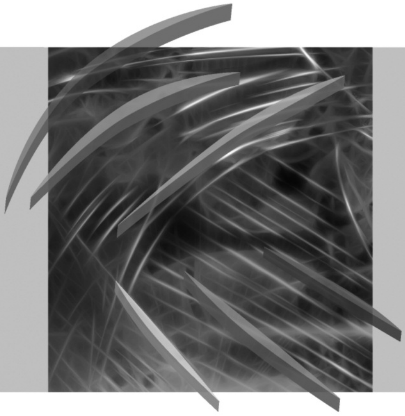
PRACE NAUKOWE
Uniwersytetu Ekonomicznego we Wrocławiu
RESEARCH PAPERS
of Wrocław University of Economics

232

Knowledge Acquisition and Management



edited by
Małgorzata Nycz
Mieczysław Lech Owoc

Publishing House of Wrocław University of Economics
Wrocław 2011

Reviewers: Grzegorz Bartoszewicz, Witold Chmielarz, Halina Kwaśnicka,
Antoni Ligęza, Stanisław Stanek

Copy-editing: Marcin Orszulak

Layout: Barbara Łopusiewicz

Proof-reading: Barbara Łopusiewicz

Typesetting: Beata Mazur

Cover design: Beata Dębska

This publication is available at www.ibuk.pl

Abstracts of published papers are available in the international database
The Central European Journal of Social Sciences and Humanities
http://cejsh.icm.edu.pl and in The Central and Eastern European Online Library
www.ceeol.com as well as in the annotated bibliography of economic issues BazEkon
http://kangur.uek.krakow.pl/bazy_ae/bazekon/nowy/index.php

Information on submitting and reviewing papers is available
on the Publishing House's website www.wydawnictwo.ue.wroc.pl

ISSN 1899-3192
ISBN 978-83-7695-200-0

The original version: printed

Printing: Printing House TOTEM

Contents

Streszczenia

PRACE NAUKOWE UNIWERSYTETU EKONOMICZNEGO WE WROCŁAWIU nr 232
RESEARCH PAPERS OF WROCŁAW UNIVERSITY OF ECONOMICS
Knowledge Acquisition and Management ISSN 1899-3192

Tatiana V. Solodukha, Boris A. Zhelezko
Belarus State Economic University

DEVELOPING A MULTI-AGENT SYSTEM FOR E-COMMERCE

Summary: Agent technology has been claimed to be a promising tool for creating e-commerce applications. To develop a multi-agent e-commerce system effectively, the developers have to deal with several issues such as agent characteristics, agent functionalities, protocols, communication, cooperation, and coordination. Furthermore, agent-based systems should be robust, scalable, and secure. To achieve this, the development of open, stable, scalable, and reliable architectures that allow agents to discover each other, communicate and offer services to one another is required. JADE (Java Agent Development Environment), an open source software framework, is currently the best tool used for developing multi-agent systems. In this paper, our aim is to present our experience in designing and developing an e-commerce multi-agent system with JADE.

Keywords: agent technology, e-commerce, multi-agent systems, JADE.

1. Introduction

During the last few years, there has been a growth of interest in the potential of agent technology in the context of e-commerce. Agent technology is considered by researchers as one of the most useful and powerful technologies to model real-world e-commerce business processes. Agents are computational systems that are capable of autonomous, reactive, and proactive behaviour, and are also able to communicate and may cooperate with each other. A multi-agent system (MAS) is defined as a network of loosely-coupled computational autonomous agents who can perform actions, have resources at their disposal, and possess knowledge, capabilities, or skills.

MAS approaches the problem using the well-proven tools from game theory, economics, and biology. It supplements these with ideas and algorithms from artificial intelligence research, distributed systems, reasoning methods, search methods, and machine learning.

Agent technology is often claimed to be the best approach for automating e-commerce business processes. However, it is difficult to find successful large-scale agent-based e-commerce applications to confirm this claim. This paper addresses this issue by discussing the development of an e-commerce system using an agent platform – JADE.

2. The methodology for building MAS

Over the past few years, there have been several attempts at creating tools and methodologies for building such systems. Unfortunately, many of the methodologies have focused on single agent architectures or have not been adequately supported by automated toolsets. Thus, the aim of this research is to develop methodology for analysing, designing, and developing MAS for e-commerce.

In this research, agents coordinate their actions via conversations to accomplish individual and community goals. The general flow of the methodology follows the seven steps shown in Figure 1.

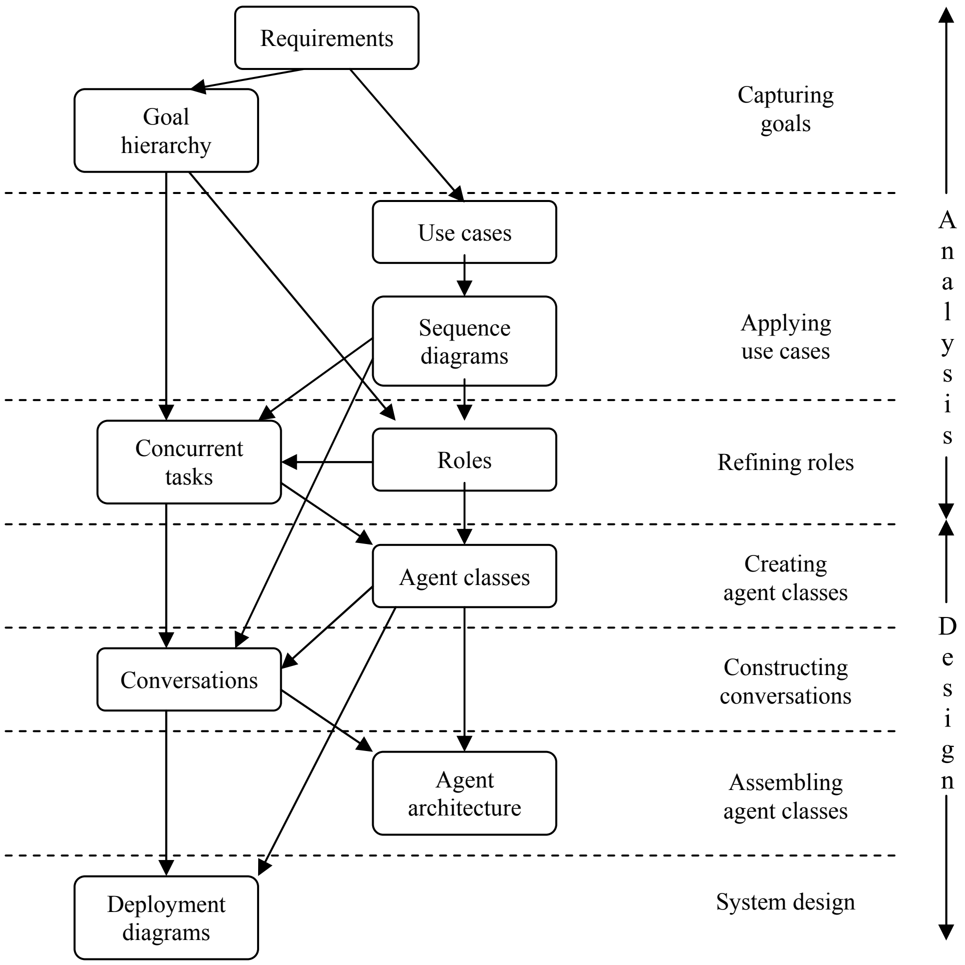


Figure 1. The methodology for building MAS

Source: authors' own study.

The methodology is a seven-step process that guides a designer in transforming a set of requirements into a successively more concrete sequence of models. By analyzing the system as a set of roles and tasks, a system designer is naturally led to the definition of autonomous, pro-active agents that coordinate their actions to solve the overall system goals.

The methodology begins in the analysis phase by capturing the essence of an initial system context in a structured set of goals. Next, a set of use cases is captured and transformed into sequence diagrams so that desired event sequences will be designed into the system. Finally, the goals are combined to form roles, which include tasks that describe how roles satisfy their associated goals. In the design phase, roles are combined to define agent classes and tasks are used to identify conversations between the classes. To complete the agent design, the internal agent architecture is chosen and actions are mapped to functions in the architecture. Finally, the run-time structure of the system is defined in a deployment diagram and implementation choices such as language and communication framework are made. Following the methodology is relatively simple, yet it is flexible enough to allow for a variety of solutions.

The rounded rectangles on the left side denote the models used in each step. The goal of the methodology is to guide a system developer from an initial system specification to a MAS implementation. This is accomplished by directing the designer through this set of inter-related system models. Although the majority of the methodology's models are graphical, the underlying semantics clearly defines specific relationships between the various model components.

The methodology is designed to be applied iteratively. Under normal circum-stances, designer moves through each step multiple times, moving back and forth between models to ensure each model is complete and consistent. While this is com-mon practice using most design methodologies, our methodology was specifically designed to support this process by formally capturing the relationships between the models.

By automating the methodology models, these relationships are captured and enforced thus supporting the designer's ability to freely move between steps. The result is consistency between various models and a system design that satisfies the original system goals. The methodology is independent of a particular multi-agent system architecture, agent architecture, programming language, or communication framework. Systems designed using this methodology can be implemented in a variety ways. For example, a system could be designed and implemented so that included a heterogeneous mix of agent architectures and used any of a number of existing agent communication frameworks. The ultimate goal of the methodology is the automatic generation of code that is correct with respect to the original system specification. With such a capability, the methodology could work hand in hand with flexible and efficient runtime environments such as JADE.

The main reason for the JADE agent platform selection was the fact that JADE is one of the best modern agent environments. JADE is open-source, it is FIPA compliant and runs on a variety of operating systems, including Windows and Linux. Furthermore, it has very good scalability. It was shown that it can easily scale up to 1 500 agents and 300 000 messages. The goal of JADE is to simplify the development of MAS, while ensuring standard compliance through a comprehensive set of system services and agents in compliance with the FIPA specifications: naming service and yellow-page service, message transport and parsing service, and a library of FIPA interaction protocols ready to be used.

As described in the recent surveys, multi-agent technology should have an important economical impact, by bringing efficiency to businesses (and thus improving their profitability), as well as benefiting individual users. MAS are expected to assure price fairness and reduce the negotiation time. This research indicates that most currently existing automated trading systems are not robust enough to become the foundation of the next generation of e-commerce. The claim that there is a lot more work to be done to develop real-life agent-based support for e-commerce is further supported by the fact that it is almost impossible to point out an existing large-scale implementation of an e-commerce agent system.

On the other hand, modern agent environments allow building and experimenting with large-scale agent systems. Therefore, we have set up a goal of developing, implementing and experiment with MAS for e-commerce. In this paper we present a system with a multitude of agents that play variety of roles and interact with each other (system skeleton).

3. System description

As a result of the present research, the model of the e-commerce platform targeted at online trading and auction systems, e-markets, and private trading exchanges was developed. The platform may support e-commerce activity across a wide range of markets and industries. Whether it is buying or selling products or services, domestically or internationally, the system will meet company's needs. Software agents lie at the heart of the platform, automating the business processes that match buyers and sellers (see Figure 2).

The e-commerce model acts as a distributed marketplace that hosts e-shops and allows e-clients to visit them and purchase products. Clients have an option to negotiate with shops, to bid for products and to choose a shop from which to make a purchase. Conversely, shops may be approached "instantly" by multiple clients and, consequently, through auction-type mechanisms, have an option to choose a buyer. At this stage, we assume that a price is the only factor determining a purchase.

The system supports dynamic agent creation and destruction and agent migration to engage in negotiations. The top level conceptual architecture of the system illustrating proposed types of agents and their interactions in a particular configuration is shown in Figure 3.

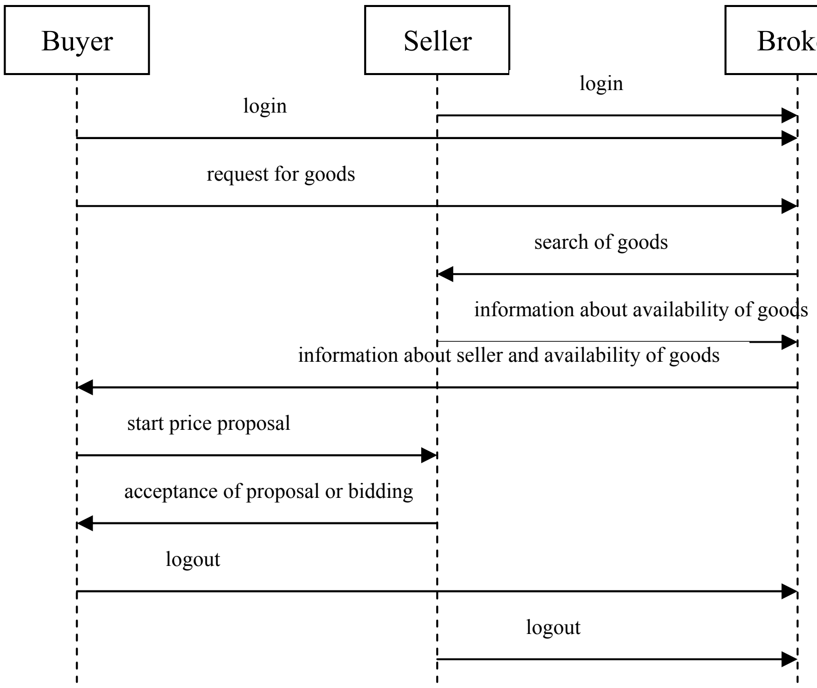

Buyer
Seller
Broker
login
login
request for goods
search of goods
information about availability of goods
information about seller and availability of goods
start price proposal
acceptance of proposal or bidding
logout
logout

Figure 2. Agent interaction

Source: authors' own study.

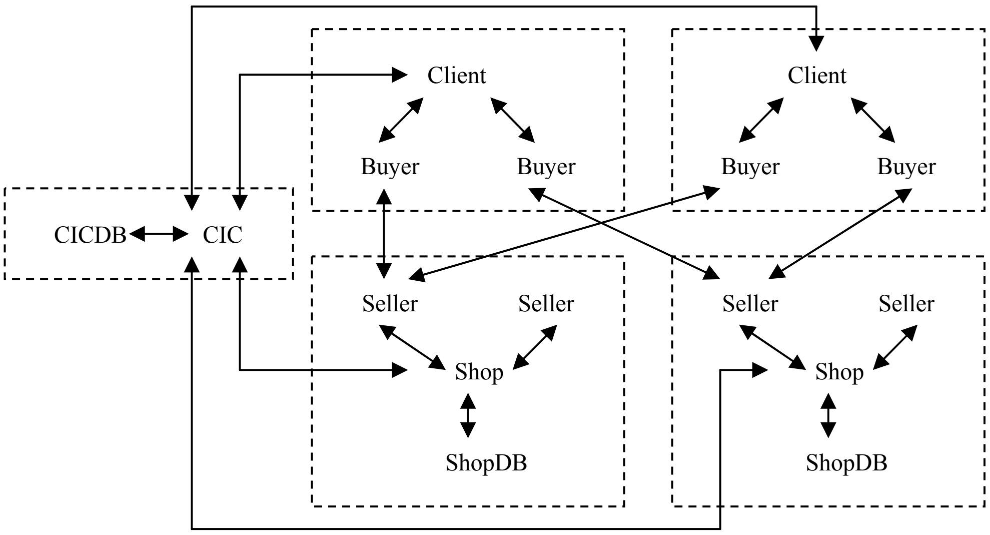


Figure 3. The conceptual architecture of MAS for e-commerce

Source: authors' own study.

A *Client* agent acts within the marketplace on behalf of a buyer. A merchant will create a *Shop* agent, responsible for advertising and selling products within the marketplace. After being created both *Shop* and *Client* agents register with a *CIC* agent (broker) to be able to operate within the marketplace. Returning agents will receive their existing IDs. In this way, support for the future goal of agent behavior adaptability is provided. Agents in the system are able to recognize status of their counterparts and differentiate their behaviour depending if this is a "returning" or a "new" agent that they interact with.

There is only one *Client Information Center* (*CIC*) agent in the system. It is responsible for storing, managing, and providing information about all "participants." To be able to participate in the marketplace, all *Shop* and *Client* agents must register with a *CIC* agent, which stores information in the *Client Information Database* (*CICDB*). The *CICDB* combines the function of *client registry*, by storing information about and unique IDs for all users and of *yellow pages*, by storing information about of all shops known in the marketplace. Thus, *Client* agents (new and returning) communicate with the *CIC* agent to find out which stores are available in the system at any given time.

A *Client* agent is created for each customer that is using the system. Each *Client* agent creates an appropriate number of "slave" *Negotiation* agents with the "buyer role" (*Buyer* agent). One *Buyer* agent is created for each store, within the marketplace, selling sought goods. On the supply side, a single *Shop* agent is created for each merchant in the system and it is responsible for creating a slave *Negotiation* agent with the "seller role" (*Seller* agent) for each product sold by the merchant.

Finally, *Database* agents are responsible for performing all database operations (updates and queries). For each database in the system, one database agent is created. Currently, there are two databases in the system: a single *CICDB* (operated by the *CICDB* agent) containing the information about clients, shops, and product catalogues, and a single *Shop Database* (*ShopDB*) operated by a *ShopDB* agent storing information about sales and available supplies for each merchant registered within the system.

The central part of the system operation is comprised of price negotiations. *Buyer* agents negotiate price with *Seller* agents. For this purpose, *Buyer* agents migrate to the e-stores known by a *CIC* agent to carry sought after commodity. In the case of multiple *Buyer* agents attempting at purchasing the same item, they may compete at an auction. The results of price negotiations are send to a *Client* agent, who decides where to attempt at making a purchase. Note that the system is fully asynchronous and thus an attempt at making a purchase does not have to result in a success as by the time the offer is made other *Buyer* agents may have already purchased the last available item.

4. Usage scenario

A session with the system starts with the merchants and customers creating *Shop* and *Client* agents. A *Client* agent obtains the name of the product of interest and a reserve price. A *Shop* agent obtains a list of pairs (product, reserve price) and the negotiation protocol that is to be used for interactions with incoming *Buyer* agents. The reserve price of a *Client* agent is the maximum price that it agrees to pay for the product. The reserve price of a *Shop* agent is the minimum price at which it is to agree to sell a specified product. In the future, *Client* and *Shop* agents will have access to a collection of strategies to be used depending on the context of unfolding negotiations.

A *Shop* agent creates *Seller* agents, one *Seller* agent for each product sold. *Seller* agents await incoming *Buyer* agents interested in buying their products and upon their arrival engage in negotiations with them.

Let us now describe what happens in the marketplace after a customer has made a purchase request, until a request is completed.

(1) As specified above, a *Client* agent registers with a *CIC* agent. It obtains a new ID if it is a new *Client* or recovers its original ID if it is a returning *Client.* The information that an agent with a given ID is active in the marketplace is stored in the *CICDB* database (this step involves interactions between a *CIC* agent and a *CICDB* agent).

(2) A *Client* agent queries a *CIC* agent to obtain the list of *Shop* agents selling the product it is expected to purchase. For each *Shop* agent on this list, it creates a *Buyer* agent to negotiate conditions of purchase.

(3) *Buyer* agents migrate to *Shop* agent sites and query *Shop* agents about the negotiation protocol used in a given e-store and which *Seller* agent they should negotiate with. Then, *Buyer* agents dynamically load appropriate negotiation protocols (and, in the future, strategy modules) and subscribe to the designated *Seller* agent, waiting for the negotiation process to start.

(4) A *Seller* agent checks periodically (currently in one-minute intervals) for the set of *Buyer* agents that subscribed to bid for its product. If this set is nonempty, it starts an auction. At the end of an auction, a *Seller* agent informs a *Shop* agent about the winner. *Shop* agents are recording the auctions winners and inform the corresponding *Client* agents that a purchase is possible. The decision to buy and where to buy from is made by a *Client* agent, depending on the winning offers made by *Shop* agents.

(5) A *Client* agent obtains results of auctions from *Shop* agents, finds the best negotiated price and makes an attempt at purchasing the product by informing the corresponding *Shop* agent about the decision to buy. When the confirmation is received, it informs the customer about the result of its request: success or failure of purchase, the shop where the purchase was made from and the negotiated price. Note

that there are various strategies that could be employed by a *Client* agent in order to decide where to buy from, and there is an associated risk also. For example, it may decide for the best offer, the safest offer or the most trusted offer, etc.

5. Conclusion

Recent advances in software engineering, business process management, and computational intelligence resulted in methods and techniques for developing advanced e-commerce applications as well as supporting automating e-commerce business processes. Despite this fact, up to now, the most successful e-commerce systems have been still based on humans to make the most important decisions in various activities within an e-business transaction. Therefore, we have set up a goal of developing, implementing, and experimenting with an agent-based e-commerce system.

We designed the methodology for building MAS, which is independent of a particular MAS architecture, agent architecture, programming language, or communication framework. Systems designed using this methodology can be implemented in a variety ways.

In this paper we also have presented basic features of an e-commerce modelling agent system which we are in the process of developing. At this stage, its capabilities are limited, but we have already considered a number of future research directions that we plan to pursue.

1) Currently, price is the only factor determining purchase. Other factors, such as the speed of delivery, trust, history of involvement with a given merchant should be also taken into account.

2) At present, only shops can advertise available commodities. We plan to extend this to the scenario in which also clients will be able to advertise their "needs."

3) We will complete implementation of negotiation protocols.

References

Paprzycki M. et al. (2004), Implementing agents capable of dynamic negotiations, [in:] D. Petcu et al. (eds.), *Proceedings of SYNASC04: Symbolic and Numeric Algorithms for Scientific Computing*, Mirton Press, Timisoara, Romania, pp. 369-380.

Solodukha T.V., Sosnovskiy O.A., Zhelezko B.A. (2009), Multi-agent systems for e-commerce. Pattern recognition and information processing (PRIP) '2009, [in:] *Proceedings of the 10th International Conference (19-21 May, 2009, Minsk, Belarus)*, Publishing Center of BSU, Minsk, pp. 354-358.

Srivastava V., Mohapattra P.K.J. (2003), PLAMUN: a platform for multi-user negotiation, *Electronic Commerce Research and Applications*, Vol. 2, No. 3, pp. 339-349.

Websites

http://jade.tilab.com

BUDOWA SYSTEMÓW WIELOAGENTOWYCH NA POTRZEBY HANDLU ELEKTRONICZNEGO

Streszczenie: Technologia agentowa zdaje się być obiecująca na potrzeby tworzenia aplikacji handlu elektronicznego. Aby efektywnie budować wieloagentowe systemy e-commerce, ich twórcy muszą zmierzyć się z wieloma problemami, takimi jak charakterystyki agentów, ich funkcjonalność, protokoły, komunikacja, współdziałanie i koordynacja. Dodatkowo systemy agentowe musza być wydajne, skalowalne oraz bezpieczne. Aby to osiągnąć musimy korzystać z otwartej, stabilnej i godnej zaufania architektury, która pozwala agentom komunikować się i oferować sobie wzajemnie usługi. JADE (*Java Agent Development Environment*), czyli platforma *open-source* jest obecnie najlepszym narzędziem do budowy Systemów wieloagentowych. W tym artykule celem autora jest prezentacja własnego doświadczenia w projektowaniu i budowie wieloagentowych systemów handlu elektorncznego przy użyciu platformy JADE.

Słowa kluczowe: technologia agentowa, *e-commerce*, systemy wieloagentowe, JADE.