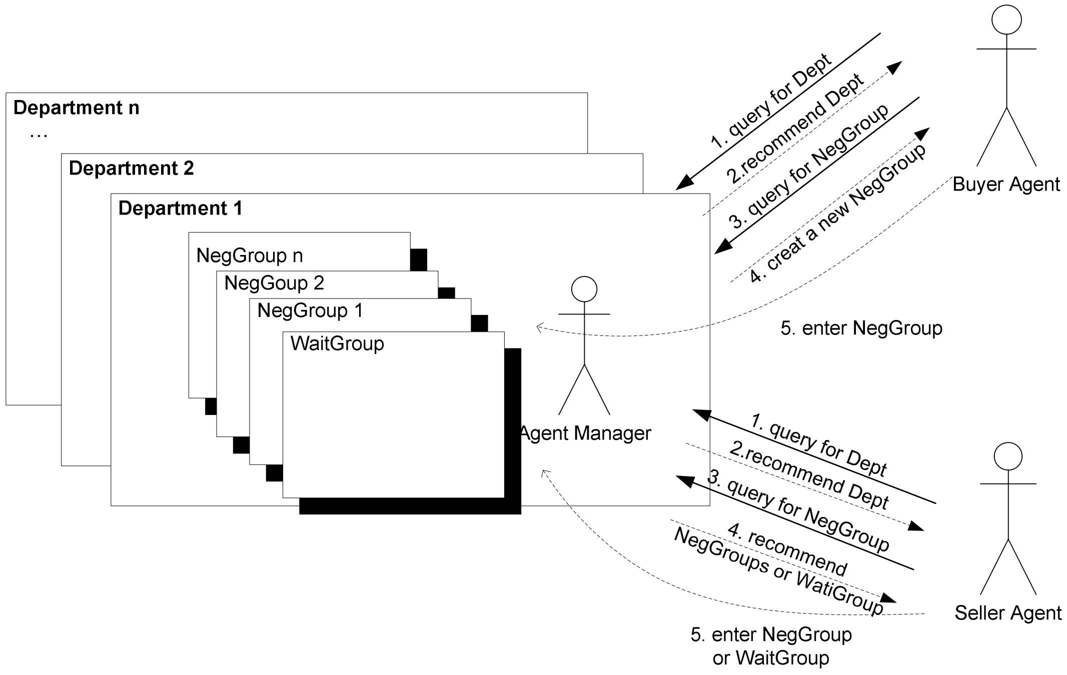


Fig.4 Conceptual Design for Matching Shopping Agent

The primary tasks of DB Broker include, getting new Shopping Agents by periodically scanning the DB and notifying new Shopping Agents to Agent Manage. Also, it serves to get feedback from users regarding their decisions about Informal Contracts by periodically scanning the DB and to inform Contract Manager on the feedback. Further, it keeps track on any changes of item attributes and Shopping Agents delegation constraints and informs Agent Manager about these changes.

Similarly, the primary tasks of Contract Manager are to save Informal Contracts and Formal Contracts into the DB and to inform users about the Informal Contract details, users' feedback towards Informal Contracts and Formal Contract details.

In the case of NCM, there is an additional Management Agent, referred to as Data Packager, in the negotiation system. Its responsibilities are to receive the standard XML documents describing Shopping Agents from distributed e-marketplaces in the cluster, unpack the XML documents and extract the relevant data into the DB. Further, it packs Informal Contract information to standard XML documents and transmits them back to the relevant e-marketplaces.

Although it is an open organization and admits new Shopping Agents at any time after the negotiation system is launched, the organization is well structured and any specific agents are restricted to work in certain space. According to product categories, AANS is divided into several departments. Each department is made of a Waiting Group and multiple Negotiation Groups. For any new Buyer Agent, a new Negotiation Group will be created for it. On the other hand, for a new Seller Agent, the system will search existing Buyer Agents that can match with it and send it to the Negotiation Groups that the matched Buyer Agents are located, or send it to a Waiting Group if no matching Buyer Agents are found.

Unlike Flat Organization, agents in CFO cannot freely communicate with others, but only can communicate with part of them. To a Buyer Agent, the Negotiation Group in which it is located is its working space and it can directly contact any Seller Agents within this Negotiation Group. However, to a Seller Agent, the whole department is its working space, and it can directly contact with any matched Buyer Agents within the department.

Apart from the advantages of Flat Organization, CFO provides some additional benefits to the system like, providing an elegant solution to the ontology problem.

When creating departments, the ontology describing the departments is defined according to product categories. It reduces the system overhead by searching the matched department, groups and counterparts for Shopping Agents in advance.

Also, it minimizes the complexity of negotiation processing by replacing the m-to-m (multiple Seller Agents to multiple Buyer Agents) negotiation model in Flat Organization by the 1-to-m (one Buyer Agent to multiple Seller Agents) negotiation model in CFO. In other words, only 1-to-m negotiation is executed within each Negotiation Group. On the other hand, every Seller Agent can participate in the negotiation activities of multiple matched Negotiation Groups in the same department and negotiate with multiple Buyer Agents in these Negotiation Groups simultaneously. Therefore, CFO logically realizes an m-to-m negotiation model, although it is a complex 1-to-m model within a Negotiation Group.

Further, the model meets the reality of buyer’s market in the real business world by conducting Buyer Agent centralized 1-to-m negotiation in every Negotiation Group. Buyer Agent is in control of the communication in a Negotiation Group and therefore negotiation implementation is relatively simple and efficient. But the disadvantage is that Seller Agents can only respond upon the requests of Buyer Agents.

Finally, it simplifies the extension from a centralized computing system to a distributed computing system by assigning different departments, even different Groups in the same departments, to a cluster of servers for distributing tremendous business requests. Thus, this model subsumes inherent scalability.

3.3.2 Agent Communication Model

Most MASs use specially designed communication protocols that best fit the agent architectures, organizations, and scenarios of these systems. The advantage of using special protocols for MASs is high efficiency. Such systems, however, cannot converse with agents which do not support that specialized communication infrastructures. To void such limitations, several multi-agent systems support generic communication protocols such as KQML and FIPA-ACL and provide generic communication modules. For all distributed computing systems, there are three main attributes that are relevant to MAS communication.

The first one is symmetry. The pure asymmetric communication is well supported by operating systems and programming languages and such implementations are simple and efficient. But the drawback is caused by asymmetry between agents: one is in control of communication, while the other can only respond upon a request. Certainly, it is not an optimal choice in an open Flat Organization. On the other hand, the pure symmetric communication is more appropriate for Flat Organization, however, increases protocol complexity and may slow down communication.

Considering the strengths and weaknesses of the two communication models and the mixture characters of CFO, we judiciously combine pure asymmetric communication and pure symmetric communication into a Half-Symmetry Communication (HSC) Model. All Management Agents in AANS can interact with others by symmetric model, which implies there are fair cooperation relations among Management Agents. On the other hand, within every Negotiation Group, Buyer Agents are in control of the interaction, and therefore asymmetric communication model is implemented among Shopping Agents. The objective of asymmetric communication in Negotiation Groups is to simulate buyer’s market model and simplify negotiation process. The communication between Management Agents and Shopping Agents is also executed symmetrically, which ensures the assistant help and management commands to be efficiently delivered from Manage Agents to Shopping Agents. Simultaneously, the negotiation feedback can be timely transmitted from Shopping Agents to Management Agents.

The second attribute is message recipients. Messages in a network may be sent to a single address, to multiple addresses (multicast) or to all (broadcast). Open MAS usually implements peer-to-peer or multicast communication and closed MAS commonly uses broadcast communication. In our negotiation system, most communication is conducted by peer-to-peer, but in some special cases, Buyer Agents also send messages to all Seller Agents within its Negotiation Room. From the whole negotiation system’s point of view, it is a multicast communication. But if we observe each Negotiation Group as a sub-ecosystem, it can be considered as a broadcast communication. Therefore, our negotiation system uses both peer-to-peer and multicast/broadcast communication models.

The third attribute is connection types. Connection-oriented communication and connectionless communication are both implemented in multi-agent systems. Connectionless communication is sufficient when task execution is loosely coordinated and where concurrency is not important. On the other hand, connection-oriented communication is preferred when dependent tasks are performed concurrently by multiple agents, and coordination is necessary during execution. Because the task of individual agent is dependently executed and highly requires the coordination among the agents, we choose connection-oriented communication for the negotiation system.

3.3.3 System Openness

The openness of MAS refers to an ability of introducing additional agents into the system in excess to the agents that initially existed. We categorize MAS openness into the following cases.

In a dynamic open system, agents can leave and enter the system dynamically during the run time, without explicit global notification to other agents. The advantage of such openness in the system is able to dynamically adjust itself to changes in the environment and tasks. But the disadvantage of this extreme openness is that more system services and computing consumption are needed and implementation of a robust agent location mechanism becomes mandatory.

On the other hand, in a static open system, agents can be added into the system without re-starting it, but with all of the agents are notified on such an addition, or they all hold in advance a list of prospective additional agents. This kind of openness decreases the need for an agent location mechanism and reduces the execution and computation complexities. However, the flexibility of the system and its ability to adjust itself to dynamic changes are restricted.

In an off-line open system, the strictest one in the category, new agents can only be added off-line by halting the system, updating connection information, and re-booting the agent system. Excess infrastructure services and additional computation is not needed in this case. Thus, among these three, the off-line open system is least flexible.

For our AANS, we consider a Half-Dynamic Openness (HDO) model, which is a trade-off between the dynamic and static open models described above. In our MAS, Management Agents are well defined in advance and therefore every agent in the system holds a list of the prospective Management Agents. On the other hand, Shopping Agents are unpredictable and may enter or leave at any time. To resolve this problem, Agent Manager, one of the Management Agents, is created as a coordinator to manage Shopping Agents and selectively notify Shopping Agent's status to the concerned Management Agents in different negotiation stages, rather than notifying Management Agents globally. A location mechanism is provided by the agent platform and makes the selective notification efficient and accurate.

3.3.4 Infrastructure Services

The infrastructure services are fundamentally provided by a multi-agent platform, Madkit (Multi-Agent Development Kit) [22], which is inseparable from AANS.

MadKit is a Java multi-agent platform built upon an organizational model. It provides general agent facilities, including lifecycle management, naming services, location mechanism, message passing, distribution, synchronization, etc, and allows various customizations and high heterogeneity in agent architectures and communication languages. This versatile agent platform has the following main features:

- Cross-platform
- No pre-requisite on agent model
- Can support many simultaneous communication models
- Transparent distributed mode
- Can host multiple, heterogeneous agent applications
- Componential and flexible graphical agent interfaces

The Madkit platform architecture is rooted in Agent-Group-Role (AGR) model which is developed in the context of the AALAADIN project [22]. Madkit both implements and uses this model for its own management. Fig.5 presents a diagram of this AGR model.

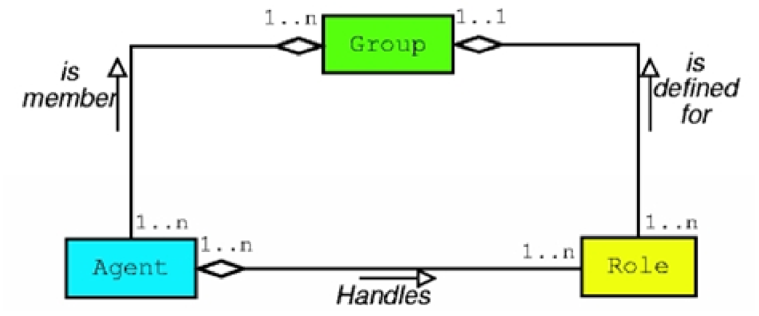


Fig.5 Agent-Group-Role (AGR) Model

Agent is only specified as an active communicating entity which plays roles within groups. This agent definition is intentionally general to allow agent designers to adopt the most accurate definition of agent-hood relative to their application. The agent designer is responsible for choosing the most appropriate agent model as internal architecture.

Group is defined as an atomic set of agent aggregation. Each agent is part of one or more groups. In its most basic form, the group is only a way to tag a set of agents. In a more developed form, in conjunction with the role definition, it may represent any usual MAS. An agent can be a member of multiple groups at the same time. A major point of AALAADIN groups is that they can freely overlap. A group can be founded by any agent.

Role is an abstract representation of an agent function, service or identification within a group. Each agent can handle multiple roles, and each role handled by an agent is local to a group. Handling a role in a group must be requested by the candidate agent and is not necessarily awarded. Abstract communication schemes are thus defined from roles.

The ARG model is not a static description of an agent organization. It also allows defining rules to specify the part of the dynamics of the agent organization. In addition to the three core concepts, the platform adds three design principles, micro-kernel architecture, agentification of services and component model for graphical interface.

MadKit itself is a set of packages of Java classes that implements the agent kernel, the various libraries of messages, probes and agents. It also includes a graphical development environment and many system and demonstration agents.

The basic philosophy of the MadKit architecture is to use wherever possible the platform for its own management: any service besides those assured by the micro-kernel are handled by agents. Thus the platform is not an agent platform according to the classical sense. The reduced size of the agent kernel, combined with the principle of modular services managed by agents enable a range of multiple, scalable platforms with simultaneous specialized agent models libraries. Fig.6 represents the architecture of Madkit platform.

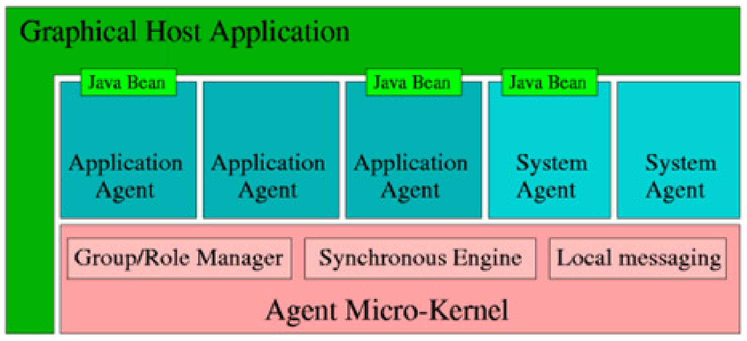


Fig.6 Madkit Architecture

3.3.5 Negotiation Algorithms & Strategies

A key problem of all the previous agent-based shopping assistant systems is that they are too focused on one aspect of the transaction, namely price. But it is very often the case that neither buyers nor sellers have price as their only concern. Especially buyers often interest in non-aspects of the purchase (such as warranty, delivery time, and reputation of the seller). Therefore, we believe multi-issues automated negotiation will become the dominant model of operation for shopping assistant agents

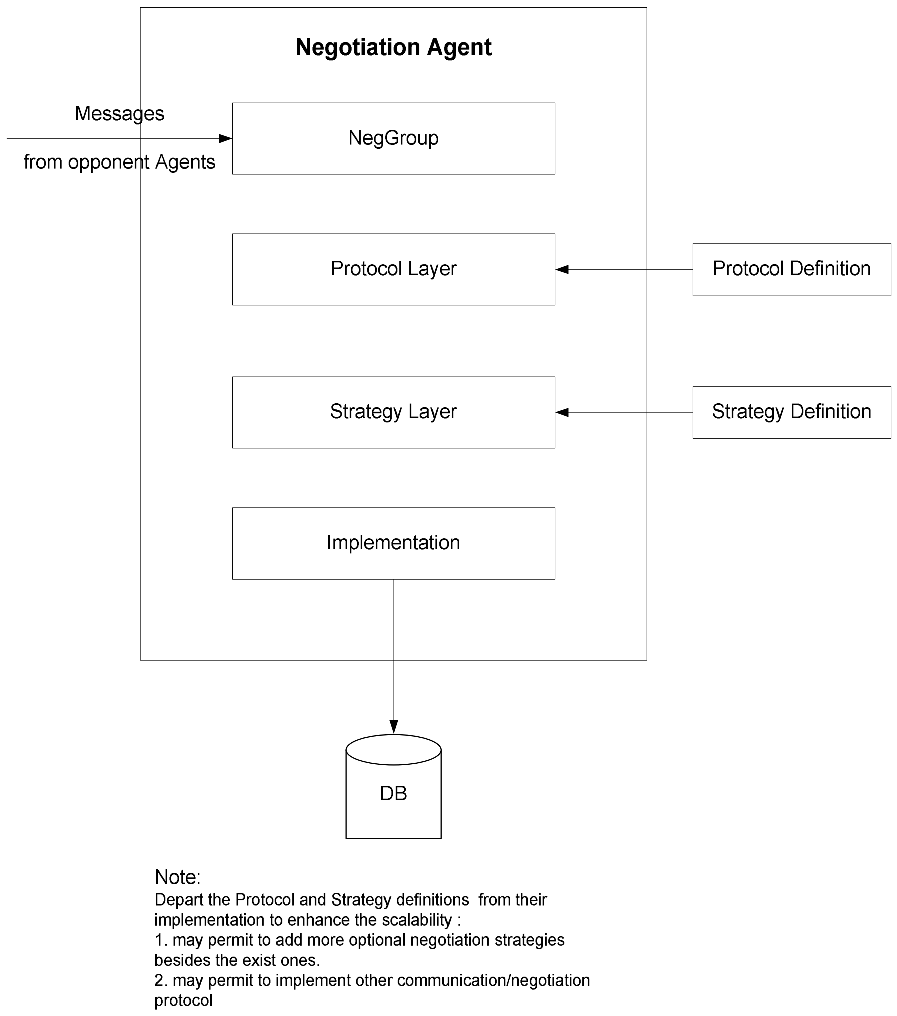

Negotiation Agent
Messages
from opponent Agents
NegGroup
Protocol Layer
Protocol Definition
Strategy Layer
Strategy Definition
Implementation
DB
Note:
Depart the Protocol and Strategy definitions from their implementation to enhance the scalability :
1. may permit to add more optional negotiation strategies besides the exist ones.
2. may permit to implement other communication/negotiation protocol

Fig.7 Negotiation Agent Architecture

Negotiation strategies and their properties depend heavily on the specific characteristics of the scenarios under consideration. They are the specifications of the sequence of actions (usually offers and responses) the agent plans to make during the negotiation. There will usually be many strategies that are compatible with a particular protocol, each of which strategies may produce a very different outcome. The choice of strategy to use is thus a function not just of the specifics of the negotiation scenario, but also the protocol in use.

To make negotiation protocols and strategies scalable for update or extension, we modularize the architecture of negotiable Shopping Agents, separating the negotiation implementation from negotiation protocol definition and strategy definition. Fig.7. represents it.

The most frequently used model of negotiation is Bilateral Negotiation Model [23]. The bilateral model, a seller agent and a buyer agent, having contradictory demands, must exchange proposals in order to reach a deal. Each agent has a public initial proposal, as well as a private border proposal, which is a maximum limit for Buyer Agents and minimum limit for Seller Agents that must be respected in reaching a deal, indicating in Fig.8. Each agent's border proposal values are normally kept hidden from its opponents.

But in our agent-based negotiation system, we reinvent the usual Bilateral Negotiation Model and defined a Half- Bilateral Negotiation Model (HBNM) as shown in Fig. 9. In HBNM, only Seller Agents are required to offer initial (desired selling price) and consequent proposals to all matched Buyer Agents till reaching the border proposals (minimum selling price). For Buyer Agents, though they also bear the initial-point (desired purchase price) and border-point (maximum purchase price) in mind, but do not exchange them publicly as proposals. The initial and border points of Buyer Agents are used to evaluate the proposals from Seller Agents. In other words, Buyer Agents are in control of the negotiation, waiting for all matched Seller Agents' proposals, evaluating and comparing received proposals and accepting the best one or continuously asking for better ones. While, though Seller Agents are in a passive situation, we still reserve Seller Agents the right to offer proposals according to their interests and strategies. When a peer Buyer Agent asks for better proposals, the Seller Agent definitely can refuse it and wait for other Buyer Agents' responses. The extension of HBNM is to meet the reality of buyer's market in the real business world by conducting the Buyer Agent centralized 1-to-m negotiation within Negotiation Groups. Also, because Buyer Agent is in control of the conversation in a Negotiation Group, the negotiation protocol is relatively simple and efficient, significantly reducing the system complexity and communication overhead. But the disadvantage, compared with Bilateral Negotiation Model, is that Seller Agents can only respond upon the requests of Buyer Agents.

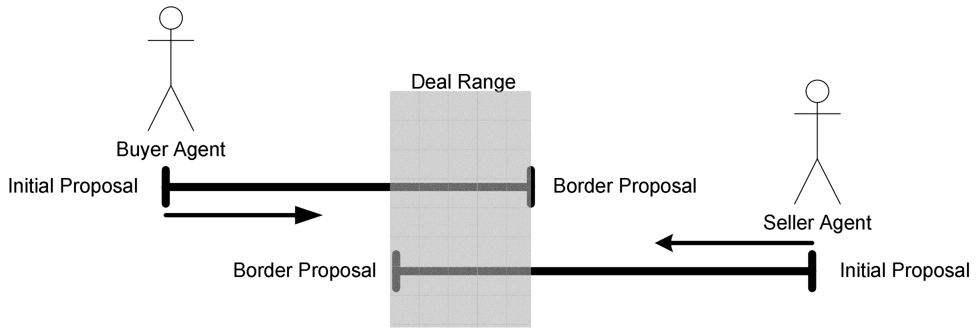


Fig.8 Bilateral Negotiation Model

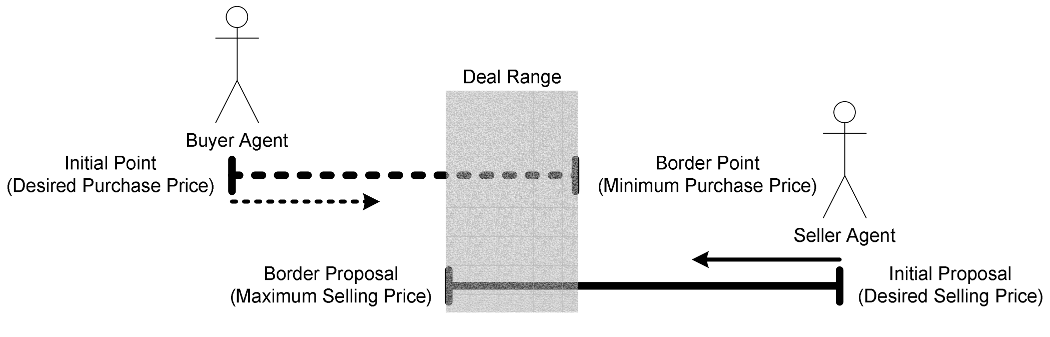


Fig.9 Half- Bilateral Negotiation Model

3.3.5.1 Proposal Evaluation

In our SHCM negotiation system, Buyer Agents take four factors of sellers' proposals into consideration, including price, brand, used time and seller's reputation. The evaluation value of a received proposal i (P_i) can be represented as:

$$Ep(Pi) = (v_{pi}, v_{bi}, v_{ti}, v_{ri}) \qquad v_{pi}, v_{bi}, v_{ti}, v_{ri} \in [0,1] \tag{1}$$

Where v_{pi} , v_{bi}, v_{ti}, v_{ri} are respectively the evaluation values of the price factor P_i, the brand factor V_i, the using time factor T_i and the reputation factor R_i.

The four factors considered in the negotiation processing are defined below.

Price Factor

$$v_{pi} = \left(\frac{p_{\max}^{B} - p_{p}^{S}}{p_{\max}^{B}}\right) \tag{2}$$

Where $p_{\max}^{B}$ is the maximum price that the Buyer Agent can accept and p_{p}^{S} is the present price that the Seller Agent is offering.

Brand Factor

$$v_{bi} = \frac{n+1-m}{n} \tag{3}$$

Where n is the number of the acceptable brands of the Buyer Agent and m is the ranking order of the selling product's brand according to the Buyer Agent's favor (*1* indicates the most favorable brand and *n* indicates the least favorable brand.). The precondition of this formula is that the selling product's brand must fall into the Buyer Agent's acceptable brand list.

Using Time Factor

$$v_{ti} = \frac{t_{\max}^{B} - t^{S}}{t_{\max}^{B}} \text{ iff } t^{S} \le \mathrm{t}_{\max}^{B} \tag{4}$$

Where $t_{\max}^{B}$ is the maximum using time of the selling product that the Buyer Agent can accept and t^{S} is the using time of the selling product.

To get the evaluation value $E_p(P_i)$ of the proposal P_i, besides the values of the four negotiation factors, we also need to know the weight values of these four factors ranked by the user. We define the weight value of the price factor as w_p, the weight value of the branch factor as w_b, the weight value of the using time factor as w_t and the weight value of the reputation factor as w_r. $w_p, w_b, w_t, w_r \in [0,1]$, and $w_p + w_b + w_t + w_r = 1$.

Assume the Buyer Agent considers i $(1 \le i \le 4)$ factors and gives them priorities p $(1 \le p \le 4)$ respectively. 1 indicates the highest priority, and 4 is the lowest one. Then, w_p can be determined simply like:

$$\text{If } i = 1,\ w_1 = 1\ ; \tag{5}$$

$$\text{If } i = 2,\ \left\{ \begin{matrix} w_1 = \frac{2}{3} \\ w_2 = \frac{1}{3} \end{matrix} \right\}; \tag{6}$$

$$\text{If } i = 3,\ \left\{ \begin{matrix} w_1 = \frac{4}{7} \\ w_2 = \frac{2}{7} \\ w_3 = \frac{1}{7} \end{matrix} \right\}; \tag{7}$$

$$\text{If } i = 4, \begin{Bmatrix} w_1 = \dfrac{8}{15} \\ w_2 = \dfrac{4}{15} \\ w_3 = \dfrac{2}{15} \\ w_4 = \dfrac{1}{15} \end{Bmatrix} \quad (8)$$

Therefore, $Ep(Pi)$, the evaluation value of P_i, can be represented by:

$$Ep(Pi) = \frac{(v_{pi} \times w_p + v_{bi} \times w_b + v_{ti} \times w_t + v_{ri} \times w_r)}{4} \quad \text{Where } w_p + w_b + w_t + w_r = 1 \quad (9)$$

After a Buyer Agent receives a proposal, it will calculate $Ep(Pi)$ of the received proposal as evaluation benchmark. If there are more than one matched Seller Agent in the Buyer Agent's Negotiation Group, the Buyer Agent will receive more than one proposal in a round of negotiation. Then the Buyer Agent needs to evaluate every proposal and choose the one with the highest $Ep(Pi)$.

3.3.5.2 Behavior Rules and Strategies

Now we turn to the actual implemented behavior of Shopping Agents. It should be obvious that the actual behavior of Shopping Agents needs to correspond closely to the users' perception of how they will behave; if not, then the users are sure to be disappointed. This is a general challenging facing agent research, namely to make sure that agents lives up to the user's expectation. Given this, then we explain the actual implemented behavior in our system.

Every Buyer Agent in the Negotiation Group calls for proposal as the Negotiation Group is initialized. Then all the matched Seller Agents will reside in this group and send their proposals to the Buyer Agent by means of peer-to-peer. The proposals include the values of the four negotiation factors discussed in the former section. After evaluating all the proposals and comparing the evaluation results, the Buyer Agent sorts out the optimal one in this round. If the optimal proposal's evaluation value is acceptable according the Buyer Agent's negotiation strategy which is specified by the user, the Buyer Agent will accept this offer and make an Informal Contract with the Seller Agent who gives this offer. Otherwise, the Buyer Agent will broadcast the offer price of this optimal proposal within the Negotiation Group and call for better proposals.

Every Seller Agent sends the same proposal to all the matched Buyer Agents which reside within the same Department. If all Buyer Agents matched with the Seller Agent have called for new proposals and the theory offer price $p_t^S(t)$ of the Seller Agent providing to all matched Buyer Agents at time *t*, is equal to or less than all prices of the optimal proposals received by these Buyer Agents, then the Seller Agent will offer a new proposal to all matched Buyer Agents. The Seller Agent's present offer price in the new proposal will be set equal to the current theory offer price and is obviously lower than its previous offer price, but will never be set lower than the minimum acceptable price the user has specified. Conversely, if $p_t^S(t)$ is higher than any offer prices of the optimal proposals received by the Buyer Agents, the Seller Agent will refuse to provide new offers and insist on the current one.

In this thesis, we propose three strategies for Seller Agents and Buyer Agents respectively which allow users to specify the negotiation 'behavior' and the judgment making 'behavior' of their agents. Further, because agents' negotiation implementation is separated from the negotiation protocol definition and the strategy definition, we can upgrade the existing strategies or create new strategies without changing other parts of the system.

Next, we describe the three strategies for Seller Agents and Buyer Agents.

Seller Agent

For a Seller Agent, the offer price can be adjusted over time according to its theory offer price $p_t^S(t)$ at time *t* and other real-time factors. The theory offer price $p_t^S(t)$ of our proposed strategies is extended from Kasbah [14] that follows some functions of time and map to the following curves.

Assume:

$p_{\min}^S$: The minimum selling price that the Seller Agent can offer, i.e. the worst price

$p_{\max}^S$: The maximum selling price that the Seller Agent can offer, i.e. the initial price

t_{r1}^S : The time the Seller Agent receives the first proposal request from Buyer Agents

d^S : The deadline of the Seller Agent

t : The current time

- $p_t^S(t)$ of the Urgent Strategy

$$p_t^S(t) = p_{\max}^S - (p_{\max}^S - p_{\min}^S) \times (\frac{t - t_{r1}^S}{d^S - t_{r1}^S}) \qquad (10)$$

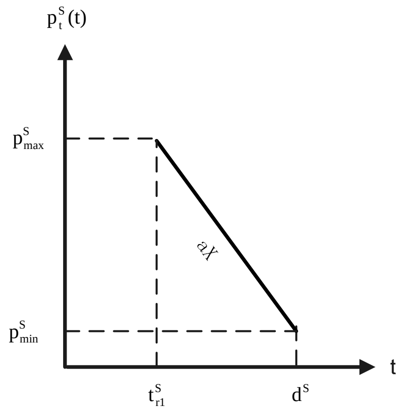


Fig.10 Theory Offer Price for Urgent Strategy

- $p_t^S(t)$ of the Normal Strategy

$$p_t^S(t) = p_{\max}^S - (p_{\max}^S - p_{\min}^S) \times (\frac{t - t_{r1}^S}{d^S - t_{r1}^S})^2 \tag{11}$$

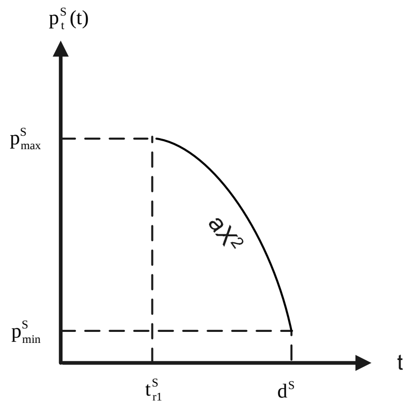


Fig.11 Theory Offer Price for Normal Strategy

- $p_t^S(t)$ of Patient Strategy

$$p_t^S(t) = p_{\max}^S - (p_{\max}^S - p_{\min}^S) \times (\frac{t - t_{r1}^S}{d^S - t_{r1}^S})^3 \tag{12}$$

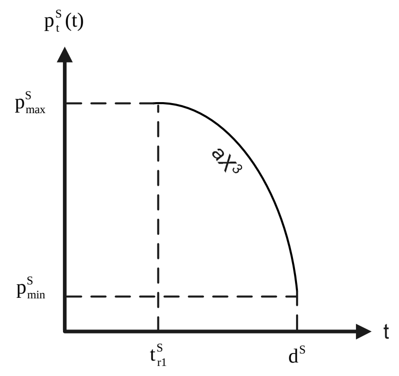


Fig.12 Theory Offer Price for Patient Strategy

Buyer Agent

For a Buyer Agent, if it receives an eligible proposal from a Seller Agent, an Informal Contract will be reached between these two agents; If the Buyer Agent receives multiple eligible proposals from multiple Seller Agents in the same round of negotiation, the Buyer Agent will choose the optimal one by comparing these proposals' evaluation values.

Assume:

f_t : The time factor

d^B : The deadline of the Buyer Agent

t_{p1}^B : The time the Buyer Agent receives the first proposal from Seller Agents

t_d^B : The time the buyer delegates the negotiation task to the Buyer Agent

A^B : The Buyer Agent's attitude towards the optimal proposal received in the current round of negotiation

We define f_t as,

$$f_t = \frac{(d^B - t_{p1}^B)}{(d^B - t_d^B)} \tag{13}$$

- Urgent Strategy

$$A^B = \begin{cases} \text{Accept } (Pi)_{opt} \ , & \text{if } \ \forall Ep(Pi)_{opt} \geq 0.4 \times f_t \\ \text{Refuse and await new proposals,} & \text{otherwise} \end{cases} \tag{14}$$

- Normal Strategy

$$A^B = \begin{cases} \text{Accept } (Pi)_{opt} \ , & \text{if } \ \forall Ep(Pi)_{opt} \geq 0.6 \times f_t \\ \text{Refuse and await new proposals,} & \text{otherwise} \end{cases} \tag{15}$$

- Patient Strategy

$$A^B = \begin{cases} \text{Accept } (Pi)_{opt} \ , & \text{if } \ \forall Ep(Pi)_{opt} \geq 0.8 \times f_t \\ \text{Refuse and await new proposals,} & \text{otherwise} \end{cases} \tag{16}$$

For both Seller Agents and Buyer Agents, if the running time has exceeded their deadlines set by the agent users, i.e. $t \geq d^S$, or $t \geq d^B$, the system will automatically cancel the overdue Seller Agents or Buyer Agents, notify their users and call for further requests, which may be terminating the agent delegation or modifying the agent's attributes.

3.3.6 Other Issues

Besides the negotiation organizations, strategies and algorithms, the key problems of AANS, the automated negotiation e-marketplace also makes efforts to solve other issues in e-trading. Here, we primarily discuss two of them, predicting negotiation stages and evaluating users' reputation.

3.3.6.1 Prediction of the Negotiation Stages

Even though the negotiation process conducted among intelligent agents can be performed without human-labor, agent users may still expect to know how the negotiation is going during the course and then decide whether it is necessary to change the delegation bonds or even cancel the delegation. To indicate how far a negotiation activity goes, we define three stages for an active negotiation, Almost_Done, Middle_Way and Just_Begin. The evaluation of negotiation stages will be made according to agent roles (Buyer Agent or Seller Agent) and negotiation strategies (Urgent, Normal and Patient) as follows.

(1) For Buyer Agents

Assume $Ep(Pi)_{opt}$ is the evaluation value of the optimal proposal that the Buyer Agent receives in the current round of negotiation.

If the Buyer Agent is taking the Urgent Negotiation Strategy, then

$$Negotiation\_Stage_u = \left\{ \begin{array}{l} Just\_Begin, 0 \le Ep(P_i)_{opt} < 0.133 \\ Middle\_Way, 0.133 \le Ep(P_i)_{opt} < 0.267 \\ Almost\_done, 0.267 \le Ep(P_i)_{opt} < 0.4 \end{array} \right\} ; \tag{17}$$

If the Buyer Agent is taking the Normal Negotiation Strategy, then

$$Negotiation\_Stage_n = \left\{ \begin{array}{l} Just\_Begin, 0 \le Ep(P_i)_{opt} < 0.2 \\ Middle\_Way, 0.2 \le Ep(P_i)_{opt} < 0.4 \\ Almost\_Done, 0.4 \le Ep(P_i)_{opt} < 0.6 \end{array} \right\} ; \tag{18}$$

If the Buyer Agent is taking the Patient Negotiation Strategy, then

$$Negotiation\_Stage_p = \left\{ \begin{array}{l} Just\_Begin, 0 \le Ep(P_i)_{opt} < 0.267 \\ Middle\_Way, 0.267 \le Ep(P_i)_{opt} < 0.533 \\ Almost\_Done, 0.533 \le Ep(P_i)_{opt} < 0.8 \end{array} \right\} ; \tag{19}$$

(2) For Seller Agents

Unlike Buyer Agents who may care about more than price, we assume Seller Agents only seek the possible highest selling price for its product. Because Buyer Agents do not disclose the negotiation factors to others, Seller Agents only can evaluate the negotiation stage according to the price feedback, which is the price of the optimal proposal received by Buyer Agents in the latest round of negotiation. Firstly, we define some criteria:

c_s : The Senior Point, standing for the boundary between Middle_Way Stage and Almost_Done Stage

c_j : The Junior Point, standing for the boundary between Just_Begin Stage and Middle_Way Stage

c_p : The Primary Point, standing for the boundary between Initial Point and Just_Begin Stage

$p_{\min}^{S}$: The minimum selling price that the Seller Agent can offer

$p_{\max}^{S}$: The maximum selling price that the Seller Agent can offer, i.e. the initial price

p_{p}^{S} : The present price that the Seller Agent is offering

p_{opt}^{B} : The offering price of the optimal proposal received by the Buyer Agent in the latest round of negotiation

Therefore,

$$\left\{ \begin{aligned} & c_p = p_{\max}^{S} \\ & c_j = p_{opt}^{B} + \frac{2 \times (p_{opt}^{B} - p_{\min}^{S})}{3} \\ & c_s = p_{opt}^{B} + \frac{p_{opt}^{B} - p_{\min}^{S}}{3} \end{aligned} \right\} \tag{20}$$

Then, the Seller Agent evaluates the negotiation stage for its user by using the following expressions,

$$Negotiation\_Stage = \left\{ \begin{array}{ll} Just\_Begin, & c_j \le p_p^S \le c_p \\ Middle\_Way, & c_s \le p_p^S \le c_j \\ Almost\_Done, & p_p^S \le p_{opt}^B (*) \; or \; p_{opt}^B \le p_p^S \le c_s \end{array} \right\} \qquad (21)$$

* happens in the case that the Seller Agent's offer price is the lowest one among all proposals received by the Buyer Agent, but $Ep(Pi)$ of this proposal is not the highest one, because the Buyer agent does not only consider the price factor. Hence, $Ep(Pi) < Ep(Pi)_{opt}$, but $p_p^S < p_{opt}^B$.

It is obvious that the negotiation stage prediction makes less sense to Seller Agents' users than to Buyer Agents' users.

3.3.6.2 Reputation Mechanism

Without doubt, business activities can be significantly benefited by e-commerce approach, but the downturn of .COM also tells us that many non-technical problems are still restricting the development of it. One of the most critical problems is trust issue which means how people can trust other parties participating in the trade. In the AANS, another trust problem is raised between shopping assistants (software agents) and human traders. Because of time limits, no measures are taken in the current SHCM version regarding it, but further observation will be discussed in the Chapter 5.

Users' reputation becomes a common concern in today's online trading. Much dissatisfaction after trading occurs when one cannot get proper products or services in time. Although many online commercial auction systems provide rating mechanism for users to rate their counterparts, such as "e-Bay" and "EachNet", there can be problems in them all the same, including fake transactions between two friends to rate high reputation mark for each other. To solve this problem, we take some measures in our online marketplace:

- Users can not modify their identity information once their accounts begin to work, but other less critical particulars can be freely changed, such as phone number, address, etc.
- More value of the goods in trading, more important the related rating is considered.
- For more than one times of rating between two participants, only the latest one is useful when computing the current reputation rate.
- Ratings from reputed users will be considered more importantly than those from others with lower reputation.

To achieve the above objectives, we propose a reputation evaluation mechanism in SHCM system. First, we suppose the user who is evaluated has made n transactions in SHCM totally, and the nth is the latest one.

Assume:

R : The current reputation value of the evaluated user

$r_c(i)$: The reputation value of the counterpart in past trades (not including the latest one) which the evaluated user has participated

$v_c(i)$: The rating value that the counterpart evaluates the past trade (not including the latest one) which the evaluated user have participated. $v_c(i) \in \{-1, -0.5, 0, 0.5, 1\}$

d : The decrease factor, used to reduce the influence of the previous rating so that system can count more on the recent rating values. $0 < d \leq 1$

$p_{trade}(i)$: The trading price of the product related to this transaction rating

$p_v(i)$: The product value

We define the product value according to its trading price as follows:

$$p_v = \left\{\begin{array}{ll} 0.1 & \text{if } p_{trade} \le 100; \\ 0.2 & \text{if } 100 < p_{trade} \le 500; \\ 0.4 & \text{if } 500 < p_{trade} \le 1000; \\ 0.8 & \text{if } 1000 < p_{trade} \le 2000; \\ 1.0 & \text{if } p_{trade} > 2000; \end{array}\right\} \tag{22}$$

The current reputation value of the evaluated user is defined as follows.

$$\mathrm{R} = \frac{d \times \sum_{\mathrm{i=1}}^{\mathrm{n-1}} [\, r_c(i) \times v_c(i) \times p_v(i)] \; + \; r_c(n) \times v_c(n) \times p_v(n)}{\mathrm{n}} \qquad (1 \le R \le -1, \quad \mathrm{i = 1, 2, ...n\text{-}1}) \tag{23}$$

Therefore, the reputation rank of users can be given as the following expression.

$$G_r = \left\{\begin{array}{ll} 1 & \text{if } -1 \le \mathrm{R} < -0.66 \\ 2 & \text{if } -0.66 \le \mathrm{R} < -0.33 \\ 3 & \text{if } -0.33 \le \mathrm{R} < 0.33 \\ 4 & \text{if } 0.33 \le \mathrm{R} < 0.66 \\ 5 & \text{if } 0.66 \le \mathrm{R} \le 1 \end{array}\right\} \tag{24}$$

Chapter 4 Implementation

There is no commonly accepted methodology and practice for evaluating negotiation performance of agent-based e-commerce systems because of the intrinsic complexity of the task. We attempt to build an automated negotiation system that simulates the real world negotiation conditions, consume reasonable amount of system resources and is understandable to human users. Further, this negotiation system is exclusively designed and built for the SHCM. Though this negotiation scenario is designed specifically for the SHCM, its modularized design and implementation make the negotiation system loosely coupled with the front-end e-marketplace and provide adaptability, scalability and easy-to-use for constructing dynamic e-marketplaces.

4.1 Selection of Agent Platform and Programming Approach

The key element to the system architecture is the Agent Platform. An Agent Platform (AP) [24] provides an infrastructure in which agents can be deployed. An agent must be registered on a platform in order to interact with other agents on that platform or on other platforms. Minimally, an AP consists of three capability sets: an Agent Communication Channel, an Agent Management System and a Directory Facilitator.

We have explored some programming platforms dedicated to creating MAS; some of them are oriented towards communication between distributed systems, as some others are more oriented towards the building of simulation models. For platform selection, we primarily compared three open source multi-agent platforms:

(1) COMAS(Co-Ordination in Multi-Agent Systems) [25]

COMAS belongs to the second category - simulation platform, which implements resource coordination as well as goal coordination. Each agent can make a resource request to another agent for resource coordination. Additionally, each agent passes the state change information to the other agent after it has performed an action to achieve goal interaction coordination. Therefore, this platform is established with specificity in the domain of natural resources management.

(2) ASDK (Aglets Software Development Kit) [26]

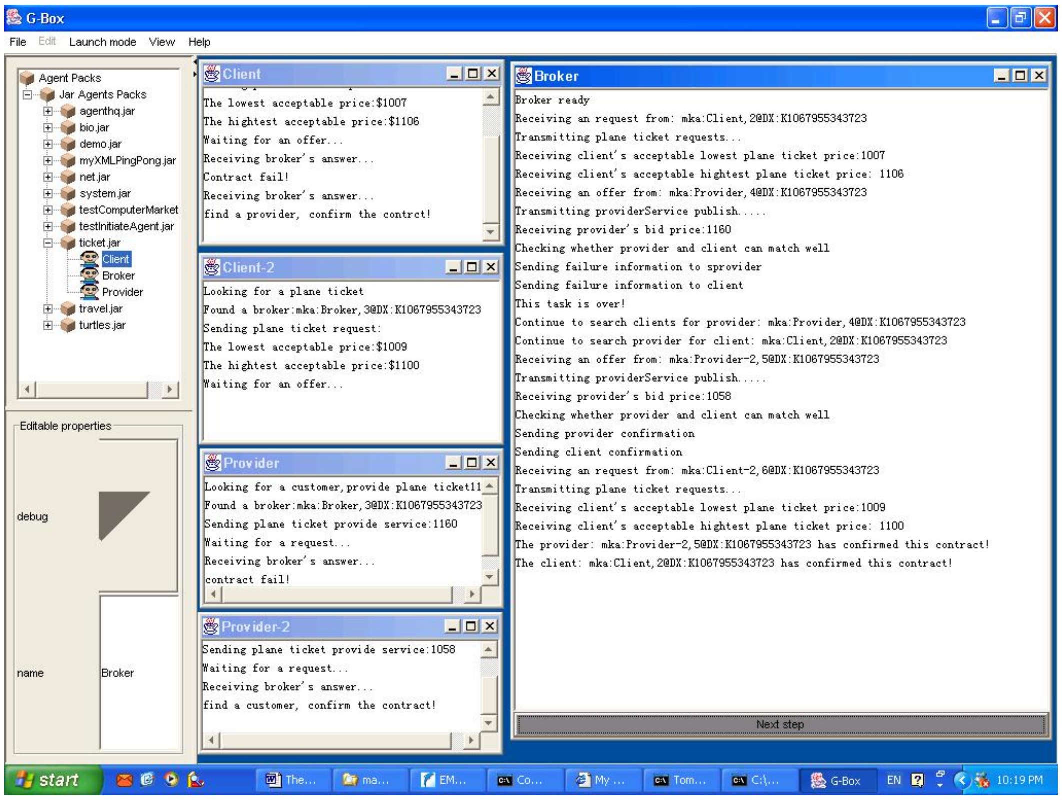
Aglets is a framework and environment developed by IBM for researching and developing mobile Internet agents in Java, belonging to part of IBM worldwide research. The focus here is mobile capability of software agents. Aglets are Java objects that can move from one host on the Internet to another, introducing program code that can be transported along with state information. That is, an aglet that executes on one host can suddenly halt execution, dispatch itself to a remote host, and resume execution there. When the aglet moves, it takes along its program code as well as its data. This is also open source software freely downloaded from IBM.

(3) MadKit(Multi-Agent Development Kit) [27]

MadKit is a versatile Java agent platform, designed to support heterogeneous agent and communication models, and host multiple distributed applications. The platform architecture is based on a minimalist agent kernel decoupled from specific agent or communication models. Services like distributed message passing, migration or monitoring are provided by individual agents for maximal flexibility. A componential graphical interface model enables variations in platform looks and classes of usage.

MadKit is built upon an organizational model. It provides general agent facilities (lifecycle management, message passing, distribution, etc.), and allows high heterogeneity in agent architectures, communication languages, and various customizations. The AGR (agent-group-role) organization model is rooted in this platform for building large, scale, heterogeneous systems. To avoid agent-based viewpoint, AGR organization can be only regarded as a structural relationship among a group of agents. In addition to the three core concepts, the platform adds three design principles, the micro-kernel architecture, agentification of services and the graphic component model.

MadKit itself is a set of packages of Java classes that implements the agent kernel, the various libraries of messages, probes and agents. It also includes a graphical development environment and standard agent models. Fig.13 shows the Graphical User Interface of MadKit, G-Box.



Client
```
The lowest acceptable price:$1007
The hightest acceptable price:$1106
Waiting for an offer...
Receiving broker's answer...
Contract fail!
Receiving broker's answer...
find a provider, confirm the contrct!
```

Client-2
```
Looking for a plane ticket
Found a broker:mka:Broker,3@DX:K1067955343723
Sending plane ticket request:
The lowest acceptable price:$1009
The hightest acceptable price:$1100
Waiting for an offer...
```

Provider
```
Looking for a customer,provide plane ticket11
Found a broker:mka:Broker,3@DX:K1067955343723
Sending plane ticket provide service:1160
Waiting for a request...
Receiving broker's answer...
contract fail!
```

Provider-2
```
Sending plane ticket provide service:1058
Waiting for a request...
Receiving broker's answer...
find a customer, confirm the contract!
```

Broker
```
Broker ready
Receiving an request from: mka:Client,2@DX:K1067955343723
Transmitting plane ticket requests...
Receiving client's acceptable lowest plane ticket price:1007
Receiving client's acceptable hightest plane ticket price: 1106
Receiving an offer from: mka:Provider,4@DX:K1067955343723
Transmitting providerService publish.....
Receiving provider's bid price:1160
Checking whether provider and client can match well
Sending failure information to sprovider
Sending failure information to client
This task is over!
Continue to search clients for provider: mka:Provider,4@DX:K1067955343723
Continue to search provider for client: mka:Client,2@DX:K1067955343723
Receiving an offer from: mka:Provider-2,5@DX:K1067955343723
Transmitting providerService publish.....
Receiving provider's bid price:1058
Checking whether provider and client can match well
Sending provider confirmation
Sending client confirmation
Receiving an request from: mka:Client-2,6@DX:K1067955343723
Transmitting plane ticket requests...
Receiving client's acceptable lowest plane ticket price:1009
Receiving client's acceptable hightest plane ticket price: 1100
The provider: mka:Provider-2,5@DX:K1067955343723 has confirmed this contract!
The client: mka:Client,2@DX:K1067955343723 has confirmed this contract!
```

Fig.13 G-Box - The Graphical User Interface of MadKit

Because COMAS platform is dedicated for resource coordination, it is not a very scalable architecture for e-commerce applications. In addition, there are not many development guidelines and application demos available on the Internet, it is hard to evaluate COMAS' usability and get technical for our implementation. Thus, COMAS is not taken into consideration for SHCM.

For ASDK, though it is a Java-based open source software from IBM, with a relatively high reputation, it is also not proper for SHCM due to its specificity with mobile agents. There are a few reasons why we do not consider building this negotiation system with an agent mobility approach.

Firstly, there is no need for Shopping Agents to move on the Internet [28] either for the LSM or NCM of AANS. Often, the typical Client/Server application communicates via requests and responses, which require a round trip trek across the network, therefore moving code to implement locally is more cost-efficient. However, in our SHCM system, cross-network communication between client-end and server-end is not frequent and most complicated negotiation processing work is executed automatically in the server. Only when some Informal Contracts are achieved or the Shopping Agents are expired, the system will contact client-end users again for further demands.

Secondly, negotiation is not a time-sensitive task. Users usually will not and need not wait for the negotiation result in front of their computers all the time. In fact, to save the time for online searching, automatic searching and matching is one of the important reasons why people use shopping assistants. In other words, the client-end users need not always be available after the agent delegation is activated, therefore email, pagers, SMS… are more suitable approaches than real-time network connection for the latter communication with users. Thus, Shopping Agents absolutely can work locally on the server and need not roam over the Internet.

Finally, the distinguishing characteristic of the mobile agent architecture is the mobility of the code. This mobility of the software code will raise the problems of communication efficiency, reliability and security, as well we as programming complexity. A centralized negotiation processor which need not frequent communication with clients is more applicable and suitable for the SHCM scenario.

Therefore, instead of choosing a mobile agent platform, we make Madkit 2.0.1 (the latest version of Madkit is 3.1b5 released in March, 2003) the infrastructure of the AANS.

4.2 Implementations of SHCM

In this section, we consider both the LSM and NCM models and discuss on implementation issues such as requirement analysis, system design and modeling.

4.2.1 Local Service Model

4.2.1.1 Requirement Analysis

For LSM of SHCM, the major functions provided to the e-marketplace users are divided into two parts, normal services and special services. The normal services are those ordinary ones most e-marketplaces provide and the special service is the ANS that our proposed system exclusively provides. We will describe the system requirements of these two parts as below.

(1) Requirements of normal services

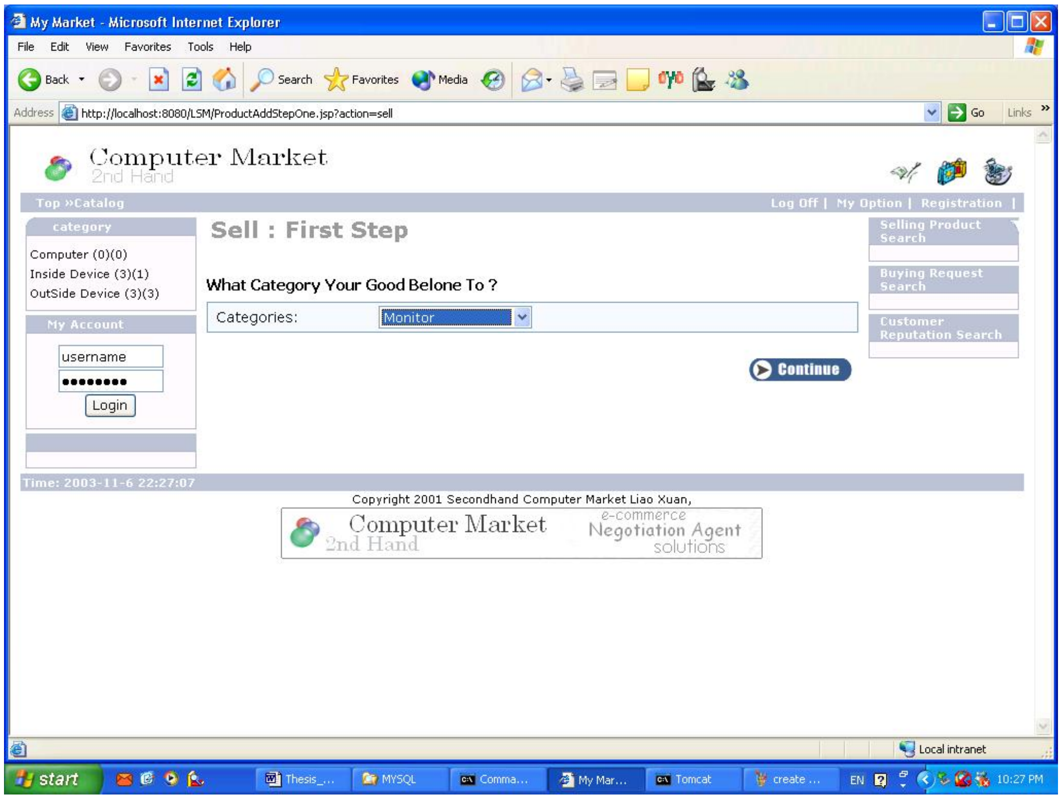
The business requirements of traditional services are presented in the following table.

Requirements	**Normal Users**	**Members (customers)**	**Remark**
Browse Item	✓	✓	
Search Item	✓	✓	
Register	✓		Users will be requested to fill a Customer Particular Sheet (CPS) to create a unique account. The CPS should include username, first name, last name, gender, password, IC, date of birth, phone number, and postal address, etc.
View/Modify Customer Information	N/A	✓	Some identified information should not be modified after creating for the sake of reputation concerns, such as ID, first name, last name), but most changeful items, such as phone number, email, postal address can be modified.
Selling/Buying	N/A	✓	Customers will be requested to fill an Item

Items			Requirement Sheet (IRS) to specify the attributes of a selling product or a buying request, not including the constraints of negotiation delegation. The major attributes in IRS are item category, item models, optimal selling/purchase price, brands, used time (measure of an extent to which the item is new), selling/buying deadline, brief description, etc.
Modify/Cancel Items	N/A	✓	

Table1 Business Requirements of Normal Services

The following Fig.14, Fig.15 and Fig.16 have demonstrated the web interface for filling Item Requirement Sheet (IRS) and Customer Particular Sheet (CPS).


Address: http://localhost:8080/LSM/ProductAddStepOne.jsp?action=sell

Sell : First Step

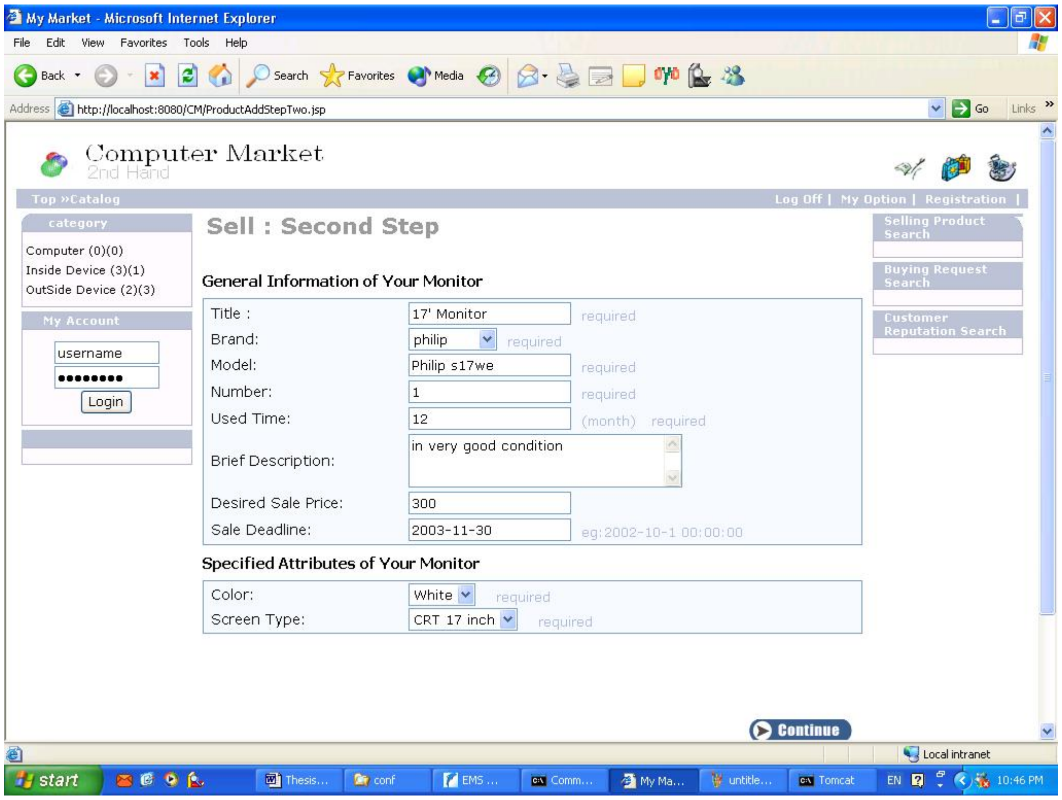
What Category Your Good Belone To ?

Categories: Monitor

Time: 2003-11-6 22:27:07

Copyright 2001 Secondhand Computer Market Liao Xuan,

Fig.14 Item Requirement Sheet (IRS) --- First Step



My Market - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost:8080/CM/ProductAddStepTwo.jsp
Computer Market
2nd Hand
Top »Catalog
Log Off | My Option | Registration |
category
Computer (0)(0)
Inside Device (3)(1)
OutSide Device (2)(3)
My Account
Login
Selling Product Search
Buying Request Search
Customer Reputation Search
Continue

Sell : Second Step

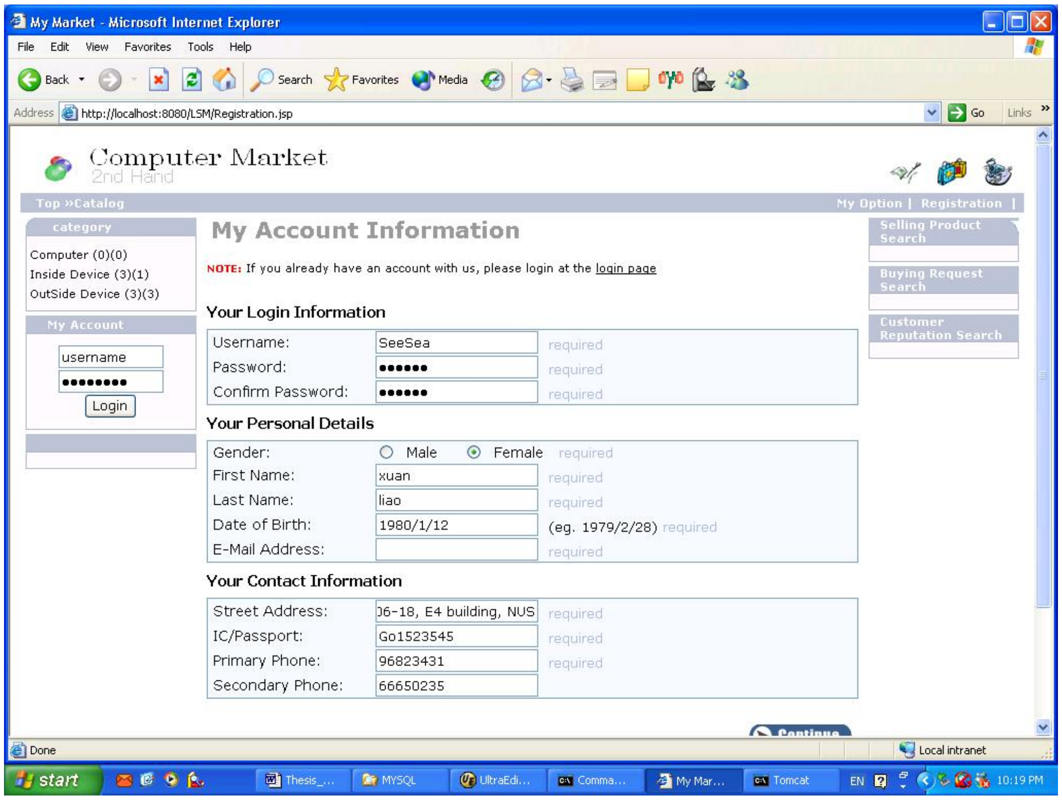
General Information of Your Monitor

Title :	17' Monitor	required
Brand:	philip	required
Model:	Philip s17we	required
Number:	1	required
Used Time:	12	(month) required
Brief Description:	in very good condition	
Desired Sale Price:	300	
Sale Deadline:	2003-11-30	eg:2002-10-1 00:00:00

Specified Attributes of Your Monitor

Color:	White	required
Screen Type:	CRT 17 inch	required

Fig.15 Item Requirement Sheet (IRS) --- Second Step



My Market - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost:8080/LSM/Registration.jsp
Computer Market
2nd Hand
Top »Catalog
My Option | Registration |
category
Computer (0)(0)
Inside Device (3)(1)
OutSide Device (3)(3)
My Account
Login
Selling Product Search
Buying Request Search
Customer Reputation Search
Done

My Account Information

NOTE: If you already have an account with us, please login at the login page

Your Login Information

Username:	SeeSea	required
Password:	••••••	required
Confirm Password:	••••••	required

Your Personal Details

Gender:	○ Male [✓ Female]	required
First Name:	xuan	required
Last Name:	liao	required
Date of Birth:	1980/1/12	(eg. 1979/2/28) required
E-Mail Address:		required

Your Contact Information

Street Address:	06-18, E4 building, NUS	required
IC/Passport:	G01523545	required
Primary Phone:	96823431	required
Secondary Phone:	66650235	

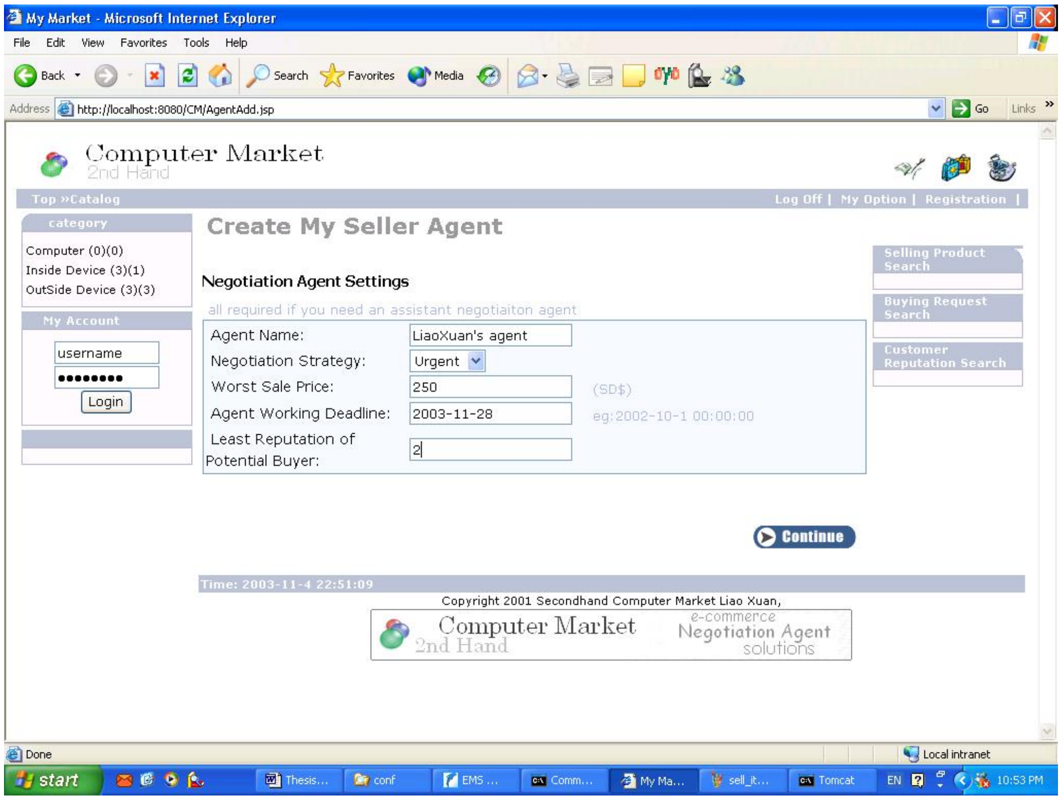
Fig.16 Customer Particular Sheet (CPS)

(2) Requirements of ANS

When customers add a new item to the e-marketplace, they can just let this item list online or continue to delegate this item to a Shopping Agent. That means ANS is an optional service for any customer. Usually, seven actions may happen if customers choose this advanced service.

Action 1: Create Agents

To create a Shopping Agent and delegate negotiation constraints to the agent, SHCM requests the customer who wants to make use of ANS to fill an Agent Delegation Sheet (ADS) shown in Fig.17. All important data for negotiation are included in this sheet. For the customer who sells goods, his/her Shopping Agent is called as a Seller Agent and for the one who purchases goods, his/her Shopping Agent is called as a Buyer Agent. ADS should include attributes, buyer/seller agent name, working deadline, worst purchase/selling price, negotiation strategy and reputation of the counterpart.


My Market - Microsoft Internet Explorer

http://localhost:8080/CM/AgentAdd.jsp

Computer Market 2nd Hand

Top »Catalog | Log Off | My Option | Registration |

category: Computer (0)(0), Inside Device (3)(1), OutSide Device (3)(3)

My Account: username, Login

Create My Seller Agent

Negotiation Agent Settings

all required if you need an assistant negotiaiton agent

Agent Name:	LiaoXuan's agent	
Negotiation Strategy:	Urgent	
Worst Sale Price:	250	(SD$)
Agent Working Deadline:	2003-11-28	eg:2002-10-1 00:00:00
Least Reputation of Potential Buyer:	2	

Continue

Selling Product Search | Buying Request Search | Customer Reputation Search

Time: 2003-11-4 22:51:09

Copyright 2001 Secondhand Computer Market Liao Xuan,

Computer Market 2nd Hand — e-commerce Negotiation Agent solutions

Fig.17 Agent Delegation Sheet (ADS)

Besides the above constraints, the attributes about items are all necessary information for agent negotiation. Among all these attributes, only price, brands, using time and counterpart’s reputation are optional negotiation factors whereas, others are static constraints for group matching ahead of negotiation.

“Create Agent” action happens in two cases. The first case is that customers delegate the negotiation task to Shopping Agents for new items. It permits a customer to create multiple Seller Agents for one selling product, or multiple Buyer Agents for one buying request. The second one is that customers want to change the delegation constraints, such as negotiation strategy and agent working deadline, the previous agent running under the old constraints will be “killed” (terminated) and a new agent will be created for the new constraints.

SHCM permits one user to create more than one Seller Agent for one selling product, or more than one Buyer Agent for one buying request; it also permits users to own more than one agent, either Seller Agents or Buyer Agents, running for different selling products or buying requests.

Action 2: Monitor Agents

To increase the visibility of agents' work, SHCM gives agent users the right to monitor their Shopping Agents and to check the related static and dynamic information. The static information includes agent type, agent name, agent id, delegation constraints, and related item attributes. The dynamic information includes agent status, present offering price and negotiation stage prediction.

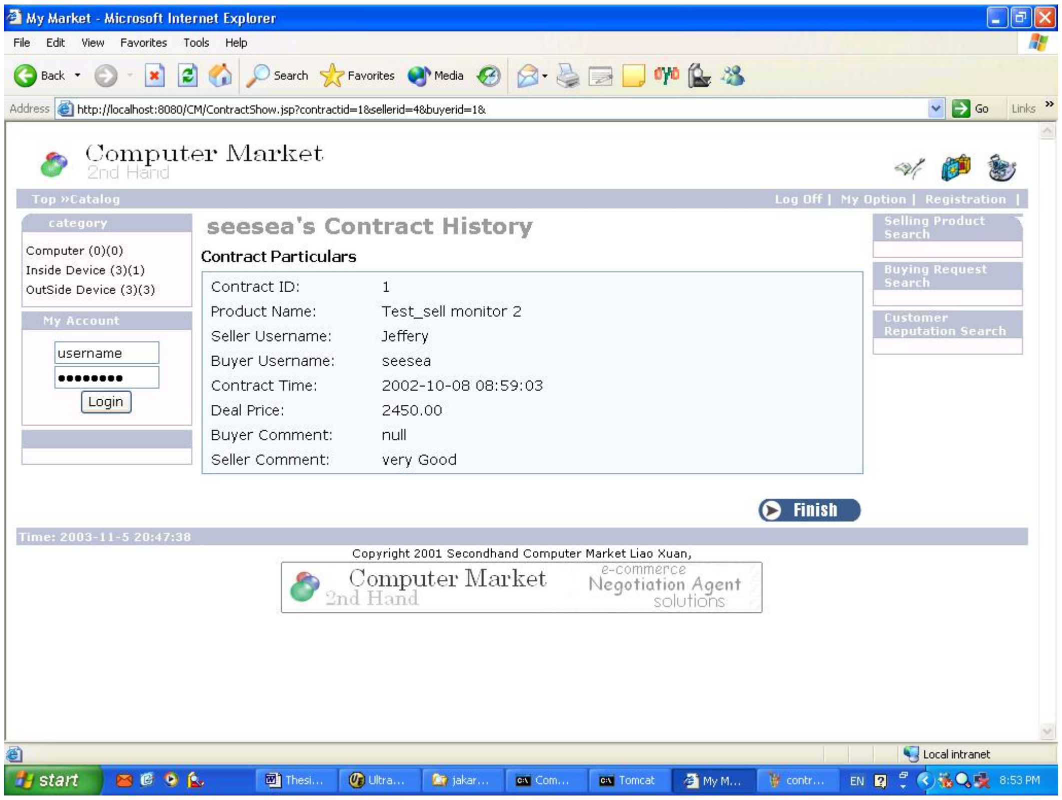
Action 3: Cancel Agents

"Cancel Agents" only means to cancel Shopping Agent delegation, but does not mean to cancel the related items in e-marketplaces. Only if the customer chooses "Delete Product/Request", the corresponding items will be eliminated from the e-marketplace.

It may be noted that "Modify Agents" action will not happen in SHCM forever, since any changes of agent attributes or related item attributes will need a new agent to be created. The new agent will be identified by a new ID and negotiation process will be re-started.

Action 4: Check Contract History

SHCM will list all the successful contracts that are made in AANS and all the Informal Contracts that participating customers have not presented attitudes. The primary information includes contract id, contract time, contract price, attributes of the trading product, basic information of participated customers, the rating values and comments for the Formal Contracts, and the attitudes and comments for the Informal Contracts. Fig.18 is the history record for a Formal Contract.



My Market - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address
Go Links
Computer Market
2nd Hand
Top »Catalog
Log Off | My Option | Registration |
category
Computer (0)(0)
Inside Device (3)(1)
OutSide Device (3)(3)
My Account
username
Login
Selling Product Search
Buying Request Search
Customer Reputation Search
Finish
Local intranet
start
Thesi... Ultra... jakar... Com... Tomcat My M... contr...
EN 8:53 PM

http://localhost:8080/CM/ContractShow.jsp?contractid=1&sellerid=4&buyerid=1&

seesea's Contract History

Contract Particulars

Contract ID:	1
Product Name:	Test_sell monitor 2
Seller Username:	Jeffery
Buyer Username:	seesea
Contract Time:	2002-10-08 08:59:03
Deal Price:	2450.00
Buyer Comment:	null
Seller Comment:	very Good

Time: 2003-11-5 20:47:38

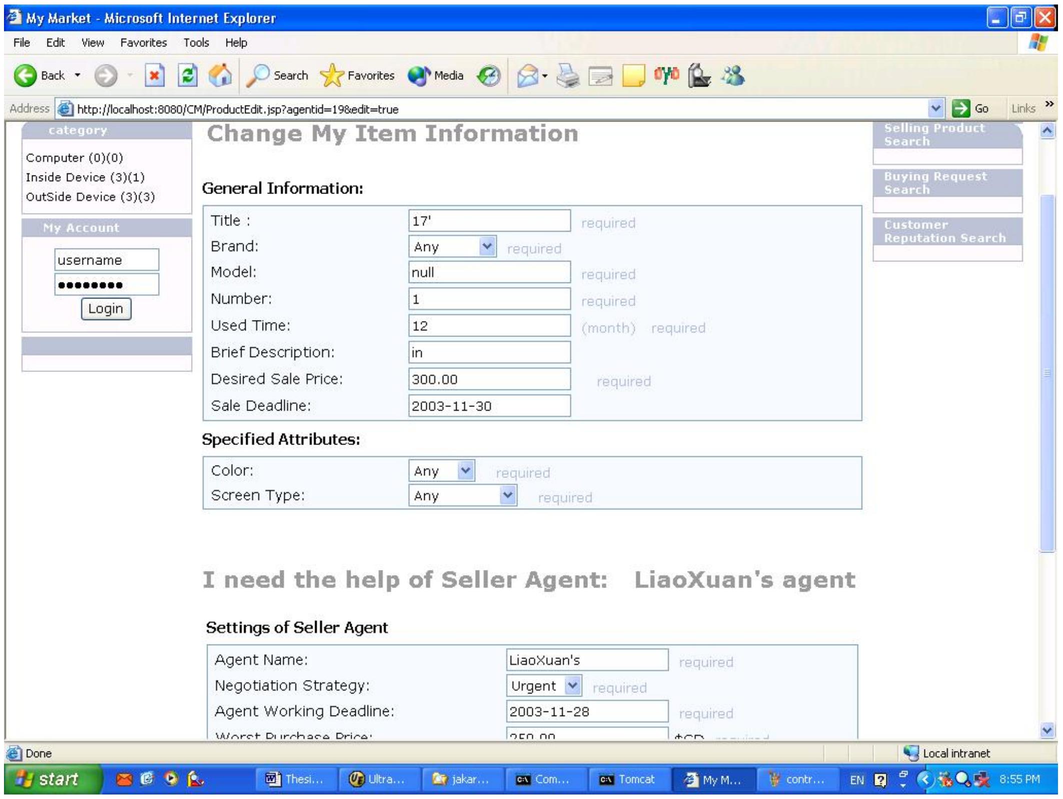
Copyright 2001 Secondhand Computer Market Liao Xuan,

Computer Market 2nd Hand e-commerce Negotiation Agent solutions

Fig.18 Contract History Record

Action 5: Modify Items (Selling Products/Buying Requests)

Even if items are delegated to Shopping Agents or not, customers can change item attributes at any time. The system will display the IRS filled previously by the customer as in Fig.19 and allow the customer to modify. Here too, it may be noted that if an item has been delegated to some Shopping Agent, once the item’s attributes are changed, all correlated Shopping Agents will be terminated in the negotiation system and the customer will be queried whether or not they want to create new agents for the modified item.


My Market - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost:8080/CM/ProductEdit.jsp?agentid=19&edit=true
Go Links
category
Computer (0)(0)
Inside Device (3)(1)
OutSide Device (3)(3)
My Account
username
Login
Selling Product Search
Buying Request Search
Customer Reputation Search
Done
Local intranet
start
8:55 PM

Change My Item Information

General Information:

Title :	17'	required
Brand:	Any	required
Model:	null	required
Number:	1	required
Used Time:	12	(month) required
Brief Description:	in	
Desired Sale Price:	300.00	required
Sale Deadline:	2003-11-30	

Specified Attributes:

Color:	Any	required
Screen Type:	Any	required

I need the help of Seller Agent: LiaoXuan's agent

Settings of Seller Agent

Agent Name:	LiaoXuan's	required
Negotiation Strategy:	Urgent	required
Agent Working Deadline:	2003-11-28	required
Worst Purchase Price:	250.00	$CD required

Fig.19 Item Information Changes Interface

Action 6: Response to Informal Contracts

Once SHCM makes any successful Informal Contract, the system will automatically notify the participating customers to judge it and make the decision. Though the Informal Contract absolutely meets the customers' requirements, customers may still feel unsatisfied with the result and therefore have the right to refuse it.

The system notifies both the participating customers of an Informal Contract to make further decisions by e-mails. If any party refuses, the Informal Contract will be deemed "Fail" and that party that has chosen "Refuse" will be asked to brief the reason. Then the system will send both parties a "failure message" with reasons stated by customer(s) and query customers whether they want to negotiate again under the same or different delegation constraints, or definitely want to cancel the delegation. If both parties agree on the Informal Contract, the system will send them a “success message” and convert the Informal Contract to a Formal Contract.

Action 7: Comments on Contracts

SHCM does not supply any services for payment and delivery issues and we assume that the participants will make the transaction privately. After making transaction in private, SHCM will request users to give feedback towards this contract. The required feedback includes comments and rating for the service quality of Shopping Agents and counterpart’s credit in this transaction.

4.2.1.2 System Design & Modeling

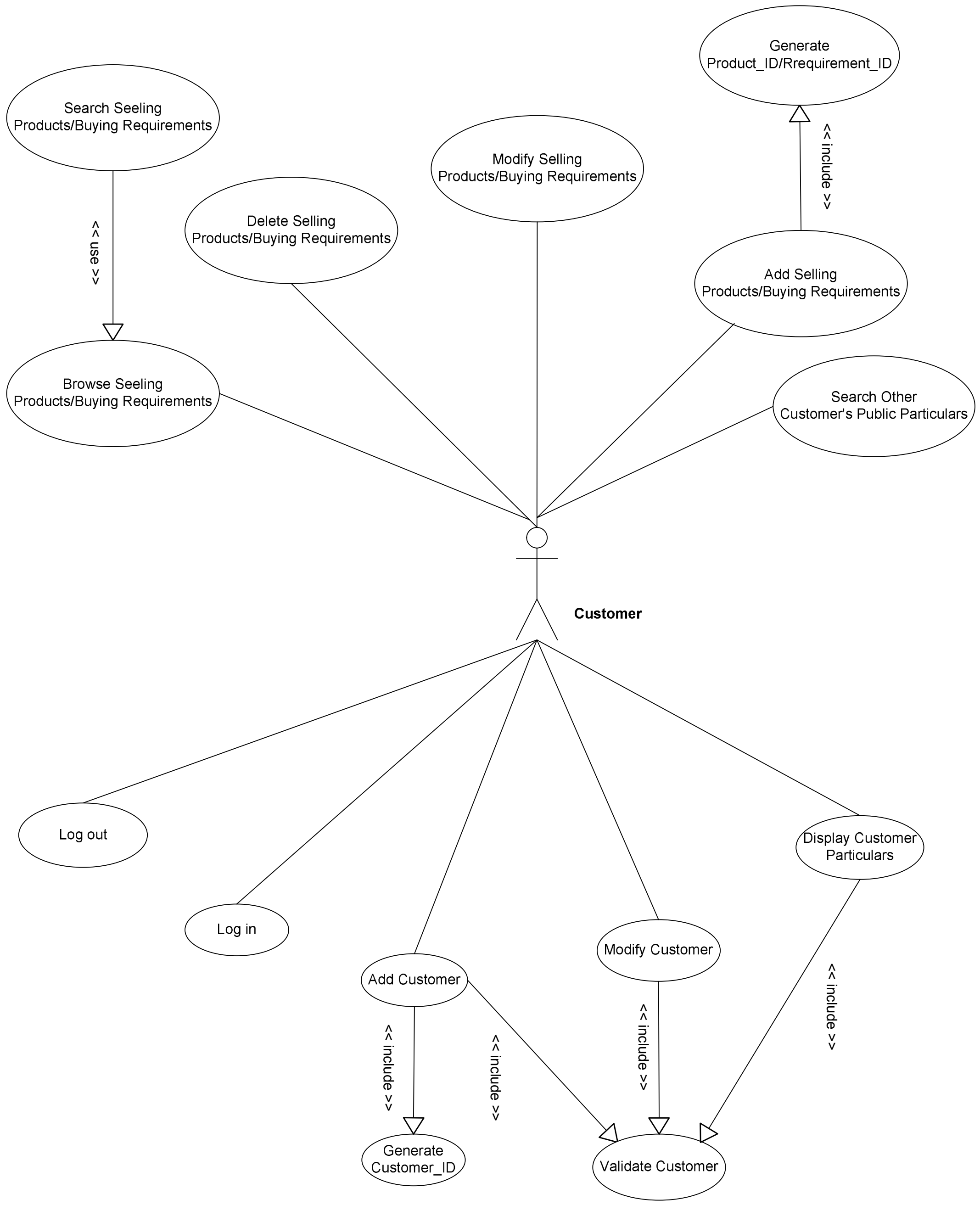


Fig.20 Use Case Diagram of Traditional E-Marketplace Sub-System in LSM

(1) Agent UML

In the past, research on agent-based software engineering had been widely lacking touch with the world of industrial software development. For the acceptance of agent technology in industry, FIPA (the Foundation of Intelligent Physical Agents) and OMG (Object Management Group) relate the object-oriented software development to the agent-based software development and proposed AGENT UML [29], an extension of the standard UML (Unified Modeling Language). For the same sake, we make use of AGENT UML as the tools to analyze and design the whole SHCM system, including the AANS.

(2) System Architecture

Fig.1 demonstrates the system architecture for LSM. The normal services analyzed in Section 4.2.1.1 are modularized into the traditional e-marketplace sub-system. The ANS is independently plugged into an AANS as a plug-in module of the major e-marketplace.

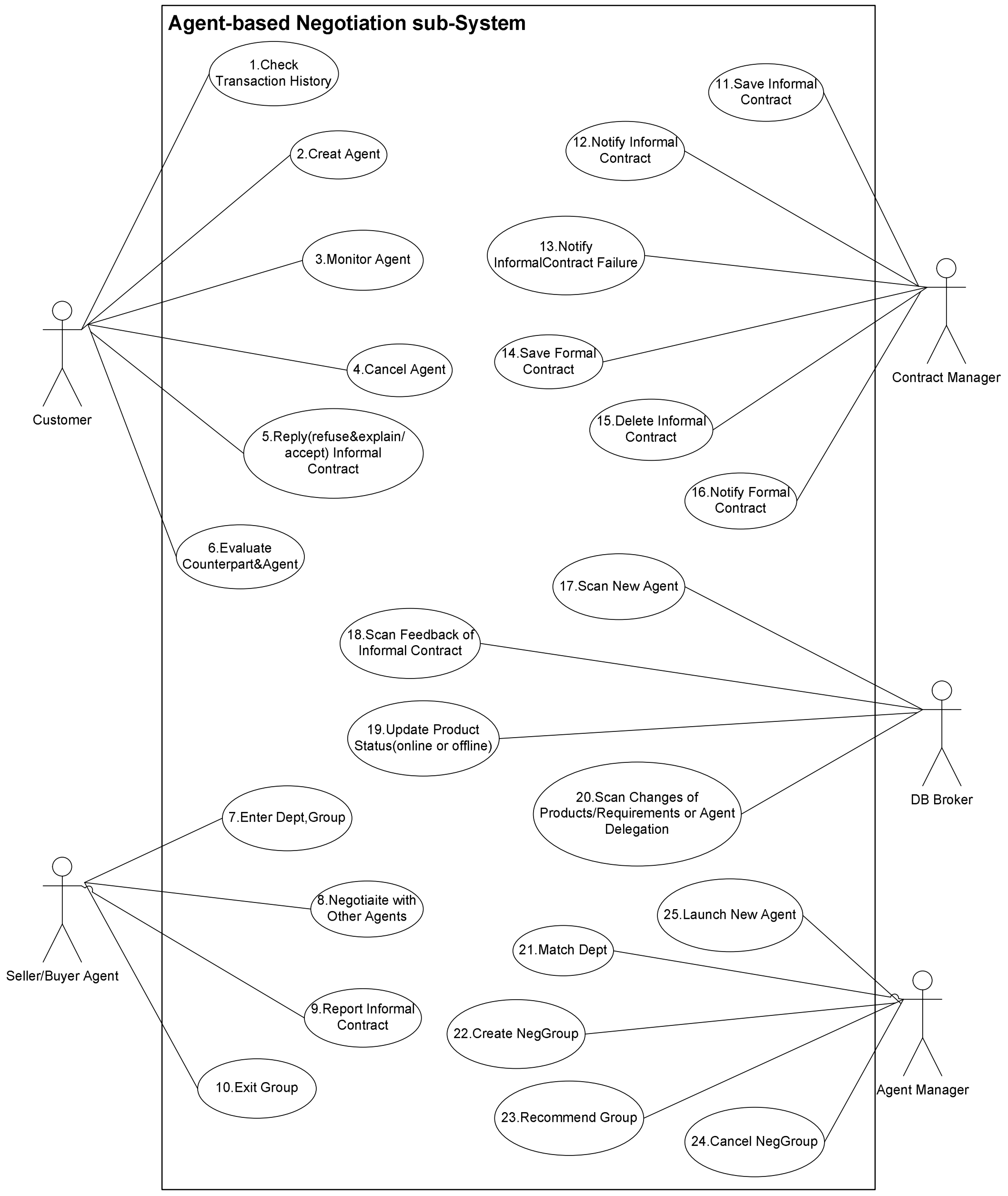

Agent-based Negotiation sub-System
1.Check Transaction History
2.Creat Agent
3.Monitor Agent
4.Cancel Agent
5.Reply(refuse&explain/ accept) Informal Contract
6.Evaluate Counterpart&Agent
7.Enter Dept,Group
8.Negotiaite with Other Agents
9.Report Informal Contract
10.Exit Group
11.Save Informal Contract
12.Notify Informal Contract
13.Notify InformalContract Failure
14.Save Formal Contract
15.Delete Informal Contract
16.Notify Formal Contract
17.Scan New Agent
18.Scan Feedback of Informal Contract
19.Update Product Status(online or offline)
20.Scan Changes of Products/Requirements or Agent Delegation
21.Match Dept
22.Create NegGroup
23.Recommend Group
24.Cancel NegGroup
25.Launch New Agent
Customer
Seller/Buyer Agent
Contract Manager
DB Broker
Agent Manager

Fig.21 Use Case Diagram of Agent-Based Automated Negotiation Sub-System

(3) Use Case Diagram

To identify and partition system functionality, we draw two use case diagrams [22] for the two sub-systems, the traditional e-marketplace sub-system and the agent-based automated negotiation sub-system.

For the traditional e-marketplace sub-system, only one actor, Customer, is defined to represent customers of the e-marketplace and correspondingly fourteen use cases are defined to express Customer’s behaviors, as shown in Fig.20.

For AANS, six actors and twenty-four use cases are defined, as shown in Fig.21. Except for Customer stands for human customers, all the rest actors stand for the corresponding software agents. Actors of Seller Agent and Buyer Agent present the negotiation players in this sub-system. Actors of Agent Manager, DB Broker and Contract Manager are the management staff who organize and administrate the actors of Seller Agent and Buyer Agent for conducting automatic negotiation. Once AANS is started, all Management Agents will be automatically created by the system and in their places all the time. If any Management Agents break down, the whole negotiation system will collapse. On the other hand, Shopping Agents are created and terminated dynamically by Agent Manager on demand of customers. It may be noted that the collapse of any Shopping Agents will not affect the whole system except themselves.

(4) Sequence Diagrams

The definition of communication protocols is part of the specification of the dynamical model of an agent system. To focus on agent working system, we only demonstrate the agent interaction protocols but omit the interaction between users and the e-marketplace. In Agent UML, Agent Interaction Model (AIM) is captured by interaction diagrams, sequence diagrams and activity diagrams. Because in AANS, messaging communication is the major activity during the negotiation, we only focus on the sequence diagrams in this thesis to describe the detailed interaction processes among agents. Fig.22, 23, 24, 25 demonstrate the pre-negotiation and post-negotiation processes among Management Agents and Shopping Agents, and Fig.26, 27, 28 represent the negotiation processes among Management Agents and Shopping Agents.

Fig.22 represents the process that the Agent Manager helps a new Seller Agent get into a proper Department and matched Negotiation Groups. It also represents the process a Seller Agent informs the Contract Manager as it achieves a successful Informal Contract with a Buyer Agent, and the Contract Manager saves the Informal Contract data into DB.

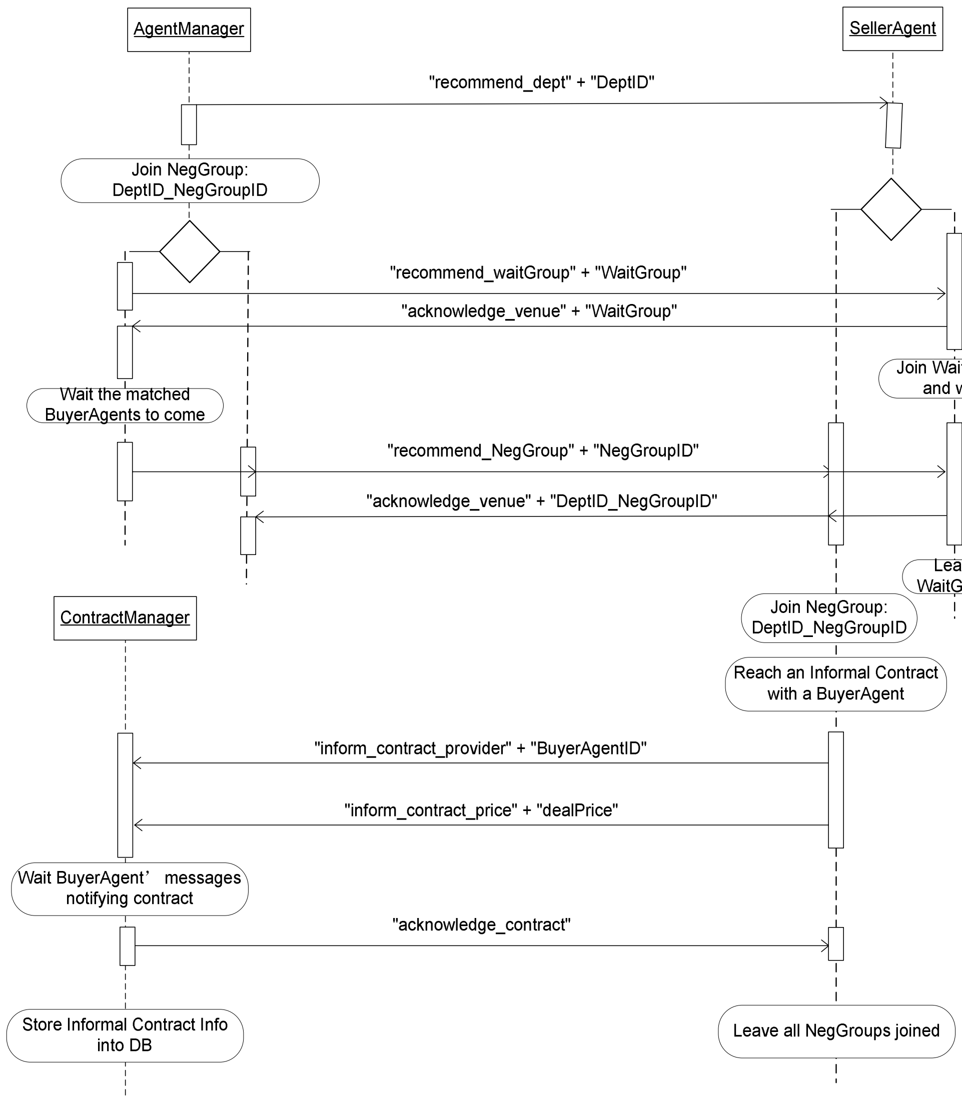

AgentManager
SellerAgent
"recommend_dept" + "DeptID"
Join NegGroup: DeptID_NegGroupID
"recommend_waitGroup" + "WaitGroup"
"acknowledge_venue" + "WaitGroup"
Join WaitGroup and wait
Wait the matched BuyerAgents to come
"recommend_NegGroup" + "NegGroupID"
"acknowledge_venue" + "DeptID_NegGroupID"
Leave WaitGroup
ContractManager
Join NegGroup: DeptID_NegGroupID
Reach an Informal Contract with a BuyerAgent
"inform_contract_provider" + "BuyerAgentID"
"inform_contract_price" + "dealPrice"
Wait BuyerAgent' messages notifying contract
"acknowledge_contract"
Store Informal Contract Info into DB
Leave all NegGroups joined

Fig.22 Communication Protocol & Message Format
(Agent Manager, Contract Manager --- Seller Agent)

Fig.23 represents the process that the Agent Manager helps a new Buyer Agent get into a proper Department and creates a new Negotiation Group for it. Additionally, it represents the process that the Buyer Agent informs Contract Manager as it achieves a successful Informal Contract and then Contract Manager saves the Informal Contract information into DB.

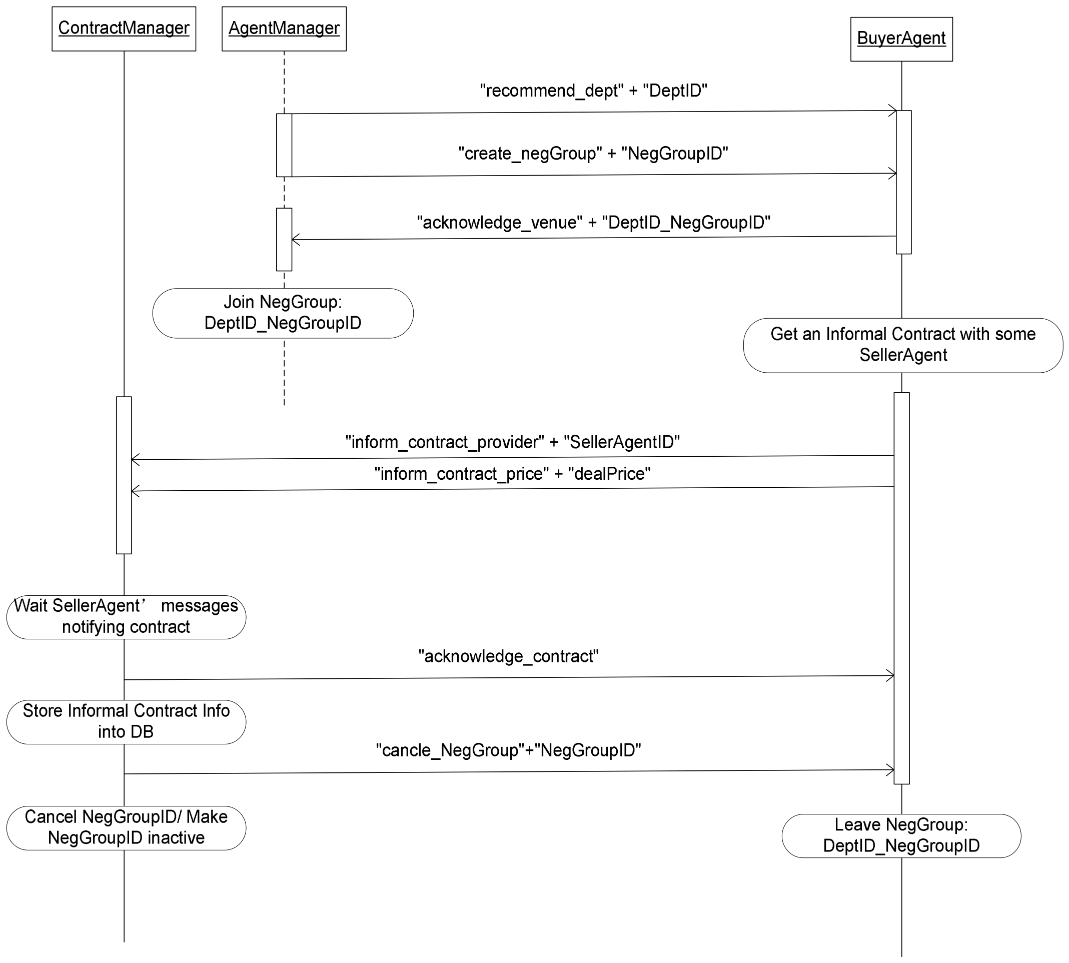

ContractManager
AgentManager
BuyerAgent
"recommend_dept" + "DeptID"
"create_negGroup" + "NegGroupID"
"acknowledge_venue" + "DeptID_NegGroupID"
Join NegGroup: DeptID_NegGroupID
Get an Informal Contract with some SellerAgent
"inform_contract_provider" + "SellerAgentID"
"inform_contract_price" + "dealPrice"
Wait SellerAgent' messages notifying contract
"acknowledge_contract"
Store Informal Contract Info into DB
"cancle_NegGroup"+"NegGroupID"
Cancel NegGroupID/ Make NegGroupID inactive
Leave NegGroup: DeptID_NegGroupID

Fig.23 Communication Protocol & Message Format
(Agent Manager, Contract Manager --- Buyer Agent)

Fig.24 is the cooperation sequence diagram between DB Broker and Agent Manager. DB Broker scans DB periodically to check new Shopping Agents. Once DB Broker has any new finding, it will notify Agent Manager the type and ID of the new agent. Then Agent Manager will activate the shopping agent and lead it into AANS. Apart from scanning new agents, DB Broker also take charge of identifying overdue Shopping Agents. Once DB Broker finds out that any running agents have exceeded the deadline set by their users, DB Broker will inform Agent Manager to “kill” (terminate) them.

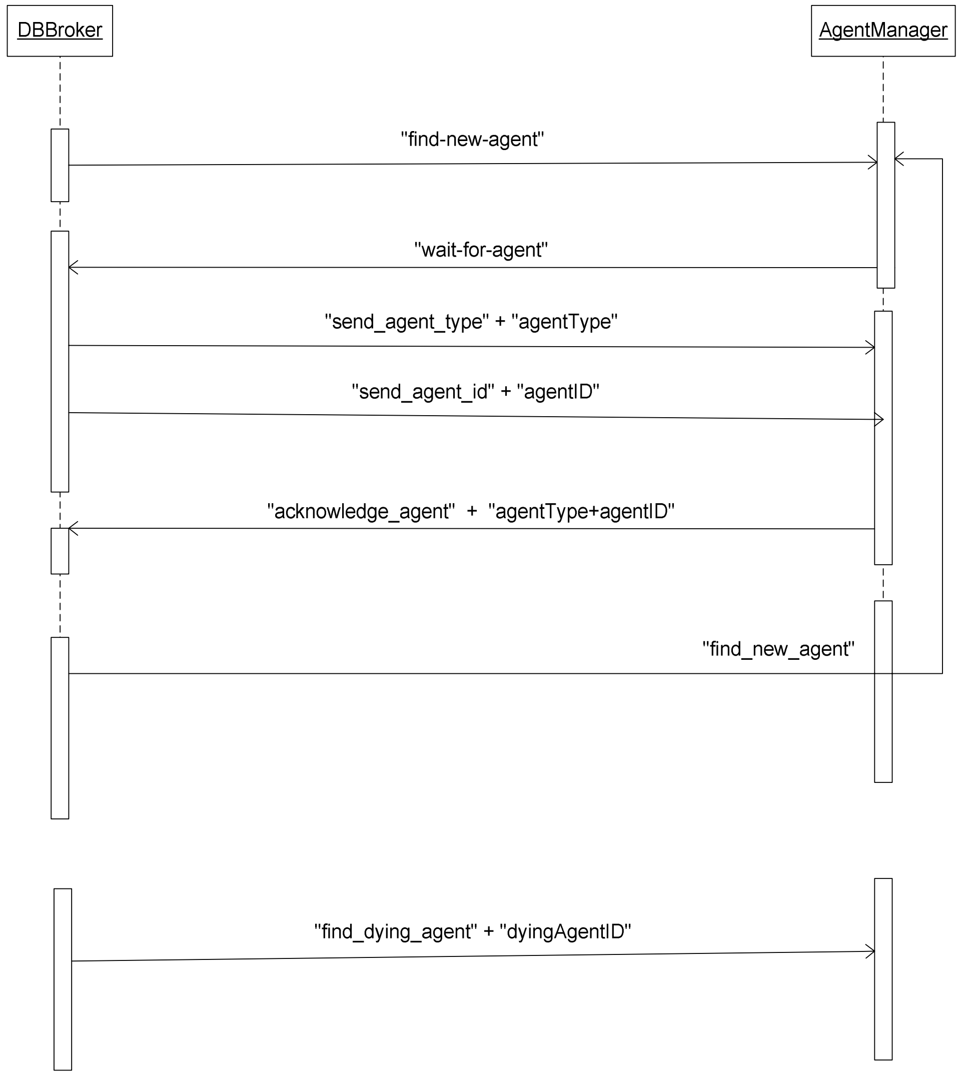

DBBroker
AgentManager
"find-new-agent"
"wait-for-agent"
"send_agent_type" + "agentType"
"send_agent_id" + "agentID"
"acknowledge_agent" + "agentType+agentID"
"find_new_agent"
"find_dying_agent" + "dyingAgentID"

Fig.24 Communication Protocol & Message Format
(DB Broker --- Agent Manager)

Fig.25 is the cooperation sequence diagram among DB Broker, Contract Manager and a Buyer Agent. If the Buyer Agent successfully achieves an Informal Contract with a Seller Agent, the Buyer Agent will send messages to notify Contract Manager about it. Then Contract Manager saves the Informal Contract information into DB, informs the related customers of this contract via emails and asks for the customers' attitudes. The customers' feedback will be automatically recorded in DB, which can be found out by DB Broker's periodic scan. Once DB Broker knows any Informal Contract that has been replied by both participants, it will message Contract Manager. Then Contract Manager processes the Informal Contract according to the customers' responses.

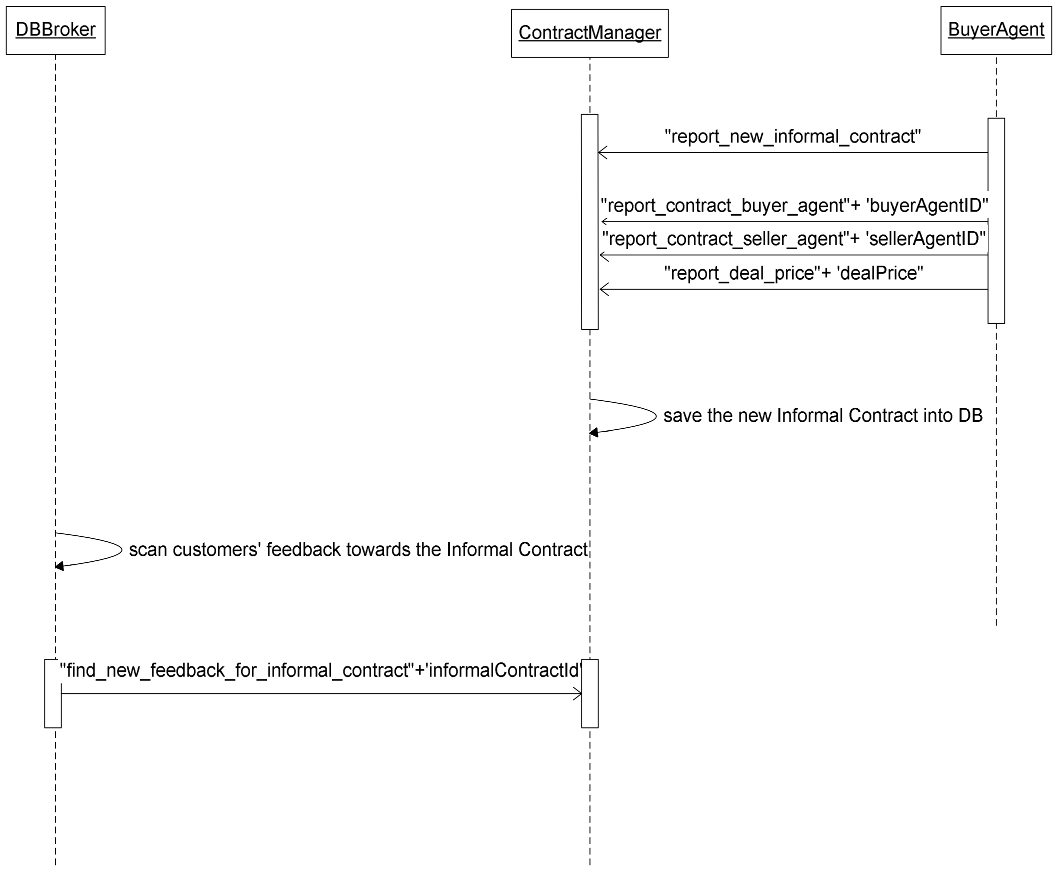
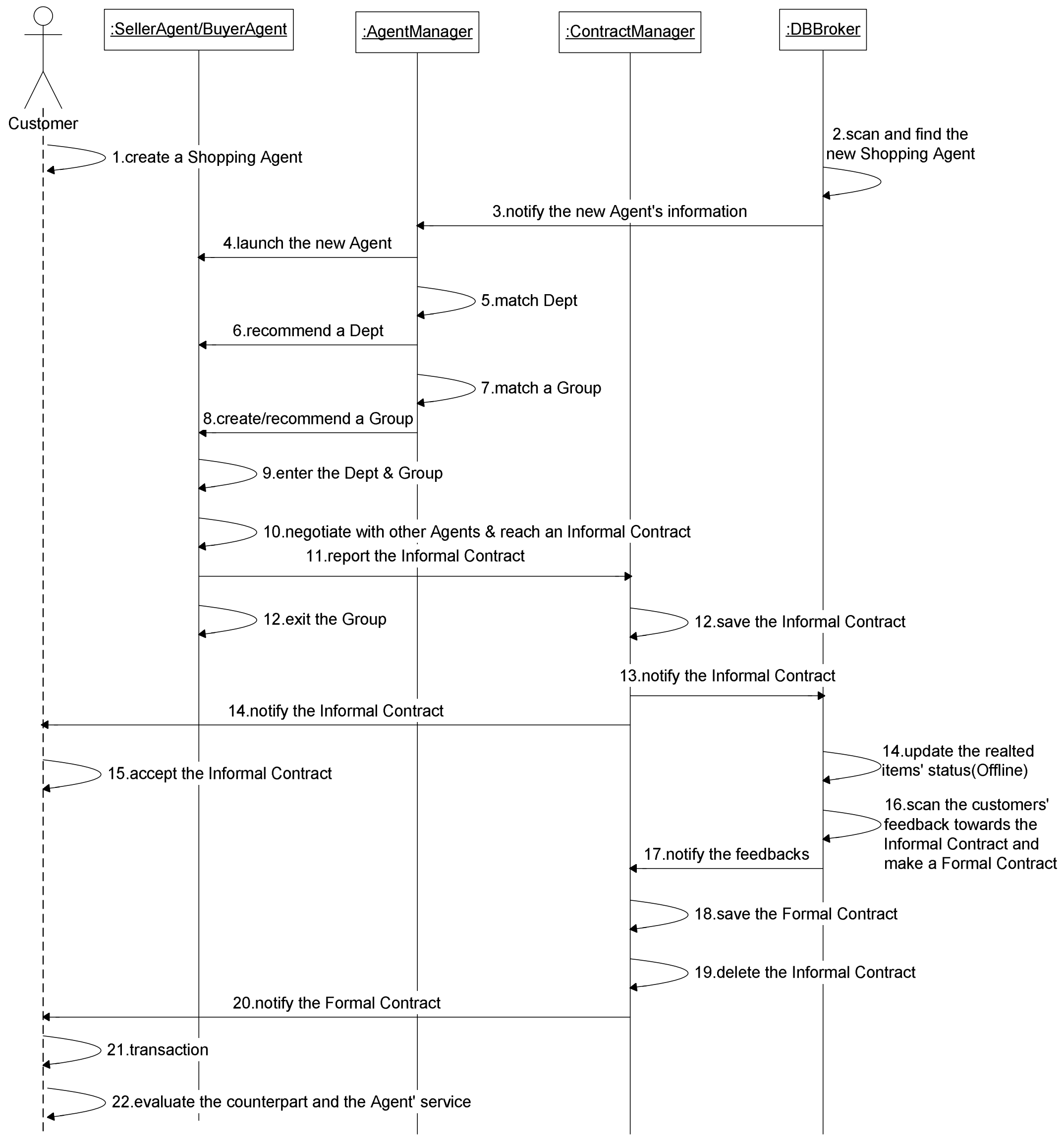

DBBroker
ContractManager
BuyerAgent
"report_new_informal_contract"
"report_contract_buyer_agent"+ 'buyerAgentID"
"report_contract_seller_agent"+ 'sellerAgentID"
"report_deal_price"+ 'dealPrice"
save the new Informal Contract into DB
scan customers' feedback towards the Informal Contract
"find_new_feedback_for_informal_contract"+'informalContractId'

Fig.25 Communication Protocol & Message Format
(DB Broker --- Contact Manager – Buyer Agent)
1. Get a new Informal Contract 2. Notify customers' feedback for the Informal Contract

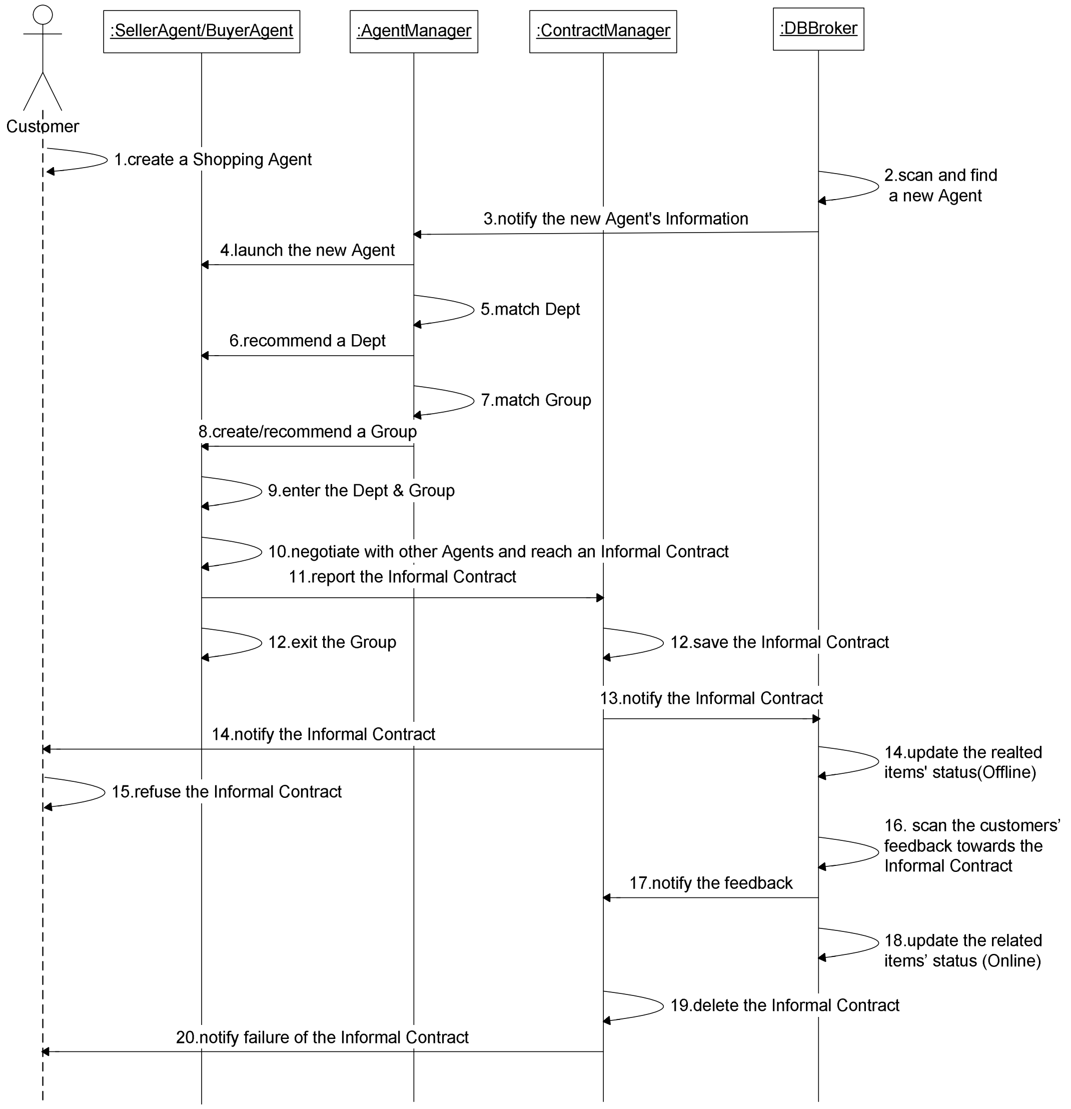
Fig.26 represents the complete scenario of achieving a Formal Contract.



Customer
:SellerAgent/BuyerAgent
:AgentManager
:ContractManager
:DBBroker
1.create a Shopping Agent
2.scan and find the new Shopping Agent
3.notify the new Agent's information
4.launch the new Agent
5.match Dept
6.recommend a Dept
7.match a Group
8.create/recommend a Group
9.enter the Dept & Group
10.negotiate with other Agents & reach an Informal Contract
11.report the Informal Contract
12.exit the Group
12.save the Informal Contract
13.notify the Informal Contract
14.notify the Informal Contract
14.update the realted items' status(Offline)
15.accept the Informal Contract
16.scan the customers' feedback towards the Informal Contract and make a Formal Contract
17.notify the feedbacks
18.save the Formal Contract
19.delete the Informal Contract
20.notify the Formal Contract
21.transaction
22.evaluate the counterpart and the Agent' service

Fig.26 Sequence Diagram of Negotiation Process
(Scenario 1: Get a successful Formal Contract)

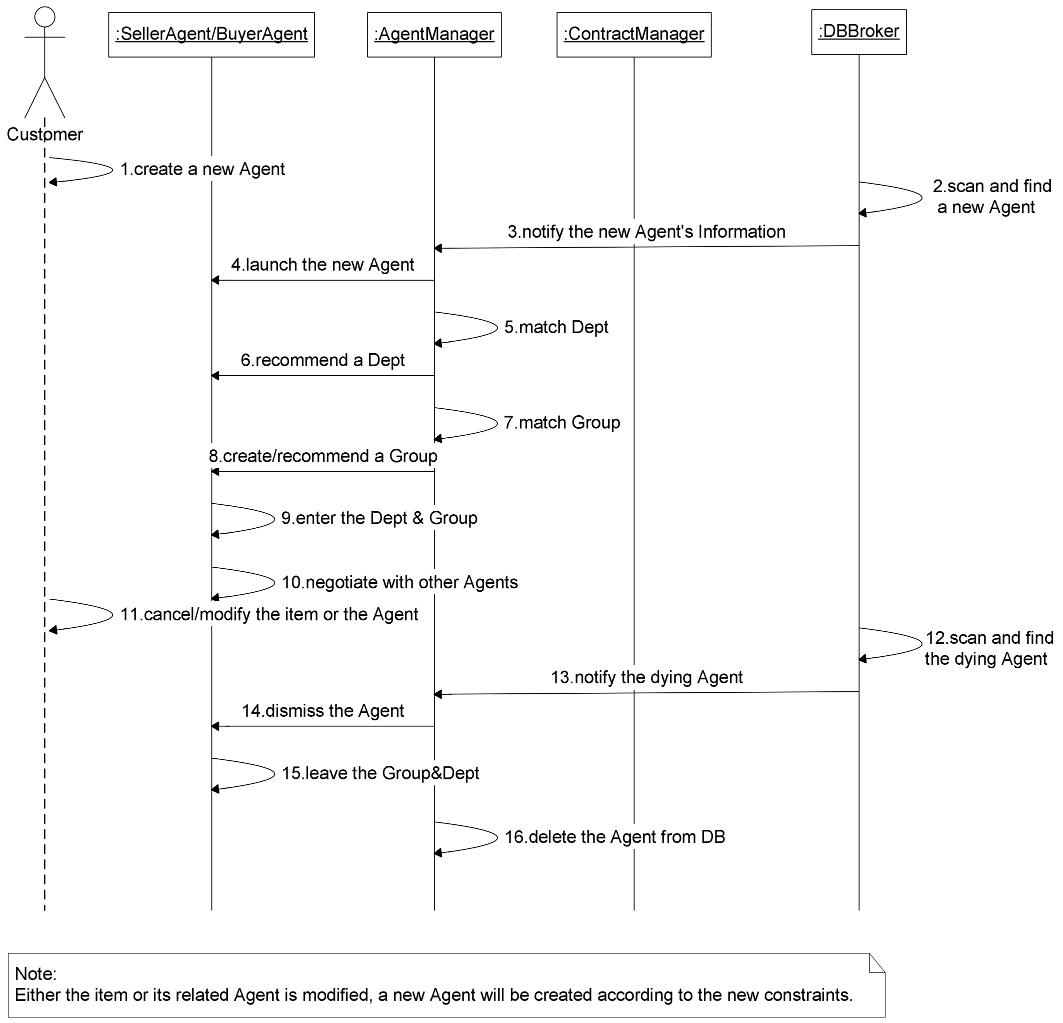
Fig.27 represents the complete scenario of achieving a successful Informal Contract by Shopping Agents, but refused by one or both agent users.



:SellerAgent/BuyerAgent
:AgentManager
:ContractManager
:DBBroker
Customer
1.create a Shopping Agent
2.scan and find a new Agent
3.notify the new Agent's Information
4.launch the new Agent
5.match Dept
6.recommend a Dept
7.match Group
8.create/recommend a Group
9.enter the Dept & Group
10.negotiate with other Agents and reach an Informal Contract
11.report the Informal Contract
12.exit the Group
12.save the Informal Contract
13.notify the Informal Contract
14.notify the Informal Contract
14.update the realted items' status(Offline)
15.refuse the Informal Contract
16. scan the customers' feedback towards the Informal Contract
17.notify the feedback
18.update the related items' status (Online)
19.delete the Informal Contract
20.notify failure of the Informal Contract

Fig.27 Sequence Diagram of Negotiation Process
(Scenario 2: Get successful Informal Contract, but refused by customers)

The scenario in Fig.28 describes the scenario that the customer cancels or modifies the selling/buying request or agent delegation constraints. In any of the above cases, the related active Shopping Agents will be noted as "dying" in DB. After DB Broker finds out the new "dying" agents by periodic scanning, it will notify Agent Manager to terminate the "dying" agents in AANS.



:SellerAgent/BuyerAgent
:AgentManager
:ContractManager
:DBBroker
Customer
1.create a new Agent
2.scan and find a new Agent
3.notify the new Agent's Information
4.launch the new Agent
5.match Dept
6.recommend a Dept
7.match Group
8.create/recommend a Group
9.enter the Dept & Group
10.negotiate with other Agents
11.cancel/modify the item or the Agent
12.scan and find the dying Agent
13.notify the dying Agent
14.dismiss the Agent
15.leave the Group&Dept
16.delete the Agent from DB
Note:
Either the item or its related Agent is modified, a new Agent will be created according to the new constraints.

Fig.28 Sequence Diagram of Negotiation Process
(Scenario 3: Customer cancels or modifies agents in the midway of negotiation)

(5) Class Diagram

The Class Diagram shows classes and their relationships. Classes define the types of objects that exist within the system. Classes can have attributes which are primitive data members of objects and operations which define functions that can be performed on the object. The visibility of attributes and operations to other objects can be defined as their signatures including types, default values, parameters, parameter types and return types.

Associations between classes show what links can exist between objects and define constraints on those links including the relative quantity of instances linked by an association. Class Diagrams can also show packages which group classes, dependencies between classes and dependencies between the packages which contain them.

After design and analysis of the system structure and typical scenarios, we develop the class diagrams for the traditional e-marketplace sub-system and AANS respectively, shown in Fig.29 and Fig.30.

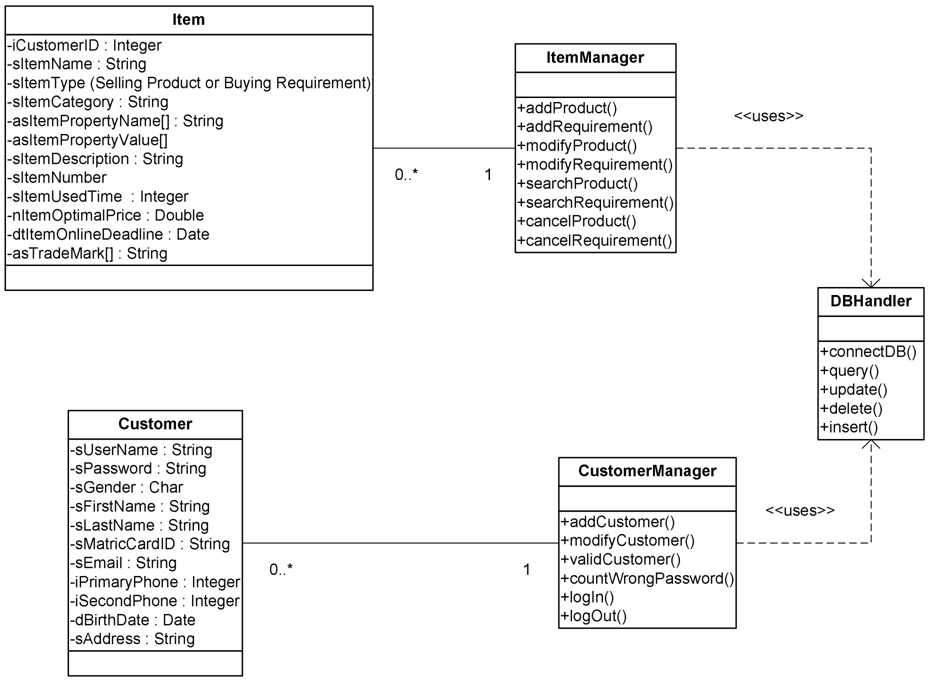


Fig.29. Class Diagram of E-Marketplace System

Please note that in Fig.29 Class Diagram of E-Marketplace System, Class Item represents Selling Products and Buying Requests, attribute sItemType can identify them. nOptimalPrice is the maximum selling price for Selling Products, or the minimum purchase price for Buying Request. For Selling Products, sItemUsedTime is the time the selling product has been used when on sale, and for Buying Requests, it is the maximum used time of the products that the buyer can accept. Trademark attribute indicates the specific brand of Selling Products and the optional brands (may be more than one) specified by Buying Requests.

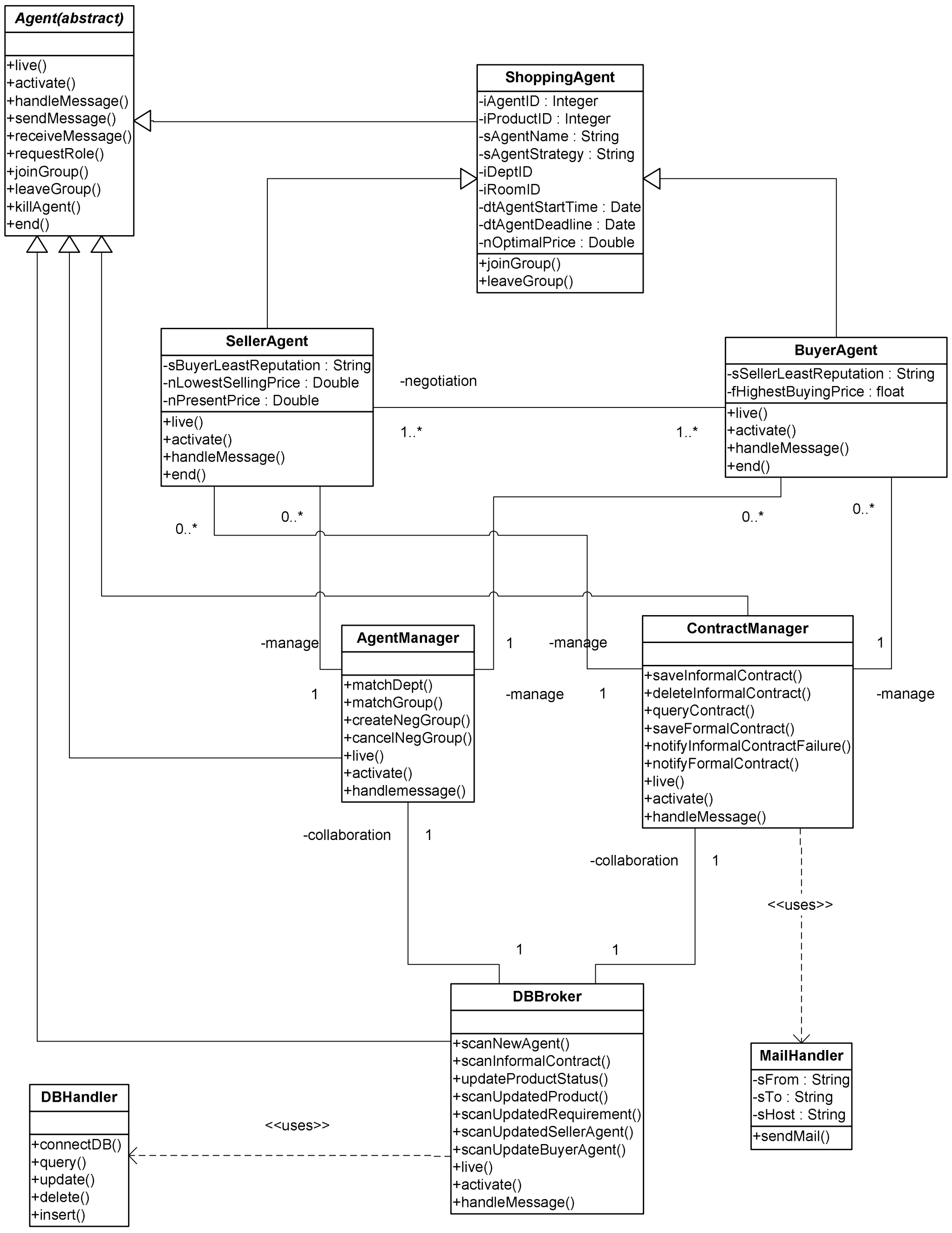


Fig.30 Class Diagram of Agent-Based Automated Negotiation System

Class Diagram Fig.30 represents that most classes in AANS are designed based on actors' responsibilities, i.e. responsibility-oriented. The actor classes include Class Seller Agent, Class Buyer Agent, Class Agent Manager, Class DB Broker and Class Contract Manger. Only two accessorial classes - Class DB Handler and Class Mail

Handler are created as system assistant classes. Although these two classes do not generate from the actors of the AANS, but also implement specific tasks and take specific responsibilities. Therefore, the class diagrams in AANS is typically responsibility-oriented rather than response-oriented, which fully applies the OOD(Object Oriented Design) methodology.

4.2.2 Network Cluster Model

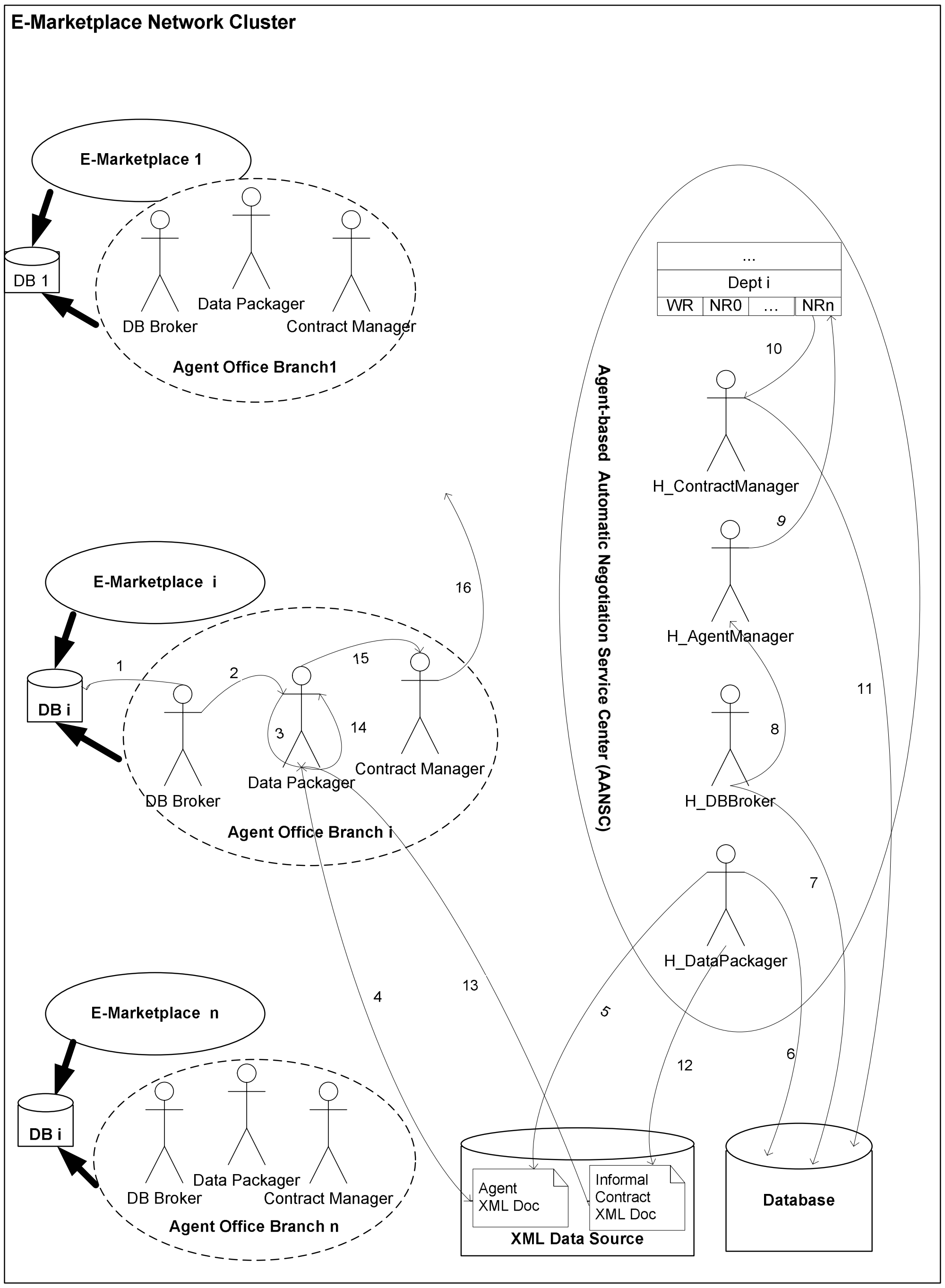
4.2.2.1 Requirement Analysis

In SHCM, the differences between LSM and NCM are not in terms of their functionalities, but in terms of the system structures and working models. It implies that the differences are almost invisible to normal customers. Customers still logon to the e-marketplace they register to sell or buy stuff and then create assistant Shopping Agents. They will not be aware (rather, need not be aware) of the model for AANS and where their Shopping Agents locate. The only difference customers can feel is the range of available trading peers. In a LSM system, available trading peers of a customer are only from the same e-marketplace. But in a NCM system, available trading peers may be the members of any e-marketplace within the e-marketplace cluster. Therefore, the business requirements of LSM and NCM are almost identical and will not be particularly analyzed again in this thesis.

4.2.2.2 System Design & Modeling

(1) System Architecture

As shown in Fig.31, compared with LSM, we abstract out the plug-in modular, AANS, from every individual e-marketplace and build it as a centralized AANSC in NCM. This model provides a light-weight ANS solution for e-marketplaces. Any e-marketplace that has computer product categories can easily add this new advanced service to its system by joining this cluster. Furthermore, no Shopping Agents will run in e-marketplaces, which reduces system consumption for negotiation computing. Only a simple AOB sub-system needs to be plugged into the heritage e-marketplace to communicate with AANSC. In a nut-shell, the whole cluster consists of several traditional e-marketplace sub-systems with several AOB sub-systems and one AANSC system.



1. scan new agent
2. notify new agent
3. pack agent data into XML doc
4. transmit agent XML doc
5. receive and unpack agent XML doc
6. save agent data into AANSC DB
7. scan new agent
8. notify new agent
9. launch new agent
10. notify new Informal Contract
11. save new contract
12. pack contract data into XML Doc
13. transmit back contract XML doc
14. receive, unpack and save contract doc
15. inform new Informal Contract
16. handle contract with customers

Fig.31 Architecture and Working Flow of Network Cluster Model

The working flow among agents of NCM also is demonstrated in Fig.31. Similar with LSM, DB Broker of every ABO, one of the Management Agents, will periodically scan branch DBs for searching new Shopping Agent records. Once found, DB Broker messages Data Packager who will then consolidate all information of the Shopping Agent into a standard XML document. Then the agent-related XML document is transmitted to ANNSC through socket communication. After arriving to AANSC, the agent-related XML document will be unpacked and parsed by the Data Packager of AANSC and the useful information will be saved into the DB of AANSC. Then, the DB Broker of AANSC will discover the new Shopping Agent record a while later when it scans the DB of AANSC and will inform Agent Manager there to launch the new Shopping Agent. The rest processes of agent matching, negotiating, and contract making are all identical with those in LSM. Once any new Informal Contracts are achieved, the contract relevant information will be saved into the DB of AANSC.

Unlike LSM, Contract Manager of NCM in AANSC will not take the responsibility of notifying participating customers about Informal Contracts. Instead, Data Packager packs Informal Contracts into standard XML documents and transmits them back to the corresponding AOBs through socket communication. Data Packager in the AOBs will receive Informal Contract XML documents, parse them and finally saves Informal Contract related information into the branch e-marketplace DB. The contract-related notification work will be conducted by Contract Manager in every AOB and the rest communication procedure with customers is identical with that in LSM.

(2) Use Case Diagram

Because the traditional e-marketplace system in NCM is the same with that in LSM, we do not demonstrate the use case diagram of this sub-system again. We will only be presenting the use case diagrams of AOB and AANSC.

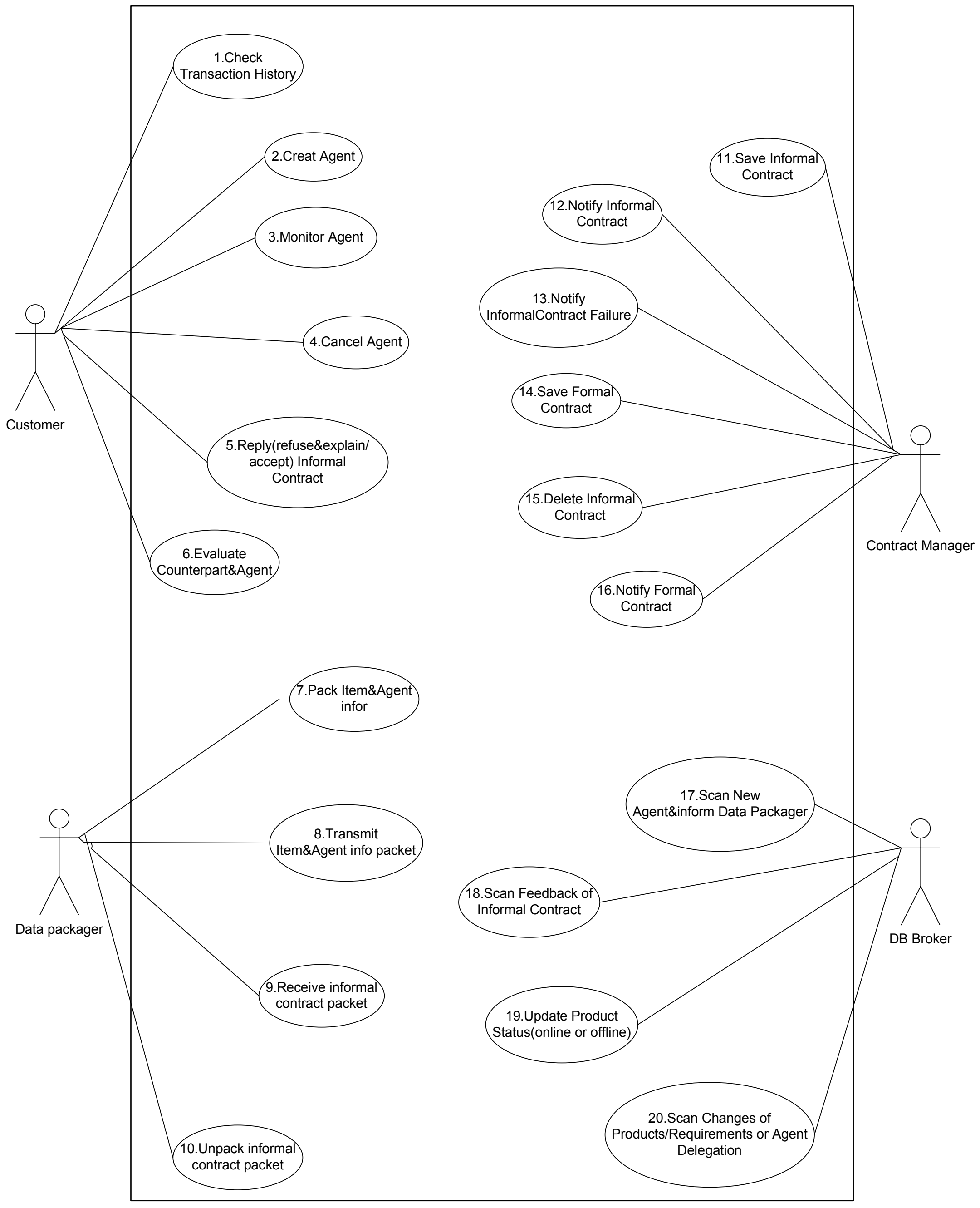

1.Check Transaction History
2.Creat Agent
3.Monitor Agent
4.Cancel Agent
5.Reply(refuse&explain/ accept) Informal Contract
6.Evaluate Counterpart&Agent
Customer
11.Save Informal Contract
12.Notify Informal Contract
13.Notify InformalContract Failure
14.Save Formal Contract
15.Delete Informal Contract
16.Notify Formal Contract
Contract Manager
7.Pack Item&Agent infor
8.Transmit Item&Agent info packet
9.Receive informal contract packet
10.Unpack informal contract packet
Data packager
17.Scan New Agent&inform Data Packager
18.Scan Feedback of Informal Contract
19.Update Product Status(online or offline)
20.Scan Changes of Products/Requirements or Agent Delegation
DB Broker

Fig.32 Use Case Diagram of the Agent Office Branch Sub-System in E-Marketplace

For AOB sub-system in every distributed e-marketplace, four Actors and twenty Use Cases are defined as shown in Fig.32. Except for Actor Customer stands for e-marketplace customers, all other Actors stand for software agents, including Actor DB Broker, Actor Contract Manager and Actor Data Packager. We can see in AOB, all agent Actors are Management Agents, but not any Shopping Agent Actors. It is consistent with the previous claim that no Shopping Agents run at local e-marketplaces. The Management Agents take the responsibility of observing and packing new agent data into standardized XML documents, sending the XML documents to AANSC, receiving Informal Contract XML documents and seeking relevant customers’ attitudes.

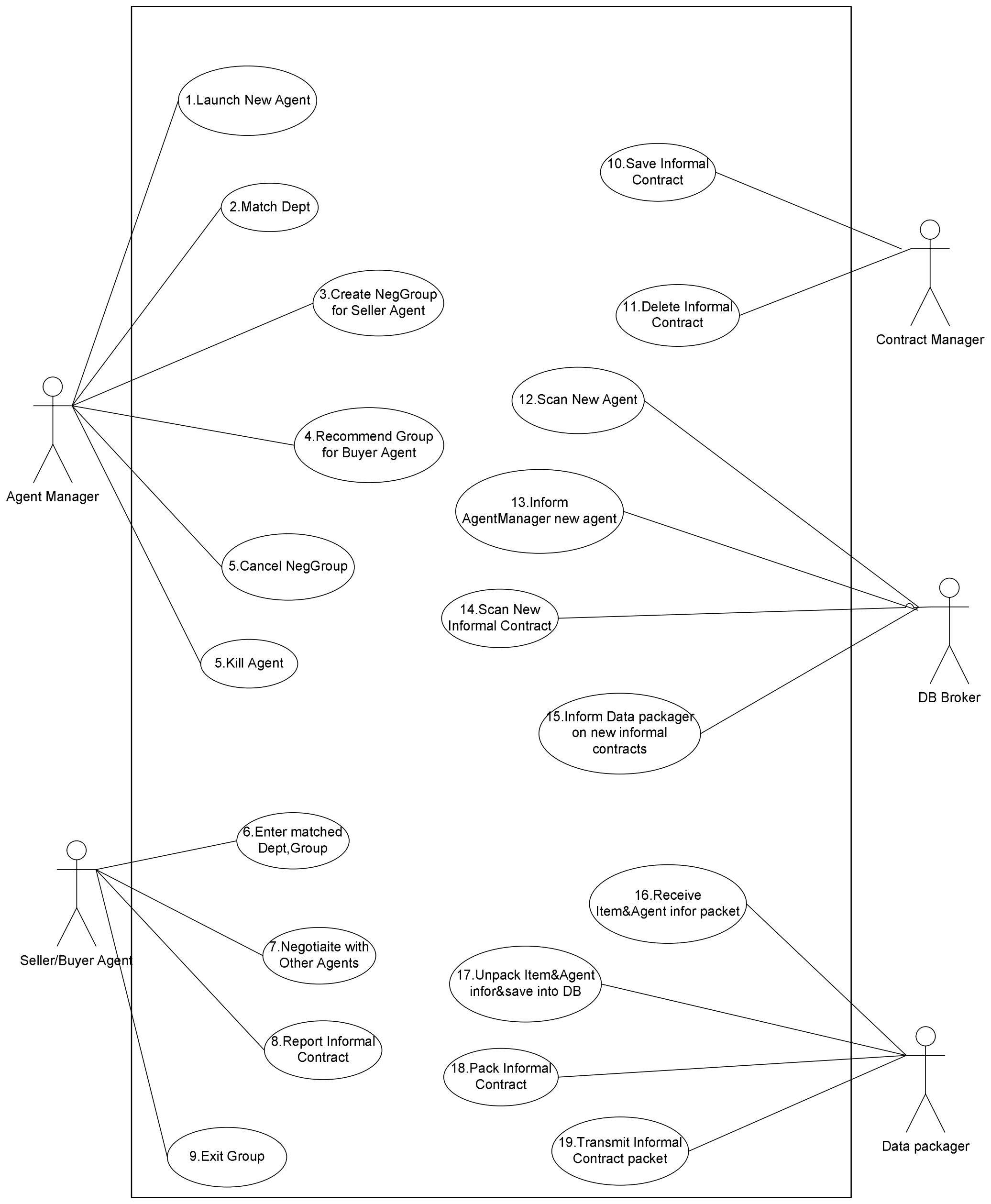

1.Launch New Agent
2.Match Dept
3.Create NegGroup for Seller Agent
4.Recommend Group for Buyer Agent
5.Cancel NegGroup
5.Kill Agent
Agent Manager
10.Save Informal Contract
11.Delete Informal Contract
Contract Manager
12.Scan New Agent
13.Inform AgentManager new agent
14.Scan New Informal Contract
15.Inform Data packager on new informal contracts
DB Broker
6.Enter matched Dept,Group
7.Negotiaite with Other Agents
8.Report Informal Contract
9.Exit Group
Seller/Buyer Agent
16.Receive Item&Agent infor packet
17.Unpack Item&Agent infor&save into DB
18.Pack Informal Contract
19.Transmit Informal Contract packet
Data packager

Fig.33 Use Case Diagram of Agent-Based Automated Negotiation Service Center

AANSC is shown as Fig.33. All six Actors designed in it are software agent Actors, including Actor Seller Agent, Actor Buyer Agent, Actor Agent Manager, Actor DB Broker, Actor Contract Manager and Actor Data Packager. The automated negotiation will be organized and conducted by these Actors in AANSC. The Use Cases of each actor are similar with the parallel ones in LSM and only Actor Data Packager is a new member of NCM, which is corresponding to Data Packager Agent. Further, ANS will not take charge of interacting with end-customers but focuses on the core business of negotiation processing.

(3) Sequence Diagrams

In this part, we only demonstrate the interaction and message exchange among intelligent agents within AOB, within AANSC, and between AOB and AANSC. Because messaging communication is the major activities in the AANS during the trading, we only focus on the sequence diagrams that describe the specific interaction processes among these agents. Fig.34, 35, 36 present three typical scenarios happened in this course.

Fig.34 presents Scenario 1, which happens after a new shopping agent is created in the e-marketplace branch. DB Broker of AOB finds it when scanning DB and then DB Broker informs Data Packager within the same branch about the new agent. Then, Data Packager will pack the shopping agent data into a standardized XML document and transmits it to the remote AANSC, indicated as headquarter (HQ) in diagrams. The remote Data Packager of AANSC will receive the coming XML document, unpack and parse the useful information for next-step processing.

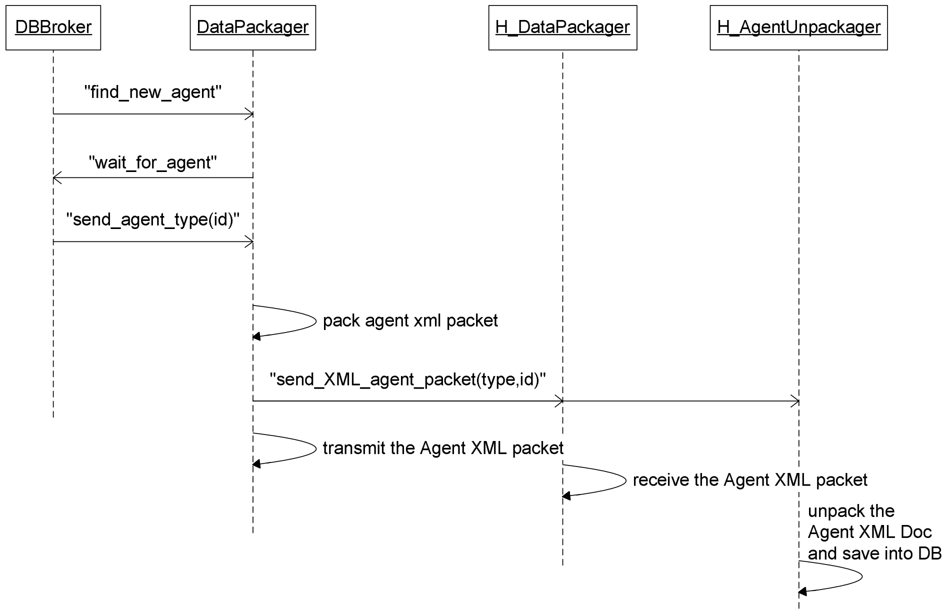


Fig.34 Sequence Diagram of Agent Office Branch of the Network Cluster Model
(Scenario 1: Branch--HQ management collaboration for scanning and delivering new agents)

Fig.35 presents the second scenario that takes place within AANSC when a new Shopping Agent comes. DB Broker of AANSC will find the new Shopping Agent records and message Agent Manager. Then, Agent Manager leads the Shopping Agent into a proper department based on its category attribute. For a Buyer Agent, a new Negotiation Group will be created specially for it. For a Seller Agent, Agent Manager will match the suitable Negotiation Groups or Waiting Group for it. After registration, the new Shopping Agents will commence to conduct negotiation activities within the matched Negotiation Groups.

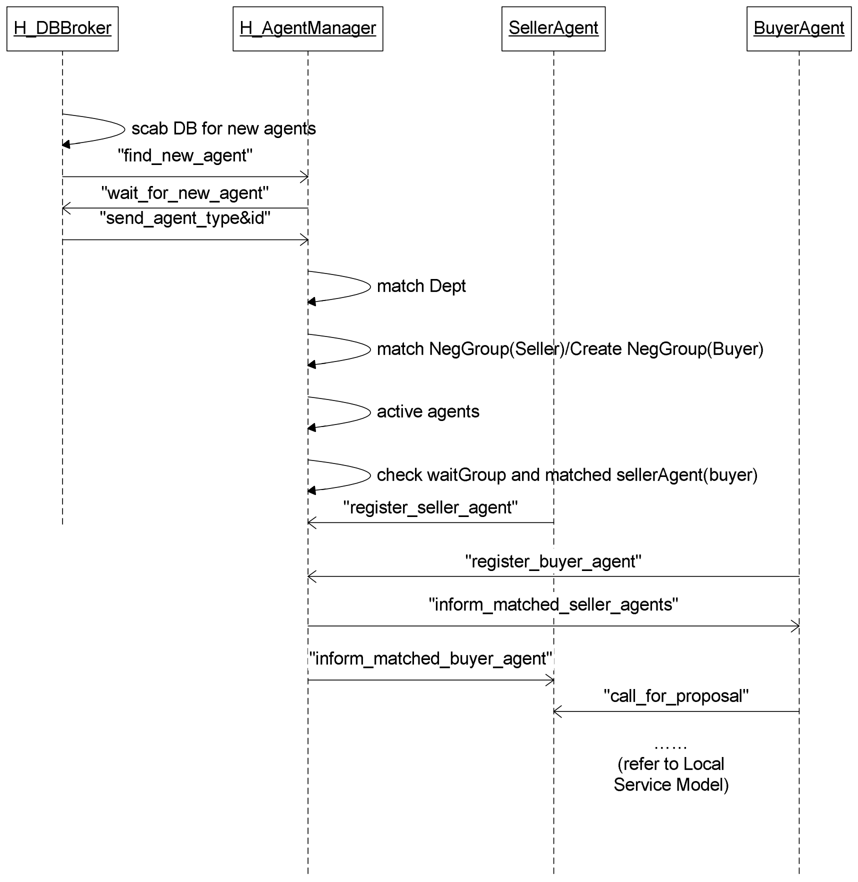


Fig.35 Sequence Diagram of Agent Office Branch of Network Cluster Model
(Scenario 2: HQ management collaboration upon a new agent's coming)

Fig.36 presents scenario 3 that an Informal Contract is achieved in AANSC and is sent back to the related e-marketplace branches. Once a successful Informal Contract is completed by AANSC, Contract Manager in AANSC will save all the relevant data into AANSC DB and pack the data into a standardized XML document. The Informal Contract XML document is then sent back to the branch e-marketplaces involved in this contract. Data Packager in branches will unpack the received Informal Contract XML document and save it into the branch DB. As found by DB Broker of the branch, the information about the successful Informal Contract will be soon sent to the participating customers by emails and the branch e-marketplace will consequently process the feedbacks from end-customers. Whether a Formal Contract can be achieved or not depends on the attitudes of the both trading customers.

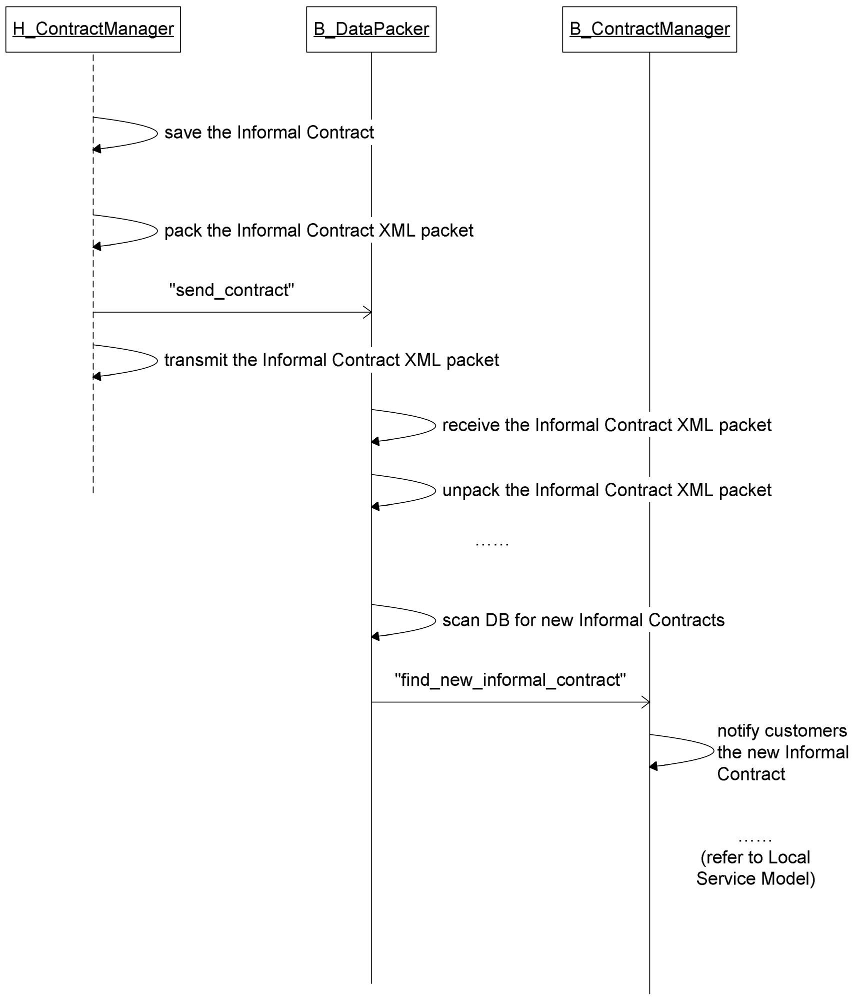

H_ContractManager
B_DataPacker
B_ContractManager
save the Informal Contract
pack the Informal Contract XML packet
"send_contract"
transmit the Informal Contract XML packet
receive the Informal Contract XML packet
unpack the Informal Contract XML packet
……
scan DB for new Informal Contracts
"find_new_informal_contract"
notify customers the new Informal Contract
…… (refer to Local Service Model)

Fig.36 Sequence Diagram of Agent Office Branch of Network Cluster Model
(Scenario 3: Branch--HQ management collaboration for new Informal Contract transport)

(4) Class Diagrams

The Class Diagram for the traditional e-Marketplace sub-system in NCM resembles that in LSM, because similar functions are provided in the e-Marketplace sub-system for both two models. Thus, we do not describe it again, only focusing on the Class Diagrams of negotiation related sub-systems.

As shown in Fig.37, Class Diagram of Agent Branch Sub-System in E-Marketplace, there are three major classes, Class DataPackager, Class DBBroker, and Class ContractManager, which are particularly designed for Actor Data Packager, Actor DB Broker and Actor Contract Manager in this sub-system (please see Use Case Diagrams). These three major classes all inherited from Class Agent that exists in the Madkit platform kernel and take the responsibilities of the three actors respectively.

To facilitate the agent classes, we also design some assistant classes, including Class SellerAgentPacker, Class BuyerAgentPacker, Class BranchTransmitter, Class BranchReceiver, Class B_ContractUnpacker, Class SAXParser, Class DBHandler and Class MailHandler. We can see that five out of eight assistant classes in this diagram are used by Class DataPackager, which indicates the critical role of Actor Data Packager in AOB. In fact, the primary task of this sub-system is to pack and unpack standardized XML documents, and to transmit and receive them between the local system and the central negotiation center.

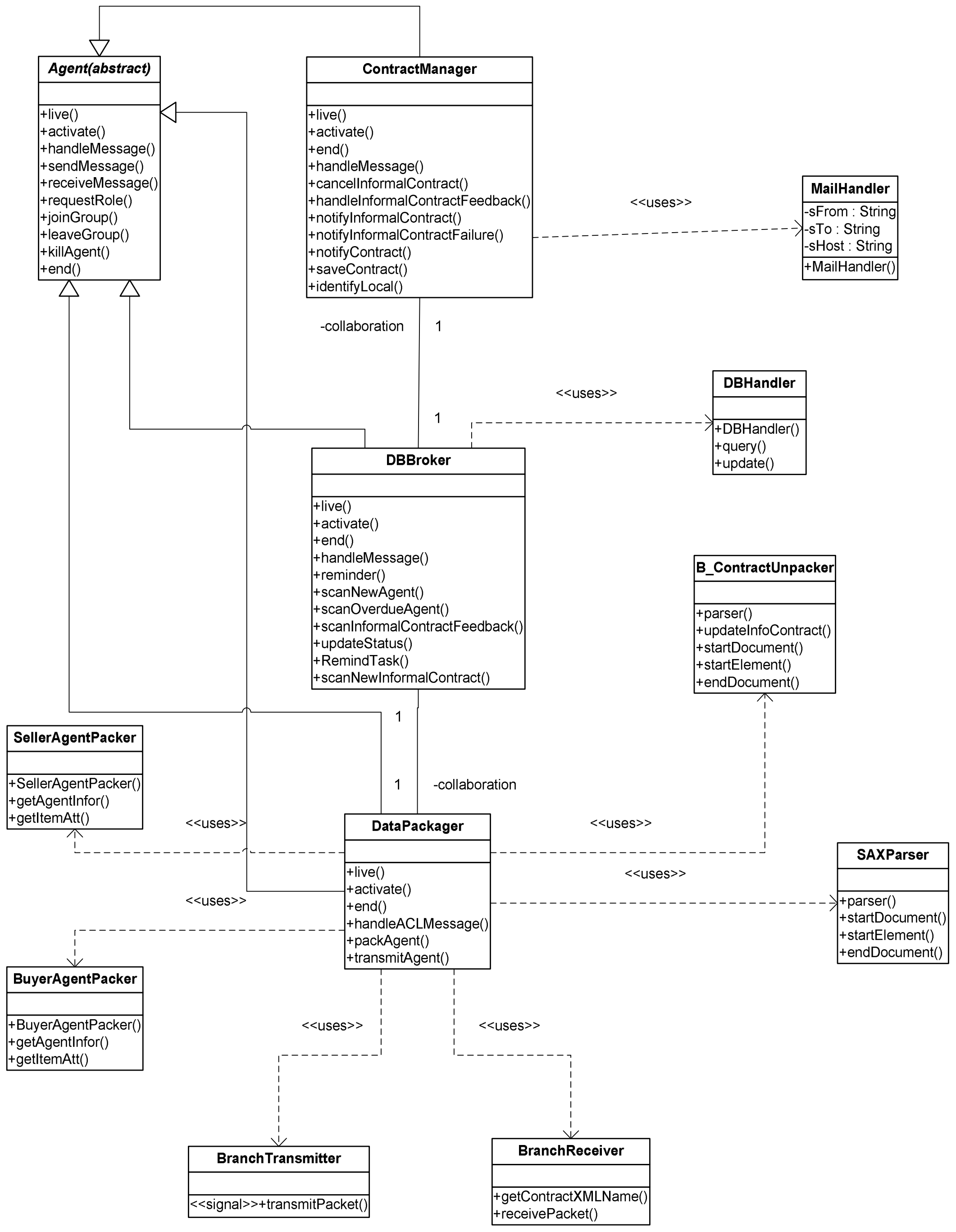


Fig.37 Class Diagram of Agent Office Branch Sub-System in E-Marketplace

In Class SellerAgentPacker and Class BuyerAgentPacker, the XML document schemes and formats for Shopping Agent data and contract data are carefully designed and strictly constrained. Required information on negotiation processes and contract making will be packed into the XML documents in the unified format that all members in this network cluster agree on. Class BranchTransmitter and Class BranchReceiver are the communicators in this sub-system, helping Class DataPackager send and receive XML documents.

To coordinate transmitting XML documents, we do not use socket, but use the internal messaging system and Agent/Role/Group Model of the Madkit platform.

The messaging systems in the Madkit platform are defined by inheritance from a generic Message class. Thus specific messages can be defined for intra-group communication, and allows a group to have its specific communication attributes. Messages receivers and senders are identified by their AgentAddress. MadKit do not define interaction mechanism, which can be defined on an ad-hoc basis, or built in a specific agent model library. Therefore, the coordination between AOB and AANSC can be supported by the intra-group communication mechanism provided by Madkit.

We just simply register the Management Agents in AOB and AANSC in the same Group, Office_Staff, and assign distinct Roles to these Management Agents. Once the distributed Madkit kernels are synchronized before starting the whole system, the Management Agents in Group ‘Office-Staff’ will automatically be visible to any peers within this Group. Therefore, they can send and receive message to their peers by identifying the individual AgentAddress, no need of the socket connection process.

For XML documents transmission, the messaging mechanism of Madkit is not powerful and robust enough to transmit a mass of XML documents carrying all important information for AANS. But the Socket communication supported by Java networking computing provides us a standard and widely used approach. The following part will give some brief introduction about socket communication.

Though URLs and URLConnections provide a relatively high-level mechanism for accessing resources on the Internet, sometimes our programs require lower-level network communication, for example, when we want to write a client-server application. In client-server applications, the server provides some service, such as processing database queries or sending out current offer prices. The client uses the service provided by the server, either displaying database query results to the user or making purchase recommendations to an investor. The communication that occurs between the client and the server must be reliable. That is, no data can be dropped and it must arrive on the client side in the same order in which the server sent it.

TCP provides a reliable, point-to-point communication channel to those Client/Server applications on the Internet to communicate with each other. To communicate over TCP, a client program and a server program establish a connection to one another. Each program binds a socket to its end of the connection. To communicate, the client and the server each reads from and writes to the socket bound to the connection.

A socket is one end-point of a two-way communication link between two programs running on the network. Socket classes are used to represent the connection between a client program and a server program. The java.net package provides two classes, Class Socket and Class ServerSocket, which implement the client side of the connection and the server side of the connection, respectively.

Thus, we design two classes running at AOB and AANSC to implement the connection between them. When sending files from AOB to AANSC, AOB is the client side and NAASC is the server side. Conversely, when sending files from NAASC to AOB, NAASC is the client side and AOB is the server side. Once Management Agents at both ends have been ready for delivering XML documents, the socket communication programs will get to connect and implement the transmission work.

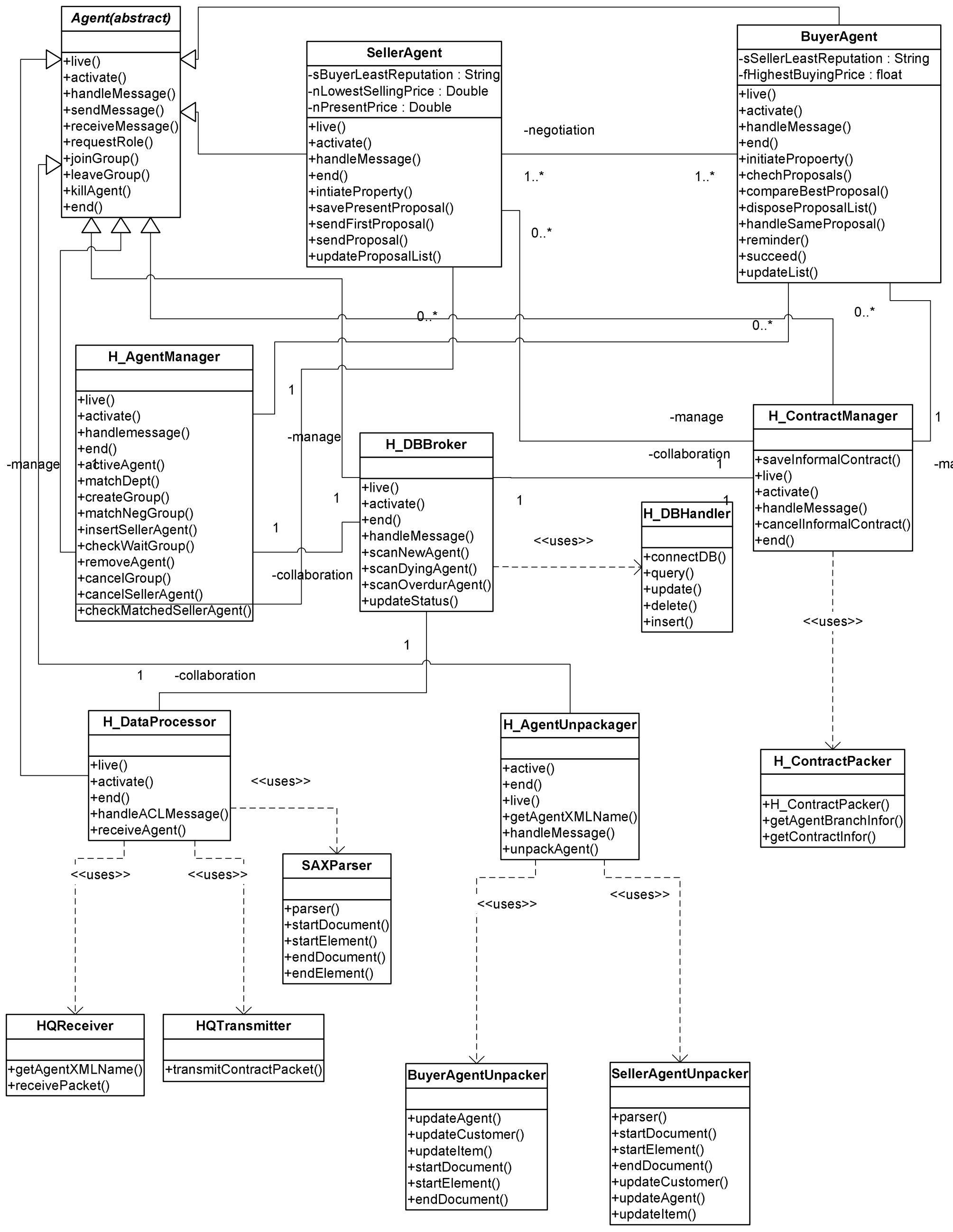


Fig.38 Class Diagram of the Agent-Based Automated Negotiation Service Center

As shown in Fig.38, the Class Diagram of AANSC presents some agent classes we can see in the Agent-Based Negotiation Sub-System of LSM, including Class SellerAgent, Class BuyerAgent, Class AgentManager, Class DBBroker and Class ContractManager. There are also some agents specially designed for NCM, primarily for processing XML documents. The key class is the Class H_DataProcessor and Class

H_AgentUnpackager. The Class H_DataProcessor together with its assistant classes, Class SAXParser, Class HQTansmitter and Class HQReceiver, take the responsibility of transmitting and receiving XML documents by socket. Class H_AgentUnpackager together with assistant classes, BuyerAgentUnpacker and Class SellerAgentUnpacker, unpack all the coming XML documents of Shopping Agent data and save them into the Central DB. Further, an assistant class also is created for Class H_ContractManager and Class H_ContractPacker, which specially packs the Informal Contract data into the standardized XML documents. And then, Class HQTansmitter will send them back to the corresponding AOBs.

4.3 Tests and Results

The software systems for these two models, LSM and NCM, are both implemented and we have run the systems in all of the scenarios defined in 4.2.1.2 and 4.2.2.2. All the test cases, including boundary and stress test cases, unit test cases, integration test cases and final acceptance test cases are created in the design phase and fully executed after completing unit implementation, unit integration and system integration, respectively. Unit test cases are defined based on the system architecture design, use case diagrams and class diagrams. Boundary and stress test cases, integration test cases and final acceptance test cases are defined based on the software requirement analysis and the scenarios specified in sequence diagrams.

The testing results from LSM are largely as expected and the whole system works very stable. ‘Stable’ means the system, including the traditional e-marketplace and the plug-in AANS, can be independently launched correctly and works well along with each other by sharing DB. ‘Stable’ also means tens of Shopping Agents can be simultaneously activated in the MAS, conduct negotiation for their users' interests following the communication protocol predefined and reach qualified Informal Contracts for their users. Other normal functions, such as user account management, search engine, etc., all can work properly in tests.

On the other hand, the results from NCM are also encouraging although they are not as good as those from LSM. In most cases, agents can work well together and achieve right contracts if the total number of agents is not more than twenty around and the number of the branch e-marketplaces within the cluster is not more than three. Once more e-marketplaces join the cluster or more agents get activated in AANSC, some unexpected errors may happen. The most frequently happened error is that the Management Agents in the e-marketplaces that newly join the cluster can not smoothly interact with the Management Agents in AANSC, such as losing messages on the way. These unexpected errors can be interpreted by the previous analysis on NCM in 3.2. The primary reason for these unstable problems is that agents in e-marketplaces and AANSC need to collaborate and communicate remotely within the cluster, which increases the requirement for agents' interoperability and synchronization. Thus, our agent programs need further optimization. On the other hand, better support for agents' interoperability and synchronization is also expected from the agent infrastructure platform, Madkit, whose new version is on the way already.

Chapter 5 Conclusions & Future Works

This thesis has described two models for building an Agent-Based Automated Negotiation System (AANS) for e-marketplaces, Local Service Model (LSM) and Network Cluster Model (NCM), where agents buy and sell goods and negotiate with interested parties in behalf of their users. In LSM, AANS belongs to the e-marketplace. Though the negotiation system can not be shared with other similar e-trading systems, it is independently plugged into the e-marketplace as an optional service component. Therefore, the modularized design makes this advanced service system easy to be assembled and dismantled to any similar e-marketplace as an additional module with little integration effort.

After demonstrating the two models for supporting the agent-based negotiation service, we described the design of the multi-agent organization for the negotiation system, agent communication protocols, system openness, infrastructure services, negotiation strategies, negotiation algorithms and some other issues.

Firstly, we extend Flat Organization to Complex Flat Organization for the open MAS, which are supported by an agent location mechanism. Agents running in it are divided into two categories, Management Agents and Shopping Agents. Management Agents are ‘full-time staff’, which should be active if only the MAS is running, of our MAS and dedicate to maintain the system and administrate Shopping Agents within it. On the other hand, Shopping Agents, who are only active for specific sales or purchase delegations, have limited working life, and are assigned to various departments and groups to conduct negotiation. Shopping Agents are under the control of Management

Agents and can not freely communicate with other Shopping Agents in the negotiation system. To any Shopping Agent, only the counterparty Shopping Agents matched by the Agent Manager are accessible. That means a Seller Agent only can negotiate with matched Buyer Agents in the same Negotiation Group and a Buyer Agent only can negotiate with matched Seller Agents in the same Department. This organization design makes a tradeoff between the openness and dynamism of Flat Organization and the low communication cost of Hierarchical Organization.

Secondly, considering the mixed characters of Complex Flat Organization, we designed a mixed Symmetry & Asymmetry Communication Model, using connection-oriented peer-to-peer and broadcast/multicast communication approaches. When the negotiation system is started and connected to the network, all Management Agents can symmetrically communicate with each other by means of peer-to-peer; On the other hand, Buyer Agents asymmetrically communicate with Seller Agents in the same Negotiation Groups, by means of peer-to-peer and multicasting.

Thirdly, the negotiation MAS is Half-Dynamic Openness. Management Agents are well determined in advance, but Shopping Agents are unpredictable and may enter or leave the MAS at any time. The infrastructure services are supported by the multi-agent platform Madkit (Multi-Agent Development Kit), a Java multi-agent platform. It provides general agent facilities, including lifecycle management, naming services, location mechanism, message passing, distribution, synchronization, etc, and allows high heterogeneity in agent architectures and communication languages, and various customizations.

Fourthly, to make the negotiation protocol and strategies scalable, we modularized the architecture of negotiable Shopping Agents by separating the negotiation implementation from negotiation protocol definition and strategy definition. We extended a Half-Bilateral Negotiation Model for the negotiation protocol, three optional negotiation strategies and respective algorithms.

Finally, we designed a prediction algorithm for negotiation, evaluating how far the negotiation is going towards the objective. Furthermore, we discussed a critical problem in e-commerce application, users' reputation, and designed a simple but effective reputation evaluation mechanism.

To implement the two models, NCM and LSM, we developed two different versions for SHCM. Especially for the collaboration mechanisms of the distributed servers in NCM, implementation using XML plays a key role in terms of collaboration and integration of the distributed servers in this model.

In fact, the two versions of SHCM are two separate systems, experienced two different system analysis, design, modeling, coding and testing processes, and can be used separately. For LSM of SHCM, we specifically described the system requirements of the traditional E-Marketplace Sub-Systems and the Agent-Based Automated Negotiation Sub-System, and demonstrated the system design of these two sub-systems by presenting Use Case Diagrams, Sequence Diagrams and Class Diagrams. For NCM of SHCM, we only focused on the Agent Office Branch Sub-System in e-marketplaces and the centralized Agent-Based Automated Negotiation Service Center (AANSC) , neglecting the system analysis and design of the traditional E-Marketplace

Sub-Systems which is similar with that in LSM. Also, we made use of Use Case Diagrams, Sequence Diagrams, and Class Diagrams to demonstrate the implementation details, especially explaining the collaboration mechanisms for the distributed servers in the network cluster. XML is the core technology in terms of communication and cooperation, since all important data are packed into the standardized XML documents and transmitted within the cluster.

Finally, we described and analyzed the testing results for these two software systems. The results from LSM are very satisfying in terms of the stable work performance. Also, the system in NCM model is workable but interoperability errors may happen as the population of agents and e-marketplaces increases.

However, our proposed architectures, organizations, protocols, algorithms, as well as the scenarios dedicated to the two models still have some disadvantages.

First of all, the agent communication protocol we used is not a standard and general protocol. The communication protocol we proposed in this thesis is tailored for the specific scenario of this project to reduce computing overhead and enhancing interaction efficiency. It is hardly to extend it to support multi-agent systems for other goods category without modification.

Then, because of the limited time, the negotiation strategies and algorithms designed in this thesis are still simple and raw, which is for easing implementation. The negotiation algorithm curves which are executed by the Shopping Agents are all the transformations of basic mathematic curves, including line, square and cube.

Also, SHCM only focuses on the key component of the whole system, AANS, and neglects many other issues which will help increase the trust of the trading platform, such as issues on security, legal, privacy and so on.

Next, our MAS is implemented on an excellent open source multi-agent platform-Madkit, making use of its given mechanisms, such as lifecycle management, naming services, location mechanism, message passing, distribution, synchronization, heterogeneity, etc. however, Madkit is not robust enough to support a large number of agents (typically in the order of hundreds or thousands) launched simultaneously, otherwise the kernel of Madkit will have a greater risk of collapsing under such an overloaded situations.

Finally, the wide application of our proposed negotiation system needs the industry support for a uniform agent platform and specification. There are several existing multi-agent platforms, but most are only used for research and simulation. It seems the uniform agent platform will not emerge in the very near future.

The solutions for the above-mentioned issues would be challenging for the deployment of software agent-mediated e-marketplace applications.

References

[1] C. Sierra and F. Dignum, "Agent-Mediated Electronic Commerce: Scientific and Technological Roadmap", Lecture Notes in Artificial Intelligence, Vol. 1991, Springer-Verlag, pp. 1-18 (2001)

[2] Chris Preist, "Algorithm Design for Agents Which Participate in Multiple Simultaneous Auctions", Trusted E-Services Laboratory, Hewlett-Packard Company, (2000)

[3] Alessio R. Lomuscio, Michael Wooldridge and Nicholas R. Jennings, "A Classification Scheme for Negotiation in Electronic Commerce", International Journal of Group Decision and Negotiation 12 (1) pp. 31-56 (2003)

[4] Alessio R. Lomuscio, Michael Wooldridge and Nicholas R. Jennings, "Automated Negotiation in Agent-Mediated Electronic Commerce", UK (Dec. 1999)

[5] Charles Petrie, "Agent-Based Software Engineering", Agent-Oriented Software Engineering: First International Workshop, AOSE 2000, pp. 59-76, Limerick, Ireland (Jun. 2000)

[6] http://www.madkit.org/

[7] http://www.chemconnect.com/

[8] http://www.orbitz.com/

[9] http://www.priceline.com/

[10] http://www.ebay.com/

[11] http://www.bn.com/

[12] M. Tsvetovovatyy and M. Gini, "Toward a Virtual Marketplace: Architectures and Strategies", Proceedings of the First International Conference on the Practical

Application of Intelligent Agents and Multi-Agent Technology, pp. 597-613, London, UK (1996)

[13] A. Chacez, D. Dreilinger, R. Guttman and P. Maes, “A Real-life Experiment in Creating an Agent Marketplace”, Software Agents and Soft Computing: Towards Enhancing Machine Intelligence: Concepts and Applications, pp. 160-182, Berlin, New York (1997)

[14] A. Chavez and P. Maes, “Kasbah: An Agent Marketplace for Buying and Selling Goods”, Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology, pp. 75090, London, UK (1996)

[15] Gioogos Zacharia, Alevandros Moukas, Robert Guttmen and Pattie Maes, “An Agent System for Comparative Shopping at the Point of Sale”, Technologies for the Information Society: Developments and Opportunities, Amsterdam: IOS Press (1998)

[16] Shim Yoon, Ju Young Yun, SooWoong Kim, “Jangter: A Novel Agent-Based Electronic Marketplace”, Agent Mediated Electronic Commerce II: Towards Next-Generation Agent-Based Electronic Commerce Systems, pp. 102-112, New York (2000)

[17] Simon Parsons, Carles Sierra, and Nick Jennings, “Agents that Reason and Negotiate by Arguing”, Journal of Logic and Computation, Vol. 8 Nos1-3, pp. 261-292 (1998)

[18] Maria Joao Viamonte and Carlos Ramos, “A Model for an Electronic Marketplace” Agent Mediated Electronic Commerce: the European agentlink perspective, Berlin, New York (2001)

[19] Noyda Matos and Carles Sierra, “Evolutionary Computing and Negotiating Agents”, Agent Mediated Electronic Commerce: First International Workshop on Agent Mediated Electronic Trading, AMET-98 Minneapolis, MN, USA (1998)

[20] Carles Sierra, Peyman Faratin and Nick R. Jennings, “A Service-Oriented Negotiation Model between Autonomous Agents”, Multi-Agent Rationality: 8th European Workshop on Modeling Autonomous Agents in a Multi-Agent World, MAAMAW'97, Ronneby, Sweden (1997)

[21] Onn Shehory, “Software Architecture Attributes of Multi-Agent Systems”, IBM Research Lab, Israel (2000)

[22] Jacques Ferber, Olivier Gutknecht and Fabien Michel, “MadKit Development Guide v3.1”, Madkit Project Team, France (2002)

[22] Olivier Gutknecht and Jacques Ferber, “The MADKIT Agent Platform Architecture”, Madkit Project Team, France (May 2000)

[23] Gustavo E. de Paula, Francisco S. Ramos and Geber L. Ramalho, “Bilateral Negotiation Model for Agent-Mediated Electronic Commerce”, Agent-Mediated Electronic Commerce III, pp. 1-14, New York (Jun. 2000)

[24] Agent Platform Special Interest Group of the Object Management Group (OMG), “Agent Technology Green Paper”, Available at http://www.objs.com/agent/index.html (Sep. 2000)

[25] http://www.cs.wright.edu/~mcox/COMAS/

[26] http://www.trl.ibm.com/aglets/index_e.htm/

[27] http://www.madkit.org/

[28] Todd Sundsted, “Agents on the move, Bolster Your Client Applications by Adding Agent Mobility”, Available at http://www.javaworld.com

[29] Bernhard Bauer, Jorg P. Muller and James Odell, “Agent UML: A Formalism for Specifying Multi-agent Software Systems”, AOSE (Agent-Based Software Engineering) 2000, pp. 45-58, Limerick, Ireland (Jun. 2000)

[30] James J.Odell, H.Van Dyke Parunak and Bernhard Bauer, “Represeting Agent Interaction Protocols in UML“, AOSE (Agent-based Software Engineering) 2000, pp. 45-58, Limerick, Ireland (Jun. 2000)

[31] Ralph Depke, Reiko Heckel, Jochen Malte Kuster, “Agent-Oriented Modeling with Graph Transformation”, AOSE (Agent-based Software Engineering) 2000, pp. 45-58, Limerick, Ireland (Jun. 2000)

[32] Jurgen Lind, “Issues in Agent-Oriented Software Engineering”, Agent-based Software Engineering (AOSE) 2000, pp. 45-58, Limerick, Ireland (Jun. 2000)

[33] Joakim Eriksson, Niclas Finne and Sverker Janson, “SICS Marektspace – An Agent-Based Market Infrastructure“, Agent Mediated Electronic Commerce: First International Workshop on Agent Mediated Electronic Trading, AMET-98 Minneapolis, pp. 41-53, MN, USA (1998)

[34] Amy R. Greenwald and Jeffrey O. Kephart, “Shopbots and Pricebots”, Agent Mediated Electronic Commerce II: Towards Next-Generation Agent-Based Electronic Commerce Systems, pp. 1-23, New York (2000)

[35] N. R. Jennings, P. Faratin, A. R. Lomuscio, S. Parsons, C. Sierra and M. Wooldridge, “Automated Negotiation: Prospects, Methods and Challenges”, International Journal of Group Decision and Negotiation 10 (2) pp. 199-215 (2001)

[36] H. Jurgen Muller, “Negotiation Principles”, Foundation of Distribution Artificial Intelligence, pp. 211-229, New York (1996)

[37] Martin Fowler and Kendall Scott, “UML Distilled, Applying the Standard Object Modeling Language”, Reading, MA, Addison-Wesley (1997)

[38] Robert H. Guttman and Pattie Maes, “Cooperatived vs. Competitive Multi-Agent Negotiations in Retails Electronic Commerce”, Cooperative Information Agents (CIA)

II: Learning, Mobility and Electronic Commerce for Information Discovery on the Internet: Second International Workshop, CIA '98, pp. 135-147, France (1998)

[39] M. Tsvetovat, K. Sycara, Y. Chen and J. Ying, "Customer Coalitions in the Electronic Marketplace", Proceedings of the Fourth International Conference on Autonomous Agents, pp. 263 – 264, Barcelona, Spain (Jun. 2000)

[40] Simon Parson and Nick R. Jennings, "Argumentation and Multi-Agent Decision Making", Proceedings of the American Association for Artificial Intelligence (AAAI) Spring Symposium on Interactive and Mixed-Initiative Decision Making, pp.89-91, Stanford, USA (1998)

[41] Martin Beer, Mark d'Inverno, Michael Luck, Nick Jennings, Chis Preist and Michael Schroeder, "Negotiaiton in Multi-Agent Systems", Knowledge Engineering Review 14(3) pp. 285-289, UK (1999)

[42] N. R. Jennings, S. Parsons, P. Noriega and C. Sierra, "On Argumentation-Based Negotiation", Proceedings of the International Workshop on Multi-Agent Systems, Boston, USA (1998)

[43] Shamimabi Paurobally and Jim Cunningham, "Specifying the Process and States of Negotiation", Agent Mediated Electronic Commerce: the European Agentlink Perspective, pp. 61-77, Berlin (2001)

[44] C. Sierra, N. R. Jennings, P. Noriega and S.Parsons, "A Framework for Argumentation-Based Negotiation", Intelligent Agents IV: Agent Theories, Architectures, and Languages (ATAL): 4th International Workshop, ATAL'97, pp. 177-192, Providence, Rhode Island, USA (July 1997)

[45] N. R. Jennings, T. J. Norman, P. Faratin, P. O'Brien and B. Odgers, "Autonomous Agents for Business Process Management", International Journal of Applied Artificial Intelligence, 14 (2) pp. 145-189, UK (2000)

[46] Yannis Labrou and Tim Finin, "A Semantics Approach for KQML – A General Purpose Communication Language for Software Agents", Third International Conference on Knowledge and Information Management (CIKM-94), Gaithersburg, MD (Nov. 1994)

[47] Peter Fingar, "A CEO's Guide to eCommerce Using Object-Oriented Intelligent Agent Technology", Available at: http://home1.gte.net/pfingar/eba.htm (Jun. 1998)

[48] Stan Franklin and Art Graesser, "Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents", Proceedings of the Third Internation Workshop on Agent Theories Architectures, and Languages, published as Intelligent Agents III, pp. 21-35, Springer-Verlag (1997)

Author’s Publication

[1] Liao Xuan and Bharadwaj Veeravalli, “Agent-Based Automated Negotiation System for e-Marketplaces: Local Service Model and Network Cluster Model”, submitted to Electronic Commerce Research and Applications, Elsevier Publishers, Aug. 2003