Information Flow in a Distributed Agent-Based Online Auction System

Sorin Ilie, Costin Bădică
University of Craiova
Software Engineering Department
Bvd.Decebal 107, Craiova, 200440, Romania

Liviu Sandu, Raluca Sbora
University of Craiova
Software Engineering Department
Bvd.Decebal 107, Craiova, 200440, Romania

Amelia Bădică
University of Craiova
Business Information Systems Department
A.I.Cuza 13, Craiova, 200585, Romania

Maria Ganzha, Marcin Paprzycki
Systems Research Institute
Polish Academy of Sciences
ul. Newelska 6, 01-447 Warszawa, Poland

ABSTRACT

Recently we were involved in the development of a distributed agent-based English auction server. The focus of the work was on emphasizing the advantages brought by the multi-agent systems technology to the high-level of abstraction, modularity and performance of the server architecture, and its implementation. Less effort was spent on the analysis of its external functionalities and usability, as well as on approaching the implementation of a realistic system for online auctions. Therefore, in this paper we present our solution, in terms of information flow management, and its relation to the functionalities of the system for online auctions that incorporates our server. The main outcome of this work is a clean specification of the information exchanges between the agent and non-agent software components of the system. Special attention is also provided for the interoperability of the different data communication, and agent and non-agent software technologies that are employed for the implementation of the system.

Categories and Subject Descriptors

D.2.11 [**Software**]: Software Architectures

General Terms

Software Design, Multi-Agent System

1. INTRODUCTION

The vision of e-commerce automation proposes the development of global agent-based e-commerce environments that will enable dynamic trading between business partners. In particular, increasing the level of automation of negotiations became an important issue to be resolved to allow the engagement of business partners, either individuals or business organizations, into nontrivial and dynamic business relationships.

While there exists a large number of negotiation mechanisms, understood as techniques for reaching an agreement over an issue (see, for instance results produced by the EU COST Action "Agreement Technologies" [1]), in the context of e-commerce, the most often considered are auctions [5], [11]. They represent a special class of negotiations with many applications in conducting e-business transactions [12]. In particular auctions have been established to be particularly useful for trading in the following areas: spectrum licenses, electricity markets, emission rights, airports takeoff and landing slots, exploitation rights of natural resources, selling of collectibles, antiques, luxury and second-hand products, a.o.

One of the interesting side-effects of the raise of the Internet is, rapidly increasing, interest in online auctions. In this context, many types of online applications for auctions were proposed including: auction directories, auction tops, meta-auctions, and auction servers [6]. Recently the research focus was set on the development of more process-generic, flexible and reusable auction solutions, with an increased potential for applicability to the B2B sector. Here, application of agent-based services orientation proposed as a new approach that takes the idea of an auction service from the human-driven web world to the software agents' world [3].

Our recent work in this area was focused on the study, design and implementation of an open, flexible infrastructure for service-based automated negotiations in agent systems. The following two objectives were set for this project:

(i) To emphasize the advantages of using multi-agent systems (MAS hereafter) [2] by developing a realistic auction server. Here the focus was set on employing fun-

[1]http://www.agreement-technologies.eu/

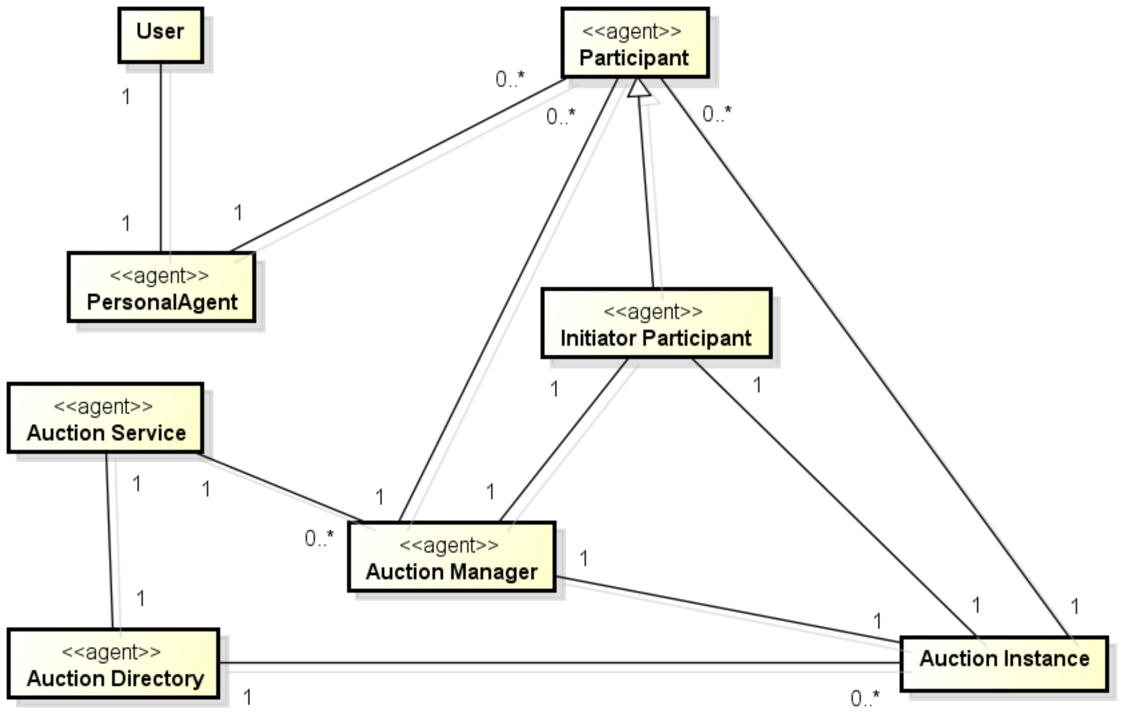
User
<<agent>> Participant
<<agent>> PersonalAgent
<<agent>> Initiator Participant
<<agent>> Auction Service
<<agent>> Auction Manager
<<agent>> Auction Directory
Auction Instance
1
0..*

Figure 1: Relationships between users, agent types and auctions on the auction server.

damental software engineering principles (abstraction and modularity), as well as on evaluating and improving the performance and scalability of the implementation. Software engineering principles, as well as initial performance assessment were presented in [6]. The scalability and performance aspects were further investigated and improved by realization of a cluster-based implementation of the auction server. Obtained initial results were presented and evaluated in [14].

(ii) To develop a tool that can be used for online auctions in B2C and B2B systems. The first goal of this objective is addressed in this paper, by focusing on the details of incorporating our agent-based auction server into a Web-based application for online auctions. The second aspect of this goal is left as future work.

Our agent-based auction server solution combines the best features of: (i) generic software framework for automated negotiations [1]; (ii) market architecture for auction development [13]; (iii) rule-based declarative representation of auction mechanisms [5]; (iv) special computing nodes available in active networks and realized by means of proxy agents [9]. Consequently, it provides certain advantages, including: service orientation, openness, flexibility, generality, and scalability (see [6] and [14] for more details on these aspects).

The focus of our previous work was on emphasizing the advantages brought to the automated negotiations by the MAS technology. Specifically, we have focused our attention on high-level of abstraction, modularity and performance of the auction server architecture. Here, we are concerned with the analysis of its external functionalities and usability, as well as issues involved in their implementation into a realistic system for online auctions. We present the details of our solution in terms of the information flow management and its relation to the functionalities of a system for online auctions that incorporates the server. Furthermore, we provide a clean specification of the information exchanges between the agent and non-agent software components of the system, which is particularly interesting from the software engineering point of view. In this context, special attention is also given to the interoperability of the different data communication and software technologies that were utilized while implementing the system.

In this way we contribute to the research concerning usability of agent-based e-commerce solutions. Note that, while very attractive, the complete automation of e-commerce processes is probably impossible to achieve, and therefore the human user involvement through an appropriate online system will be necessary in most scenarios. However, design and implementation of auction servers of the type discussed in this paper may push use of autonomous beyond simple "sniper agents," like these available for the Allegro and eBay C2C / B2C platforms (for more details, see [2]).

The paper is structured as follows. In Section 2 we describe the background of our auction server that is necessary for understanding the contribution of this paper. Next (in Section 3), we propose the design of a Web system for online auctions that incorporates the proposed auction server. Here, the discussion is focused on three aspects: (i) system architecture, (ii) design details of the Web layer as well as of the interfacing of the agent and non-agent software, and (iii) interaction protocols. We follow with a brief review of related works (in Section 4), and conclude by pointing to future works.

[2]http://www.snajper.net/

2. BACKGROUND

Let us start by briefly reviewing the architecture and interaction protocols of our auction server that was configured to support single-item English auctions. The initial architecture and the agent interaction protocols of the server was introduced in [6]. In [14] we proposed an improved architecture that enables the deployment of the server on a computer cluster.

The auction server was designed to support the innovative concept of a *generic agent-based auction service.* It is composed of a MAS representing a society of cooperating agents that also interact with the external environment, using standardized agent interaction protocols. Furthermore, the MAS is scalable and therefore it can be distributed efficiently on a computer network, to guarantee system performance.

The core software infrastructure of the server introduced in [14] contains the following types of agents; illustrated by the class diagram in Figure 1:

- *Auction Service* agent that manages all the active auctions on the server.
- *Auction Manager* agent that manages a single active auction on the server (also known as the *Auction Instance*).
- *Auction Directory* agent that manages the registry of active auctions, as well as the identifiers of their associated *Auction Manager* agents.
- *Personal Agent* acts as an interface between the server and a given user. Each user connected to the server is represented on the server by a *Personal Agent.* Usually this agent is controlled by the user through an external user interface either directly (i.e. the *Personal Agent* can incorporate a user interface) or via a binding software. We assume that there is a default one-to-one mapping between the user name and the name of the corresponding *Personal Agent* (these two names are actually the same).
- *Participant* agent represents a user registered to a particular auction. For each user registered to participate in an auction there is an associated *Participant* agent on the server. The *Participant* agents associated to a given user directly report to and are controlled by his or her *Personal Agent.* A special type of participant is the *Initiator Participant*, representing the *Participant* agent that initiated the auction. Note that in an English auction the initiator participant has the role of a *seller*, while the remaining participants have the role of a *buyer.*

Additionally, the server contains the following types of agents that are responsible for management of resources, and enhancement of the server performance (see, also, [14]):

- *Proxy* agent handles the bids received from a subset of *Participant* agents. The *Participant* agents are split into disjoint groups, and each group is managed by a single *Proxy* agent. *Proxy* and *Participant* agents are linked into a balanced two-level hierarchical structure rooted at the *Auction Manager* such that the total number of *Proxy* agents equals (or at least, it is almost equal to) the number of *Participant* agents that are linked to each *Proxy* agent. Obviously, during the bidding part of the auction, *Participant* agents communicate heavily with their *Proxy* agents, while *Proxy* agents pass on to the *Auction Manager* agent only the relevant bids, while the other bids are filtered out and processed locally, thus reducing the amount of messages handled by the *Auction Manager* and in this enhancing the server response time.
- When the server is installed on a computer network, the *Computer Manager* agents are responsible for management of *Participant* and *Proxy* agents on each available machine. Here, the *Resource Manager* contains a registry of all the agents of type *Computer Manager* within the system.

The initial experimental results (see, [14]) show that our hierarchical scheme of structuring the server using *Proxy* agents and the proposed simple balancing scheme is effective, and has good scalability when the server is distributed on multiple machines.

The interaction between the *Personal Agent* and the agents of the auction server is governed by a set of agent interaction protocols. These protocols specify the set of correct message exchanges that can occur during a conversation carried out between several agents. The auction server uses interaction protocols for:

- *The initiation of an auction.* A conversation for creation of a new auction that is started by a *Personal Agent* that interacts with the *Auction Service.* The effect of this interaction is creation of a new *Auction Manager*, a new *Participant* agent (actually an *Initiator Participant*), and a new *Auction Instance.* The *Auction Instance* is also registered with the *Auction Directory.* The *Participant* agent is also linked to the *Personal Agent* that represents the user on the auction server.
- *Searching for auctions.* A conversation for searching for an auction according to a given set of criteria is started by the *Personal Agent* that interacts with the *Auction Service.* The actual search is performed with the help of the *Auction Directory* agent and the resulting list of auctions is returned to the *Personal Agent.*
- *Registration to an auction.* In order to start a conversation for joining an auction, the *Personal Agent* must hold the identifier of the corresponding *Auction Manager.* This usually happens after a conversation for searching for active auctions. The *Personal Agent* chooses the auction (and consequently the associated *Auction Manager*) that fulfills its requirements and then interacts with the *Auction Manager* by submitting its intention for registration with that auction. If the registration is successful then a new *Participant* agent is created on the server and at the end of the conversation its identifier is returned by the *Auction Manager* to the *Personal Agent.*
- *Participation in an auction.* This assumes the ability to bid and to obtain the price quotation information. In order to start a conversation for submitting a bid in an auction the *Personal Agent* must hold the identifier of the corresponding *Participant* agent that

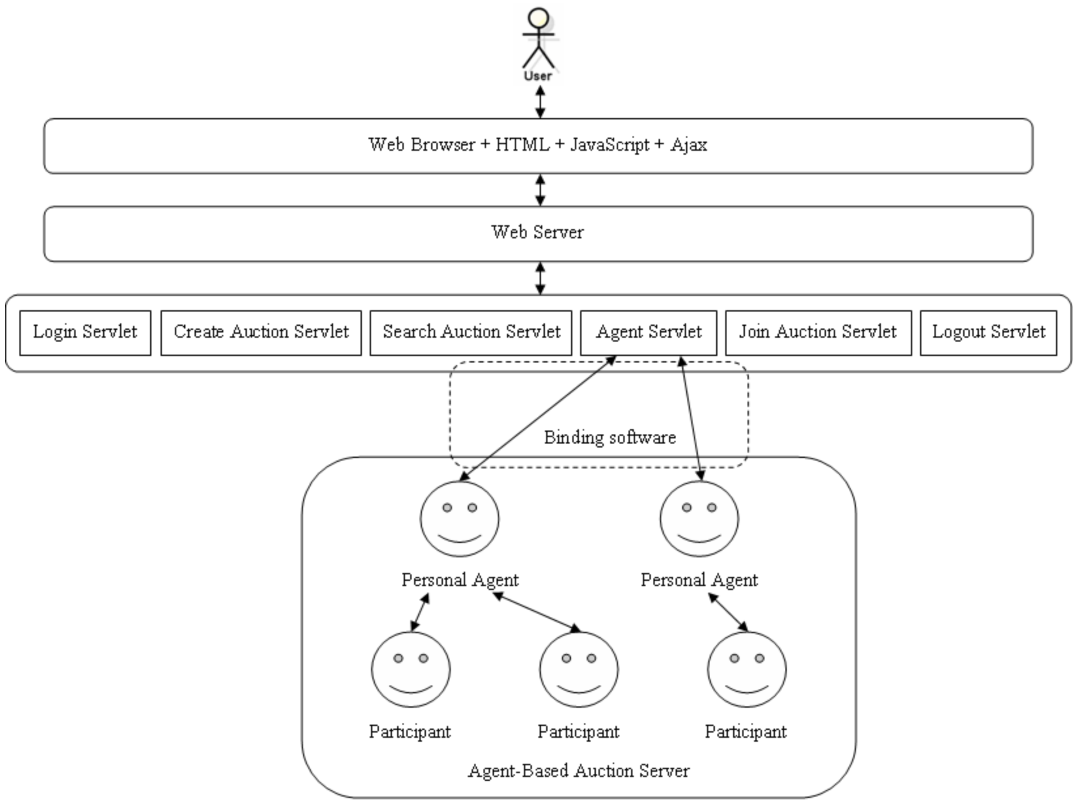

Figure 2: System architecture.

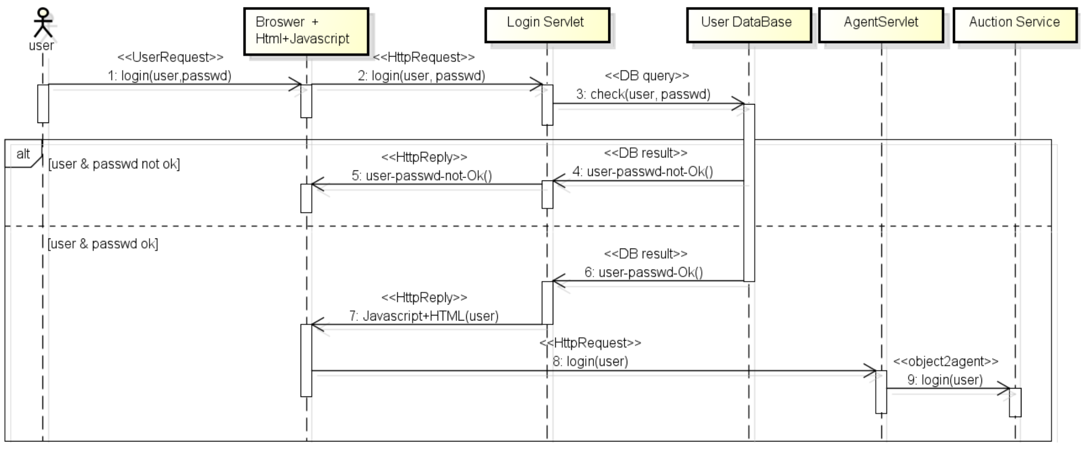

Figure 3: User login.

represents him/her in that auction and must be registered with that auction. The *Personal Agent* submits a bid by interacting with the *Participant* agent. In our model the *Personal Agent* automatically obtains the price quotation information using a "push" model through notifications received from the *Participant* agent [18]. These notifications are automatically spread by the *Auction Manager* to the auction participants whenever the auction status was changed and the visibility rules (depending on the auction type) are enabled for that participant type.

3. SYSTEM DESIGN

The system design consists of three parts: system architecture, interaction protocols, and design details of the system components.

3.1 System Architecture

The auction server was implemented using the JADE multi-agent platform [2]. For experimenting with the usability of this auction server we have developed an online system equipped with a Web-based GUI that allows human users to create, search for and participate in English auctions. The architecture of the application follows the classical separation between the client side, comprising the human user equipped with a Web browser, and the server side comprising a Web server that interacts with the auction server. Therefore we designed and developed a software for binding the Web server part that is non-agent software with the auction server that consists solely of JADE-based agent software.

The system has a multi-tier architecture composed of the following layers, as illustrated in Figure 2:

- *User layer.* This layer represents the client part of the system that consists of a Web browser combined with HTML content including JavaScript code that is downloaded from the Web server.

- *Web layer.* This layer supports the user interaction functionalities. It consists of a Web server enhanced with a set of Java servlets that implement the user functionalities of the online application.

- *Binding layer.* This layer is represented by the software that enables the interfacing of the non-agent Web server software with the agent-based auction server. This software is encapsulated into a special servlet called *Agent servlet* that is able to communicate with the JADE platform.

- *Agent layer.* This layer represents the agent-based auction server that was built on top of JADE platform.

3.2 Design Details

3.2.1 User and Web Layers

The *Web layer* is responsible for management of users and their accounts, while the auction server (i.e. the *Agent layer*) is responsible for the auction management. With this separation of functionalities, the *Web layer* will support the interaction of the application with the human user, via a Web-based GUI that is based on HTML, Asynchronous JavaScript, and XML (also known as AJAX).

As we did not create a functionality for user authentication in the agent-based auction server, we had to provide a solution for this problem at the level of the Web layer. So, in our prototype system, the Web layer is responsible for user authentication (login and logout functionalities) and management of user accounts. The addition of this functionality requires the Web layer to maintain a separate database for the management of user account information.

At the *User layer*, the servlets must provide HTML responses with information extracted from the *Agent layer* using software from the *Binding layer*. This operation can be time consuming and thus it can slow down the load time of the Web page. This issue was addressed by inserting JavaScript code into the HTML responses. This code allows the Web page to update itself quickly and efficiently using asynchronous requests. The JavaScript code issues automatic, or user generated, HTTP requests to the *Agent servlet*, which was configured to reply with the XML responses. Note that the *Agent servlet* is the only servlet of the *Web layer* that can communicate with the agents on the *Agent layer* (via the *Binding layer*).

3.2.2 Binding Layer

The *Binding layer* communicates with the *Agent layer* using a specialized software. This software benefits from the JADE facilities for interfacing agent and non-agent software materialized as the *JadeGateway* class ([10]). The interface is achieved using agents (also known as the *Gateway* agents) that are created locally by the *Agent servlet*. One *Gateway* agent is created for each user logged into the system. These local agents are created in a local container that is connected to the agent platform that hosts the auction server. This container is created and started together with the *Web layer*.

Whenever a new user is logged into the system, the *Agent servlet* automatically creates a local *Gateway* agent with the role of relaying messages from the *Web layer* to the agents on the auction server. Then the *Agent servlet* locally creates and passes a serializable object (called "blackboard object," in [10]) to the *Gateway* agent assigned to the current user. The *Gateway* agent then sends the message using the JADE messaging functionality to an agent located on the auction server, according to one of the interaction protocols presented in section 3.3. This type of interaction is marked with the ≪*object2agent*≫ stereotype in Figures 3, 4, 5, 6, and 7. Conversely, whenever an agent present at the auction server must send an information to the *Web layer* it will use the JADE messaging to send this information to the corresponding *Gateway* agent located in the *Agent servlet*. Then the *Gateway* agent will invoke a method to update the "blackboard object" and thus achieving the correct transfer of the information to the *Agent servlet*. This type of interaction is marked with the ≪*agent2object*≫ stereotype in Figures 5 and 6.

Note that, whenever the JavaScript code of the Web page asynchronously requests an information update from the *Agent servlet* the servlet will respond with a message containing the relevant data extracted from the "blackboard" object and represented in the XML format. This type of interaction is represented with the ≪*XML response*≫ stereotype in Figures 5 and 6.

3.3 Interaction Protocols

The *user login* operation is detailed in Figure 3. The first

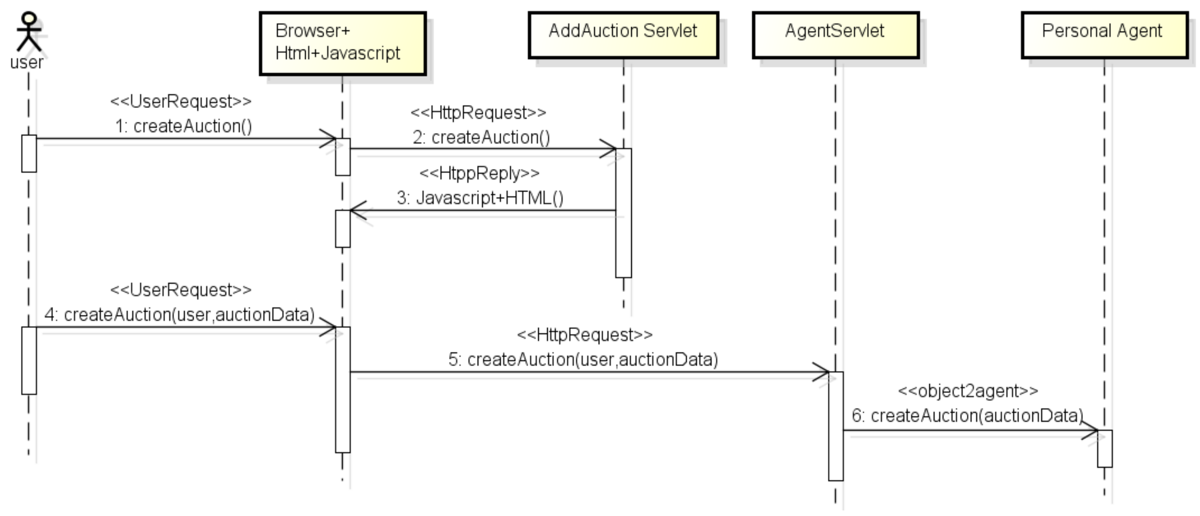

Figure 4: Initiation of an auction.

part (interactions numbered from 1 to 7) achieves the authentication function. If the authentication is successful (i.e. the interactions proceed according to the to the branch consisting of messages 6 and 7) the *Auction Service* agent is notified accordingly, via the interaction consisting of messages 8 and 9.

The operation of *initiating an auction* is detailed in Figure 4. The first 3 interactions activate the user menu for setting the auction data. The next 3 interactions support the function of creating a new auction. The actual creation is achieved after interaction 6. Note that we assume that when a new auction is created the user is already logged in and its *Personal Agent* is active.

The operation of *searching and registering at an active auction* is detailed in Figure 5. The first 3 interactions activate the user menu for setting the search criteria for the desired auction. The next 8 interactions (numbered from 4 to 11) support the function of searching auctions available in the auction directory. This is achieved with the help of the *Auction Directory* agent, residing on the auction server. Then the user chooses the desired auction (this is achieved by interaction 12). Note that after the execution of this action the name of the corresponding *Auction Manager* agent that represents the auction selected by the user becomes known to the user. Consequently, the last 6 interactions (numbered from 13 to 18) allow the user to join the desired auction.

The operation of *bidding in an auction* is detailed in Figure 6. The first 3 interactions activate the user menu for setting the bid data. The next 3 interactions (numbered from 4 to 7) support the function of submitting the bid to the auction server. Note that we assume that at this point the user knows the identifier of the auction where (s)he wishes to submit the bid to (parameter *auction*). The last 4 interactions (numbered from 8 to 11) allow the user to visualize the quotes of the auctions where (s)he is subscribed. In particular, for an English auction, the user can check if a given bid was accepted or not, by the auction server.

The *user logout* operation is detailed in Figure 7. Similarly to the login operation, the auction server is notified that the user is leaving the system. However, note that in this case, the notification is sent directly to the *Personal Agent* that represents the user.

Finally, it is important to observe that there is an interesting relationship (not shown in Figures 3 and 7) between the *Auction Service* and the *Personal Agent* that represents a specific human user on the auction server. This is the result of the fact that the *Personal Agent* has a very important role, by controlling the user participation in auctions, even when the user might be disconnected from the online system. This fact has two important consequences:

(I) During the login operation the system must check if the user already has an active *Personal Agent* on the auction server, and if not it must create one. We assume that this operation is achieved by the *Auction Service* agent. So, the *Auction Service* agent has the responsibility of creating and setting up of a new *Personal Agent* according to the user requirements. For example, such requirements could include: name or description of the sought-after product, the required quantity, the available budget, the maximum price that is allowed to bid, as well as a deadline for buying the product. Note that the *Personal Agent* can be endowed with a certain level of “controlled autonomy” that enables it to act by subscribing to appropriate auctions and bidding accordingly (via suitable *Participant* agents) in order to meet the user requirements and/or maximize his or her profit.

(II) During the logout operation the system must inform the *Personal Agent* that the user has left the system. However, the *Personal Agent* can behave more or less autonomously (according to the user settings and requirements) in representing the user preferences and interests. So, the *Personal Agent* can autonomously decide to go offline in situations when, for example, there are no more active auctions in which the user is participating or, more generally, when a certain user goal is achieved or abandoned. Alternatively the *Personal Agent* can decide to continue its execution on

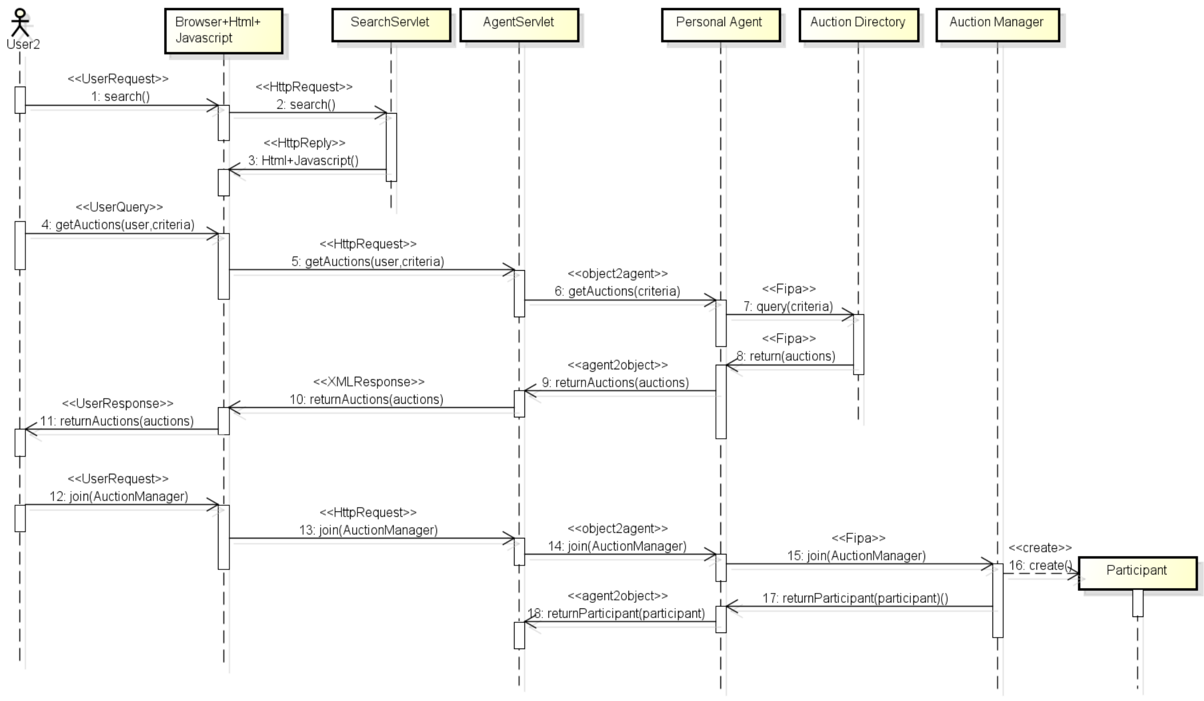

Figure 5: Searching and joining an auction.

the server, depending on the situation. Nevertheless, we set the requirement that the *Personal Agent* must always notify the *Auction Service* that it will go offline before doing so, such that if the user logins again onto the system then the *Auction Service* will be able to create and setup a new *Personal Agent* accordingly.

4. RELATED WORK

The interest in the development of online software systems for online negotiations, with a special focus on online auctions, increased significantly during the last 15 years. Traditionally, auctions were utilized for trading support in economic markets in offline as well as in online environments. Recently auctions started to be applied in market environments for trading resources for utility computing, including grids and clouds [4].

One of the first and most influential works in the area of auction servers for online applications is the Michigan Internet AuctionBot introduced in [19]. This is a versatile and robust server for online auctions supporting both agent-oriented and human-oriented auction execution. The Michigan Internet AuctionBot introduced the principles of software design for supporting flexible auction mechanisms, including: separation of the user interface from the core auction engine, the capability of running concurrently multiple auctions, as well as the abstraction of the auction process. Most of these principles are currently employed by state-of-the-art auction servers. We also included them in our work, as well as expanded and adapted them to allow the deployment of our agent-based auction server on a computer cluster while providing the user interaction through a Web interface.

In [15], the authors proposed an Internet-based negotiation server for e-commerce applications. Although this work does not explicitly address the auction mechanisms (the focus is on bargaining, instead) and the use of software agent technologies, it is of interest to our approach for at least the following reasons: (i) the system is conceptualized as a replicable service that can be multiply instantiated by complementing standard Web server software, i.e. quite similarly to our proposal; (ii) the system incorporates methods of event-based rule processing and constraint satisfaction for checking negotiation proposals and implementation of negotiation strategies which, although they are not the focus of this paper, were also employed in our previous work ([5, 6]).

The authors of [8] propose an agent-based modeling of the New York Stock Exchange specialized system. Although this work clearly differentiates from our own work, as the focus is not on the development of an online system incorporating an auction server, but rather on the agent-based modeling of the complex interactions occurring in the New York Stock Exchange specialist system, there are also similarities. First, their modeling addresses a non-trivial class of auctions – continuous double auctions and, second, the modeling could be further expanded to cover the development of an e-service system as part of the New York Stock Exchange.

In paper [7], the authors introduce the e-Game tool that supports the design and implementation of electronic market simulation games inspired by the real life problems. These simulations can also incorporate various types of auctions, and they were used for teaching purposes. The e-Game tool provides both Web and agent interfaces, similarly to our system. Nevertheless, differently from our work, the aspects related to software engineering principles, performance and scalability were not addressed.

In [16], the authors present the principles of constructing

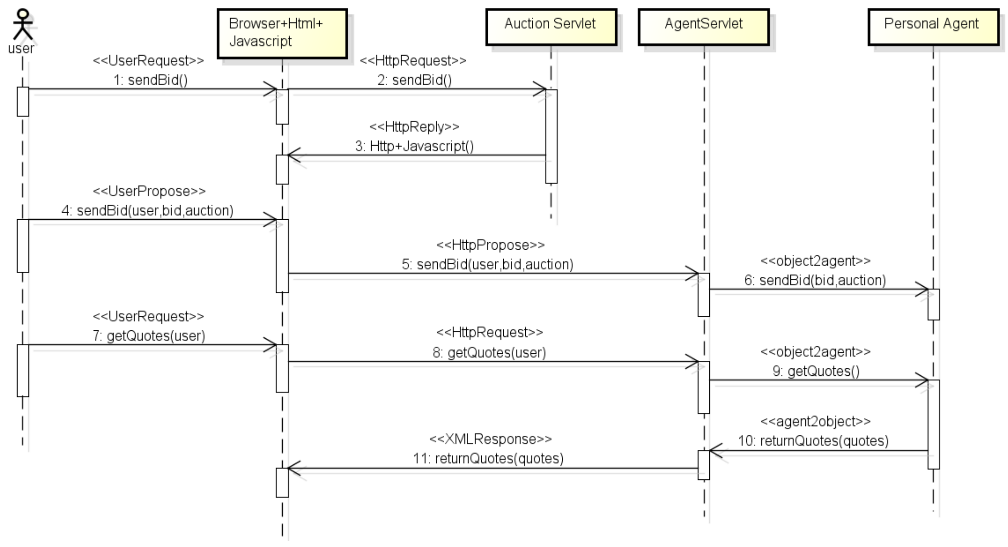
user
Browser+Html+Javascript
Auction Servlet
AgentServlet
Personal Agent
<<UserRequest>> 1: sendBid()
<<HttpRequest>> 2: sendBid()
<<HttpReply>> 3: Http+Javascript()
<<UserPropose>> 4: sendBid(user,bid,auction)
<<HttpPropose>> 5: sendBid(user,bid,auction)
<<object2agent>> 6: sendBid(bid,auction)
<<UserRequest>> 7: getQuotes(user)
<<HttpRequest>> 8: getQuotes(user)
<<object2agent>> 9: getQuotes()
<<agent2object>> 10: returnQuotes(quotes)
<<XMLResponse>> 11: returnQuotes(quotes)

Figure 6: Participating in an auction.

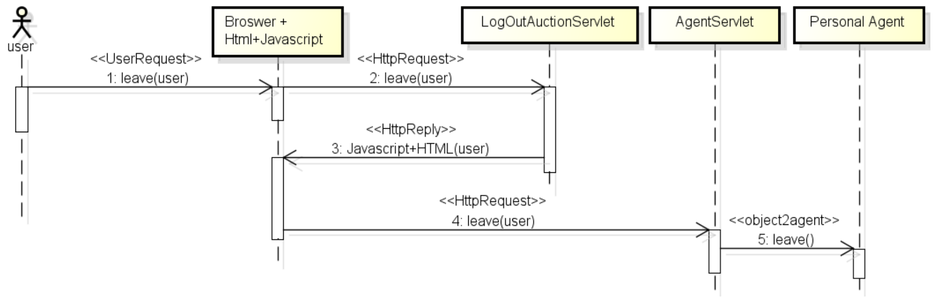
user
Broswer + Html+Javascript
LogOutAuctionServlet
AgentServlet
Personal Agent
<<UserRequest>> 1: leave(user)
<<HttpRequest>> 2: leave(user)
<<HttpReply>> 3: Javascript+HTML(user)
<<HttpRequest>> 4: leave(user)
<<object2agent>> 5: leave()

Figure 7: User logout.

online auction systems that were employed for building the Research Auction Server for performing both simulated and real auctions. However, many details are lacking from their description, especially those related to the interaction protocols. Moreover, although the "agent" metaphor was used for system design, we noticed that the development of the Research Auction Server did not actually use software agent technologies.

A generic online auction server was presented in [20]. The server supports a flexible bidding language based on the OR/XOR formulae. Although, apparently there are many similarities with our own work, the details of the design and implementation of the system are actually lacking; only a listing of available technologies is provided. In particular, the interaction protocols and the details of the interfacing of agent and non-agent software are not described.

In [17], the authors introduce a configurable auction server for resource allocation in the grid. The server design addresses the heterogeneity of the grid environment by allowing the dynamic configuration of the auction mechanism to meet the application requirements. Note that in that work the authors did not highlight the software engineering aspects to support user interaction, as we did in the present paper.

5. CONCLUSIONS AND FUTURE WORK

This paper presents our design and implementation of an online auction system. The system provides a Web-based GUI for the agent-based auction server. We outlined the main functionalities of the system, as well as their design and implementation, in terms of system architecture, design details and interaction protocols. The main outcome of our work is a clean specification of the Web-based and agent-based software layers of our system, as well as of their software interfaces. As future work we plan to expand our design by providing a Web services interface to our agent-based auction server.

6. ACKNOWLEDGMENTS

This work was partially supported by the strategic grant POSDRU/88/1.5/S/50783, Project ID50783 (2009), co-financed by the European Social Fund – Investing in People, within the Sectoral Operational Programme Human Resources Development. Collaboration between Polish and Romanian teams was supported by a bilateral grant "Agent-Based Service Negotiation in Computational Grids".

7. REFERENCES

[1] C. Bartolini, C. Preist, and N. Jennings. A software

framework for automated negotiation. In *Lecture Notes in Computer Science*, volume 3390, pages 213–235. Springer, 2005.

[2] F. L. Bellifemine, G. Caire, and D. Greenwood. *Developing Multi-Agent Systems with JADE*. John Wiley & Sons Ltd, 2007.

[3] M. Benyoucef and S. Rinderle. Modeling e-negotiation processes for a service oriented architecture. *Group Decision and Negotiation*, 15(5):449–467, 2006.

[4] J. Broberg, S. Venugopal, and R. Buyya. Market-oriented Grids and Utility Computing: The State-of-the-art and Future Directions. *Journal of Grid Computing*, 6(3):255–276, 2008.

[5] C. Bădică, M. Ganzha, and M. Paprzycki. Implementing rule-based automated price negotiation in an agent system. *Journal of Universal Computer Science*, 13(2):244–266, 2007.

[6] A. Dobriceanu, L. Biscu, A. Bădică, and C. Bădică. The design and implementation of an agent-based auction service. *International Journal of Agent-Oriented Software Engineering (IJAOSE)*, 3:116–134, 2009.

[7] M. Fasli and M. Michalakopoulos. e-game: A platform for developing auction-based market simulations. *Decision Support Systems*, 44(2):469–481, 2008.

[8] K. Griggs and R. Wild. Intelligent support for sophisticated e-commerce services: An agent-based auction framework modeled after the new york stock exchange specialist system. *e-Service Journal*, 2(2):87–104, 2003.

[9] J. Hillston and L. Kloul. Performance investigation of an on-line auction system. *Concurrency and Computation: Practice and Experience*, 13(1):23–41, 2001.

[10] V. Kelemen. Jade tutorial. simple example for using the jadegateway class, 2006. http://jade.cselt.it/doc/tutorials/JadeGateway.pdf.

[11] A. R. Lomuscio, M. Wooldridge, and N. R. Jennings. A classification scheme for negotiation in electronic commerce. *Group Decision and Negotiation*, 12:31–56, 2003.

[12] A. Ockenfels, D. H. Reiley, and A. Sadrieh. Online auctions. In T. Hendershott, editor, *Economics and Information Systems*, pages 571–628. Emerald Group Publishing, 2006.

[13] D. Rolli, S. Luckner, H. Gimpel, and C. Weinhardt. A descriptive auction language. *Electronic Markets*, 16(1):51–62, 2006.

[14] L. Sandu, R. Sbora, S. Ilie, and C. Bădică. Scalable distributed agent based english auction server. In *Proceedings of the 15th International Conference on System Theory, Control and Computing*, pages 496–501, 2011.

[15] S. Y. W. Su, C. Huang, J. Hammer, Y. Huang, H. Li, Liuwang, Y. Liu, C. Pluempitiwiriyawej, M. Lee, and H. Lam. An internet-based negotiation server for e-commerce. *The VLDB Journal*, 10:72–90, 2001.

[16] J. Trevathan, W. Read, and R. Balingit. Online auction software fundamentals. *International Proceedings of Computer Science and Information technology*, 2:254–259, 2009.

[17] X. Vilajosana, R. Krishnaswamy, and J. M. Marquès. Design of a configurable auction server for resource allocation in grid. In *Proceedings of International Conference on Complex, Intelligent and Software Intensive Systems, CISIS'09*, pages 396–401, 2009.

[18] K. Wasielewska, M. Gawinecki, M. Paprzycki, M. Ganzha, and P. Kobzdej. Optimizing blackboard implementation of agent-conducted auctions. *IADIS International Journal on WWW/Internet*, 6(1):50–60, 2008.

[19] P. R. Wurman, M. P. Wellman, and W. E. Walsh. The michigan internet auctionbot: A configurable auction server for human and software agents. In *Second International Conference on Autonomous Agents, Agents-98*, pages 301–308, 1998.

[20] D.-Q. Yao, H. Qiao, and H. Qiao. A generic internet trading framework for online auctions. In A. Becker, editor, *Electronic Commerce: Concepts, Methodologies, Tools, and Applications*, pages 163–177. IGI Global, 2008.