University of Crete
School of Sciences and Engineering
Department of Computer Science

A Semantic Marketplace of Peers Hosting Negotiating Intelligent Agents

Patkos Theodore

Master of Science Thesis

November 2004

Heraklion, Crete

Μια Αγορά Διαπραγματευόμενων Ευφυών Πρακτόρων Λογισμικού, Δομημένων σε ένα Ομότιμο Σημασιολογικό Δίκτυο

Πάτκος Θεόδωρος
Μεταπτυχιακή Εργασία

Τμήμα Επιστήμης Υπολογιστών
Πανεπιστήμιο Κρήτης

Περίληψη

Τα τελευταία χρόνια παρατηρείται ραγδαία αύξηση στο πλήθος των διαθέσιμων πληροφοριών και υπηρεσιών που προσφέρονται δικτυακά, συμβάλλοντας στη δημιουργία ιδιαίτερα πολύπλοκων διαδικασιών σε ό,τι αφορά την ανεύρεση και εμπορία πόρων. Ο ανθρώπινος παράγοντας εξακολουθεί να είναι ενεργά εμπλεκόμενος σε όλες τις φάσεις διαπραγμάτευσης ηλεκτρονικού εμπορίου. Επιπλέον, η υπάρχουσα οικονομική και εμπορική δραστηριότητα δομείται πάνω σε ένα ανοιχτό, κατανεμημένο, ετερογενές και, συχνά, αναξιόπιστο περιβάλλον. Συνέπεια αυτών είναι οι περισσότερες ερευνητικές προσπάθειες να προσανατολίζονται στην επίτευξη διαλειτουργικότητας μεταξύ των εμπλεκόμενων πλευρών και στην αυτοματοποίηση της επιτέλεσης εργασιών, βασιζόμενες σε κλιμακώσιμες υποδομές.

Η παρούσα εργασία προτείνει μία σχεδίαση που ενοποιεί τρεις σημαντικές τεχνολογίες, για την διευθέτηση θεμάτων των εφαρμογών ηλεκτρονικού εμπορίου επόμενης γενιάς: την τεχνολογία αυτόνομων, ευφυών πρακτόρων λογισμικού, τα ομότιμα δίκτυα και το Σημασιολογικό Ιστό. Η SeMPHoNIA είναι μία αρχιτεκτονική μιας ιδεατής αγοράς πρακτόρων λογισμικού, η οποία χρησιμοποιεί γνώση από RDF βάσεις δεδομένων προϊόντων, σε ένα ανοιχτό ομότιμο περιβάλλον. Η πλατφόρμα προδιαγράφει και υλοποιεί όλα τα βασικά στάδια της διαδικασίας δικτυακών εμπορικών συναλλαγών, υποστηρίζοντας τους χρήστες στην επίτευξη συμφωνιών με αυτοματοποιημένο τρόπο.

Η υλοποίηση της προσέγγισής μας περιγράφεται στο πλαίσιο σεναρίων δημοπρασιών. Η πλατφόρμα σχεδιάστηκε να είναι ιδιαίτερα επεκτάσιμη, ώστε να επιτρέπει τον πειραματισμό με τις τεχνολογίες που εμπεριέχει. Η εργασία αποσκοπεί να επιδείξει τα οφέλη της αρμονικής συνεργασίας των τριών τεχνολογιών. Για το λόγο αυτό και παρουσιάζεται μια εκτενής αξιολόγηση της απόδοσης του συστήματος υπό ποικίλες καταστάσεις ελέγχου.

Επόπτης: Πλεξουσάκης Δημήτριος
Αναπληρωτής Καθηγητής

A Semantic Marketplace of Peers Hosting Negotiating Intelligent Agents

Patkos Theodoros
Master of Science Thesis

Department of Computer Science
University of Crete

Abstract

Recent years have seen an enormous increase in the amount of information and services accessible online, causing the task of discovering and trading resources to become highly complex. Still, human users are actively involved in all phases of the e-Commerce interaction process. Moreover, the current economic trading sphere is structured on top of an open, distributed, heterogeneous and, most often, unreliable environment. As a result, research efforts are oriented towards achieving interoperability between negotiating parties and automation in job execution, based on scalable infrastructures.

This thesis proposes a design that integrates three prominent technologies for addressing issues of next generation e-Commerce applications; autonomous intelligent software agents, peer-to-peer networking and the Semantic Web. SeMPHoNIA (SEmantic Marketplace of Peers HOsting Negotiating Intelligent Agents) is an architecture for an agent-based virtual marketplace, utilizing knowledge from queryable RDF product repositories, in an open peer-to-peer environment. The platform defines and implements all the basic stages of the overall process of e-trading, facilitating human users in closing deals in an automated manner.

The implementation of our approach is demonstrated in the context of auction scenarios. The platform is designed to be highly extensible to allow experimentation with the technologies involved. The standpoint of this thesis is the significance of collaboration of those three technologies. In addition, a thorough evaluation of the system's performance, under various test cases, is demonstrated.

Supervisor: Plexousakis Dimitris
Associate Professor

Ευχαριστίες

Ολοκληρώνοντας μια διετή προσπάθεια στο πανεπιστήμιο Κρήτης αποκόμισα πολύ περισσότερα από έναν τίτλο σπουδών. Αυτό το οφείλω πρωτίστως στον επόπτη καθηγητή μου, κ. Δημήτρη Πλεξουσάκη, τόσο για την ευκαιρία που μου έδωσε να ασχοληθώ με ένα πολύ ενδιαφέρον αντικείμενο όσο και για την ουσιαστική συνεργασία που είχαμε σε όλη τη διάρκεια της πορείας μου. Γνώρισα έναν τρόπο σκέψης και εργασίας που θα αποτελέσει για μένα πρότυπο για τις μελλοντικές μου δραστηριότητες.

Θα ήθελα, επίσης, να ευχαριστήσω δύο άτομα που έδειξαν αμέριστη διάθεση να μοιραστούν μαζί μου χρόνο και γνώσεις, συχνά παραμερίζοντας τη δική τους δουλειά. Τη Σοφία Αλεξάκη και το Σταύρο Σαχτούρη, που οι συζητήσεις μας μου έλυσαν πλήθος αποριών.

Τέλος, ευχαριστώ τους φίλους μου σε Ηράκλειο και Θεσσαλονίκη, αλλά και την οικογένειά μου, στην οποία και αφιερώνω τούτη την εργασία, για τη συμπαράσταση και την ψυχολογική υποστήριξη που μου έδειξαν σε κάθε στιγμή.

List of Figures

List of Tables

Chapter 1

INTRODUCTION

The emergence and rapid development of electronic commerce has influenced many fields of human activity and business industry. Ranging from consumer-to-consumer (C2C) applications to sophisticated business-to-consumer (B2C) and world-wide business-to-business (B2B) transactions, e-Commerce provides a “gravity well” which pulls a variety of diverse technologies and novel research efforts into closer collaboration. Recent years have seen an enormous increase in the role of information technology in markets, in particular the emergence of electronic marketplaces [16], which represent places where clients and suppliers meet to communicate, exchange information and do business. This is a dynamic and constantly evolving field, revealing characteristics and conventions that differ from those employed in the offline economy market. As electronic negotiation and trading becomes part of our daily life changing the way customers and retailers interact, the need for identifying key aspects of this fluid environment is substantial for harnessing its full potential and developing efficient applications across the global market.

The current economic trading sphere is structured on top of an open, distributed, heterogeneous and most often unreliable and unpredictable environment. Companies need scalable infrastructures, integrated information systems and flexible tools to react to changing conditions, exploit technological developments and extend the scope of merchant impact in such a dynamic market. Customers, on the other hand, participate in increasing numbers in e-Commerce transactions, modulating a plethora of requirements. The study on shopping experience has concluded in numerous consumer buying behavior (CBB) models, such as the Bettman model [15], the Howard-Sheth model [47], the Engel-Blackwell model [28], and the Andreasen model [3], which all recognize a similar list of six fundamental stages:

a) consumer requirement identification,
b) product brokering,
c) merchant brokering,
d) negotiation,
e) purchase execution and delivery and
f) after-sale services and evaluation.

Human participants are still actively involved in all stages of the buying process. As the trend of e-Commerce continues though, an inevitable growth in the number and features of on-line markets is observed, causing the task of monitoring and effective decision-making to become complex and time-consuming for humans. By increasing the degree of heterogeneity and sophistication on both the business and the customer side, interoperability and automation of execution are going to be the most challenging tasks that next generation e-Commerce applications will face. Without the ability to interpret correctly the meaning of negotiation terms and to support robust and dynamic execution in all trading stages, participants will lack the equipment to handle the conditions of the competitive international e-market climate.

As an enabling step towards achieving interoperation of data and automation of services in open dynamic environments, researchers adapt different approaches proposing new technologies or enhancing the existent. Prominent technologies are the intelligent software agents, peer-to-peer networks and the Semantic Web.

1.1 Agent Technology

We agree with Klusch's appraisal [59] stating that with the advance of computing technology in terms of computational speed and popularity, intelligent software applications known as agents, will become a potential area of development for e-business. It is widely anticipated that agent technology, supple as it is, may have a profound effect in the process of a typical e-shopping scenario. Agents are autonomous software entities that accomplish tasks on behalf of their owner. They exhibit certain properties, as identified in [102], such as proactivity, reactivity, autonomy, social ability and mobility, which differentiate them from traditional software. Specialized agents, revealing deliberation and autonomous behavior, may represent the users, the offering services or the information resources addressing issues that may range from simple and iterated procedures to highly complex and important tasks. However, while agent technology represents a promising medium for conceptualizing negotiations and implementing software, understanding its limitations is equally important. To realize their full potential, agents need to function in groups, elaborating forms of cooperation and social activity. Jennings in [50] has identified the need of understanding the impact of sociality, which influences the link between the behavior of the individual agents and that of the overall system. For the efficient implementation of such behavior certain hurdles have to be overcome. The society of agents that synthesize a multi-agent system (MAS) needs to have a shared understanding of the rules and the permitted actions that describe the conditions of the negotiations taking place. Advanced agent-communication languages (ACL), conversation policies and negotiation protocols constitute an important topic in allowing meaningful communication and interaction between heterogeneous agents in modern marketplaces.

1.2 Semantic Web Technology

The most essential step towards enabling interoperability in open environments is attempted by the emergence of the Semantic Web. Communication between entities, such as software agents or web services that belong to diverse providers, requires a common apprehension of the vocabulary and the terms of participation in the negotiation session. Moreover, current discovery mechanisms depend on keyword searching, rather than matching the capabilities of the resources with the meaning of the queries, resulting in inefficient schemes. These issues refer to what is called the semantic aspect of information. Tim Berners-Lee et al. in [14] envision the Semantic Web as an extension of the current web, in which information is given well-defined meaning, better enabling computers and people to work in cooperation. By defining the semantics of terms and resources, information is given uniform and concise description, independent of the syntactic representation, widening its accessibility from closed groups to open and diverse societies. The semantic representation of knowledge is captured and shared at an ontological level, which formalizes terminology by defining resources, concepts and relations between them. Semantic Web technology is expected to influence many fields of Web activity, but the impact in e-Commerce transactions specifically will revolutionize the design and development of on-line trading applications.

1.3 Peer-to-Peer Technology

Still, to move to next-generation e-Commerce applications, we need to improve the performance of the networks that represent the medium through which all stages of e-trading are evolving. Current systems are based on centralized infrastructures that suffer from a number of drawbacks, because they introduce single points of failure, expose vulnerability to malicious attacks and generate performance bottlenecks and hotspots on the network. Peer-to-peer systems are considered the next evolutionary step in networking since they comprise features that address problems of traditional centralized approaches. Their paradigm was made popular as file-sharing applications, such as Napster [68] and Gnutella [41], but peer-to-peer networks are much more than that. Their essence, compared to client-server architectures, is that the roles of provider and requester of services or resources are interchangeable between all nodes in the network and these nodes directly connect with others to exploit their resources without central intervention of any kind (server, broker etc.). Their reliability and high connectivity make them suitable for dynamic environments and for a variety of applications. Such applications could range from information exchange and resource sharing, two crucial functions of present-day organizations, but difficult to exploit due to their increasingly decentralized nature [92], to query routing or, even, Web Service discovery infrastructures, which currently rely on centralized schemes, such as UDDI registries [95]. On the other hand, it is natural for such a rapidly developing technology to suffer from certain problems, primarily related to scalability, as well as search times and completeness on the network. Research efforts are oriented towards new peer-to-peer architectures and efficient broadcast mechanisms to alleviate those problems.

1.4 State-of-the-art Approaches on Electronic Negotiation and Trading

Up until now, many researchers have identified both the prospective benefits and the limitations of the aforementioned technologies, working towards improved approaches. Experience has revealed that, in most cases, the proper combination of the strength of state-of-the-art techniques may eliminate drawbacks that appear by each of them individually, while giving an impetus to their advantages. Previous studies have demonstrated the profits of such consolidations. Indeed, projects like Nuin [101], TAGA [109], O_3F [67], ITTALKS [56] or [91] promote the use of agents as an enabling technology for exploiting and delivering semantic services and descriptions. Moreover, Semantic Web combined with peer-to-peer computing is considered a highly innovative, as well as, a requisite step towards information retrieval purposes in open, dynamic environments. In the Edutella [69], METEOR-S [96] and InfoQuilt [6] projects the importance of metadata descriptions is demonstrated, when querying or searching for information resources stored on numerous peers on a network. We should not, however, disregard the benefits in functionality and performance, when structuring peer-to-peer architectures that utilize the abilities of software agents for accomplishing intelligent tasks. Homayounfar et al. in [46] claim that agent technology is the crossing point where Artificial Intelligence and distributed systems meet each other.

1.5 Motivations and Goals

This thesis introduces the design and implementation of a system, called SeMPHoNIA, for addressing many of current e-trading issues. The SeMPHoNIA project attempts to proceed one step beyond the state-of-the-art. Our goal is to integrate and exploit all three technologies, namely intelligent software agents, peer-to-peer systems and the Semantic Web, into a unified platform for demonstrating the many benefits that arise from their collaboration. SeMPHoNIA is an architecture for an agent-based virtual marketplace and is intended as a platform for research in multi-agent systems, ontologies, peer-to-peer networking and automatic negotiation, as well as, an application for experimentation with these technologies. The platform defines and implements the basic stages of the overall process of e-trading by closing deals in a semi-automated manner, utilizing knowledge from queryable product repositories in an open peer-to-peer environment. SeMPHoNIA could be considered as what [45] describes as the third key actor in agent-mediated e-Commerce applications, apart from buyers and sellers; the “market owner”, an environment that sets and controls the rules, in which buyers and sellers trade. We want the platform to be a useful device for researchers practicing new approaches, therefore flexibility and extensibility were essential features of our design.

The implementation of our approach is demonstrated in the context of auction scenarios. On-line auctions represent a more specific type of negotiation governed by predefined protocols and have rapidly achieved enormous popularity as a common interaction medium both for humans and software agents on the internet. In SeMPHoNIA, users are offered the ability to participate in multiple auctions at the same time, when the result of one auction may affect the action taken for the other. The platform is intended to facilitate users in discovering and bidding across multiple auctions with varying start and end times and varying protocols, attempting to ensure that at most one of the desired items will be purchased agreeing the best, for the customer, deal. Our implementation currently supports three types of auctions, namely English, Vickrey and a hybrid peer-to-peer auction, featuring a variety of characteristics (i.e., single/multi-round, centralized/non-centralized, private/public value, first/second price). SeMPHoNIA utilizes peer-to-peer architecture and semantic components as a mechanism for auction discovery, as well as, intelligent agents for participation and automatic negotiation in these auctions. Still, auction scenarios are just a paradigm of multi-agent negotiation. Our intention is to encourage researchers and application designers to explore and experiment with aspects of other domains that go beyond auction theory, such as automated negotiation in general, Semantic Web, e-Commerce, publication and discovery of resources and Web Services etc..

In the following sections we will describe the design and implementation of SeMPHoNIA platform, compare it to existing approaches and measure its performance through various test cases. Chapter 2 summarizes the basic terms and principles of agent technology, Semantic Web, peer-to-peer systems and on-line auction houses. It, also, presents work carried out in similar projects to ours. A detailed analysis of SeMPHoNIA's architecture is described in Chapter 3. It covers issues concerning its structure, design, basic components and examples of interaction. Additional functionalities of the platform are presented in Chapter 4. Chapter 5 explains the basic steps for creating auction services and exploiting the system's features. In includes screenshots and walkthrough examples. Because special care has been given in providing an extensible design to our platform, Chapter 6 identifies aspects of SeMPHoNIA's architecture that can be extended to integrate even broader characteristics and allow experimentation with numerous issues. Chapter 7 explains certain implementation parameters allowing the reader to shape a complete image of the system. An evaluation of the platform's performance is presented in Chapter 8, attempting an interpretation of the results obtained. Finally, Chapter 9 concludes with some final remarks and gives suggestions for future work.

Chapter 2 BACKGROUND AND RELATED WORK

This section reviews the basic terms and principles of technologies used in this thesis. It explains the general features of software agents, Semantic Web and peer-to-peer systems and describes existing Internet-based auction sites, identifying their limitations. In addition, a survey of related work is presented, attempting a comparison with our platform.

2.1 Agent Technology and Standards

Recently there has been much interest in the role of dynamic negotiation in electronic business transactions. Negotiations in industries are often inefficient due to the diversity of intellectual background of the negotiation parties, the many variables involved and the complex interactions. For such negotiations to be effectively automated, certain issues have to be addressed. Multi-agent systems (MAS) offer an innovative approach towards reducing the tremendous time and human resources invested in negotiations, since they are particularly suitable for resolving fragmented mechanisms.

2.1.1 Defining Software Agents

Since the beginning of recorded history, people fascinated with the idea of non-human agencies that would be appointed the task of accomplishing trivial objectives on behalf of their human owners (popular examples include androids, humanoids, robots and many others). Software agents represent the implementation of such notions in software technology and, in particular, in distributed systems and artificial intelligence.

Although no standard definition for software agents has been provided, one could comprehensively define agents as: *entities (software programs) in a system that can react autonomously (without user's interference) to an incoming task (problem) from another object, make rational decisions and return proper results (solution).* Agents may be autonomous and intelligent entities, which reside on nodes on a network, get the problem from the users, discover resources and services, consult with each other, offer solutions, learn from past experience and update their knowledge based on an autonomous conclusion [34].

Software agents exhibit the following characteristics that differentiate them from other software programs:

- They perform their task autonomously, being capable of making decisions about what actions to take without constantly referring to the user.

- They are knowledgeable about their objectives.

- They are proactive, responding appropriately to the prevailing circumstances in dynamic and unpredictable environments [102].

- They are reactive, acting in anticipation of future goals and available information and not driven directly by commands [102].

- They are able to adapt, learn, modify their behavior and update their knowledge bases though experience (learning process) [98].

- They elaborate forms of social interaction, consulting each other and working together in performing a job [102], [64].

- They are mobile, moving around an electronic network.

- They break a problem into low-level tasks and offer a solution compiled from the different results.

- They are deliberative, acting intelligently based on rational conclusions [103].

In addition to this, certain assumptions have to be made when building a multi-agent system [102]:

- veracity is the assumption that an agent will not knowingly communicate false information,

- benevolence is the assumption that agents do not have conflicting goals, and that every agent will therefore always try to do what is asked of it

- rationality is the assumption that an agent will act in order to achieve its goals, and will not act in such a way as to prevent its goals being achieved – at least insofar as its beliefs permit

2.1.2 Comparison with closed systems

In general the technology of autonomous software agents is most suitable for applications in which knowledge can be well formulated. We can identify certain advantages that agent-based computing offers in comparison to closed systems:

- Higher productivity: With agents the amount of work that users can accomplish is much higher than with a closed system.

- Distributed computing: With agent-based computing the idea of distributed computing becomes possible. Applications can be developed, made up of multiple agents, that perform their specified tasks over a network of computers.

- Economical: Instead of having a single computer with extremely large computing power developers can use agent-based computing to utilize a network of computers with the equivalent computing power.

- Less network traffic: With today's RPC model commands are scattered across the network from computer to computer, increasing network traffic. With agents, users can send once a relatively small agent across the network to perform the work.

On the other hand, agent technology still has a long way to cover before achieving an optimum shape, especially in the following fields:

- Security: Migrating agents can be proved to be a Trojan horse for the security of a system. The idea of allowing a program that could potentially destroy the contents of a computer is very troubling. If agent-based computing is to succeed, measures need to be taken to ensure that agents are unable to harm their host computer.

- AI improvement: More work needs to be done on the "intelligence" of these agents. For achieving high degree of automation many more issues need to be addressed and improved reasoning algorithms need to be designed.

- Semantics of negotiation: Although attempts for standardizing the agent communication languages have been made, they fail to capture the semantics of negotiation between foreigner agents that desire to interact.

A more general and flexible framework is necessary for allowing agents to dynamically negotiate the protocol of their interaction.

2.1.3 Agent-to-Agent Architecture

Multi-agent systems can be structured in numerous ways in order to achieve the level of decentralization, cooperation and job fragmentation that they desire. Approaches differ greatly in features and performance (i.e., brokering-based, distributed etc.). However, the most recent and promising architecture is the one that combines characteristics of both the peer-to-peer and agent model and is called agent-to-agent architecture [46]. SeMPHoNIA has been implemented following the general principles of that particular approach.

In an agent-to-agent environment, each node of the network can be a host for one or more agents. Each agent can have a point-to-point communication with other agents within the network. An agent-to-agent system is an advanced version of the Internet agent technology, in which agents have more flexibility and efficiency compared with the ordinary agents' models. In fact, agent-to-agent architecture is an agent-based model designed and implemented by advanced peer-to-peer features. An autonomous agent-to-agent design of a peer-to-peer system may overcome the limitations of the current peer-to-peer applications and improve the efficiency of its components [92].

The main characteristics of agent-to-agent architectures are:

- Virtual peer-to-peer platform
- Agents reside on nodes
- Agents have peer-to-peer operations and abilities

2.1.4 Agent Standards

Nowadays, the most interesting standardization efforts in the multi-agent area are undertaken by the Foundation for Intelligent Physical Agents (FIPA) [33]. FIPA is an international organization dedicated to promoting the industry of intelligent agents by openly developing specifications supporting interoperability among agents and agent-based applications.

FIPA specifications cover:

- FIPA Agent Message Transport: deals with the delivery and representation of messages across different network transport protocols.

- FIPA Agent Management: framework within which FIPA agents exist and operate. Establishes the logical reference model for the creation, registration, location, communication, migration and retirement of agents.

- FIPA Agent Communication: specifies the ACL (Agent Communication Language), along with libraries of predefined communicative act types, interaction protocols and content languages.

- FIPA Applications: contain service and ontology descriptions and case scenarios for agent-based applications.

Although FIPA uses AUML to represent its standard interaction protocols, many other projects utilize coloured Petri nets (CPNs) [60], because their formal properties facilitate the modeling of concurrent conversations in an integrated fashion. Examples of such projects are described in [100], [21], [27], [30] and [76].

In addition, the most important standardization body in the area of mobile agents is the Object Management Group (OMG) with its Mobile Agent System Interoperability Facility (MASIF) specification. The MASIF standard has been adopted as OMG technology in February 1998. It comprises several important aspects, such as agent management, mobility, naming and tracking.

2.1.5 Agent Communication

Communication between software agents is an active area of research for the agent community. The possible heterogeneity of the agents requires the use of a common set of messages, understandable by all involved parties.

A popular method of communication is by applying a commonly agreed language, namely an agent communication language (ACL). An ACL provides language primitives that implement the agent communication model. The two main ACLs – both from theoretical and practical use – are KQML [31] and FIPA-ACL [32]. Both borrow from traditional speech-act theory; messages of the agents are considered as actions with consequences on the environment [65]. Since speech acts are a human knowledge-level communication protocol, it is felt that they would be effective as an agent communication protocol, especially since agents might operate on behalf of humans. In principle, we can distinguish the ACL (e.g., KQML, FIPA-ACL) at the speech act level and the content language, which is used by the ACL (e.g., KIF, SL).

Apart from ACLs, other methods for information exchange between agents have been developed. The most widely used are the traditional Remote Procedure Calls (RPC), Java's Remote Method Invocation (RMI) and plain socket connections. Moreover, CORBA is a MASIF-compliant interface for remote interactions. It implements two protocols, the CORBA Internet Inter-ORB Protocol (IIOP) and the MAF IIOP, that provide the connectivity between agent systems of different vendors.

2.2 Semantic Web Standards and Technologies

For many people, the World Wide Web has become an indispensable means of providing and searching for information. Searching the Web in its current form is, however, often an infuriating experience since today's search engines usually provide a huge number of answers, many of which are completely irrelevant, whereas some of the more interesting answers are not found. One of the reasons for this unsatisfactory state of affairs is that existing Web resources are usually only humanly understandable: the mark-up (HTML) only provides rendering information for textual and graphical information intended for human consumption [7]. The Semantic Web [14] aims for machine-interpretable Web resources, whose information can be shared and processed both by automated tools, such as search engines, and human users.

2.2.1 A Layered Approach

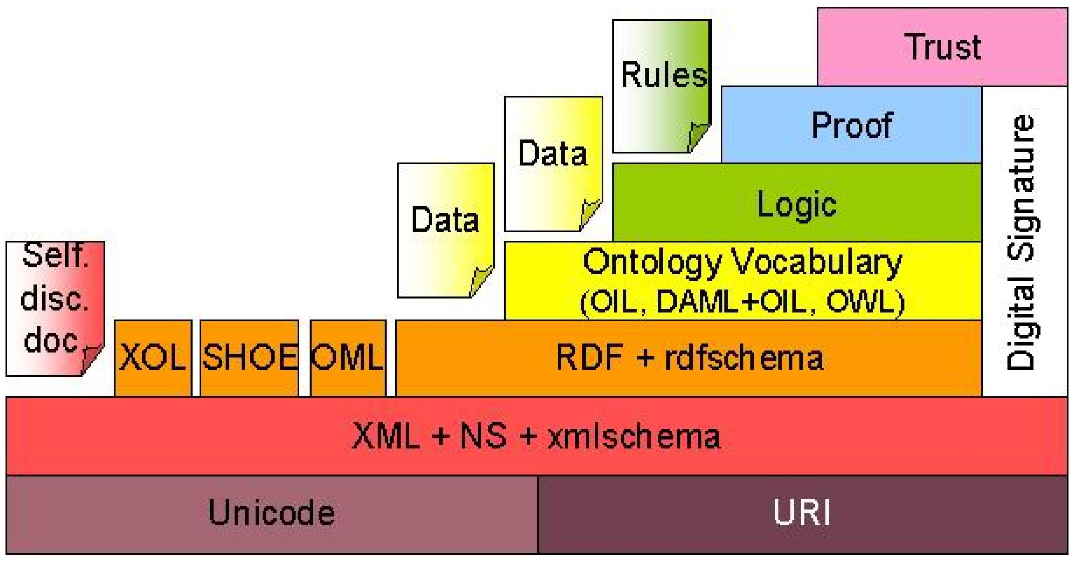


Figure 1 A layered approach to the Semantic Web.

The development of the Semantic Web proceeds in steps, each step building a layer on top of another (see figure 1). In such a structure there are some principles that should be followed [5]:

- Downward compatibility: Agents fully aware of a layer should also be able to interpret and use information written at lower levels. For example, agents aware of the semantics of OWL can take full advantage of information written in RDF and RDF Schema.

- Upward partial understanding: On the other hand, agents fully aware of a layer should take at least partial advantage of information at higher levels. For example, an agent aware only of the RDF and RDF Schema semantics can interpret knowledge written in OWL partly, by disregarding those elements that go beyond RDF and RDF Schema.

2.2.2 XML, DTD and XML Schema

The Extensible Markup Language (XML) is the universal format for structured documents and data on the Web. XML is designed to describe document types for all thinkable domains and purposes. The success of XML is primarily based on its flexibility: everybody can write a document type definition (DTD) or XML Schema to define the structure of XML documents that represent information in the form s/he desires. The purpose of a Document Type Definition is to define the building blocks of an XML document. It defines the document structure with a list of allowed elements. The same holds for XML Schema – it only defines structure, though with a richer language.

XML lets everyone create her/his own tags that annotate Web pages or sections of text on a page. Programs can make use of these tags in sophisticated ways, but the programmer has to know what the page writer uses each tag for. In short, XML allows users to add arbitrary structure to their documents but says nothing about what the structures mean. Meaning of XML-documents is intuitively clear, due to “semantic” mark-up and tags, which are domain-terms. However, computers do not have intuition. Tag-names do not provide semantics.

In essence, XML lacks a semantic model: it has only a “surface model", a tree. So, XML is not the solution for propagating semantics through the Semantic Web. It can only play the role of a "transport mechanism"; an easily machine-processable data format. Applications not only have to be aware of the DTD or XML Schema defining a class of documents, they must also be informed about the underlying semantics of tags and the meaning of the document structure. But this semantics is outside the scope of XML.

2.2.3 RDF and RDFS

The Resource Description Framework (RDF) [79] is an infrastructure that enables encoding. RDFS [18] is an abstract data model that defines relationships between entities (called resources in RDF). Statements in RDF describe resources that can be web pages or surrogates for real world objects like publications, pieces of art, persons or institutions. RDF, in combination with RDFS, offers modelling primitives that can be extended according to the needs at hand. Basic class hierarchies and relations between classes and objects are expressible in RDFS. In general, RDFS suffers from a lack of formal semantics for its modelling primitives, making interpretation of how to use them properly an error-prone process.

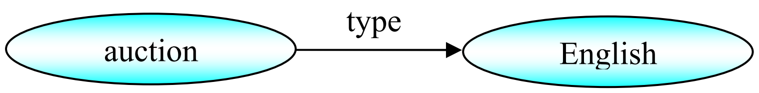


Figure 2 An example of RDF statement.

Everything in RDF is a resource (see figure 2). Resources may be related to each other or to literal (i.e., atomic) values via properties. Such a relationship represents a statement that itself may be considered a resource, i.e. reification is directly built into the RDF data model [61]. Thus, it is possible to make statements about statements. These basic notions can be easily depicted in a graphical notation that resembles semantic nets.

As a companion standard to RDF, the schema language RDFS is more important with respect to ontological modelling of domains. RDFS offers a distinguished vocabulary defined on top of RDF to allow the modelling of object models with cleanly defined semantics. The terms introduced in RDFS build the groundwork for the extensions of RDFS.

We chose RDF for SeMPHoNIA project because it is a clearer, simpler and more expressive language for labelled directed graphs than basic XML and is easily encoded in XML. Moreover, a set of well defined query languages for RDF have been developed that feature support concerning many issues. Such languages include RQL, which is utilized in our system, RDQL, Triple, SeRQL, Versa, N3 and others. Finally, RDF is supported by a wide range of stable tools for creating, validating and storing ontologies, such as the ICS-FORTH RDFSuite.

2.2.4 Ontologies

Ontologies are a key enabling technology for the Semantic Web. They interweave human understanding of symbols with their machine processability. More recently, the concept of ontology is also becoming widespread in fields, such as intelligent information integration, cooperative information systems, information retrieval, e-Commerce, and knowledge management. The reason that ontologies are becoming so popular is largely due to what they promise: a shared and common understanding of a domain that can be communicated between people and application systems. In a nutshell, ontologies are formal and consensual specifications of conceptualizations that provide a shared and common understanding of a domain, an understanding that can be communicated across people and application systems. Thus, Ontologies glue together two essential aspects that help to bring the web to its full potential [1]:

- Ontologies define formal semantics for information, consequently allowing information processing by a computer.

- Ontologies define real-world semantics, which makes it possible to link machine processable content with meaning for humans based on consensual terminologies.

Quite a large number of representation languages for representing ontologies on the web have been established over the last decade. Such languages are OIL, DAML+OIL and OWL and will be described next.

2.2.5 OIL

OIL, developed in the OntoKnowledge project, permits semantic interoperability between Web resources. Its syntax and semantics are based on existing proposals (OKBC, XOL and RDF(S)). OIL provides modeling primitives commonly used in frame-based approaches (concepts, taxonomies of concepts, relations etc.), as well as formal semantics and reasoning support found in description logic approaches (a subset of first order logic that maintains a high expressive power, together with decidability and an efficient inference mechanism).

OIL, built on top of RDF(S), has the following layers: *Core OIL* groups the OIL primitives that have a direct mapping to RDF(S) primitives; Standard *OIL* is the complete OIL model, using more primitives than the ones defined in RDF(S); *Instance OIL* adds instances of concepts and roles to the previous model; and *Heavy OIL* is the layer for future extensions of OIL. OIL's syntax is not only expressed in XML but can also be presented in ASCII.

OIL draws on three roots, Frame-based Representations, Description Logics and Web based languages. The frame-based approach employs modeling primitives based on classes (frames) with certain properties, known as attributes, which have local, rather than global, scope and are applicable to the classes they are defined for. Frames thus supply what is arguably a "natural" style and a "friendly" face to the modeler, but suffer from a lack of a well-defined semantics. Description Logics describe knowledge in terms of concepts and role restrictions to automatically derive classification hierarchies. They supply a range of concept forming operators (conjunction, disjunction, negation etc) and they also have high expressive power providing formal semantics and reasoning support. OIL inherits those characteristics and adds extra mechanisms, such as recursive class definitions and more general axioms.

2.2.6 DAML+OIL

DARPA Agent Markup Language + OIL [26] has been developed by a joint committee from the US and the European Union (IST) in the context of DAML, a

DARPA project for allowing semantic interoperability in XML. Hence, DAML+OIL shares the same objective as OIL.

DAML+OIL is built on top of RDF(S). Its name implicitly suggests that there is a tight relationship with OIL. It replaces the initial specification, which was called DAML+ONT and was also based on the OIL language. DAML+OIL has moved away from the original frame-like ideals of OIL and is, in a much stronger sense than OIL, an alternative syntax for Description Logic [1]. The idea of a "frame" is not inherent in the language and assertions in DAML+OIL are couched in terms of axioms.

2.2.7 OWL

OWL [72] is a language currently being standardized by the World Wide Web Consortium for defining Web ontologies and their associated knowledge bases [97]. In OWL, an ontology is a set of definitions of classes and properties, and constraints on the way those classes and properties can be employed. An OWL ontology may include the following elements: taxonomic relations between classes, datatype properties (descriptions of attributes of elements of classes), object properties (descriptions of relations between elements of classes), instances of classes and instances of properties.

OWL is a set of three, increasingly complex languages: *OWL Lite*, designed to satisfy users primarily needing a classification hierarchy and simple constraint features; *OWL DL*, which includes the complete OWL vocabulary interpreted under a number of simple constraints (DL corresponds to Description Logics); and *OWL Full*, which includes the complete OWL vocabulary, interpreted more broadly than in OWL DL.

2.3 Peer-to-Peer Networks

Peer-to-peer technology enables any network-aware device to provide services to another network-aware device. A device in a peer-to-peer network (figure 3) can provide access to any type of resource that it has at its disposal, be it documents, storage capacity, computing power, or even its own human operator. The peer-to-peer technology is a natural extension of the Internet's philosophy of robustness through decentralization. In the same manner that the Internet provides domain name lookup (DNS), World Wide Web, email, and other services by spreading responsibility among millions of servers, peer-to-peer has the capacity to power a whole new set of robust applications by leveraging resources spread across all corners of the Internet.

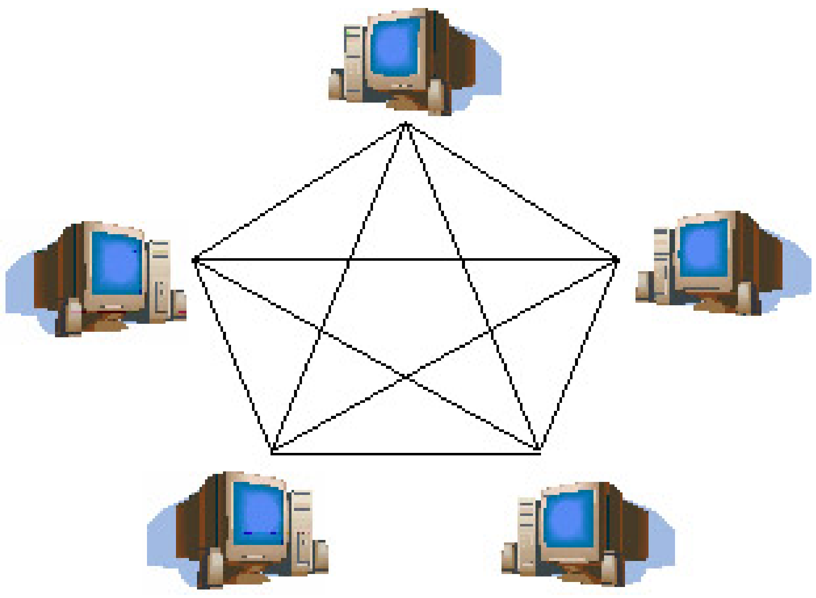


Figure 3 Peer-to-peer network.

For a better understanding on how and why an enhanced peer-to-peer architecture can play a better role in data communication and the Internet, an introduction to the traditional client/server architecture is presented.

2.3.1 Traditional Client/Server Architecture

Most Internet services are distributed using the traditional client/server architecture. In this architecture, clients connect to a server using a specific communications protocol to obtain access to a specific resource. Most of the processing involved in delivering a service usually occurs on the server, leaving the client relatively unburdened. Most popular Internet applications, including the World Wide Web, FTP, telnet, and email, use this service-delivery model.

For a harmonic and synchronized communication and data transferring among members on a network there must be some protocol to be followed. Some of the most common network protocols are:

- TCP/IP: It is the basic protocol used for data communication on the Internet and is used for implementing many other protocols. In TCP/IP data are split and coded into small "packets", which can be sent and received in a point-to-point architecture.

- HTTP: It is a convention in which Hypertext files (such as HTML files) can be downloaded from a server by clients. In Hypertext files each component (sentence or image) can point to other components and can also carry a runnable code as attachment (e.g. Java script files). These types of files should obey HTML for their structure. This method is very suitable for users with unreliable connections and also when the traffic is high, because clients change the server fast.

- FTP: This is a protocol designed for file transfer on the Internet. In this protocol any kind of file can be transferred between two nodes regardless of its content. An FTP server works faster than an HTTP server but it gives much less options to users.

Distributed systems, most of which currently work based on the client/server architecture, are an important means of data communication. Their main characteristic is that the components (hardware and software) are scattered on the network, offering resource sharing, parallel processing, higher reliability, extensibility etc. Some of the most popular technologies for developing distributed systems today involve socket, Remote Procedure Call (RPC), Java Remote Method Invocation (RMI), Common Object Request Broker Architecture (CORBA), Distributed Component Object Model (DCOM), Java Database Connectivity (JDBC), JINI.

2.3.2 Peer-to-Peer Architectures

Peer-to-peer networking grants individual machines a mechanism for providing services to each other. Peer-to-peer networks shun the centralized organization of the client/server architecture and, instead, employ a flat, highly interconnected architecture. By allowing intermittently connected computers to find each other, peer-to-peer systems enable these machines to act as both clients and servers that can determine the services available on the network and engage those services in some application-specific manner.

Most of the traditional LAN (and later the Internet) architectures have been based on the client/server concept, in which one or more computers (servers) on the net are designated to serve the other computers (clients). Although this architecture is still being widely used, it cannot fully satisfy the requirements of communication between two endpoints on the Internet. Since a server must always be present in the client/server architecture and a client can only contact with the server, direct communication between two clients is difficult. However, direct communication is important for many applications.

In peer-to-peer, users can communicate with, request help from, and serve each other without a central server. For many applications, this solution is more suitable and more efficient than the client/server architecture. Hence, peer-to-peer architecture has started to be widely adopted by designers and developers. The number of programs that are based on peer-to-peer architecture is increasing, because quite often users prefer to talk to each other more freely and be less dependent on servers for resources and services.

The main advantage of peer-to-peer networks is that they distribute the responsibility of providing services among all peers on the network; this eliminates service outages due to a single point of failure and provides a more scalable solution for offering services. In addition, peer-to-peer networks exploit available bandwidth across the entire network by using a variety of communication channels and by filling bandwidth to the "edge" of the Internet. Unlike traditional client/server communications, in which specific routes to popular destinations can become overtaxed, peer-to-peer enables communication via a variety of network routes, thereby reducing network congestion. Peer-to-peer has the capability of serving resources with high availability at a much lower cost while maximizing the use of resources from every peer connected to the peer-to-peer network. Whereas client/server solutions rely on the addition of costly bandwidth, equipment, and co-location facilities to maintain a robust solution, peer-to-peer can offer a similar level of robustness by spreading network and resource demands across the network.

From the above, we can conclude that peer-to-peer and client/server architectures differ from each other in the following aspects:

- In a peer-to-peer system communication takes place between two peers, whereas in a client/server system clients communicate only with the server.

- A peer-to-peer architecture treats all nodes equally and grants them equal privileges. In a client/server architecture, a server is a super node and other clients should follow its dominance.

- Users in a client/server system can only use server resources, whereas in a peer-to-peer system they can use other users' resources as well.

- The quantity of information held by members in a peer-to-peer system may be much bigger than that by a single server.

- Because of parallel operations, more tasks can be processed by a peer-to-peer system than by a single server.

- The waiting lists for using resources or services may be reduced in a peer-to-peer system.

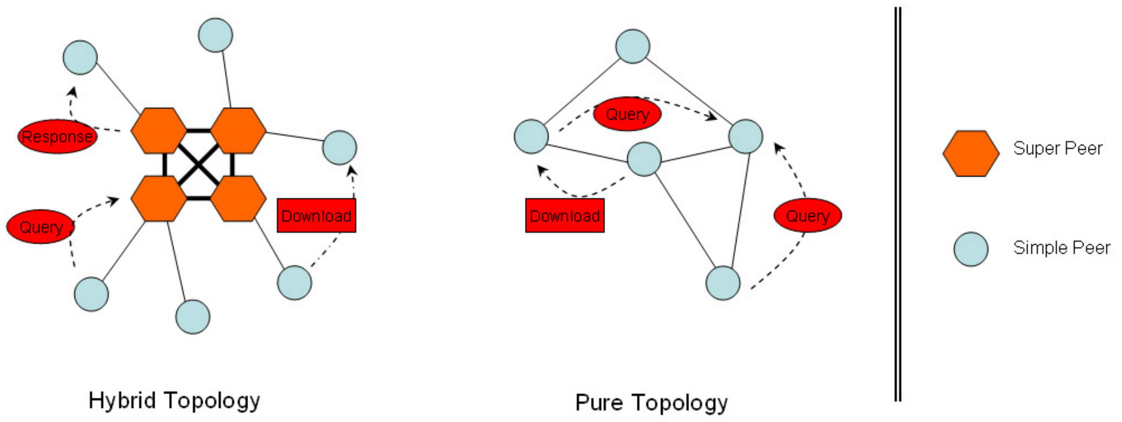


Figure 4 Basic peer-to-peer topologies.

There is as yet no concerted classification of peer-to-peer topologies, despite the fact that many attempts have been made to provide peer-to-peer network definitions and models ([1], [83]). One possible classification could consider the level of centralization of the peer-to-peer network. We can distinguish many types of peer-to-peer systems according to the level of centralization (see figure 4). In *pure* systems, such as Gnutella [41] and Freenet [36], all peers have equivalent roles and responsibilities in every issue. In *hybrid* models, such as Napster [68], a central index is present to preserve an image of all resources, but file sharing and transfer continues to occur in direct manner. Finally, *super-peer* networks, such as KaZaA, instantiate an intermediate approach between the previous two models. Super-peers are responsible to receive and forward queries, simulating the hybrid model, but their connection with other super peers is structured following the pure technique.

Nevertheless, peer-to-peer systems suffer from some disadvantages due to the redundant nature of a peer-to-peer network's structure. The distributed form of communications channels in peer-to-peer networks results in service requests that are nondeterministic in nature. For example, clients requesting the exact same resource from the peer-to-peer network might connect to entirely different machines via different communication routes, with different results. Requests sent via a peer-to-peer network might not result in an immediate response and, in some cases, might not result in any response. Resources on a peer-to-peer network can disappear at times as the clients that host those resources disconnect from the network; this is different from the services provided by the traditional Internet, which have most resources continuously available.

However, peer-to-peer can overcome all these limitations. Although resources might disappear at times, a peer-to-peer application might implement functionality to mirror the most popular resources over multiple peers, thereby providing redundant access to a resource. Greater numbers of interconnected peers reduce the likelihood that a request for a service will go unanswered. In short, the very structure of a peer-to-peer network that causes problems can be used to solve them.

2.4 On-line Auctions

Auctions, as a market mechanism, were already introduced in the ancient world by the Greeks and the Romans. Their use for determining the price and other terms of an exchange has always been popular, yet with the emergence of the Internet, an upward growth has been observed.

2.4.1 Defining Auctions

McAfee and McMillan [66] define auctions as a market institution with an explicit set of rules determining resource allocation and prices on the basis of bids from the market participants. Auctions are considered a very efficient and effective method of allocating goods or services, in dynamic situations, to the entities that value them most highly [104] and many believe that their paradigm constitutes an important economic model. Indeed, Paul Klemperer gives three reasons why the theory of auctions is relevant to economists [58]:

- A growing number of large volume transactions are realized through auctions. Examples include the many auctions organized by governments, the enterprise wide electronic procurement systems for all sorts of goods or the electronic auction markets.

- Auctions offer relatively simple mechanisms in a well-defined economic environment. They offer thus to economists a vast field for experimental research allowing to obtain empirical results that can be suitably validated.

- The theory of auctions has already revealed many scientific results in economy, which have allowed to develop new methods for pricing in competitive markets. Moreover, it has also helped to further understand complex negotiation mechanisms between vendors and buyers.

Many different taxonomies of auctions may be given. Usually, auctions are classified according to the issues that they resolve. They can range from the familiar *zero-dimensional* auctions that determine only the price of the product, *one-dimensional* auctions, that resolve price and quantity, to *multidimensional* generalizations that resolve multiple issues at once [77]. The later can be also characterized as *multi-attribute*, allowing specification of offers referring to multiple attributes of a single good [17]. In addition, *combinatorial* auctions are the ones that allow bidders to express offers for combinations of goods, aiming at an allocation that maximizes overall surplus.

Wurman et al. in [105] recognize three core activities that are common to all auctions. The first two are requisite, while the third is usually performed, and all three may be interleaved and iterated depending upon the auction rules:

- Receive bids: Bids are the messages sent by agents to indicate their willingness to participate in exchanges. In receiving a bid, the auction verifies that it satisfies the auction rules, and if so, admits it into the active set of bids.

- Clear: The central purpose of an auction is to clear the market, determining resource exchanges and corresponding payments between buyers and sellers. Clears can be triggered at scheduled times, at random times, by bidder activity or by bidder inactivity.

- Reveal intermediate information: Auctions commonly supply agents with some form of intermediate status information, such as quotes.

2.4.2 Auction Types

Auctions range vastly in types and parameters. The stereotyped image of a fast-talking person with a gavel calling out prices is but a particular special case of this class (namely, the English open outcry auction). Many other types exist, but some of them are the most common (figure 5 shows one possible classification for a small part of auction space).

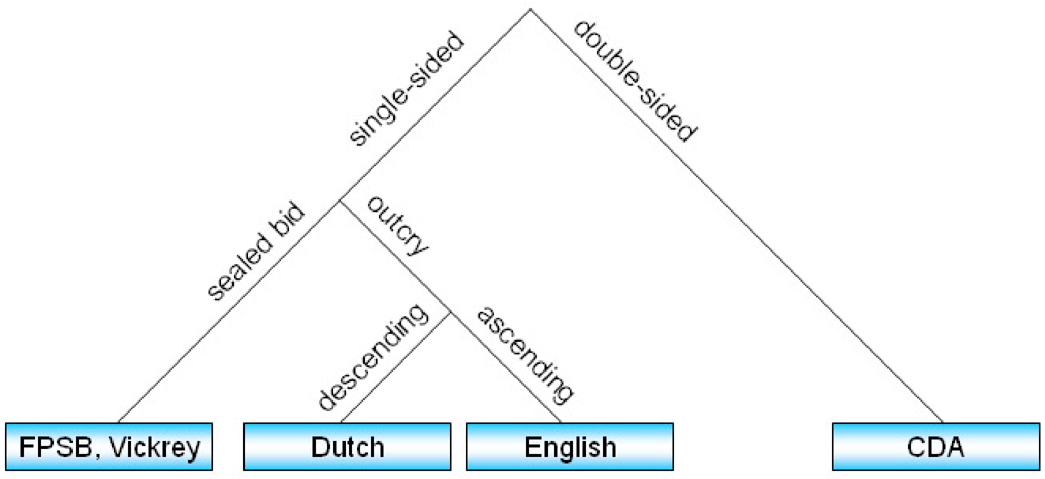


Figure 5 A classification of auctions.

The *English* auction is by far the most popular and widely used. In the English auction (first price, open-outcry) the auctioneer makes a description of the good for sale available to all bidders. It communicates to the bidders the minimum bid price that is requested to get the auction going (starting price) and the minimum increment over the current highest bid for a new bid to be accepted (bid increment). It also records private information about the minimum price that it is prepared to sell the good at (reservation price). The auctioneer then solicits progressively higher bids from the bidders until only one bidder is left. The winner claims the item at that price they last bid. English auctions have been proven as the most popular for on-line auction websites on the internet due to familiarity and simplicity of the auction protocol among consumers. The *Dutch* (descending) auction is the opposite of an English one. The auctioneer begins at an initial high price and incrementally lowers it at prescribed time interval and amount until some bidder signals acceptance acquiring the item at the current price.

For the next two types, bids are not publicly announced (sealed bid auctions) and their progress is completed in a single-round negotiation scheme. In the *first price sealed-bid* auction, each bidder submits one bid without any knowledge of the others bid. The highest bidder wins the item and pays the amount of the bid. A *Vickrey* auction (second price sealed-bid) auction is similar to a first price sealed-bid auction, however the highest bidder pays the amount for the second highest bid.

Finally, a more general type is the *double-sided* auction, which admits multiple buyers and sellers at once, allowing them to continuously update their bids/asks at any time in the trading period.

2.4.3 Auction Negotiation and Agents

The on-line auctions are an active field, where many agent researchers experiment new approaches. Agent-negotiating auctions are popular because of their potential role in e-Commerce. In physical markets dominated by standardized industrial goods and posted prices, examples of elaborate negotiation are limited to high value items (e.g., houses, collectibles, etc). In e-Commerce, on the other hand, several factors (i.e., the trend toward customization, proliferation of agents, increasing interaction between buyers and sellers) favor negotiated prices.

Auctions represent a more general approach to price determination in comparison to most on-line e-Commerce transactions, admitting a range of negotiation protocols. More specifically, auctions are basically a form of negotiation, in which a fixed auction protocol is followed. Rosenschein et al. in [80] propose five attributes that are necessary for a ‘good’ negotiation mechanism: efficiency, stability, simplicity, distributive and symmetry. All these characteristics can be found when considering an auction model for automatic negotiation [12]:

- Efficiency. To show the efficiency of an auction is often difficult, if not impossible to undertake without imposing severe restrictions such as independent private evaluations, risk neutrality, independent products to sell, etc. However, these restrictions are quite straightforward to implement in a multi-agent system. Indeed, the agents are often considered as rational entities seeking to maximize a well defined utility function based on precise rules. By imposing on architecture strict social relations between the different buyers, a simple auction model can be designed, which follows the simplifying restrictions. This will assure the efficiency of the proposed auction mechanism.

- Stability. An obvious method to achieve stability is to forbid an agent to cancel or reconsider offers once they have been submitted.

- Simplicity. Auction mechanisms are in general quite simple to implement. This is in fact one of their strength. Considering the number of messages to communicate, the only communications necessary are the offers and the responses of the vendor.

- Distributivity. At first sight, distributivity is only partially fulfilled: while there are multiple independent buyers, the vendor or auctioneer is a central entity. However, in a more complex environment, i.e. a market, different buyers and vendors may negotiate. In such an environment there is no global central entity anymore. The vendors may either act independently or coordinate their activities. This allows for multiple simultaneous auctions.

- Symmetry. The symmetry is easily achieved by assuring that all participating agents have access in the same way to all information available to them. No agent will thus be privileged with regard to others.

In economic and game theory, interactions consist of two components: a negotiation protocol and a strategy. The former defines the valid behaviour of the agents during the interaction, the interchange of messages and the rules by which the negotiators must abide (e.g. who can say what to whom at what time). The later is the method the agents employ to achieve their negotiation objectives within the specified protocol. The protocol is set at design time by the marketplace owner and is publicly known to all the participants. The strategy is designed by each individual participant and is private. Moreover the effectiveness of the strategy is very much determined by the protocol. [94] describes the three key actions which the negotiation host (auctioneer) must carry out during any negotiation process to preserve the integrity of the protocol:

- Validation: When participants submit proposals, they first need to be validated with respect to the negotiation template. The validation step consists in making sure that the proposal is a more constrained form of the agreement template. That is, the constrains over the parameters in the proposal must be tighter than the corresponding ones in the agreement template. The constraints represent acceptable values to the proposing participant.

- Protocol Checking: The proposal must be submitted according to the rules of the protocol which governs the way the negotiation takes place. These rules specify who can make proposals, when they can be made and what proposals can be submitted in relation to previous submissions. For examples, auctions often have a bid increment rule that requires any new proposal to be for a higher price than previous ones.

- Agreement formation: In an agreement is to be made, there must be at least two valid proposals which are compatible to each other. Proposals are compatible if there is an identical fully-instantiated form of each.

From the above it is clear that auctions formulate a well-defined basis for agents to structure negotiation scenarios. The application of the multi-agent paradigm to auctions can be viewed from two different points [12]:

- The mechanisms of an auction can be defined as a resource allocation problem to a set of agents. The problem may be described as a market in which there are vendor and buyer agents, which trade on items represented by the resources. These agents exhibit different acquisition capabilities that let them act differently depending on the current context or situation of the market.

- The auctions can be viewed as a process of automatic negotiation implemented as a network of intelligent agents. Buyer and vendor agents interact in an electronic market environment to trade items.

2.4.4 Auction Techniques and Algorithms

Auction algorithms can be differentiated across many parameters including, but not limited to, those concerning: the remaining time that it has left to acquire the item, the number of remaining auctions that the agent can bid in, the level of desperateness to obtain the item, the desire for bargaining the price, the price determination algorithm etc. There is a plethora of available algorithms and strategies, each of which is suitable for specific conditions and protocols. Surveying the proposed algorithms ([42], [45]) we distinguish the following cases:

- English auction: The agent's dominant strategy is to bid a small amount more than the current highest bid and stop when the user's offer limit is reached. Other approaches take into account the remaining time until the end of the auction, in order to place the bids as close to the end as possible, for increase their chances of winning, while keeping the rate of increase of the auctions maximum offer to low levels (this technique is called snipping). [29] suggests a bidding algorithm for agents participating in multi-attribute English auction.

- First-price sealed-bid auction: In general, there is no dominant bidding strategy in this auction. Here, the price of the bid and the time to stop are functions of the agent's own valuation of the item and its beliefs about the valuation of other bidders. A good strategy is to bid less than the user's true valuation, but how much less depends on the user's attitude towards risk.

- Vickrey auction: In the private value Vickrey auction, the dominant strategy is to bid the user's true valuation [81].

- Dutch auction: The Dutch auction is strategically equivalent to the first-price sealed-bid auction. This is because in both of them an agent's bid matters only if it is the highest and no relevant information is revealed during the auction process.

- Multiple auctions: This is the most interesting type from the researchers' point of view. In this type, the agent needs to monitor all the relevant auctions, decide which one to bid in and determine what to bid in order to get the goods at the best deal. There is no dominant bidding strategy, thus heuristic methods are required. Byde in [19] studies three bidding algorithms for agents that participate in several English actions with varying start and end times. [11] presents strategies and evaluations for agents confronted in a set of Dutch auctions. Preist et al. proposed a coordination algorithm designed for agents that participate in multiple simultaneous multi-agent actions [74], [75]. Jennings et al. present the design of a heuristic agent that participates across multiple English, Dutch and Vickrey auctions [51], [4]. In [20] a framework that enables an agent to make rational decisions across multiple simultaneous auctions is developed. Finally, in [71] an algorithm for combinatorial ascending auctions on the Internet is described.

An interesting event for studying agent negotiations under auction scenarios is the Trading Agent Competition (TAC) [93], an annual international forum, where software agents compete against each other in a variety of auctions. The goal is to assemble travel packages for their individual customers according to their preferences, maximizing the total satisfaction. TAC is very successful at attracting many competitors from around the world by creating an artificial domain that is simple enough to understand quickly, but complex enough to prevent a trivial winning strategy. Both universities and non-academic institutions participate in the tournament obtaining valuable experience for designing algorithms for future on-line transactions ([44], [39], [89], [88], [35]).

2.4.5 Characteristics of Existing On-line Auction Marketplaces

On-line auctions make the physical limitations of traditional auctions disappear (e.g. time, space and presence) and provide millions of globally dispersed customers with varieties of goods that can be selected within a flexible pricing mechanism [8]. The network supports inexpensive, dynamic, wide-area communication and easily handles thousands of participants and information, allowing automation of the auction protocol from both the server and bidders.

The explosion of e-Commerce has influenced greatly the auction marketplaces both in number and transactions. More and more companies are offering auction sites. As a result, the most common procedure of actually finding a product that a person is interested in on an auction site involves the tedious process of visiting one or more of the popular sites to locate the desired product to bid on. In order to get the best price, customers must monitor all of these auctions using a web browser, and place bids appropriately. Care must be taken, to ensure no more than one purchase to be made. Furthermore, if the intention is to purchase more than one item that are interdependent (a travel package for example), the task becomes almost impossible.

As a result, auction sites are beginning to offer support tools to facilitate users, such as search tools that allow customers to locate and monitor auctions or provide price trend information on popular items. Some sites running English auctions do offer a simple bidding agent. This agent resides on the auction site, and bids on the customer's behalf, based on the maximum price they are willing to pay, placing the lowest possible bid on the web site (either the reservation price, or if bidding has already started, it bids just above the current highest bid.). However, such agents are not able to participate in multiple auctions, either on the same web site or across different ones. As a result, once the agent is initiated, the customers are committed to making a purchase on the site, if possible. Locally, they may pay the lowest price they can to win the auction. Still, from a global perspective, that particular auction is unlikely to have been the best place to make a purchase.

These bidding agents have an additional disadvantage. To use them, customers must reveal the highest price they are willing to pay to the auction site. This gives the auction site information that could be used to cheat them. Furthermore, as auction sites often receive percentage commission on sales made, they also have an incentive to cheat. To do this, they would take note of the highest price their customers are willing to pay, and enter a mythical bid in the auction (a 'shill') just under this price. The agents would then place the maximum bid, and the seller has made a sale at the best possible price. The auctioneer would therefore collect their maximum possible commission. This second disadvantage could be overcome by placing agents locally on the auction participant's machine, but this would not overcome the first, more serious, problem.

2.4.6 Requirements for Supporting On-line Auctions

Next-generation on-line auction houses should resolve the limitations of existing sites and, in addition, should exploit the advantages of technological and scientific progress. The current auction process can be improved in all its phases.

Searching for products can be supported with the use of semantics. Ontological descriptions of product domains can capture more accurately their relations, allowing clients to navigate through their concept graph using semantic query languages. This will permit users to be more precise on expressing what they desire, narrowing returned results and simplifying the process of discovery.

The explosion in number of available on-line auctions necessitates a degree of automation in monitoring and bidding. Both clients and retailers should be supported in initiating and managing a session. In particular, agent technology, if utilized appropriately, can fully automate the process of participating in multiple parallel-running auctions. The question is how to make this process non-trivial for the user and easily extensible in different environments.

Moreover, auction site owners can benefit from modern technology in improving the quality of their services. Interacting with customers can and should become richer, by providing notifications of various kinds (i.e., about new products available, product offerings etc.) and advanced customization. Clients, on the other hand, request privacy and credibility. They should feel free to apply their own strategy to all auctions they participate in, instead of following the restricted rules of each site. A layer for handling heterogeneity among multiple auction houses is required.

Finally, the history of bidding of a product and the product's auction statistics have gained importance over the years, due to the increase in Internet-based auctions. This information is valuable for future auctions, for assisting bidders in specifying appropriate bids and sellers in setting the base prices. Currently none of the on-line auction sites give any information of such kind.

2.5 Related Work

SeMPHoNIA project addresses issues that extent to most phases of e-trading, exploiting technologies concerning multi-agent system, peer-to-peer networking and the Semantic Web areas. Several projects deal with subsets of those issues, but only few of them confront to the problem in its entirety. Furthermore, even fewer combine the benefits offered by all three of the aforementioned main research areas, concerning heterogeneous distributed negotiating environments. Due to the plethora of related literature we distinguish the following topics:

2.5.1 Auction Platforms and Testbeds

The Travel Agent Game in Agentcities (TAGA) [90] is a general framework for running agent-based market simulations that extends and enhances the Trading Agent Competition (TAC) [99] scenario. TAGA runs on an open multi-agent environment based on FIPA compliant platforms and uses Semantic Web languages and tools (RDF and OWL) to specify and publish the underlying common ontologies [108]. TAGA's objectives are very much in common with those of SeMPHoNIA; it offers an environment for exploring agent-based trading in dynamic markets, it supports semantic querying and publishing and it allows users to create their own reasoning mechanisms for their agents. We believe, though, that the system's design is suitable for experimenting negotiation algorithms, but lacks the ability to simulate realistic marketplace conditions. It offers a controlled environment, specialized in one trading scenario and based on a centralized infrastructure. SeMPHoNIA, on the other hand, provides openness and interoperability, because it is based on a peer-to-peer infrastructure that approximates more realistic matchmaking and trading mechanisms. In addition, the flexible design of SeMPHoNIA's agent system places emphasis on local negotiations between agents of remote peers, promoting efficiency in trading, as well.

IntelliBid [53] is an "event-trigger-rule-based" auction system over the Internet. It is structured on top of a network of specialized Web Servers, called Knowledge Web Servers, that provide various auction services to bidders and suppliers of auction sites. IntelliBid relies on events and event filters to keep users timely informed of the progress of desired auctions and, also, on rules to permit bidders to apply custom bidding strategies. Unlike SeMPHoNIA that exploits software agent technology, the approach of using events, triggers and rules introduces certain advantages, but deprives the benefits of agent communication and automatic task execution. Furthermore, IntelliBid is designed to enhance the existing scheme of Internet-based auction site structure, inheriting the handicaps of centralized web server architecture.

Other auction platforms are described in projects FAucS [24] and AuctionBot [106]. Compared to those similar projects, SeMPHoNIA offers complete solutions to most phases of e-Commerce negotiation, rather than focusing on parts of them. We argue that the combination of technologies applied in SeMPHoNIA enhance the system with flexibility and extensibility that can't be met in other platforms.

2.5.2 Automated Negotiation Issues

Outside of, but related to, the auction scenario, automated negotiation represents an important issue of the SeMPHoNIA project. Researchers of different domains study this subject from various points of view.

In [76] an infrastructure for automated negotiation is described, which uses peer-to-peer and standard agent technology for supporting multiple traders on multiple platforms. The agent interaction protocols are represented using coloured Petri Nets, while the implementation of the approach is demonstrated in the context of a card game. Communication is accomplished in two levels; at the highest level (communication between players) the notions and services of the peer-to-peer model are utilized, while at the lowest level (communication between agents of the same player) micro-agents segment the tasks and cooperate using RPCs and FIPA ACL. The general architectural principles followed in this system are similar to those of SeMPHoNIA platform, however the approach is restricted to closed environments and does not support real-world publish and discovery mechanisms.

Bartolini, Preist and Jennings in [9] present a generic interaction protocol and a general framework using this protocol, which can be parameterized by different rules to implement a variety of negotiation mechanisms. This approach intents to eliminate the need of having to explicitly hard-code the negotiation protocol in the agent's design, resulting in a more flexible multi-agent scheme. However, this work does not provide a formal semantics of the negotiation elements introduced.

Flexible negotiation is explored in [101], as well. In particular, the authors describe the Nuin agent platform, a toolkit for developing deliberative intelligent agents for Semantic Web applications, based around belief-desire-intention (BDI) principles. The objective of this project is to build a platform consistent with emerging standards of the Semantic Web and agent technology, using existing tools where applicable, to create a flexible and adaptable tool for agent designers. Agents in Nuin use first-order logic as their knowledge representation language to perform inferences and model their mental states, while the configuration of the agents themselves is defined by an RDF model. An internal interpreter processes events from the environment and, in conjunction with the agent's plans and other mental states, determines its behavior. Although agents in SeMPHoNIA embody different reasoning models, the formal approach of Nuin agent platform provides an interesting framework, from which many ideas can originate, concerning not only agent reasoning design issues, but, also, semantic integration and cross-platform interoperation issues. The framework proposed in Nuin can be applied in our platform, as well.

[91] attempts to address the problem of automated agent negotiations in open environments, where the negotiation protocols are not known beforehand, but instead the host advertises the type of protocol regulating the interaction. According to the proposed approach, a shared ontology of protocols is defined based on the idea that some general concepts are present in any negotiation protocol. Moreover, a method ontology aims at modeling knowledge concerning the interactions between agents on how to perform a task. To demonstrate a proof of concept for their approach, the authors considered the application scenario of the Trading Agent Competition (TAC) and built an ontology for English auctions using DAML+OIL. It should be noted, though, that the approach is still at an early stage of development and several other issues need to be investigated.

Similar issues are addressed in projects, such as SweetDeal [40], [63], [94], [85], [84].

2.5.3 Semantic Publish and Discovery Issues

An important aspect of SeMPHoNIA system is the mechanism for semantically publishing and discovering available resources. Many other projects deal with the same topic, following different approaches.

The work on Meteor-S project [96] is focused on semantic publication and discovery of web services. In Meteor-S Web Services Discovery Infrastructure (MWSDI) registries are categorized based on domains, while specialized ontologies maintain relationships between different domains and associate registries to them. Moreover, semantics are added to Web Service descriptions before registering them with the registries. The MWSDI framework is built on top of a peer-to-peer network, in which certain peers maintain the registries and provide their ontologies, while others are responsible for updating these ontologies and preserving consistency when new registries join the network. Semantic discovery is an aspect of SeMPHoNIA platform, too, and clustering peers based on domains is a similar approach to our notion of consortia of domain related peers (for details consult section 8.1).

InfoSleuth [10] is an agent-based information discovery and retrieval system that adopts broker agents to perform the syntactic and semantic matchmaking.

Syntactic brokering is the process of matching requests to agents on the basis of the syntax of the incoming messages, which wrap the requests; semantic brokering is the process of matching requests to agents on the basis of the requested agent capabilities or services, with the agent capabilities and services being described in a common shared ontology of attributes and constraints. This single domain-specific ontology is a shared vocabulary that all agents can use to specify advertisements and requests to the broker. The broker agent matches agents that require services with other agents that can provide those services. By maintaining a repository containing up-to-date information about the operational agents and their services, the broker enables the querying agent to locate all available agents that provide appropriate services. The service capability information in InfoSleuth is written in LDL++, a logical deduction language.

InfoQuilt system [6] investigates ontology interoperation at the phase of semantic discovery. It provides a framework for knowledge sharing and complex information request formulation over a peer-to-peer infrastructure. Peers create and maintain their own local DAML+OIL ontologies and share their definitions in the global knowledge space by uploading their KObjects (concepts) and Links (relationships) creating inter-ontological connections. Users construct IScapes (semantic information requests) to specify keywords, define relations and retrieve relevant ontologies from the peer-to-peer knowledge sharing network. InfoQuilt uses IScapes, instead of a semantic query language, to retrieve information, following an algorithm for KObject navigation and linkage. Although IScapes are based on keyword matching, the authors argue that the results acquired are more accurate, compared to those of a query language. This is a questionable matter that necessitates more research, since recent query languages, such as RQL, which is used in SeMPHoNIA, posses very powerful semantics and have proved their abilities.

Edutella [69] is a project which explores many of the issues concerning community annotation, focusing on the realm of educational metadata. The goals of the project cover query replication, mapping, mediation, and annotation. It is an open source project that builds upon the JXTA P2P Framework. The project offers a schema-based network for sharing knowledge resources (RDF-formatted). In the Edutella network every peer needs to make its metadata available as a set of RDF statements that rely to a certain schema. In order to provide interoperability, four services are offered by an Edutella peer: the Query Service, the Replication Service, the Mapping Service and the Annotation Service. The query service is the most basic service within this network. It enables a peer to register the supported metadata-fields together with supported metadata schema(s). Peers can next exchange queries using the Edutella query exchange format, which is based on Datalog semantics. The Edutella wrapper handles translations to and from this query language.

The same topic is explored in other projects, as well; SEWASIE [13], NeuroGrid [52] and [73].

In this section we introduced the basic technologies that are exploited in SeMPHoNIA project, as well as, a survey of related work. Next, we are going to describe in detail the design and architecture of SeMPHoNIA platform, explain its components and present examples of interaction.

Chapter 3 THE SeMPHoNIA ARCHITECTURE

This section describes the SeMPHoNIA platform in detail. It starts with a layered model of the system's structure, presents its basic components, explaining their features and, finally, analyzes the platform's architecture, along with examples of interaction.

3.1 Design of SeMPHoNIA

In describing the structure of our platform, we take a layered approach. The SeMPHoNIA's basic component diagram is shown in Figure 6. The platform integrates three preexisting technologies; JXTA [55] for configuring the peer-to-peer network, ICS-FORTH RDFSuite [48] for exploiting technologies of the Semantic Web and Grasshopper [38] for planning and managing the multi-agent character of the system. Details about these systems are further discussed in the following sections. No distinction is being made between the customer and the auctioneer roles in the diagram, because the system components support the same operations and information flows regardless of the user's role.

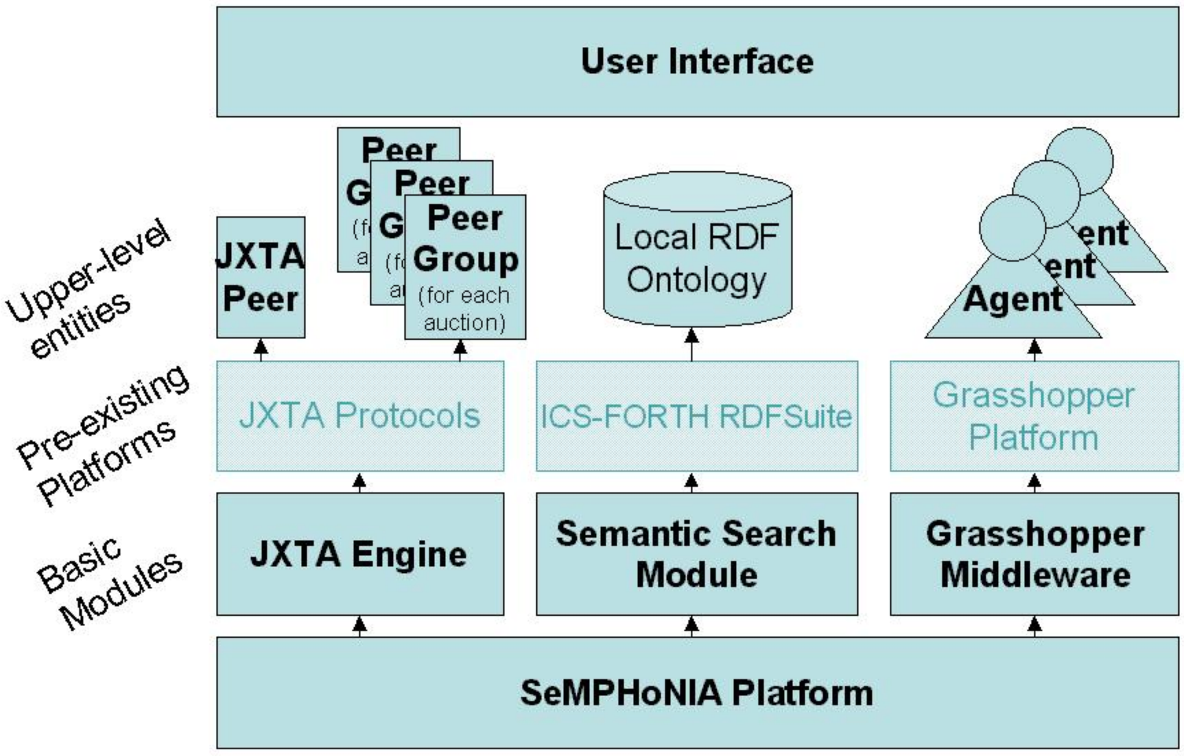


Figure 6 SeMPHoNIA structure.

The bottom layer, called *platform layer*, encapsulates the essential mechanisms for controlling the *basic service modules* and managing their inputs and outputs, in order to receive, process, filter and forward information among them. It provides core functions and unites the basic modules of the platform.

On top of the platform layer the three basic modules are defined. These modules are not interconnected, but communicate through the platform layer instead. The JXTA Engine is responsible for implementing JXTA protocols to allow the application to function as a peer, collaborate with other peers and deploy peer-to-peer services. In SeMPHoNIA, peers self-organize into series of interconnected nodes, known as peer groups, for reflecting the abstraction of auction rooms in the network. The Grasshopper middleware is the component that undertakes the role of automating the negotiation procedure of auctions by creating, controlling and monitoring software agents that represent human users. Furthermore, the Semantic Search Engine module facilitates semantic publish and discovery of products on the network, exploiting software tools provided by the ICS-FORTH RDFSuite. Communication with the ICS-FORTH RDFSuite is conducted through a specialized server, the RDFSuiteServer, that forwards RQL queries to the underlying RDF database.

At the top layer, the user interface is defined, which contains the group of modules that work together to provide an integrated user interaction with the system. It offers graphical control over the platform's functions and presents responsive information about the progress of the user's running auctions.

The following example illustrates a general interaction between the aforementioned elements. A human user participates in the SeMPHoNIA environment by initiating the application, either as a bidder or as a supplier. The platform layer activates the basic components and presents the UI main frame. It also instructs the JXTA Engine to submit an admission request for joining the network and obtaining an identity as a new peer. Acting as auctioneer, the user publishes an RDF ontology describing the selling products and their auction specifications while the system contacts the Grasshopper agent platform, with the intervention of the corresponding middleware, to create an agent to govern the auction. As a customer, the user searches the network for products by querying the published databases, using the Semantic Search Module. All client queries are forwarded to the appropriate RDF ontologies stored in multiple ICS-FORTH RDFSuite databases across the network and the results are presented on the graphical user interface. Once the desired items are found the Grasshopper middleware constructs the agents that will participate in the auctions on behave of the user and directs them to the appropriate auction places. The interface updates information in real time to inform the user about the progress of all running sessions.

3.1.1 JXTA

The JXTA platform defines a set of open, generalized peer-to-peer protocols that allow any connected device on the network – from cell phone to PDA, from PC to server – to communicate as peers, independent of the development language, operating system or network transport employed by each peer [55].

Figure 7 presents the Project JXTA software architecture, which is divided into three layers, the JXTA core layer, dealing with common peer-to-peer building blocks, the service layer, implementing popular peer-to-peer network services, and the application layer, where integrated applications are developed.

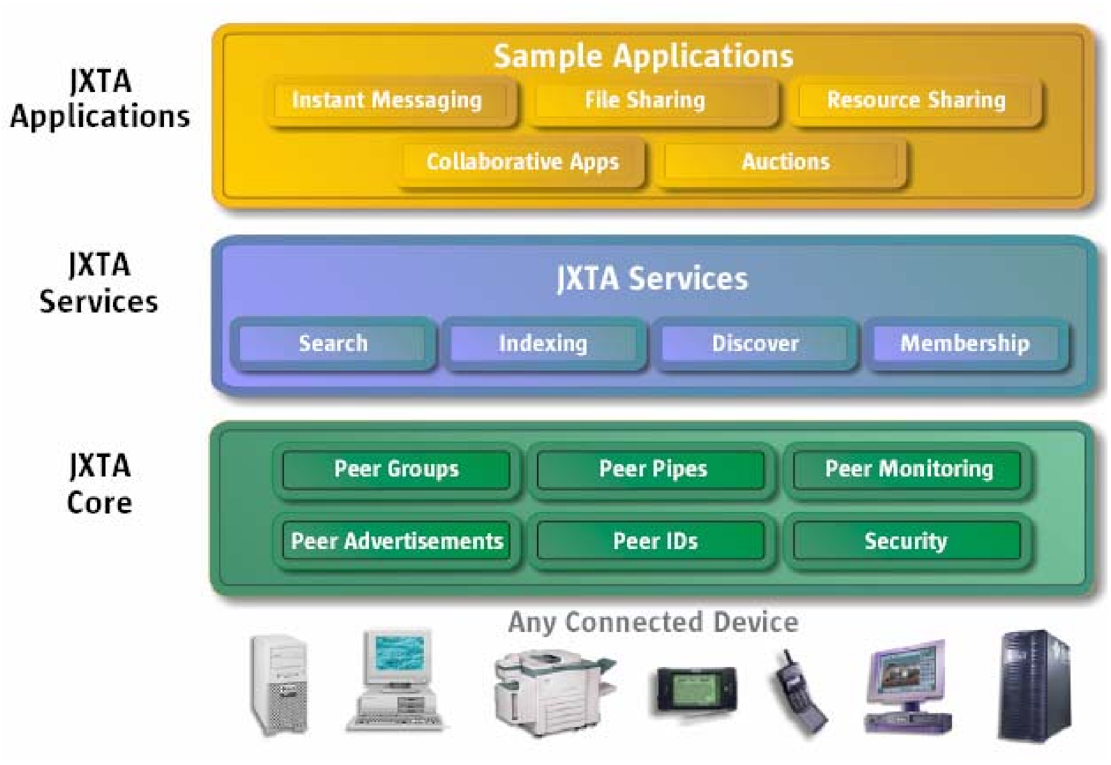


Figure 7 JXTA Software Architecture.

The fundamental notion in JXTA is a peer. JXTA peers advertise their services in language-neutral metadata structures, represented as XML documents, called *advertisements*, that enable other peers on the network to learn how to connect to and interact with a service. Advertisements are the basic unit of data exchanged between peers to provide information about available resources, such as other peers, peer groups or services. Peers discover resources by searching for their corresponding advertisement, and may cache any discovered advertisement locally for a predefined amount of time. When this time passes, the advertisement expires. Moreover, JXTA introduces a special type of peer, the *Rendezvous peer*, that is responsible for allowing a user to broadcast messages to other peers that belong to different local or private networks. These peers provide enhanced connectivity and contribute in avoiding message propagation to the entire network (message flooding).

JXTA defines a series of XML message formats, or protocols, for communication between peers. Peers use these protocols to discover each other, advertise and discover network resources, and route messages. Each of the protocols addresses exactly one fundamental aspect of peer-to-peer networking, therefore a peer can elect to implement only a subset of them to provide the desired functionality. The six basic protocols are:

- Peer Discovery Protocol – used by peers to advertise their own resources (i.e., peer, peer groups, services etc.) and discover resources from other peers. Each peer resource is described and published using an advertisement.

- Peer Information Protocol – used by peers to obtain status information (uptime, state, recent traffic etc.) from other peers.

- Peer Resolver Protocol – enables peers to send a generic query to one or more peers and receive a response to that query. Unlike Peer Discovery and Peer Information Protocols, which are used to query specific predefined information, this protocol allows peer services to define and exchange any arbitrary information they need.

- Pipe Binding Protocol – used by peers to establish a virtual communication channel, or pipe, between one or more peers, binding two or more ends of the connection (pipe endpoints).

- Endpoint Routing Protocol – used by peers to find routes (paths) to destination ports on other peers.

- Rendezvous Protocol – mechanism by which peers can subscribe or be a subscriber to a propagation service. Within a peer group, peers can be rendezvous peers or peers that are listening to rendezvous peers.

SeMPHoNIA implements the functionalities of the Peer Discovery, Peer Information and Rendezvous protocols to build JXTA Services. The platform expands to all JXTA layers displayed in figure 7. In particular, we expand certain features of the JXTA Core, such as Peer Advertisements, to customize attributes of our applications.

3.1.2 ICS-FORTH RDFSuite

The ICS-FORTH RDFSuite [48] is a suite of tools for RDF validation (Validating RDF Parser – VRP), storage (RDF Schema Specific DataBase – RDSSDB) and querying (RDF Query Language – RQL), using an object-relational DBMS [2]. Figure 8 displays an overview of ICS-FORTH RDFSuite's architecture.

The Validating RDF Parser (VRP) is a tool for analyzing, validating and processing RDF schemas and resource descriptions. The Parser analyses syntactically the statements of a given RDF/XML file according to the RDF M&S Specification. The Validator checks whether the statements contained in both RDF schemas and resource descriptions satisfy the semantic constraints derived by the RDF Schema Specifications (RDFS). The VRP is based on standard compiler generator tools for Java, namely CUP and JFlex. The stream-based parsing support of JFlex and the quick LALR grammar parsing of CUP ensure a good performance when processing large volumes of RDF descriptions. For this purpose, the VRP validation module relies on an original object representation, separating RDF schemas for their instances.

The RDF Schema Specific Data Base (RSSDB) is a persistent RDF Store for loading resource descriptions in an object-relational DBMS by exploiting the available RDF schema knowledge. It preserves the flexibility of RDF in refining schemas and enriching descriptions at any time, whilst it can store resource descriptions created according to one or more associated RDF schemas. The main goal of RSSDB schema-specific representation is the separation of the RDF schema from data information, as well as the distinction between unary and binary relations

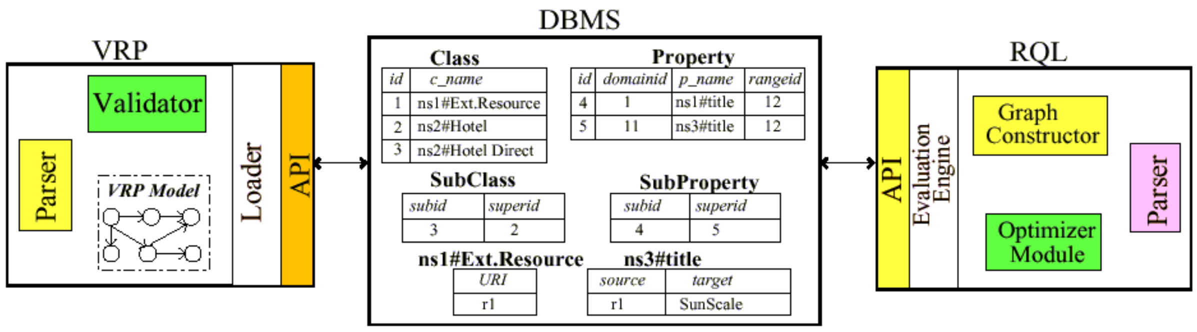


Figure 8 ICS-FORTH RDFSuite Architecture.

holding the instances of classes and properties. RSSDB has been implemented on top of an object-relational DBMS like PostgreSql. It comprises a Loading and an Update module, both implemented in Java using a number of primitive methods (APIs) for inserting, deleting and modifying RDF triples.

RQL [57] is a typed language following a functional approach, which supports generalized path expressions featuring variables on both labels for nodes (classes) and edges (properties). It relies on a formal graph model that captures the RDF modeling primitives and permits the interpretation of superimposed resource descriptions by means of one or more schemas. The novelty of RQL lies in its ability to smoothly combine schema and data querying while exploiting the taxonomies of labels and multiple classifications of resources in a transparent way. It consists of three modules, the Parser, analyzing the syntax of queries, the Graph Constructor, capturing the semantics of queries in terms of typing and interdependencies of involved expressions, and the Evaluation Engine, accessing RDF descriptions from the underlying database via SQL3 queries.

Currently, RQL is considered to be the most complete RDF query language in comparison to other popular ones (RDQL, Triple, SeRQL, Versa, N3), according to elicitations extracted from recent evaluations ([43], [22]). Considering a set of well defined criteria, RQL is the only language to provide full support for path expressions, union, difference, quantification, aggregation, namespace querying, lexical space querying, value space querying, entailment and partial support for optional path expressions, reification, collections and containers. Furthermore, it features support for orthogonality (combining a set of simple operators into powerful constructs), RDF Schema semantics, while satisfying for its readability and usability, although the lasts are very much dependent on personal taste.

3.1.3 Grasshopper

Grasshopper-2 [38] is an agent development platform that enables programmers to develop and deploy a wealth of distributed, agent-based applications written in the Java programming language. The platform is built on top of a distributed processing environment and enables users to create autonomous acting agents, able to migrate,

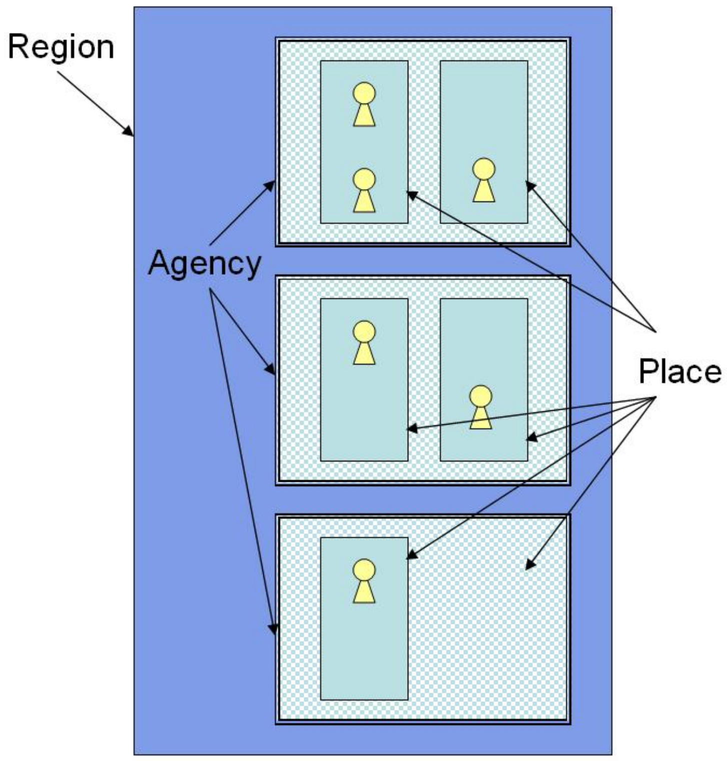


Figure 9 Grasshopper basic elements.

and to transparently locate and send messages to them. Grasshopper is conformant to the Mobile Agent System Interoperability Facility standard (MASIF) of the Object Management Group (OMG) for achieving interoperability between mobile agent platforms of different manufacturers. The system's communication service supports the following protocols: CORBA IIOP (Internet Inter-ORB Protocol), MAF IIOP, Java RMI, plain socket connection, FIPA (both ACL and Agent Management), as well as protected RMI and socket connections with SSL (Secure Socket Layer).

Grasshopper lays strong emphasis on migration and communication. Migration is different than traditional remote execution. The former implies that a program, a mobile agent in this case, is able to change its location during execution continuing at the exact point at which it has been interrupted, while the second implies that a program is sent to a remote location before its activation and remains there during its entire life time. Concerning communication issues, apart from the various protocols, Grasshopper supports several modes of inter-agent communication, such as a/synchronous, uni/multicast, dynamic and location transparent communication. The last is explained below.

The structure of the Grasshopper distributed agent environment is composed of regions, agencies, places and different types of agents, as shown in figure 9. The agencies are the actual runtime environment for mobile and stationary agents and may be initiated on different hosts of a network. Places, on the other hand, provide a logical grouping of functionality inside an agency. Agents exist and operate inside them and usually the name of the place reflects its purpose and identity. The medium, though, that facilitates the management of all the distributed components in the Grasshopper environment is the region. The region maintains information about all of the components it contains and automatically registers new ones, when they are created, to preserve a consistent image of its elements at all times. The registration services offered by regions ensure location transparent communication between entities within the distributed environment; agents need not care about the location of the desired communication peer, but only about its identity. The region traces the destination of a message, even when mobile agents travel between different places or agencies.

[37] presents the results of a technical report evaluating various agent platforms according to specific criteria, such as standard compatibilities, communication, agent mobility, security policies, availability, usability, documentation and other development issues. Grasshopper was highly recommended for the following reasons:

- It offers very good documentation, very good GUI, enjoys high acceptance of users and is used in many development projects

- It supports the MASIF and FIPA standards, as well as a plug-in for web interface and offers very good security features and logical grouping of functionality of agents

- It supports weak agent mobility with the possibility to simulate strong mobility (ability to migrate code and execution state of executing unit) and implements various communication protocols.

3.2 The SeMPHoNIA Platform

The SeMPHoNIA platform models aspects of market mechanisms that represent a common interaction medium for users on the Internet. Exploring the design space of auctions and integrating emerging technologies, the system provides a unifying framework for researchers to study issues of e-Commerce and negotiation that go beyond auction theory. Three distinct layers of functionality synthesize the platform's behavior; its semantic character, its multi-agent character and its peer-to-peer character. Before delving into more on details regarding the physiognomy of the platform as a whole, we elaborate on the different layers of abstraction and their corresponding contribution to the system.

3.2.1 Semantic Character

Traditional Web-based product searching based on keywords searching seems insufficient and inefficient in the 'sea' of information [62]. Ontologies have shown to be the right answer to knowledge structuring and different approaches are presented for modeling and sharing a particular domain by a group of people. Until now, systems based on centralized ontology schemes suffer from difficulties concerning development and maintenance [92]. The SeMPHoNIA infrastructure takes advantage of local ontologies, allowing participants to build and maintain their own RDF knowledge databases for describing products for sale. Every retailer who wishes to trade one or more products on the SeMPHoNIA network builds an RDF ontology following the conventions of the public RDF Schema for the particular domain of the corresponding product, stores it locally and publishes its location, so that other members of the network can query its descriptions.

In the SeMPHoNIA project we have developed two types of ontologies; a process ontology, which is specifically about auction-related concepts and relations, and multiple domain ontologies, which enrich product descriptions with valuable metadata to better describe their features. The former type serves transactional needs, while the latter covers informational needs for product specifications.

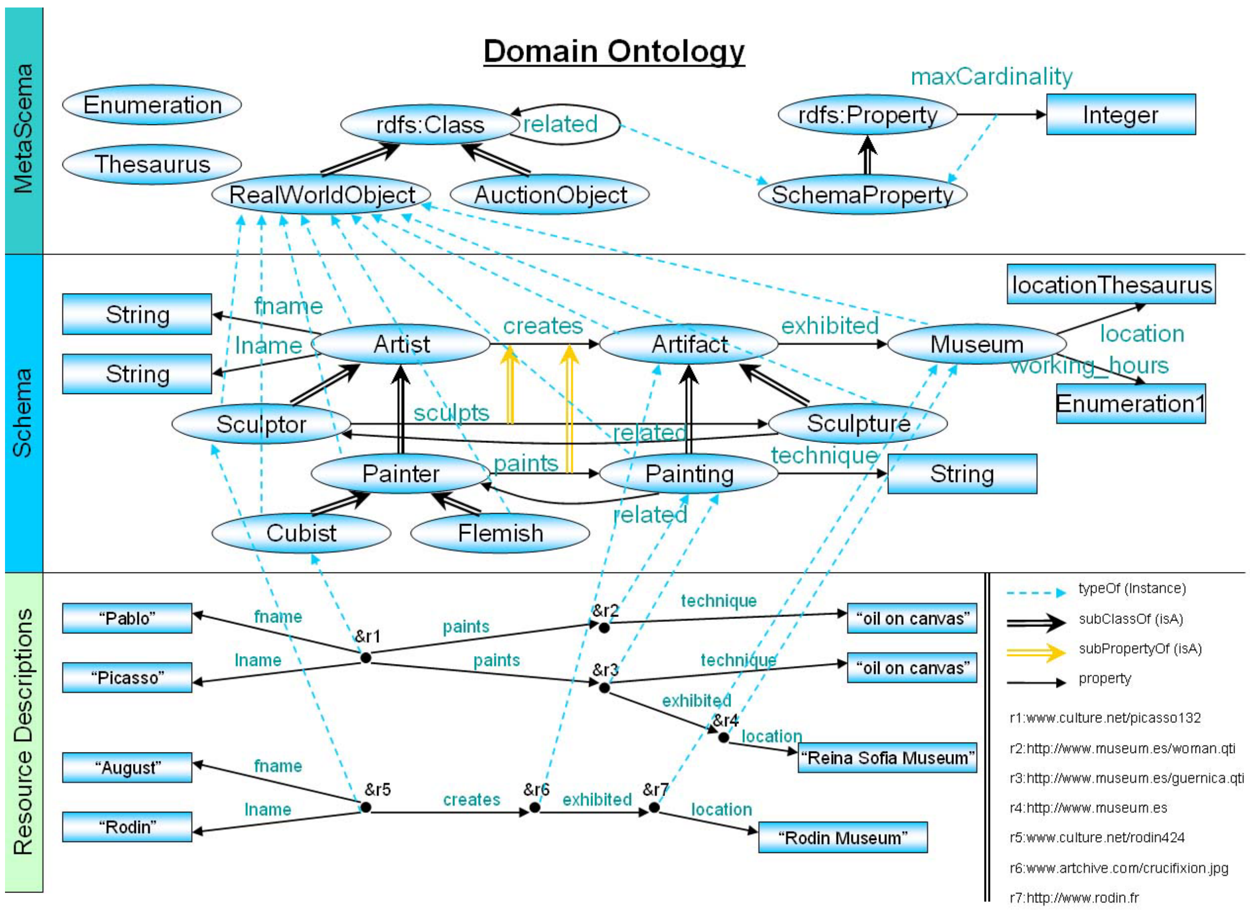


Figure 10 Artifacts domain ontology.

More specifically, each item auctioned should be related with a specific domain. A retailer describes metadata about products in the corresponding domain ontology of that product. An example of such an ontology concerning the domain of museum artifacts is shown in figure 10 [25]. The upper part of the graph depicts the default RDFS meta-schema classes, i.e. Class and Property, as well as user-defined metaclasses, specializing them (i.e., RealWorldObject, SchemaProperty). Moreover, it contains two meta-schema properties, namely *related*, which connects classes, and *maxCardinality*, which is defined on properties and has an integer value. The middle part consists of a schema intended for museum specialists, whose class definitions represent RealWorldObjects. Similar schema descriptions can be structured for other product domains, such as books, tickets, clothing etc.

The auction ontology, on the other hand, captures the characteristics of a particular auction session combining knowledge from auction protocols and other common trading concepts to specify the context in which the system operates. Roughly speaking, it is used to model all information needed for an auctioneer to initiate a new auction session and for a customer to determine a desired session based

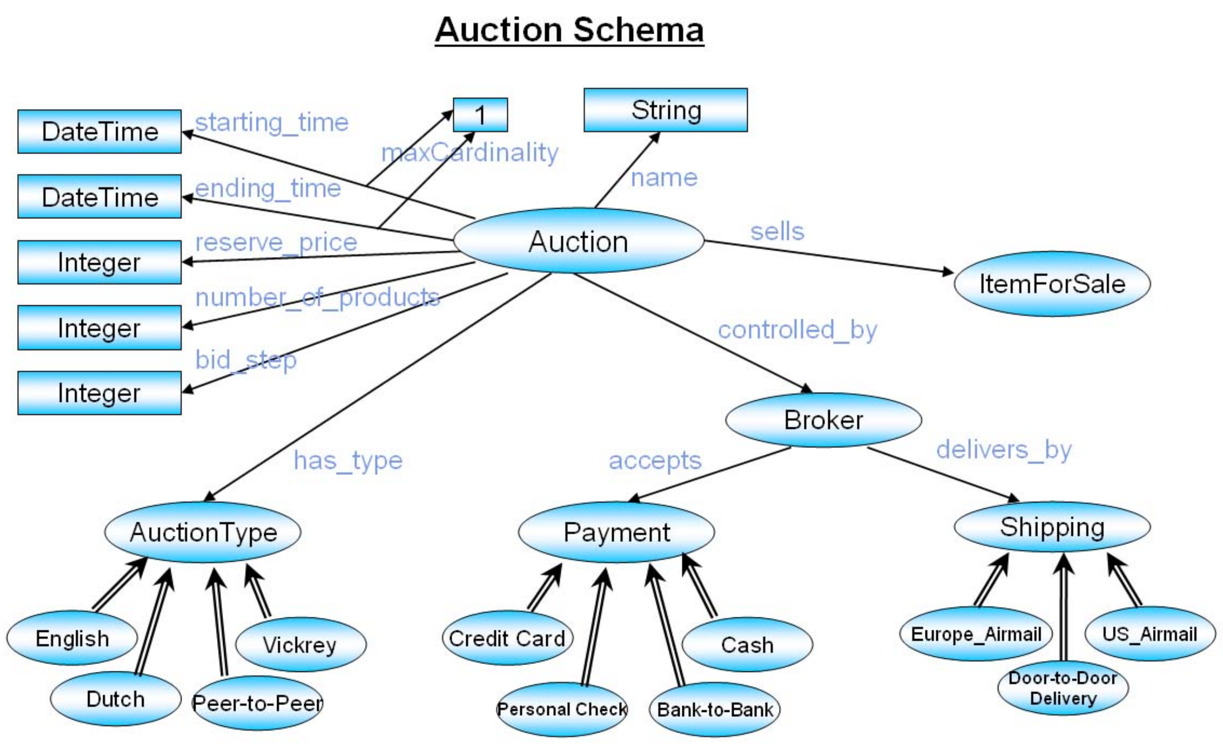


Figure 11 Auction Schema.

on criteria, such as the broker's identity, accepted payment and delivery methods, etc. Figure 11 presents a graph representation of the auction RDF Schema. An important thing to note is that the conjunction between the auction and the domain ontologies is accomplished by defining the ItemForSale class. The resource of the product, whose auction is described, should be declared both as instance of the class that captures its characteristics (domain ontology) and as instance of the ItemForSale class (auction ontology), in order for a customer to identify the ontological relation.

The announcement and publication of ontologies on the network makes SeMPHoNIA a queryable knowledge repository of RDF multi-domain ontologies. In order for customers to access that knowledge, the ability to address RQL queries on the, relevant to their interest, local databases is supported. All local RDFSuite databases connect with the SeMPHoNIA environment through a specialized type of server, namely RDFSuiteServer, which exploits a feature of RDFSuite that allows automatic query routing via a set of primitive methods (java API). The role of the RDFSuiteServer is to listen to a specific socket address for customer queries on the network, forward them to its assigned RDFSuite database, receive the answer (structured using XML format) and return it to the corresponding peer. In

SeMPHoNIA a Semantic Search Engine and a graphical user interface have been developed to assist clients in forwarding RQL queries to multiple RDFSuiteServers. The process of determining desirable products and locating auctions is accomplished in a transparent to the user manner. The queries may vary from simple resource queries (for example `ItemForSale`, which returns all product resource descriptions of the items auctioned) to more complex schema, namespace, resource description, set-based and nested queries (for example

```
SELECT     Z, W
FROM       {X}sells{Z},
           {X}controlled_by.accepts{W}
WHERE      X like “<the resource of the auction>”
```

, which returns pairs of product resource description and payment method that their supplier is willing to accept, for a given auction).

3.2.2 Multi-Agent Character

Agent technology represents a potentially novel way of conceptualizing and implementing e-Commerce transactions. The capability of agents to offer, among others, automation in job delegation and execution, coordination and advanced communication with other agents, mobility and monitoring is exploited in SeMPHoNIA to reduce the tremendous time and human resources invested in on-line trading. Specifically, agents are used to facilitate the connection of buyers and sellers and to automate the process of negotiation in the context of auction scenarios. In SeMPHoNIA, users may decide to participate in multiple auctions at the same time, when the result of one auction may affect the action taken for the other. Agents automate bidding actions and make inferences for determining the optimum path, when interrelated auctions are involved, based on the human user’s preferences and on their local knowledge. The platform currently supports three types of auctions;

English, Vickrey and a hybrid Peer-to-Peer auction (see section 4.2 for a detailed description of our peer-to-peer auction).

We recognize three basic types of agents operating in the SeMPHoNIA platform: A-, C- and CL-agents (section 4.1 outlines another type of agents, the S-agents, that follows the same implementation principles as the C-agents, but presents different functionality).

Auctioneer Agent (A-agent)

The A-agent is the auctioneer's representative in the SeMPHoNIA network. It surveils and coordinates the execution of a specific auction and is responsible for the enforcement of rules governing the negotiation among all conversing parties. Upon initialization, the agent accepts its user's preferences concerning the auction it will conduct, such as the type of the auction, it's code name, the start and end time, the reserve price, the bid increment value, etc. This information is captured in the product's domain ontology that the user has published on the network. As the auction progresses, the A-agent contacts both the participant agents and the auctioneer's platform to exchange data via a standard set of messages, whose format is specified by the negotiation interface (see also section 6.1). Finally, the A-agent declares auction termination and announces winning offers according to the negotiation rules.

Customer Agent (C-agent)

Customers in SeMPHoNIA may initiate one or more auction sessions, participating concurrently in one or more auctions in each of them. Each session has one coordinator agent, the C-agent, whose role is to manage the distinct sub-tasks that a session is decomposed into, which involve the different auctions that the customer has selected to bid in. This agent represents the user's intelligent interface to the platform because it performs the necessary actions to achieve the goal of purchasing the desired product with the best to its owner profit among all auctions that it monitors. The C-agent controls the allocation of bids across the auctions, relying on information about their progress and on its internal strategy for pursuing and maintaining its goal, but does not participate in any of them directly.

Clone Agents (CL-agents)

The CL-agents are the actual participants in auctions conducted in the SeMPHoNIA marketplace. These agents are created by the C-agent inheriting the initial knowledge concerning their user's preferences, i.e. the maximum price they are allowed to spend for an item, the number of items they should intend to acquire etc. They react to stimuli by both the A-agent, informing them about the progress of the auction they participate in, and the C-agent, instructing them to continue bidding or postpone their execution in case this serves best the session's evolution (for example when another auction shows better prospects). CL-agents are specialized according to the type of the auction that has been assigned to them (English CL-agent, Vickrey CL-agent, Peer-to-Peer CL-agent) and the bidding strategy that the user wants to follow (aggressive, passive, greedy, last-minute bidding etc.). They all possess the same initial characteristics and knowledge with their corresponding C-agent, but present differentiations in their behavior, which explains the reason for their characterization as *clones* of their C-agent.

SeMPHoNIA multi-agent system structure

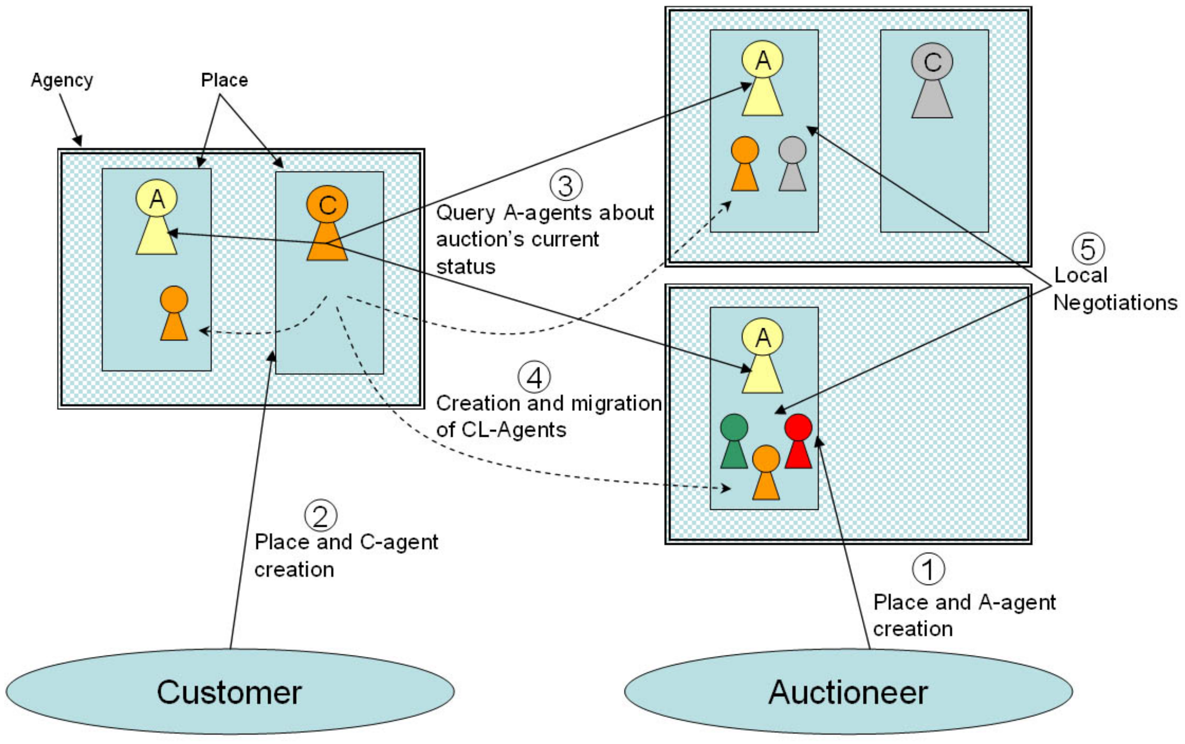

Agency
Place
A
C
A
C
A
③ Query A-agents about auction's current status
④ Creation and migration of CL-Agents
② Place and C-agent creation
① Place and A-agent creation
⑤ Local Negotiations
Customer
Auctioneer

Figure 12 Interactions occurring in SeMPHoNIA's multi-agent component.

As it has already been described, the multi-agent layer of the SeMPHoNIA infrastructure is devoted in the realization of the negotiating part of the system. The platform combines the elements of agents, agencies and places to structure a well-organized morphology for the entities that comprise this layer. The design of this structure, along with the fundamental interactions between the various components, is presented in figure 12. Certain hosts (peers) on the network offer agencies, locations where agents are instantiated and interoperate. Agencies contain user-created places that are used to segment the agent interaction space into groupings of negotiating cells.

For each new auction that an auctioneer initiates, a new place is created, named after her/his identifier followed by the code name of the session. This place hosts the A-agent, that is going to preside the auction and all the CL-agents that will register in it. Respectively, when a customer starts a new session, a place is created, possibly at a different agency than the one that the auction is running, having as name the identifier of the user followed by the name of the session. In this place the C-agent is created, which, in turn, generates the appropriate CL-agents and routes them to the places, where the auctions are actually conducted. This technique grants flexibility and efficiency to the structure, because it takes advantage of local interactions between agents during the process of negotiation in an auction.

The following example illustrates the five basic steps of communication among the SeMPHoNIA agents and the actions that are involved (see also figure 12).

1st Step: An auctioneer decides to create a new auction. The platform creates a unique place for this auction and an A-agent that operates under its owner's preferences (i.e., the auction type that the user desires, the starting and ending time of the auction, the reserve price, the increment bid step, etc.).

2nd Step: A customer has discovered three auctions in the network that s/he is willing to participate in. The platform creates a unique place for this user's session and a C-agent that operates under its owner's preferences (i.e., which auction to bid, up to which offer limit to bid, for how many product to bid, etc.).

3rd Step: After the C-agent has completed successfully its initialization, it contacts all the A-agents of the auctions in which it intends to participate. The purpose is to update its knowledge with information about the running status of these auctions (i.e., current maximum bid accepted, current number of participants, etc.).

4th Step: The C-agent creates three CL-agents, whose type depends on the auction that they will be appointed to and the bidding strategy the user prefers to follow. The CL-agents possess the knowledge concerning their owner's preferences. After their initialization, the C-agent routes the clones to the auction place that corresponds to each one of them.

5th Step: The CL-agents register in the auction and negotiate locally with the other participants and the A-agent, informing their C-agent about their progress at regular time intervals.

3.2.3 Peer-to-Peer Character

The peer-to-peer layer is the cornerstone of the SeMPHoNIA platform, upon which all its functionality is structured. SeMPHoNIA's peer-to-peer network is a computing resource sharing, knowledge sharing and asynchronous message passing system that implements the virtual auction marketplace environment. The network's topology promotes scalability, while its flexible design makes it a useful device for researchers to experiment with other trading scenarios, as well (for example Web Service discovery and composition).

Three are the basic types of SeMPHoNIA peers: customer, auctioneer and operator peers. Figure 13 shows the different types of peers and their interconnection.

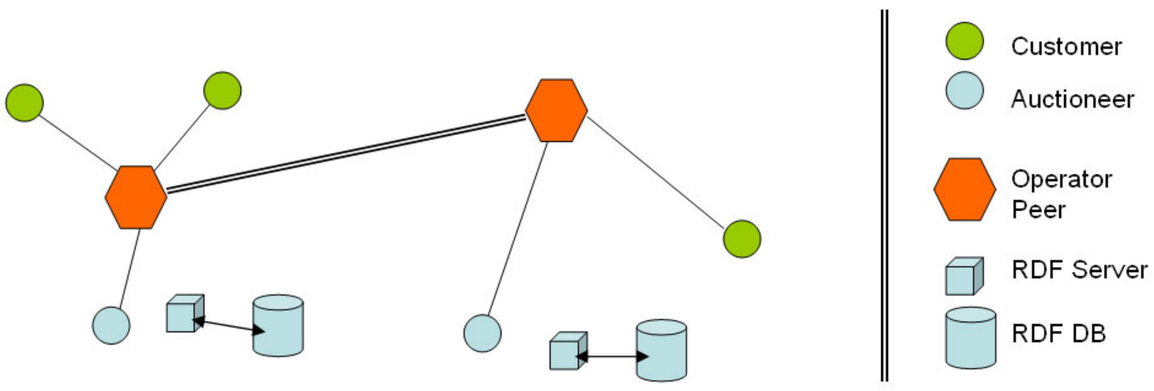


Figure 13 Elements of SeMPHoNIA's peer-to-peer component.

Customer Peers

Customer peers serve as single end users, allowing participants to obtain an identity on the network and gain access to its knowledge space. Every node on the network, including customer peers, contributes to increasing the level of connectivity to the overall network, due to its ability to cache locally XML advertisements of published resources and automatically deliver them to interested peers upon request without any need for human involvement (see section 3.1.1 for a general description of JXTA peer characteristics).

Auctioneer Peers

Auctioneer peers are equivalent to customer peers in matters of network status, but present different functionality. They provide auction services to the network and share their underlying ontologies for common use. Therefore, they are always accompanied by an ontology database, describing the selling products, along with the corresponding RDFSuiteServer, for allowing other peers to query their metadata descriptions. They publish the domain and the address of their server using XML advertisements, as it will be explained in the following section.

Operator Peers

Operator peers are a specialized type of peer in the SeMPHoNIA network infrastructure, serving a twofold role; to enhance message propagation among distant peers and private networks and to provide allocation of computing resources.

Their first goal is achieved by inheriting the characteristics of the traditional JXTA Rendezvous peer (see section 3.1.1). Acting as rendezvous peers, they function as bridges to connect different local or private networks and apply message propagation algorithms to their underlying peers, in order to avoid message flooding.

The second objective is accomplished by their assignment to provide one or more agencies to the network. This means that operator peers supply the medium, where all auction operations take place. All the agents that are created by customer and auctioneer peers live and interact in agencies offered by operator peers. This scheme allows allocation of computational needs among multiple resources across the network.

By combining the potentials and characteristics of these three types of peers, we structured the SeMPHoNIA peer-to-peer network architecture as a network that materializes scalable and flexible design, supporting loosely couple communities, where each peer can join and leave the network at will. We make the convention that operator peers are nodes with high availability offering high computing capabilities. Still, even these peers are free to disconnect without any side effects, due to the ability of agents to migrate from one place to the other. Auctions running on agencies that belong to an operator peer that needs to leave the network, migrate to another peer's agencies and continue their execution there with no loss of data.

Another important notion of the SeMPHoNIA peer-to-peer network architecture is its peer grouping concept. Peer groups are used to segment the network space into distinct communities of peers organized for a specific purpose. In SeMPHoNIA, for every auction listed on the network a new peer group is created by the auctioneer peer. Whenever a customer peer decides to participate in an auction, it must first join the corresponding auction peer group and only after the admission is granted, the peer is allowed to send its CL-agent to the auction place. The procedure of joining a peer group is simple and executed automatically by the user's platform. Each peer group has a membership policy that governs who can join. Before a peer can interact with the group, it needs to apply for membership, obtain the appropriate credentials to establish its identity within the group and then join. Auctioneer peers may create peer groups with extra capabilities, each providing its own set of services to members of the group (for example secure or private peer groups implementing private auctions etc.).

3.2.4 SeMPHoNIA platform

All the previously described entities, components and features are integrated in the SeMPHoNIA platform to implement a complete and well-defined e-trading environment. This section presents how the different layers of functionality co-exist and collaborate to constitute the overall system infrastructure.

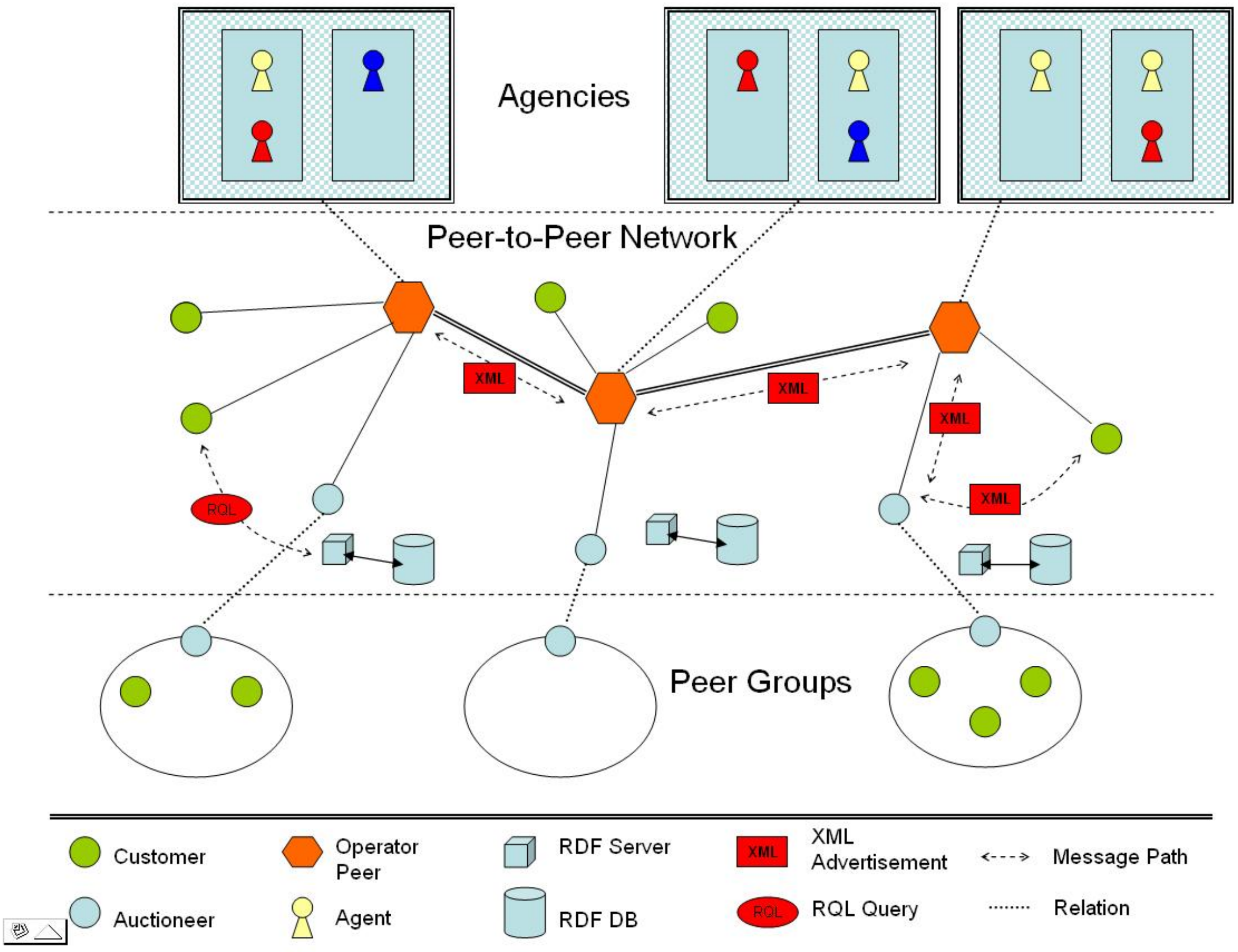


Figure 14 SeMPHoNIA platform overview.

Figure 14 displays a snapshot of the system's state at a random moment. The middle part shows a fraction of the peer-to-peer network. Customer and auctioneer peers are connected with operator peers, which in turn interconnect with each other to form a network of main channels. Advertisements travel between peers on the same local network or between operator peers and are cached locally at various nodes throughout their path. These advertisements may describe different published resources, such as product domains, ontology locations, auction peer groups, agency addresses or simply the presence of peers. The lower part of the image depicts the correspondence between auctioneer peers and their own auction peer groups, that they have created. Auctioneers may create multiple peer groups, one for each auction they conduct. This layer also presents the virtual presence of customer peers in the peer groups. Customer peers, when joining a peer group, obtain a unique identity in it that is used for interaction with other members and for security reasons. Last, the upper part of figure 14 displays the multi-agent layer of the system, which is the component that undertakes all the auction sessions. The dashed line implies the connection between an operator peer and the agencies that it grants to the system. The other peers create agents that can travel between different agencies, as well as, places for negotiating contracts on behave of their human users.

The following example illustrates a typical transaction in the SeMPHoNIA environment. Let's assume that a retailer desires to auction Olympic Game tickets on the network. The first action would be to create two ontologies, one for capturing all the necessary information concerning the specification of the tickets (for example, event type, event date, seat, etc) and one describing the auction template (for example, auction type, start and end time etc.). These ontologies should be validated by the RDFSuite and stored locally on its database. Once the preparatory work is completed, the user initiates the SeMPHoNIA Auctioneer Application for registering with the peer-to-peer network. A new peer is created and connected with an operator peer to obtain global access. In order for the system to generate the new auction session, the only information required by the human user is the address of the RDFSuiteServer; all the rest are performed automatically. These include the following actions: first of all, an XML advertisement is constructed and published that describes the ontology of the products that the auctioneer supplies (Ontology Advertisement). This advertisement is characterized by the domain of the product (tickets in this case) and holds information about the address of the specific RDFSuiteServer, in order for customer peers to contact the ontology and query its contents. The advertisement is propagated to all the neighboring peers and to the connected operator peer, which, in turn, informs other operator peers. At the same time, a peer group for that auction is created, identified by a unique Peer Group ID, while its XML advertisement (Peer Group Advertisement) is published to inform others about its existence. The auctioneer peer is the first peer to join the newly created group and, as a result, becomes the super-peer in it. Then, the system contacts one of the operator's agencies and creates a place in it, which will be the place, where the actual auction will be conducted. Finally, it queries the ontology to acquire all the necessary information for the A-agent to manage the auction and constructs the new A-agent, providing as start-up parameters this information, instructing it to initiate the new auction session. From that point on, the execution is transferred to the agents' pertinence.

At some other place on the network, a customer is interested in purchasing Olympic Game tickets. In order for him to register with the SeMPHoNIA network, he initiates the SeMPHoNIA Customer Application and a new peer is created seeking for an operator peer to connect with. The customer's first action would be to discover ontologies describing Olympic Game ticket auctions and, among the available, locate the auctions that interest him most. The first task is accomplished by searching the network for Ontology Advertisement, while the second by completing a Semantic Search Session. In order to initiate the search, the only information needed is the domain of the products he is interested in. The application constructs and sent on the network an Ontology Discovery Query message seeking for advertisements that refer to that specific domain. All responses received (Ontology Advertisements) are cached locally and contain information about the RDFSuiteServers that the customer should contact in order to access and query the auctioneer's database. Using the Semantic Search Engine graphical user interface, he is free to forward RQL queries to all the ontologies simultaneously and view the answers.

Once the customer concludes on which tickets he prefers, the next and final step would be to contact the auctions running for these tickets, join them and start a new bidding session. These tasks are preformed automatically by the application, requiring as sole input the user's preferences, concerning that session (i.e., the maximum price his is willing to pay for the tickets, the amount of tickets etc.). The application sends a Peer Group Discovery message on the network tagged with the resources of the products the customer has selected, in order for other peers to respond with advertisements that refer to the auctions running for those products. When the responses arrive, the application applies for membership on behave of the user to the corresponding peer groups and contacts a remote agency to create a place in it, which will host the C-agent that will plan and organize the session. The C-agent's initial knowledge is consistent with its owner's preferences. Moreover, after the C-agent is invoked, it creates the appropriate CL-agents and routes them to the remote places, where the auctions are conducting. At frequent time intervals, the agents inform the user about the progress of their auctions.

From that point on, communication is conducted on the multi-agent layer. The A-agent informs all CL-agents registered in its auction about new bids, which in turn inform their C-agent. The last decide on the strategy that should be followed during the session, when multiple auctions are involved, and informs the customer's application. The customer is free to terminate this session at any time, instructing his agents to stop placing bids. He is also given the ability to initiate new bidding sessions with the same or other auctions.

At any given time, peers can leave the network without causing any damage to their sessions. The agents that have been assigned the task of representing their owner's interests on the negotiations exist on agencies that operate on remote host, which are presumed to be available at most times. Even when a peer is not connected with the network, its running agents continue to live and interact at those remote agencies. Peers can reconnect at any time in the future to update their knowledge about the progress of their sessions. In addition, all the published resource advertisements remain present in the network, independent of their peer's connection status, due to the ability of other peers to cache and propagate advertisements in a transparent to the user manner.

This chapter covered numerous aspects of SeMPHoNIA's architecture; its structure and components, the way these components interact with each other, the flow of information inside the system, etc.. The next chapter will present a number of additional functionalities that SeMPHoNIA implements, in order to facilitate users in accomplishing electronic interactions and provide a complete e-Commerce experience.

Chapter 4 OTHER IMPORTANT FEATURES

SeMPHoNIA is a multi-purpose virtual market architecture that includes numerous advanced features for supporting human users in accomplishing electronic negotiation tasks. This section discusses some additional facilities that the platform integrates, which provide the supplementary infrastructure needed to become a complete and innovative system.

4.1 Statistical Agent

During negotiation, the ability of agents to adapt to conditions encountered in their world, foresee potential situations and modify their behavior accordingly is a major advantage towards improving the quality of their decision functions. An essential skill for achieving such behavior is to be able to learn from previous interactions or from their environment. Particularly in auction negotiations, the ability of agents to learn can be utilized in understanding their opponents' preferences, bidding strategies, actions or plans, utility functions and even predict their responses in order to, more efficiently, evaluate optimum counter-proposals and avoid unfavorable auction sessions. The availability of sound and unbiased feedback information is a primary requirement for enhancing this process, as it is recognized in [78]. In addition, the history of bidding of a product is vital information, when bidding across multiple auctions, to procure the best deal for the desired item. Currently, few of the on-line auction houses facilitate their customers in analyzing those data.

For that reason, a special type of agent has been developed in SeMPHoNIA, whose role it to yield feedback of previous auctions and produce valuable auction statistics. This agent, named as Statistical Agent (S-agent), automates the process of monitoring multiple auctions with varying start and end times. It follows the same interaction mechanism as C-agents; it creates clone agents that travel to the remote place, where the auction is running, recording its progress, while the master agent coordinates the session and extracts statistical data obtained by all the clones. Thus, users are provided with the option to only monitor auction sessions, relieved from the procedure of registering in them and placing bids. The platform offers them the ability to decide which auctions they prefer to monitor, regardless of their type, the items auctioned etc, using the Semantic Search Engine module for discovering on-line sessions on the network. Auction statistics may include average winning bids, price convergence behavior, equilibrium price, progress of bidding according to time etc.

The current implementation supports the dynamic provision of product auction statistics to human users. A challenge would be to create a decision-support integration mechanism between S- and C-agents that would partially or fully automate decision-making based on collected data, allowing C-agents to dynamically alter their strategy in specifying appropriate bids.

4.2 Flexible Agent Design – Peer-to-Peer Auction

In chapter 3 we have described the design of SeMPHoNIA's agent architecture for multiple auction participation, according to which one agent (C-agent) is responsible for managing the session, while a number of other agents (CL-agents) register in the auctions and place bids. It has been made clear that the client's negotiation strategy in each individual auction is expressed exclusively by the CL-agents and their inference mechanism. Therefore, a different type of CL-agent is invoked according to the type of the auction (English CL-agent, Vickrey CL-agent etc.) and the bidding strategy that the client wishes to employ (aggressive, passive, greedy, last-minute bidding etc.). The C-agent's role is to pursue the purchase of the desired product in a manner consistent with the client's preferences and agreeing the best deal among the auctions.

We argue that this design can provide high levels of flexibility and scalability to the platform. By originating custom-built CL-agents, any user is offered the ability to employ effective and novel strategies that match her/his own personal style. Custom-made CL-agents can be used by agent researchers to experiment with new AI tactics and bidding algorithms without having to re-design the whole negotiation scene; the modification of the agent's reasoning function is adequate. The only compromise that must be made is to preserve the interoperability between CL-agents and the other components of the SeMPHoNIA network (user application, C-agent, A-agents). For users with no programming background on the other hand, CL-agents could be handled as black boxes, characterized by their codified features (i.e., protocol support, implemented strategy), and exchanged between other users as separate packages to enhance their application's capabilities.

To demonstrate the power of this design we have proceeded one step further. Apart from the popular iterative English auction and the single-cycle Vickrey auction, both of which rely on a central auctioneer to direct the session, we have developed a decentralized or peer-to-peer continuous auction.

The vast majority of on-line auction houses perform centralized auctions (for example English, Dutch, Vickrey etc.), in which the clients do not negotiate with each other but rather with the auctioneer exclusively, who distributes information about offers among them. However, peer-to-peer auctions receive increasing attention, due to the absence of a central role and the drawbacks that this scheme implies. Moreover, according to [70], measuring the number of bidding rounds and the number of message rounds required to reach convergence in peer-to-peer and in traditional centralized auctions around an auctioneer, as a function of the number of trading agents, N, it has been found that:

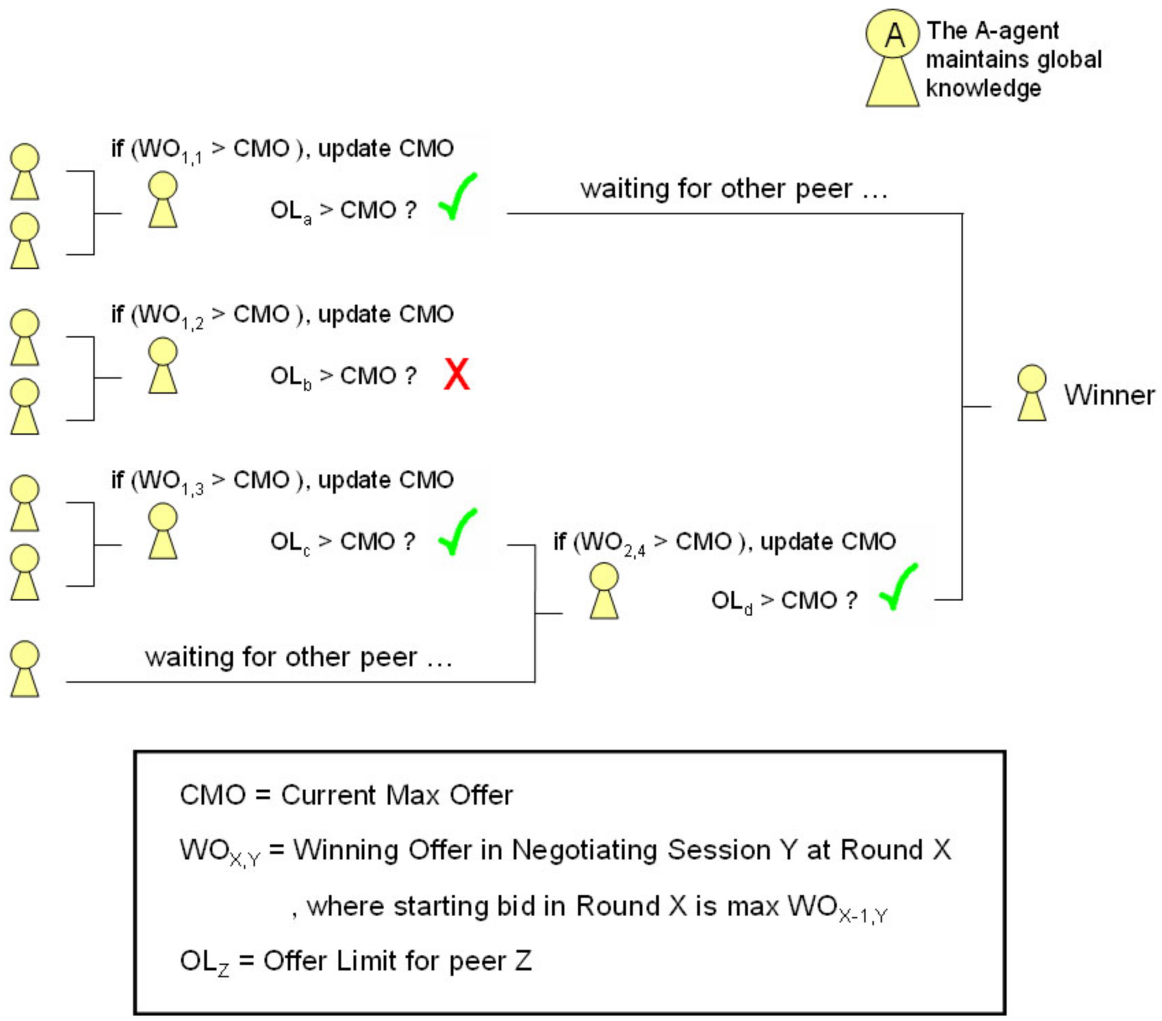
- the peer-to-peer auction displays price convergence behavior similar to that of centralized auctions,

- in bidding rounds, the rate of convergence is independent of N in both auctions and approximately two times faster in the centralized auction,

- the number of messages per bidding round to and from any entity in the peer-to-peer auction is constant, while an auctioneer must handle a number of messages in each bidding round that grows linearly with N,

- considering message round costs, the peer-to-peer auction is at least 100 times more efficient than the centralized auction.

It could also be noted that the physiognomy of peer-to-peer auctions is correlated to that of more general negotiation scenarios making them well suited to a variety of applications.


A
The A-agent maintains global knowledge
if ($WO_{1,1}$ > CMO), update CMO
OL_a > CMO ? ✓
waiting for other peer ...
if ($WO_{1,2}$ > CMO), update CMO
OL_b > CMO ? X
Winner
if ($WO_{1,3}$ > CMO), update CMO
OL_c > CMO ? ✓
if ($WO_{2,4}$ > CMO), update CMO
OL_d > CMO ? ✓
waiting for other peer ...

CMO = Current Max Offer

$WO_{X,Y}$ = Winning Offer in Negotiating Session Y at Round X

, where starting bid in Round X is max $WO_{X-1,Y}$

OL_Z = Offer Limit for peer Z

Figure 15 Peer-to-peer auction protocol.

Thus, in the context of the SeMPHoNIA project, we have developed a hybrid peer-to-peer auction, in which agents negotiate in pairs. Figure 15 displays a diagram illustrating the interactions occurring in this auction. Agents register in the auction and begin by seeking a random party to negotiate with. In each pair, the agents trade until one of them withdraws its offering permitting the other to qualify to the next round. The negotiation between any two agents is private, the winning offer, though, should be announced to the other participants and to the A-agent, in order to update, if necessary, the current maximum offer of the auction. At the beginning of each negotiation round, agents start trading considering as opening price the auction's current maximum offer. Agents that do not wish to exceed this limit do not qualify to next rounds. The A-agent, throughout the evolution of the auction, preserves a passive role; it guards the valid execution of the auction rules and maintains information about global knowledge, such as the current maximum bid and the identities of the agents that qualify from round to round. All the negotiation sessions are executed asynchronously, therefore it is possible that the current maximum offer be updated many times during one round.

The important thing to note is that the extension of the platform with new negotiation protocols, even when they present such vast differences in their operation as centralized and peer-to-peer auctions, requires no modification in the platform as a whole or in any of its elements. For the generation of the peer-to-peer auction the only requirements were the creation of the specialized peer-to-peer CL-agent and the enhancement of the A-agent's functionality to acknowledge the existence of such auctions and support their initiation. Neither the C-agent nor the A-agent needs to have the protocol hard-coded explicitly beforehand.

4.3 Synchronization Policies

For systems, such as SeMPHoNIA, that instantiate open distributed environments or manage many simultaneous auctions, synchronization between peers is a critical issue. The nature of such applications necessitates a design that tolerates network and system disruptions and realizes cross-platform accuracy utilizing careful timekeeping procedures. This section explains briefly the policies followed in SeMPHoNIA to ensure synchronization and accuracy at a satisfactory level.

Both the peer-to-peer network and the multi-agent component of the SeMPHoNIA platform are asynchronous environments, therefore a global reference time is a compulsory requirement to keep track of the flow of messages exchanged between remote entities. In SeMPHoNIA, all peers refer to Greenwich Mean Time (GMT), while all messages are timestamped to ensure the platform's fidelity. Thus, auctioneers are able to schedule events and determine the validity of message sequence occurring during them.

SeMPHoNIA applications invoke the appropriate Java objects to implement local timekeeping. Time reflection using Java classes, though, depends on the host environment of the Java Virtual Machine [49]. This approach introduces an important side-effect; peers across the network are not synchronized and even agents created by the same user, but running on different machines, maintain different time images for the same snapshot. A synchronization algorithm has been developed for that purpose that is invoked during the phase of agent creation and aims at determining the time interval between the agent's runtime environment and the host system that initiated the agent. The algorithm is based on the Ping/Pong procedure, according to which the agent sends a ping message to the user's platform and measures the time needing for the response to arrive, estimating network delays and the declination between the two clocks.

Beside that, even the design of the agent negotiation scheme supports accuracy during auction execution. The ability of CL-agents to migrate to the place, where the actual auction is conducted, provides them with the flexibility to exploit local interactions. All unpredictable message delays due to network traffic are avoided, but, most important, all agents refer to the same system clock, since they operate on the same host. In addition to this, they are able to apply accurate algorithms for dynamically calculating the length of their bidding determination cycle and improving their strategy. To do so, they take into account variables and parameters of their local host that are considered to be more stable, because they always depend on the same computing resource.

4.4 Autonomous execution

Great emphasis has been given in generating user-independent sessions that automate most of human procedures, even the ones that require a certain degree of intelligence. The platform has been designed to support human interference in sophisticated tasks and to preserve user-created executions in an autonomous manner, even when the user is not connected on the network.

Automation is accomplished by exploiting certain features of agent technology. Intelligent agents are able to inference, deliberate and communicate without the need of human guidance but rather based on a set of start-up parameters, on changes occurring in their world and on a set of inborn knowledge or beliefs [103]. Such agents are utilized in SeMPHoNIA to fully automate the user's participation in multiple auction creation, monitoring and bidding. Users (either customers or auctioneers) are free to go offline, after initiating a new session, and trust the execution on the "hands" of software agents. The platform offers a mechanism for connecting and disconnecting with the system, maintaining profile information locally for all users, so that they can keep track of their sessions' progress and recover execution whenever they desire to.

In addition, the peer-to-peer network is capable of providing a robust and consistent environment for supporting the negotiation infrastructure. All peers can leave the network at any time, even operator peers, without causing any inconveniences to the published resources or running auctions. Mechanisms, such as advertisement caching, controlled message propagation, agent migration between agencies etc. support this approach.

This chapter presented a number of additional functionalities of SeMPHoNIA platform. Their intention is to provide the means for assisting users in accomplishing electronic trading task and enhancing their interaction with advanced techniques. The next chapter describes the steps for the creation and execution of auction services in SeMPHoNIA, providing extensive walkthrough examples and screenshots.

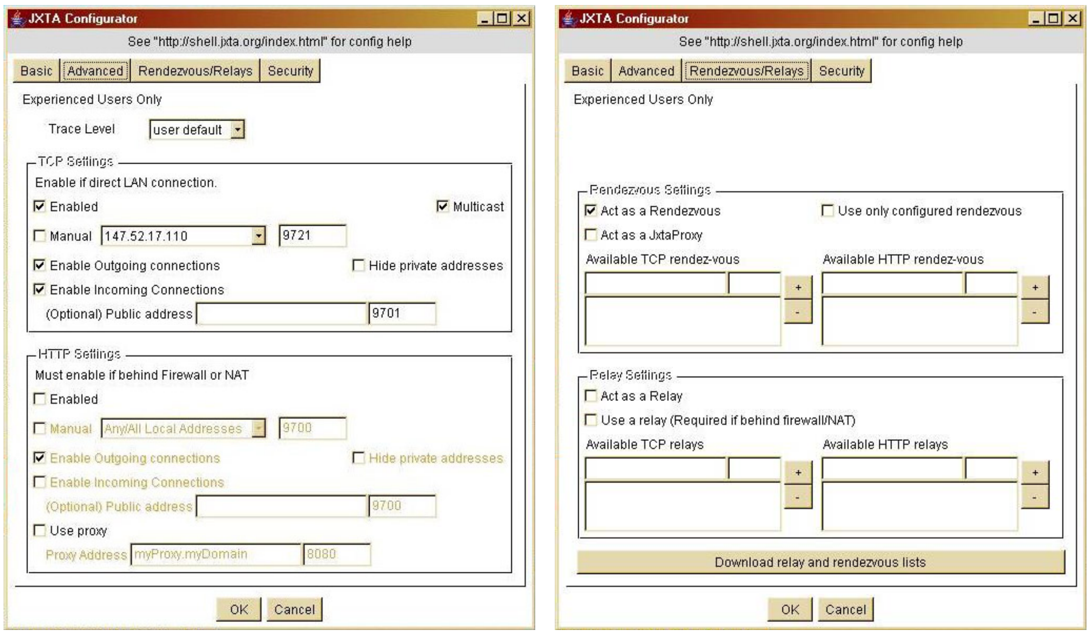
Chapter 5 CREATION OF SeMPHoNIA AUCTION SERVICES

This chapter contains directions for participating in the SeMPHoNIA environment. It outlines the steps users must take and the information needed to play the role of the operator, auctioneer or customer in the system. Extensive guidelines and screenshots are presented, concerning all available functions offered to the user. The screenshots are taken from the Windows 2000 Professional platform, therefore minor differences, regarding the window layout might be present, compared to other systems (i.e. UNIX). Apart from the graphical user interface, the SeMPHoNIA applications provide information to the user by displaying messages, warnings and errors in the command line.

5.1 Operator Application

The operator's basic contribution to the SeMPHoNIA platform is to provide other users with resources for auction execution, relying on the module of Grasshopper Agency. In parallel, it operates as a rendezvous peer in the peer-to-peer network. Prior to launching the operator application, the user must be aware of the location, where the Region, that controls all agencies in the SeMPHoNIA platform, is running.

At the first time any application that implements JXTA libraries is launched (including SeMPHoNIA's operator, auctioneer and customer applications), an auto-configuration tool (JXTA Configurator) is displayed to allow users to specify configuration information for TCP/IP and HTTP settings, rendezvous and relay peers parameters, as well as other security issues for the custom peer-to-peer network. This information is stored locally and future executions rely on it to configure the peer. For detailed instructions concerning the JXTA Configurator, please consult [54].

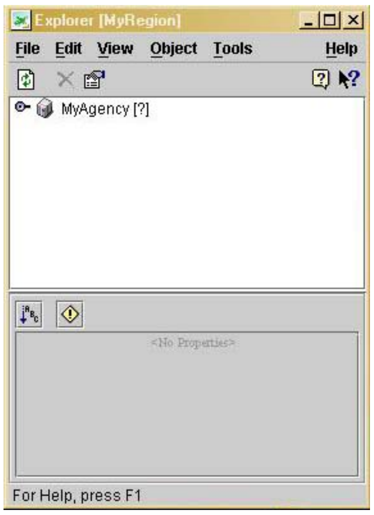

JXTA Configurator
See "http://shell.jxta.org/index.html" for config help
Basic | Advanced | Rendezvous/Relays | Security
Experienced Users Only
Trace Level: user default
TCP Settings
Enable if direct LAN connection.
☑ Enabled ☑ Multicast
☐ Manual 147.52.17.110 9721
☑ Enable Outgoing connections ☐ Hide private addresses
☑ Enable Incoming Connections
(Optional) Public address 9701
HTTP Settings
Must enable if behind Firewall or NAT
☐ Enabled
☐ Manual Any/All Local Addresses 9700
☑ Enable Outgoing connections ☐ Hide private addresses
☐ Enable Incoming Connections
(Optional) Public address 9700
☐ Use proxy
Proxy Address myProxy.myDomain 8080
OK Cancel

JXTA Configurator
See "http://shell.jxta.org/index.html" for config help
Basic | Advanced | Rendezvous/Relays | Security
Experienced Users Only
Rendezvous Settings
☑ Act as a Rendezvous ☐ Use only configured rendezvous
☐ Act as a JxtaProxy
Available TCP rendez-vous Available HTTP rendez-vous
Relay Settings
☐ Act as a Relay
☐ Use a relay (Required if behind firewall/NAT)
Available TCP relays Available HTTP relays
Download relay and rendezvous lists
OK Cancel

Figure 16 JXTA Configurator Parameters.

Figure 16 shows the "configurator window" that will be displayed the first time the operator program initiates. The user must specify the TCP and HTTP settings as shown and remember to check the "Act as Rendezvous" option, to enhance the peer's functionality.

After this task, the user is allowed to publish one or more agencies on the SeMPHoNIA network. These agencies should be launched on the local machine and be available throughout the operator peer's lifecycle. But, most important, they should be registered with the Region Registry (figure 17). This will ensure location transparent communication across all agents in the Grasshopper environment, regardless of the physical location, where they exist. The process of registering an agency with a Region is explained in the Grasshopper User's Guide [38].



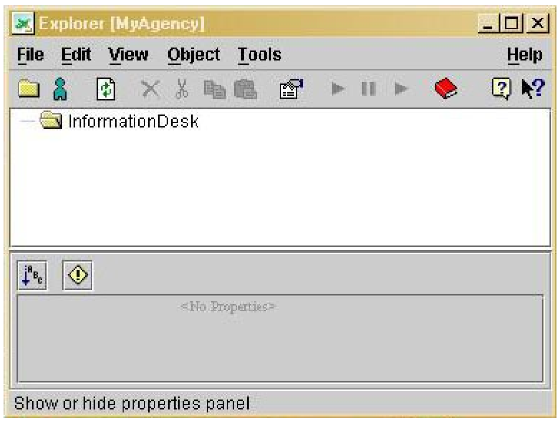
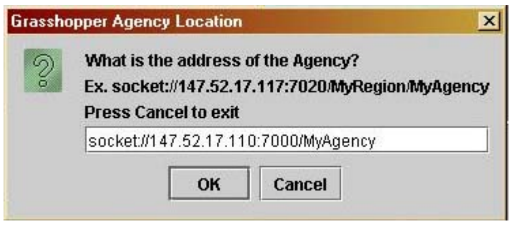

Explorer [MyAgency]
File Edit View Object Tools Help
InformationDesk
<No Properties>
Show or hide properties panel

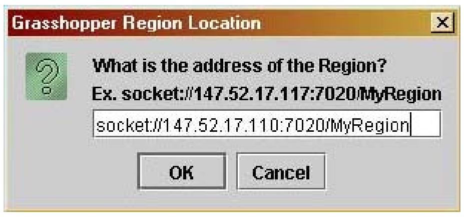
Figure 17 The 'MyAgency' agency has been successfully registered with the region.

The SeMPHoNIA Operator application requests as input the address of the Region and the addresses of the agencies that the user desires to publish. For each agency, it creates an XML advertisement containing all necessary information that other peers in the network would need, in order to communicate with the agencies, as presented in figure 18. After this task is fulfilled, the advertisements are both cached locally and published remotely. Thus, the agencies become publicly known to the network. From this point on, the human user's participation terminates and remote peers are free to search, discover and exploit those agencies, in order to create auction places and initiate agent negotiation sessions.


Grasshopper Agency Location

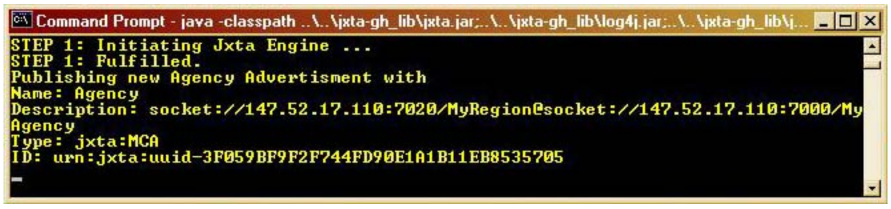
What is the address of the Agency?
Ex. socket://147.52.17.117:7020/MyRegion/MyAgency
Press Cancel to exit

socket://147.52.17.110:7000/MyAgency


Grasshopper Region Location

What is the address of the Region?
Ex. socket://147.52.17.117:7020/MyRegion

socket://147.52.17.110:7020/MyRegion


Command Prompt - java -classpath ..\..\jxta-gh_lib\jxta.jar;..\..\jxta-gh_lib\log4j.jar;..\..\jxta-gh_lib\j...

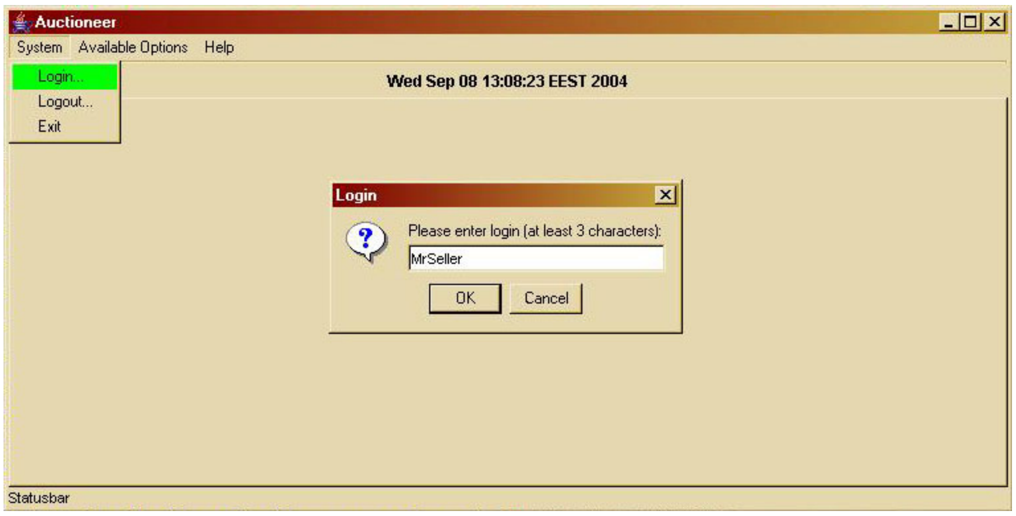
```
STEP 1: Initiating Jxta Engine ...
STEP 1: Fulfilled.
Publishing new Agency Advertisment with
Name: Agency
Description: socket://147.52.17.110:7020/MyRegion@socket://147.52.17.110:7000/My
Agency
Type: jxta:MCA
ID: urn:jxta:uuid-3F059BF9F2F744FD90E1A1B11EB8535705
_
```

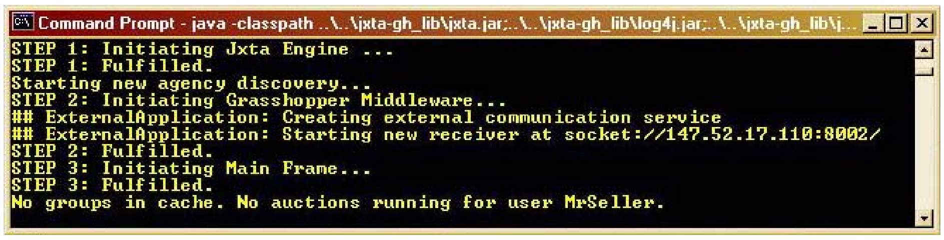
Figure 18 Publication of a new Agency Advertisement.

5.2 Auctioneer Application

In the SeMPHoNIA platform the same facilities and privileges are offered both to auctioneers and customers, therefore the interaction with the system and the mechanisms triggered by human users follow similar patterns. The auctioneer application connects with operator peers, in order to locate agencies and is able to publish one or more RDF ontologies, each containing metadata about one or more products. In addition, supports users in initiating, monitoring and terminating auction sessions. The A-agent that is responsible for surveilling and protecting the auction execution on behave of the human user is capable of handling three types of auction; English, Vickrey and Peer-to-Peer (please refer to section 3.2.2 for information about A-agents and the corresponding auction types).

The first time the auctioneer application is launched, the JXTA Configurator tool is displayed, as described in the Operator application section. The user is requested to specify the communication IP address and port for configuring the TCP settings for the peer on the network. These settings will guide the application in all future executions.



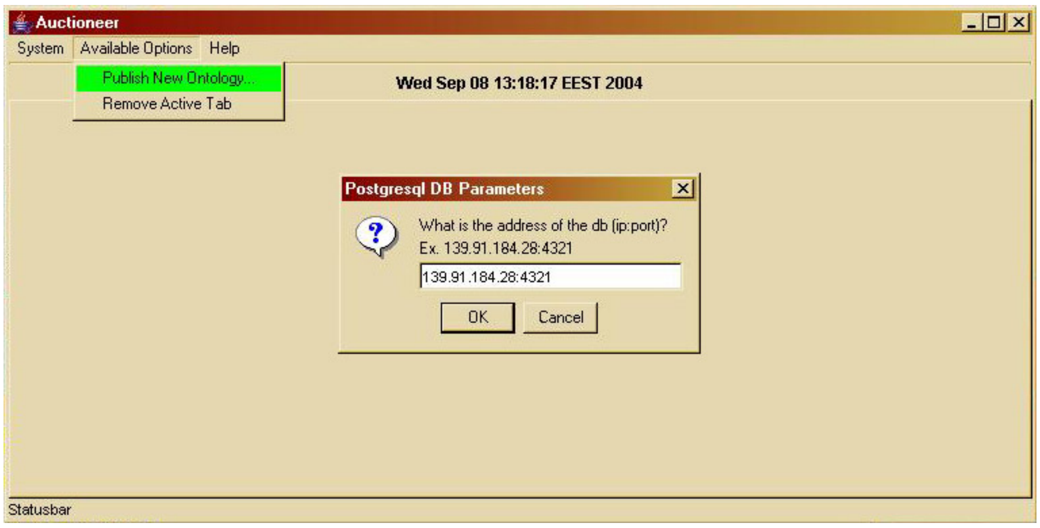

Command Prompt - java -classpath ..\..\jxta-gh_lib\jxta.jar;..\..\jxta-gh_lib\log4j.jar;..\..\jxta-gh_lib\j...

```
STEP 1: Initiating Jxta Engine ...
STEP 1: Fulfilled.
Starting new agency discovery...
STEP 2: Initiating Grasshopper Middleware...
## ExternalApplication: Creating external communication service
## ExternalApplication: Starting new receiver at socket://147.52.17.110:8002/
STEP 2: Fulfilled.
STEP 3: Initiating Main Frame...
STEP 3: Fulfilled.
No groups in cache. No auctions running for user MrSeller.
```

Figure 19 Auctioneer Login process.

Before any person can interact with the SeMPHoNIA platform, s/he must first be appropriately registered, by providing a unique identifier through the process of login (figure 19). This identifier is kept locally and is used by the system to maintain profile information for every member and regain auctions that the user might had left running in previous executions. Therefore, after the user logs in the SeMPHoNIA environment, the system seeks for the specific user's sessions in agencies across the network. The application knows where to look for, because it keeps information about peer groups that a user had created in the past, which leads to all of her/his remote auctions. Then, it informs the user about the result and triggers the A-agents' "reconnection" function, in order to inform the application about the auctions' current status. Respectively, the user may choose to either logoff or exit at any time, leaving all of her/his sessions running at the remote agencies.

The process of creating an auction in SeMPHoNIA is fully automated and human users are completely relieved of unnecessary and complicate procedures. Indeed, initiating a new auction is accomplished in one step, provided that the product's RDF ontology has already been validated and stored in the RDFSuite database. To construct a new auction service, users are just prompted to indicate the address of the RDFSuiteServer, which is responsible for mediating between the external application and the RDFSuite (figure 20). On receiving a new RDFSuiteServer address, the system initiates a set of actions that involves:



Wed Sep 08 13:18:17 EEST 2004

Postgresql DB Parameters

What is the address of the db (ip:port)?
Ex. 139.91.184.28:4321

139.91.184.28:4321

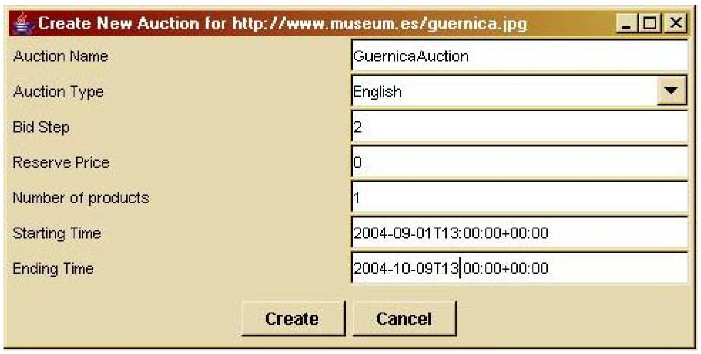
Figure 20 The creation of a new auction requires only the address of the RDFSuite database.

- Querying the RDF ontology for extracting all the necessary information for structuring the new auction, such as the resources of the items that are for sale, the type of auction (English, Vickrey, Peer-to-Peer), the start and end times of the session, the increment bid step, the reserve price etc. An example of RQL query that the system automatically generates for collecting such data is given below:

```
SELECT  Name, typeof(Type), StartTime, EndTime,
        Step, revPrice, NumOfGoods
FROM    {A}sells{Product}, {A}name{Name},
        {A}has_type{Type},
        {A}starting_time{StartTime},
        {A}ending_time{EndTime},
        {A}bid_step{Step},
        {A}reserve_price{revPrice},
        {A}number_of_products{NumOfGoods}
WHERE   Product like "<the resource of the
        product auctioned>"
```

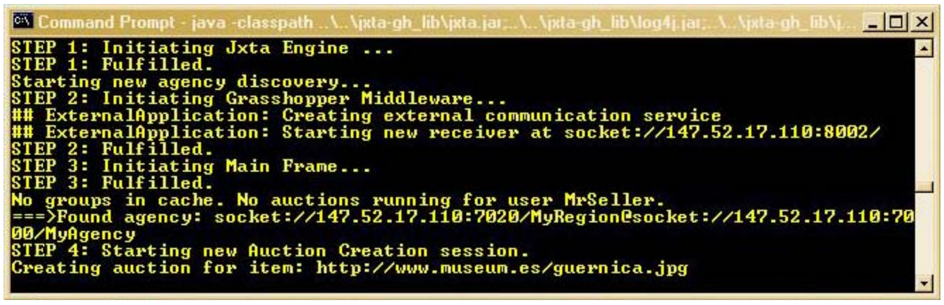
  The application also queries the ontology to determine whether the supplier intents to auction more than one different products, which means that more than one auctions are to be created.

- Then, the application constructs and publishes on the peer-to-peer network an XML advertisement that describes the ontology of the products that the auctioneer supplies (Ontology Advertisement). This advertisement is characterized by the domain of the products (artifacts, tickets etc) and preserves information about the address of the specific RDFSuiteServer, in order for customers to contact the ontology and query its contents.



Create New Auction for http://www.museum.es/guernica.jpg

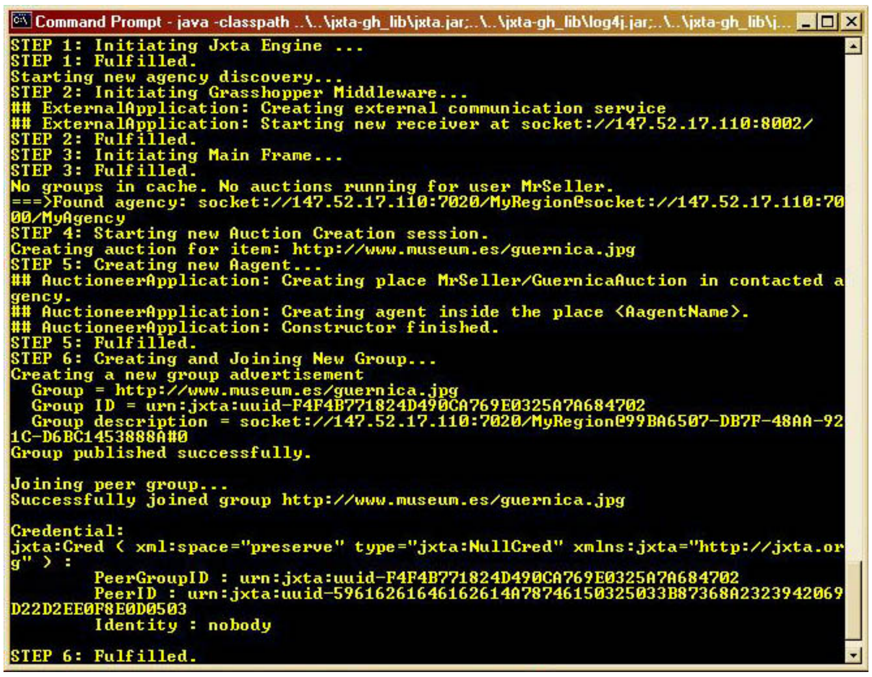
Auction Name	GuernicaAuction
Auction Type	English
Bid Step	2
Reserve Price	0
Number of products	1
Starting Time	2004-09-01T13:00:00+00:00
Ending Time	2004-10-09T13:00:00+00:00



Command Prompt - java -classpath ..\..\jxta-gh_lib\jxta.jar;..\..\jxta-gh_lib\log4j.jar;..\..\jxta-gh_lib\j...

```
STEP 1: Initiating Jxta Engine ...
STEP 1: Fulfilled.
Starting new agency discovery...
STEP 2: Initiating Grasshopper Middleware...
## ExternalApplication: Creating external communication service
## ExternalApplication: Starting new receiver at socket://147.52.17.110:8002/
STEP 2: Fulfilled.
STEP 3: Initiating Main Frame...
STEP 3: Fulfilled.
No groups in cache. No auctions running for user MrSeller.
===>Found agency: socket://147.52.17.110:7020/MyRegion@socket://147.52.17.110:70
00/MyAgency
STEP 4: Starting new Auction Creation session.
Creating auction for item: http://www.museum.es/guernica.jpg
```

Figure 21 Confirmation of data obtained by querying the RDFSuite database.



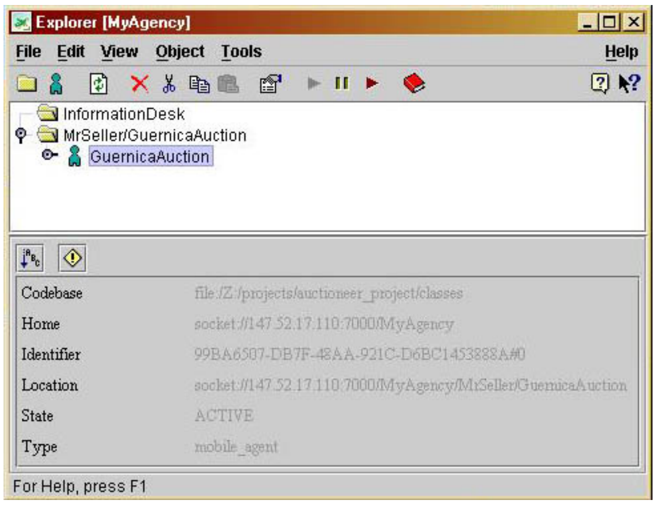
Command Prompt - java -classpath ..\..\jxta-gh_lib\jxta.jar;..\..\jxta-gh_lib\log4j.jar;..\..\jxta-gh_lib\j...

```
STEP 1: Initiating Jxta Engine ...
STEP 1: Fulfilled.
Starting new agency discovery...
STEP 2: Initiating Grasshopper Middleware...
## ExternalApplication: Creating external communication service
## ExternalApplication: Starting new receiver at socket://147.52.17.110:8002/
STEP 2: Fulfilled.
STEP 3: Initiating Main Frame...
STEP 3: Fulfilled.
No groups in cache. No auctions running for user MrSeller.
===>Found agency: socket://147.52.17.110:7020/MyRegion@socket://147.52.17.110:70
00/MyAgency
STEP 4: Starting new Auction Creation session.
Creating auction for item: http://www.museum.es/guernica.jpg
STEP 5: Creating new Aagent...
## AuctioneerApplication: Creating place MrSeller/GuernicaAuction in contacted a
gency.
## AuctioneerApplication: Creating agent inside the place <AagentName>.
## AuctioneerApplication: Constructor finished.
STEP 5: Fulfilled.
STEP 6: Creating and Joining New Group...
Creating a new group advertisement
  Group = http://www.museum.es/guernica.jpg
  Group ID = urn:jxta:uuid-F4F4B771824D490CA769E0325A7A684702
  Group description = socket://147.52.17.110:7020/MyRegion@99BA6507-DB7F-48AA-92
1C-D6BC1453888A#0
Group published successfully.

Joining peer group...
Successfully joined group http://www.museum.es/guernica.jpg

Credential:
jxta:Cred < xml:space="preserve" type="jxta:NullCred" xmlns:jxta="http://jxta.or
g" > :
        PeerGroupID : urn:jxta:uuid-F4F4B771824D490CA769E0325A7A684702
        PeerID : urn:jxta:uuid-59616261646162614A78746150325033B87368A2323942069
D22D2EE0F8E0D0503
        Identity : nobody

STEP 6: Fulfilled.
```

Figure 22 Output produced by creating the new A-agent and publishing the new Peer Group Advertisement.

- The system presents the results to the user and waits confirmation before proceeding to the following steps that affect both the agent world and the peer-to-peer network (figure 21).

- Once the confirmation is affirmative, the system contacts the remote agency and creates a place in it, which will be the place where the actual auction will be conducted (step 5 in figure 22). It also constructs a new A-agent, providing as initial parameters the information acquired by the product ontology and instructs it to start a new auction session. Figure 23 presents how the agency looks after the creation of the place and the existence of the A-agent. The last is named after the corresponding RDF resource that the user had used to characterize the auction in the ontology (property `name`).

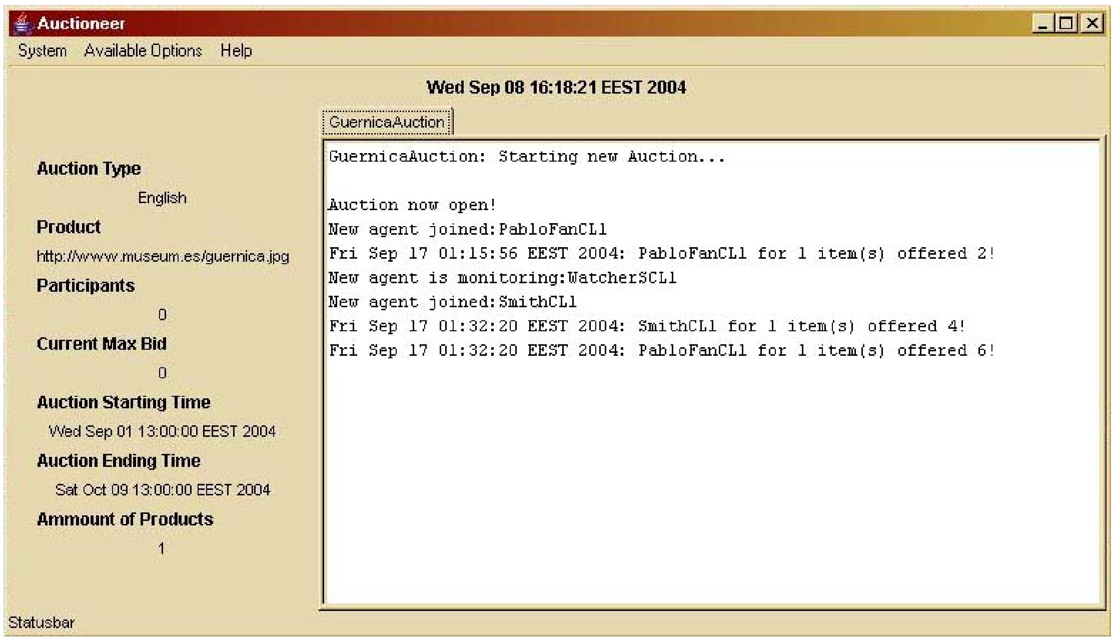


Codebase	file:/Z:/projects/auctioneer_project/classes
Home	socket://147.52.17.110:7000/MyAgency
Identifier	99BA6507-DB7F-48AA-921C-D6BC1453888A#0
Location	socket://147.52.17.110:7000/MyAgency/MrSeller/GuernicaAuction
State	ACTIVE
Type	mobile_agent

Figure 23 An A-agent has been created in the place that will host the CL-agents during the auction session.

- In addition, a new peer group in the peer-to-peer network is created, identified by a unique peer group ID (step 6 in figure 22) and its XML advertisement (Peer Group Advertisement) is published to inform others about its existence. The auctioneer peer is the first peer to join the newly created group and, as a result, becomes the super-peer in it. The system displays both the characteristics of the group (name, ID, description) and the credentials obtained by joining the peer group.

- Finally, the graphical user interface presents useful information about the new auction. Users interact with it, in order to monitor the progress of a session or terminate it. For example the left panel displays the type of auction, the number of participants in the auction, the current maximum bid etc, while the right tabbed panel displays messages about the rotation of bids offered by negotiating agents (figure 24). Data is updated in real time, whenever a change occurs and users are free to browse through all of their auctions by clicking the corresponding tab.



Wed Sep 08 16:18:21 EEST 2004

Auction Type
English
Product
http://www.museum.es/guernica.jpg
Participants
0
Current Max Bid
0
Auction Starting Time
Wed Sep 01 13:00:00 EEST 2004
Auction Ending Time
Sat Oct 09 13:00:00 EEST 2004
Ammount of Products
1

```
GuernicaAuction: Starting new Auction...

Auction now open!
New agent joined:PabloFanCL1
Fri Sep 17 01:15:56 EEST 2004: PabloFanCL1 for 1 item(s) offered 2!
New agent is monitoring:WatcherSCL1
New agent joined:SmithCL1
Fri Sep 17 01:32:20 EEST 2004: SmithCL1 for 1 item(s) offered 4!
Fri Sep 17 01:32:20 EEST 2004: PabloFanCL1 for 1 item(s) offered 6!
```

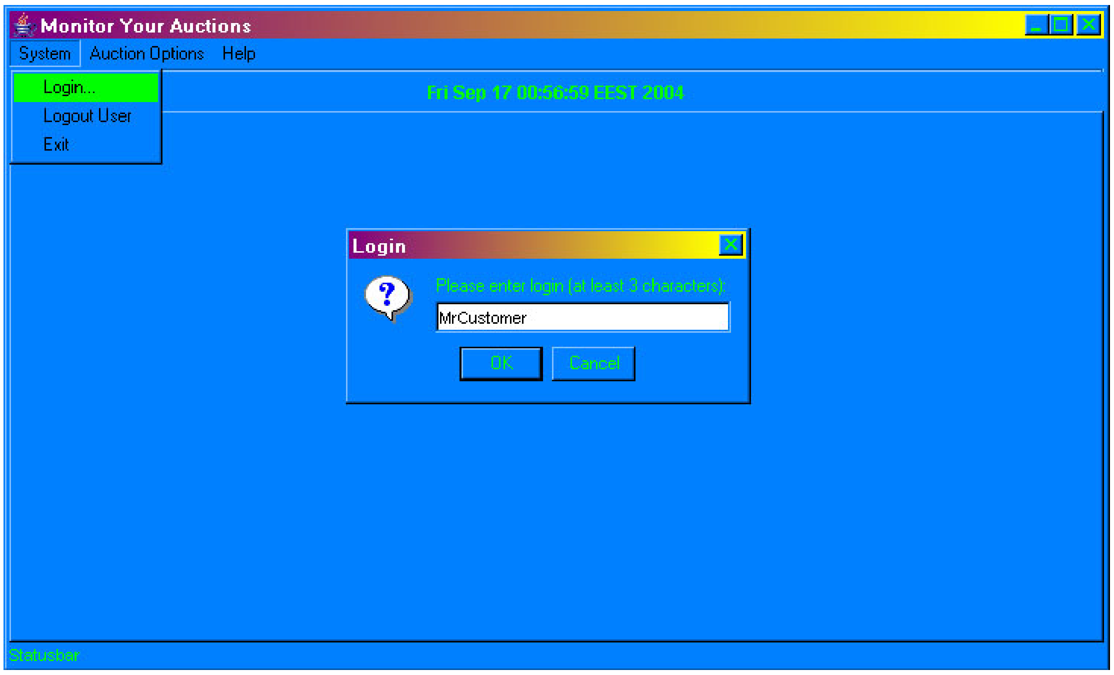
Figure 24 The Auctioneer Application graphical user interface.

The aforementioned steps are repeated for every product detected in the RDF ontology under the class `ItemForSale`. User may choose to initiate auctions for all of them at once or leave auction creation for some of the products at sometime later in the future.

Additionally, users are free to terminate an auction, after it has ended, by selecting the Remove Active Tab choice on the Available Options menu. This will cause the deletion of both the place and the A-agent at the remote agency and, also, the invalidation of the peer group's advertisement that was stored in the local cache.

5.3 Customer Application

Customers constitute the central point of interest in auction systems and in commercial systems in general, therefore SeMPHoNIA's goal is to make the tedious process of searching, participating, monitoring and analyzing auctions as simple and complexity-free as possible. The platform offers several combinations of choices to users; they can be active bidders in an auction or passive observers extracting statistical data; they can decide the bidding strategy for each auction; they are allowed to initiate one or more sessions in parallel and follow their progress; in one session they can participate in one or more auctions, aiming at winning only one of them, when the result of one auction may affect the action taken for the other. The system supports users in following the optimum path, when interrelated auctions are involved.



Fri Sep 17 00:56:59 EEST 2004

Please enter login (at least 3 characters):

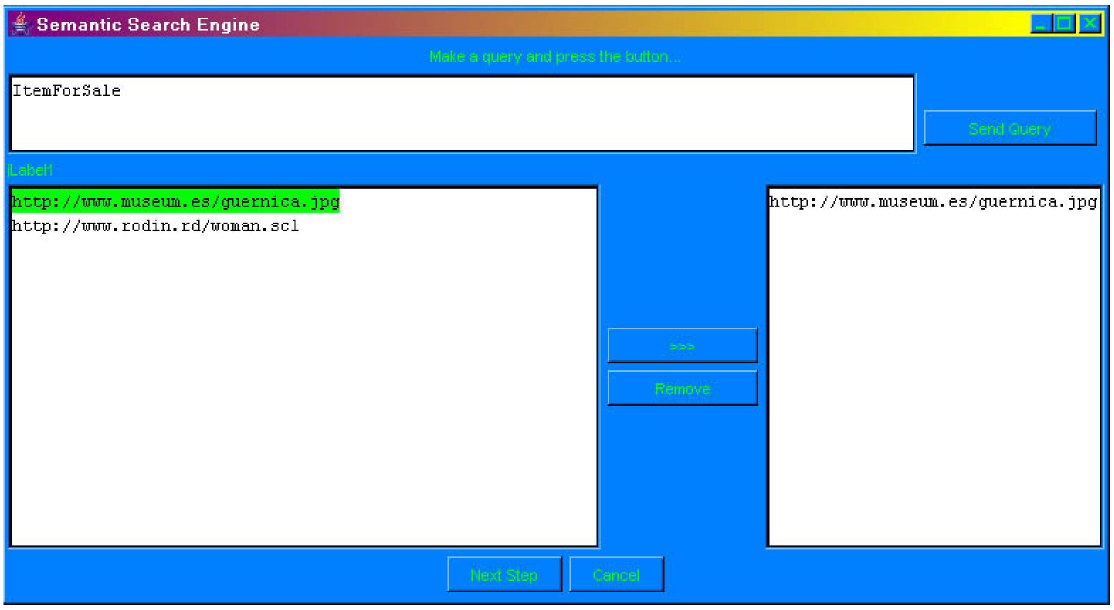
MrCustomer

Figure 25 Customer Application login process.

As with operator and auctioneer applications, the first time the customer application is launched, the JXTA Configurator tool is displayed. The user is requested to specify the communication IP address and port for configuring the TCP settings for the peer in the network. These settings will guide the application in all future executions.

Any person interacting with the SeMPHoNIA platform must provide a username through the process of login, to be appropriately registered (figure 25). This is a unique identifier that is kept locally and is used by the system to maintain profile information for every member and regain sessions that the user might had left running in previous executions. Therefore, after the user logs in the SeMPHoNIA environment, the system seeks for the specific user's sessions in agencies across the network. Then, it informs the user about the result and triggers the C-agents' "reconnection" function, in order to inform the application about the auctions' current status. C-agents preserve information about the state of their CL-agents at all times. Respectively, the user may choose to either logoff or exit, leaving all of her/his sessions running at the remote agencies.

5.3.1 SeMPHoNIA's Semantic Search



ItemForSale

http://www.museum.es/guernica.jpg
http://www.rodin.rd/woman.scl

http://www.museum.es/guernica.jpg

Figure 26 The Semantic Search Engine graphical user interface.

The first and most important step for a customer, when creating a new session, either bidding or monitoring, is to search among available products, conclude about their specifications and discover auctions offering the ones that satisfy her/his preferences. SeMPHoNIA offers a Semantic Search Engine and a graphical user interface to assist clients in locating auctions, by forwarding RQL queries to the ontologies that describe the products offered.

On the peer-to-peer network, numerous Ontology Advertisements are published and exchanged, characterized by their domain. Once the user decides on the domain that s/he is interested in, the Semantic Search Engine window is displayed (figure 26). At the same time, an Ontology Discovery Query message is sent to the network seeking for advertisements that refer to that specific domain. All the responses received are cached locally, containing information about the RDFSuiteServers that the customer can contact, in order to access and query their databases. Due to the nature of peer-to-peer networks, responses are received asynchronously, which means that the number of cached advertisements will increase as time passes.

The customer can compose RQL queries and write them in the text area located at the top of the Semantic Search window. These queries may vary from simple resource queries (for example `ItemForSale`, which returns all product resources of the items auctioned) to more complex schema, namespace, resource description, set-based and nested queries (for example

```
SELECT     Z, W
FROM       {X}sells{Z},
           {X}controlled_by.accepts{W}
```

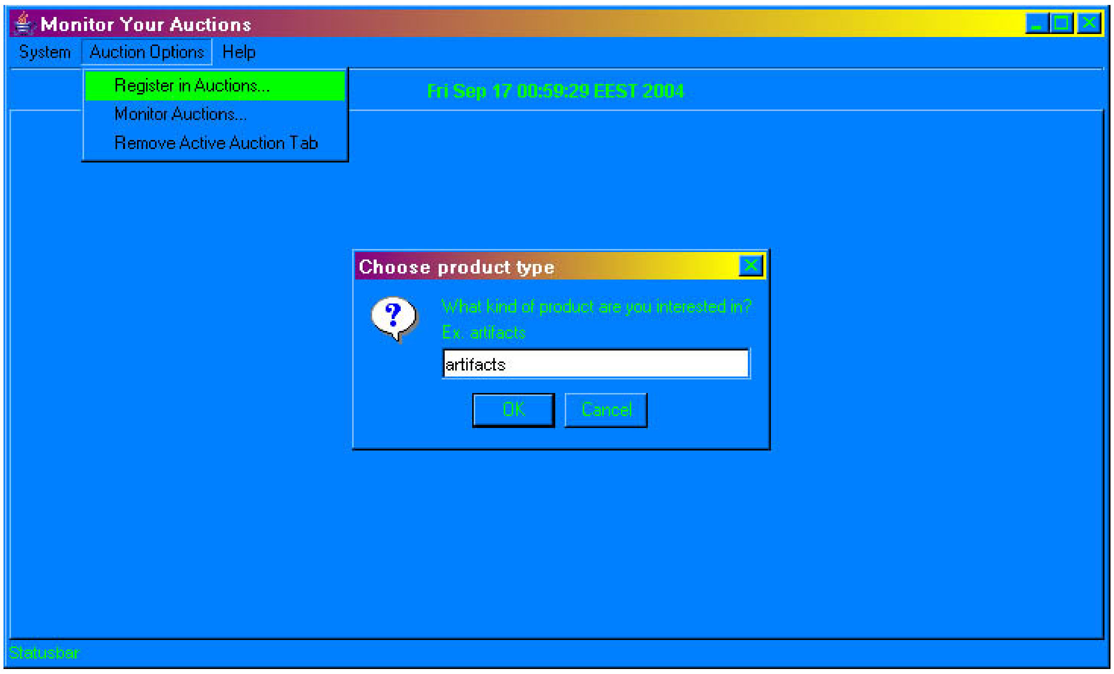
, which returns pairs of product resources and the kind of payment their supplier is willing to accept).

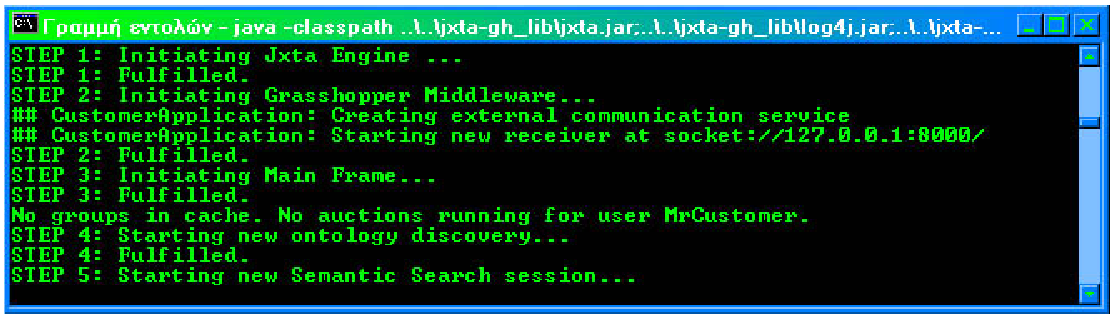
Pressing the "Sent Query" button, the system forwards the user's RQL expression to all the RDFSuiteServers, whose address is located in the local cache. They, in turn, construct the answer from the data of their own ontology and return it to the customer's application in XML format. The system presents all the answers sequentially in the main text area titled "Results".

This procedure of composing and forwarding RQL queries can be repeated as many times as needed for the user to find desirable products. The purpose is to locate the resources of those products, which will then be used as keys for discovering auctions on the network. Therefore, whenever the user finds a product of interest, s/he should place it in the right text area titled “Products”. The number of resources entered in that box indicates the number of auctions, in which s/he will participate in parallel. If the user is only interested in one product, then only one resource should be at the “Products” box. On the other hand, if the user has located multiple products, but would like to acquire only one of them, then selects the resources and places them in the box. The system will follow the progress of all of them, while the intelligence of its agent design will ensure that at most one of the products will be obtained, scoring the best utility factor possible.

5.3.2 Bidding in Auctions

The basic functionality that SeMPHoNIA’s Customer Application offers to users is to automate the process of biding in one or more auctions. To initiate a bidding session the user selects the “Register in Auction…” choice of the “Auction Options” menu. In each bidding session, the user may participate in one or more auctions and the system will attempt to win one of them, at most. Moreover, the user is free to start as many sessions as s/he desires. The first step in creating a new auction session is to specify the type of products that s/he is interested in (figure 27). This will trigger the system’s discovery mechanism to start searching the network for Ontology Advertisements dedicated to the domain that the user has specified. The next step will be the Semantic searching described previously.



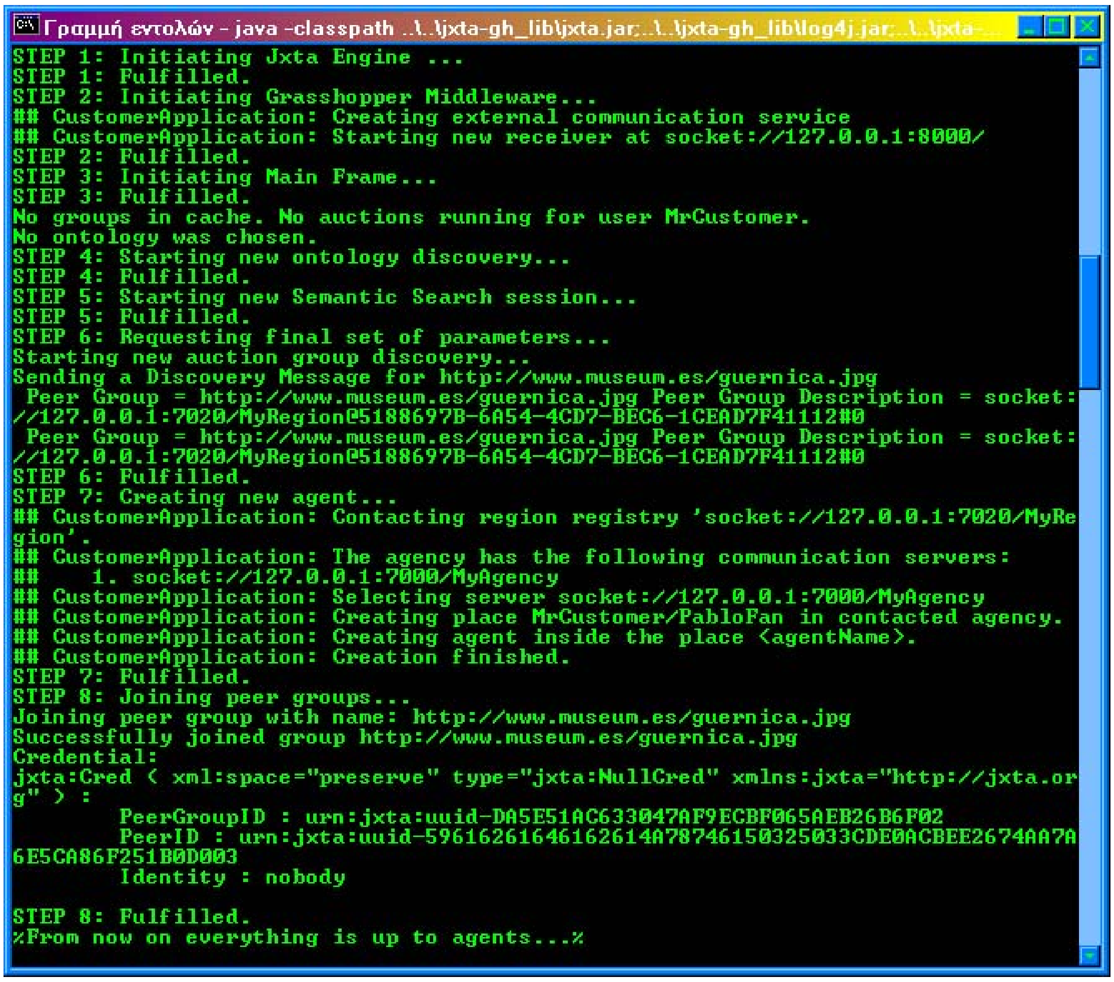

Γραμμή εντολών - java -classpath ..\..\jxta-gh_lib\jxta.jar;..\..\jxta-gh_lib\log4j.jar;..\..\jxta-...

```
STEP 1: Initiating Jxta Engine ...
STEP 1: Fulfilled.
STEP 2: Initiating Grasshopper Middleware...
## CustomerApplication: Creating external communication service
## CustomerApplication: Starting new receiver at socket://127.0.0.1:8000/
STEP 2: Fulfilled.
STEP 3: Initiating Main Frame...
STEP 3: Fulfilled.
No groups in cache. No auctions running for user MrCustomer.
STEP 4: Starting new ontology discovery...
STEP 4: Fulfilled.
STEP 5: Starting new Semantic Search session...
```

Figure 27 Initiation of a new auction bidding session.

After finishing the second step of selecting the desirable products, the customer presses the “Next Step” button and a form is displayed requesting information concerning her/his custom preferences, such as the maximum price s/he is willing to pay, the number of desirable products, a code name for the session, etc. Once the form is completed and the customer is ready to continue, her/his involvement terminates and the application activates the following set of actions:

- Sends a Peer Group Discovery message on the network tagged with the resources of the products that the customer has selected. Peers will respond with advertisements that refer to the auctions running for those products. Whenever a response arrives, the application prints a message to inform the user that the auction she/he is looking for exists and is functional (Step 6 on figure 28).

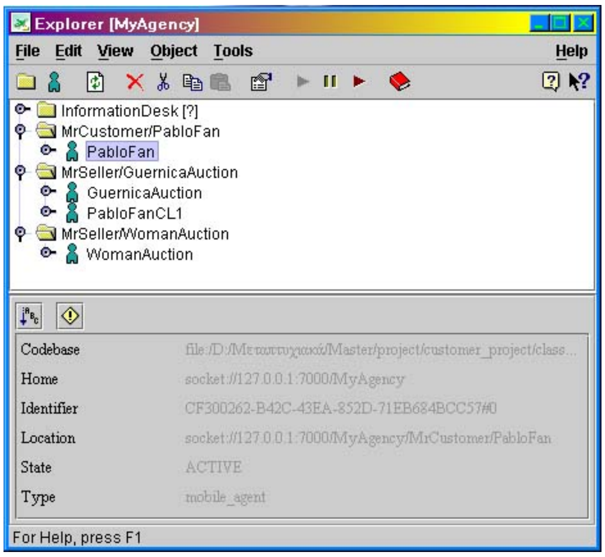


```
Γραμμή εντολών - java -classpath ..\..\jxta-gh_lib\jxta.jar;..\..\jxta-gh_lib\log4j.jar;..\..\jxta-...
STEP 1: Initiating Jxta Engine ...
STEP 1: Fulfilled.
STEP 2: Initiating Grasshopper Middleware...
## CustomerApplication: Creating external communication service
## CustomerApplication: Starting new receiver at socket://127.0.0.1:8000/
STEP 2: Fulfilled.
STEP 3: Initiating Main Frame...
STEP 3: Fulfilled.
No groups in cache. No auctions running for user MrCustomer.
No ontology was chosen.
STEP 4: Starting new ontology discovery...
STEP 4: Fulfilled.
STEP 5: Starting new Semantic Search session...
STEP 5: Fulfilled.
STEP 6: Requesting final set of parameters...
Starting new auction group discovery...
Sending a Discovery Message for http://www.museum.es/guernica.jpg
 Peer Group = http://www.museum.es/guernica.jpg Peer Group Description = socket:
//127.0.0.1:7020/MyRegion@5188697B-6A54-4CD7-BEC6-1CEAD7F41112#0
 Peer Group = http://www.museum.es/guernica.jpg Peer Group Description = socket:
//127.0.0.1:7020/MyRegion@5188697B-6A54-4CD7-BEC6-1CEAD7F41112#0
STEP 6: Fulfilled.
STEP 7: Creating new agent...
## CustomerApplication: Contacting region registry 'socket://127.0.0.1:7020/MyRe
gion'.
## CustomerApplication: The agency has the following communication servers:
##     1. socket://127.0.0.1:7000/MyAgency
## CustomerApplication: Selecting server socket://127.0.0.1:7000/MyAgency
## CustomerApplication: Creating place MrCustomer/PabloFan in contacted agency.
## CustomerApplication: Creating agent inside the place <agentName>.
## CustomerApplication: Creation finished.
STEP 7: Fulfilled.
STEP 8: Joining peer groups...
Joining peer group with name: http://www.museum.es/guernica.jpg
Successfully joined group http://www.museum.es/guernica.jpg
Credential:
jxta:Cred < xml:space="preserve" type="jxta:NullCred" xmlns:jxta="http://jxta.or
g" > :
        PeerGroupID : urn:jxta:uuid-DA5E51AC633047AF9ECBF065AEB26B6F02
        PeerID : urn:jxta:uuid-59616261646162614A78746150325033CDE0ACBEE2674AA7A
6E5CA86F251B0D003
        Identity : nobody

STEP 8: Fulfilled.
%From now on everything is up to agents...%
```

Figure 28 Output produced while discovering and joining published peer groups.

➢ Before a user can register in an auction, s/he must first join the peer group that corresponds to that auction to obtain a unique identity in it. Therefore, the application applies for membership on behave of the user to the peer groups, and only after the appropriate credentials are obtained, s/he is eligible to join the groups. Step 8 in figure 28 presents the previously described procedure that it is repeated for each of the desirable auctions.

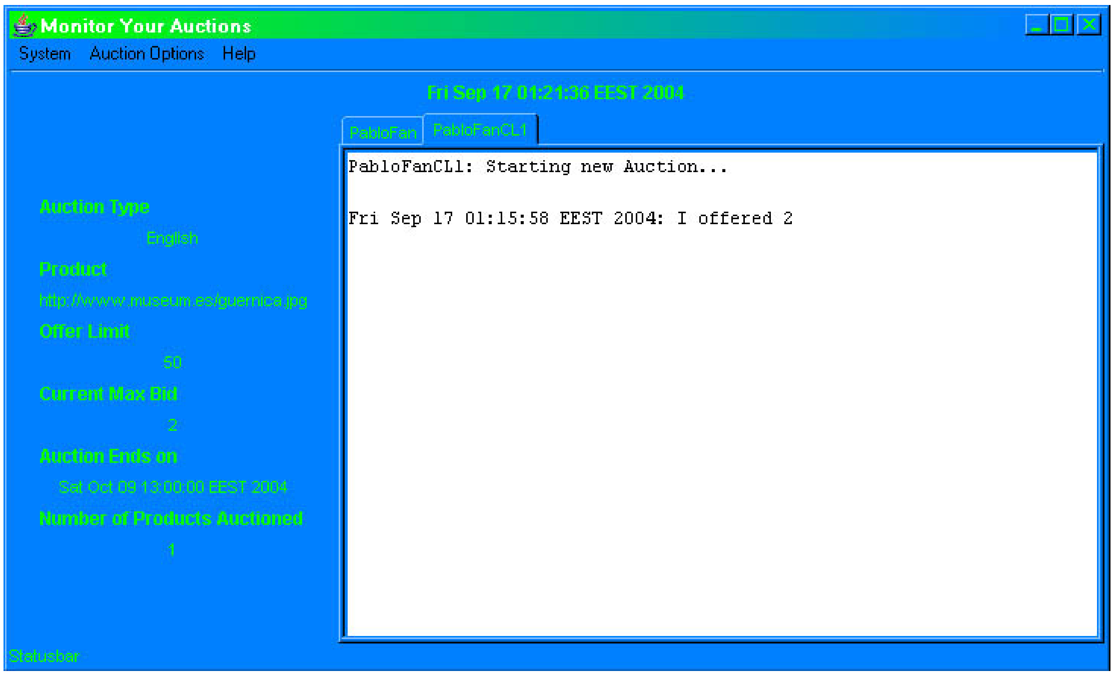
➢ In addition, the system contacts the remote agency of one of the auctions and creates a place in it, named after the username and the code name of the session. This place will host the C-agent that plans and organizes the session (step 7 in figure 28). Then, it constructs the new C-agent, providing all the necessary initial information, such as the auctions that the user is interested in and where they are located, the maximum price it should offer etc. When the C-agent is invoked, it creates the appropriate CL-agents and routes them to the remote places, where the auctions are conducting. Figure 29 presents how the agency will look after the creation of the place and the existence of the C- and CL-agents. CL-agents may need to travel to remote agencies, as well.



```
InformationDesk [?]
MrCustomer/PabloFan
    PabloFan
MrSeller/GuernicaAuction
    GuernicaAuction
    PabloFanCL1
MrSeller/WomanAuction
    WomanAuction
```

Codebase	file:/D:/Μεταπτυχιακά/Master/project/customer_project/class...
Home	socket://127.0.0.1:7000/MyAgency
Identifier	CF300262-B42C-43EA-852D-71EB684BCC57#0
Location	socket://127.0.0.1:7000/MyAgency/MrCustomer/PabloFan
State	ACTIVE
Type	mobile_agent

Figure 29 Agency containing a C-agent (PabloFan) in its place, a CL-agent in the auction place (PabloFanCL1) and two A-agents in their corresponding auction places.



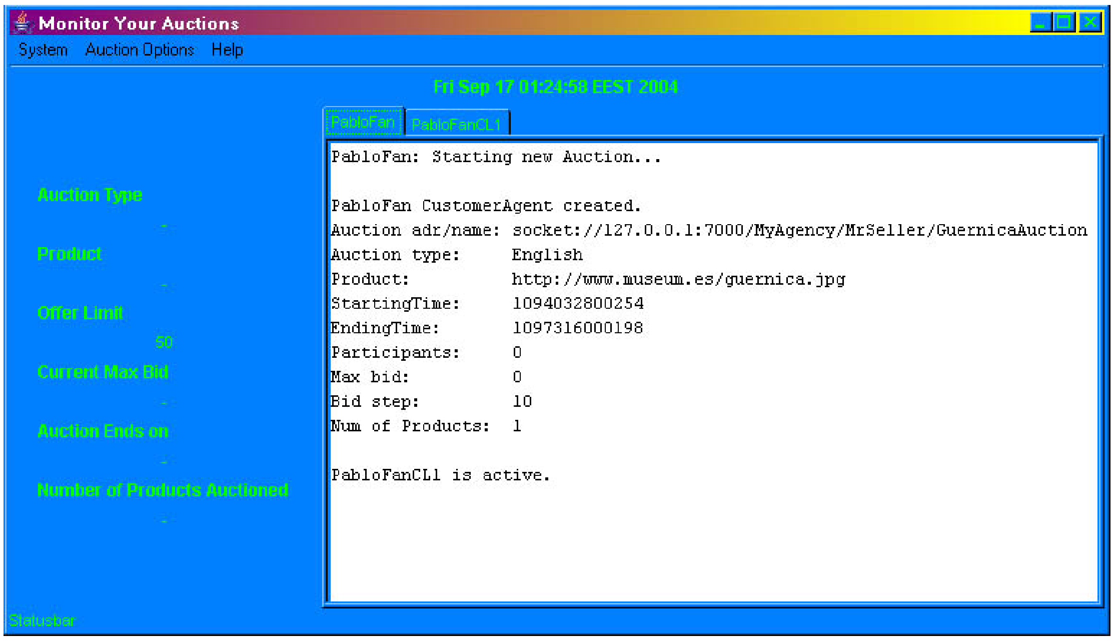
Fri Sep 17 01:21:36 EEST 2004

Auction Type
English
Product
http://www.museum.es/guernica.jpg
Offer Limit
50
Current Max Bid
2
Auction Ends on
Sat Oct 09 13:00:00 EEST 2004
Number of Products Auctioned
1

```
PabloFanCL1: Starting new Auction...

Fri Sep 17 01:15:58 EEST 2004: I offered 2
```

Figure 30 Information concerning a single auction, obtained by a CL-agent.

- Finally, the graphical user interface is updated, presenting information about the auctions the user participates in (figure 30) and the session as a whole (figure 31). The CL-agents inform the user about the progress of bidding in their auction, while the C-agent's tab displays general messages, such as which CL-agent is currently active, which is holding the maximum offer in its auction, etc. Data are updated in real time, whenever a change occurs and the user is free to browse through all of her/his auctions by clicking the corresponding tab.



```
PabloFan: Starting new Auction...

PabloFan CustomerAgent created.
Auction adr/name: socket://127.0.0.1:7000/MyAgency/MrSeller/GuernicaAuction
Auction type:     English
Product:          http://www.museum.es/guernica.jpg
StartingTime:     1094032800254
EndingTime:       1097316000198
Participants:     0
Max bid:          0
Bid step:         10
Num of Products:  1

PabloFanCL1 is active.
```

Figure 31 Information concerning the auction session, obtained by the C-agent.

5.3.3 Monitoring Auctions

A product's auction statistics is crucial information for assisting bidders in planning strategies and achieving high utility factors. SeMPHoNia provides the facility of monitoring auctions and analyzing statistics concerning the history of bidding. The procedure of starting a monitoring session is similar to the one described for initiating a bidding one (figure 32). The user selects the "Monitor Auctions…" choice of the "Auction Options" menu and invokes the Semantic Search Engine to discover available goods. S/he is free to select multiple auctions in one session and

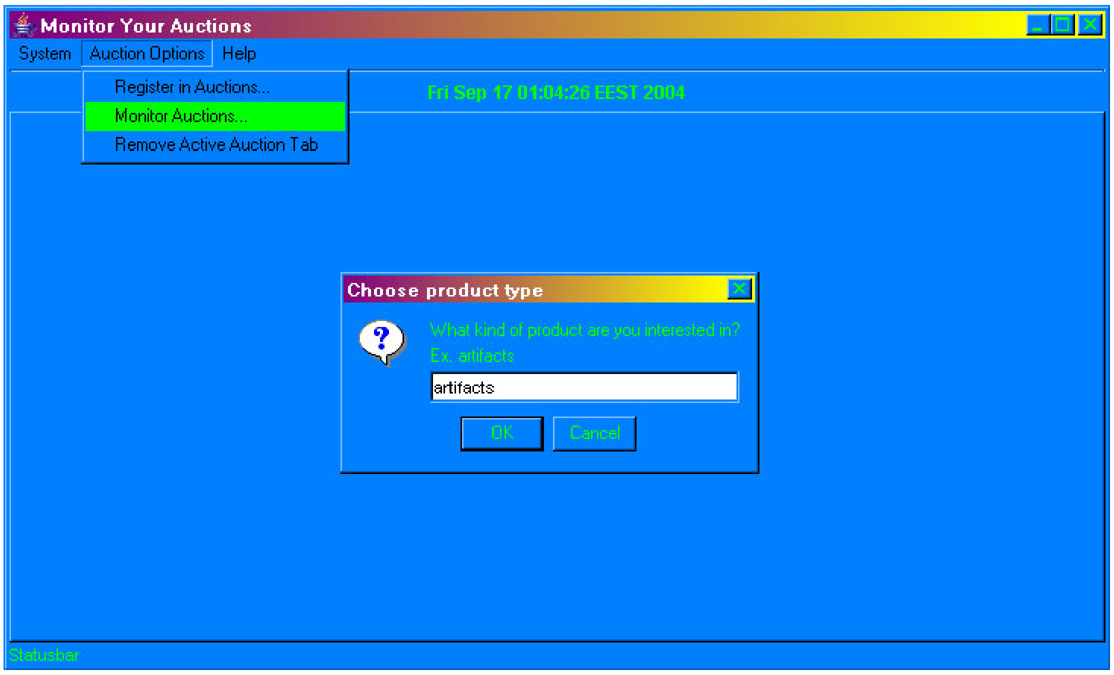


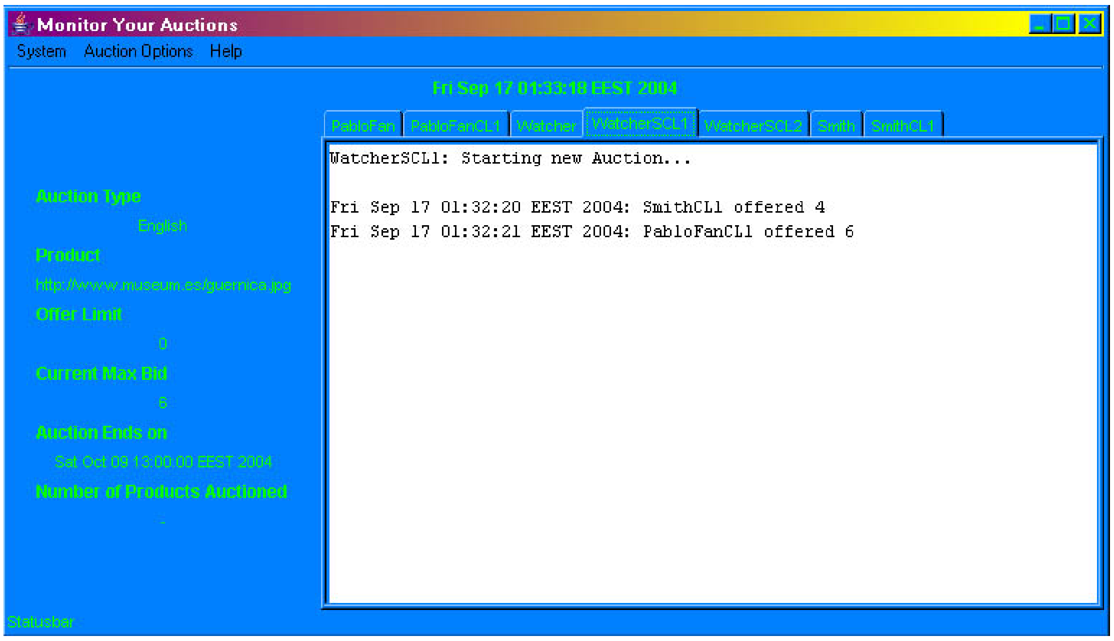
Figure 32 Initiating a new monitoring session.

the statistics will concern all of them. After providing a code name for the session, the system triggers a set of automated actions that involve:

- Sending a Peer Group Discovery message on the network tagged with the resources of the products that the customer has selected. Peers will respond with advertisements that refer to the auctions running for those products. Whenever a response arrives, the application prints a message to inform the user that the auction she/he is looking for exists and is functional.

- Applying for membership on behave of the user to the peer groups, in order to obtain the appropriate credentials and joining the group.

- Contacting the remote agency of one of the auctions and creating a place in it, named after the username and the code name of the session. This place will host the S-agent that organizes the session and analyzes the data. The system, also, constructs the new S-agent, providing all necessary initial information, such as the auctions that the user is interested in and where they are located etc. When the S-agent is



Fri Sep 17 01:33:18 EEST 2004

Auction Type
English
Product
http://www.museum.es/guernica.jpg
Offer Limit
0
Current Max Bid
6
Auction Ends on
Sat Oct 09 13:00:00 EEST 2004
Number of Products Auctioned
-

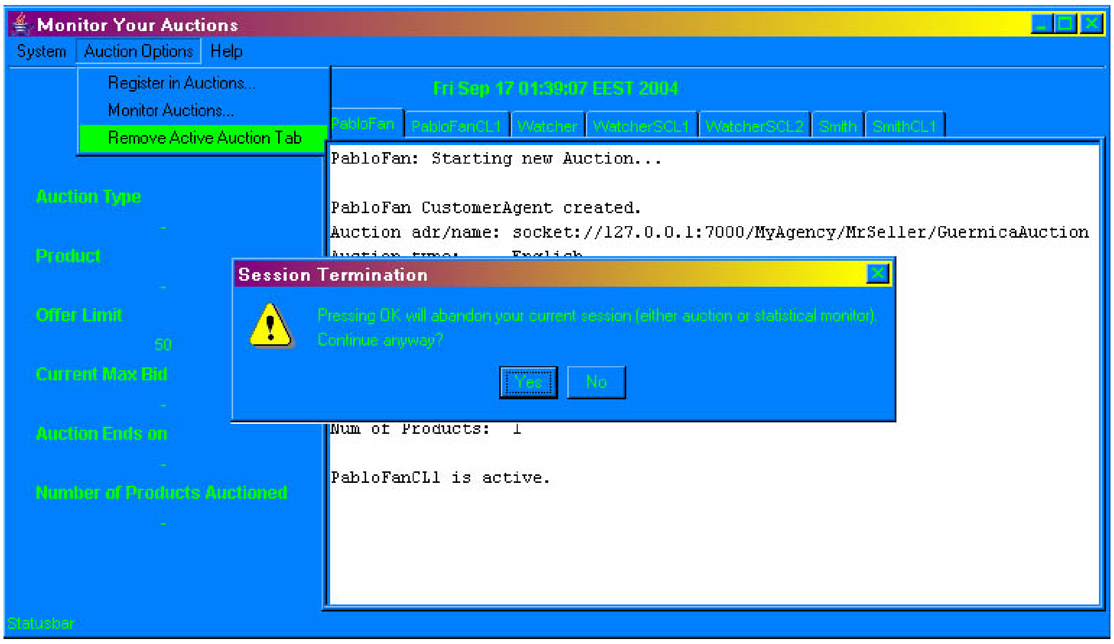
```
WatcherSCL1: Starting new Auction...

Fri Sep 17 01:32:20 EEST 2004: SmithCL1 offered 4
Fri Sep 17 01:32:21 EEST 2004: PabloFanCL1 offered 6
```

Figure 33 While monitoring a session, information is presented on the graphical user interface.

invoked, it creates the appropriate SCL-agents and routes them to the remote places, where the auctions are conducting.

➢ Finally, the graphical user interface is updated, presenting information about the auctions that the user monitors and the session as a whole (figure 33). The SCL-agents inform the user about the progress of bidding in their auction, while the S-agent's tab displays statistics, such as the average winning bids, price convergence behavior, equilibrium price, progress of bidding according to time etc. Data are updated in real time, whenever a change occurs and the user is free to browse through all of her/his auctions by clicking the corresponding tab.

5.3.4 Abandoning Auctions



```
PabloFan: Starting new Auction...

PabloFan CustomerAgent created.
Auction adr/name: socket://127.0.0.1:7000/MyAgency/MrSeller/GuernicaAuction
```

Pressing OK will abandon your current session (either auction or statistical monitor). Continue anyway?

```
Num of Products:  1

PabloFanCL1 is active.
```

Figure 34 When erasing a C-agent and its auction session, a warning message is displayed.

The user is free to terminate a session, either bidding or monitoring, and abandon the auctions involved. For that purpose, the "Remove Active Auctions tab" choice of the "Auction Options" menu is granted. Terminating a session will cause the corresponding place to be erased and the C- or S-agent along with their clones to be deleted. In addition, the graphical user interface is updated removing the current tab from the main frame. There are certain restrictions that one should consider, though:

- To terminate a session, the user must remove the tab of the C- or S-agent. In such a case, a warning message appears, to prevent any unintentional actions (figure 34).

- Removing the tab of a CL- or SCL-agent will only erase the corresponding tab. The agents continue to exist and operate. This means that a user is only allowed to terminate a session as a whole and not one of its auctions. Users can use this function to lighten their interface from multiple auction tabs. When the user reconnects with the system, all the tabs will re-appear.

- Abandoning an auction will cause the user's registration to be invalidated. However, if at that particular moment s/he was the maximum bidder in the auction, her/his offer continues to exist, until someone else outbids it. If the auction ends with no other offer, the user is considered the winner, despite the fact that the session has been terminated.

This chapter summarized the way users can interact with SeMPHoNIA platform and explained the basic steps, along with the procedures triggered, for executing auction services. Next, we are going to explain some implementation principles followed in the creation of the system.

Chapter 6

IMPLEMENTATION

The SeMPHoNIA platform is entirely programmed in Java, in order to minimize dependencies on the underlying operating system and to be consistent with existing software. JDK 1.4 or higher is required for most of its components. The customer, auctioneer and operator applications, as well as, the JXTA and Grasshopper components have been implemented on Windows 2000 and NT platforms, while the RDFSuite tools (VRP, RSSDB, PostgreSQL, RQL) and the RDFSuiteServer run on Solaris 8 (SunOS 5.8), RedHat Linux 7.3, Mandrake Linux 8.1, and Debian Linux 3.0 platforms. All forms and modules used for user interface have been implemented mainly using Java's Swing toolkit and, in less extent, Java's Abstract Windowing Toolkit (AWT) elements.

6.1 Agent Communication

A set of standard messages that are exchanged between SeMPHoNIA agents is implemented to describe the minimum required communication needs of the multi-agent layer of the platform. The message invocation is triggered by events occurring in the auction world (auction opening or termination etc.) or by inferences of their own or their master's reasoning mechanism. The description of those messages and their parameters are provided in the context of interfaces that the agents implement in order to ensure cross-platform interoperability, necessary for building a unifying framework. The messages formulate a negotiation core for defining auction activities and are always independent of the particular type of the auction. They are classified according to the negotiating parties involved as follows:

6.1.1 Auctioneer Generated Messages

<u>A-agent to CL-agents</u>

Messages that the A-agent broadcasts to all participants of an auction to initiate it and manage its execution.

StartOfAuctionNotification(OpeningPrice):

Informs the participants that the auction session has opened with the given opening price.

EndOfAuctionNotification(Winner, WinningBid, ProductQuantity):

Informs about the termination of an auction, along with the winning agent and the winning offer.

NewBidNotification(MaxBidAgentName, NewBid, ProductQuantity):

Whenever a new offer is accepted and verified as valid, the auctioneer informs the rest of the participants. In auction, like Vickrey, that the offers are private, the auctioneer passes the *null* value as the corresponding parameter.

6.1.2 Customer Generated Messages

<u>C-agent to CL-agents</u>

Messages sent by the C-agent to each clone individually for managing a session and instructing the course of action that they should follow.

Activate():

Activates a standing-by CL-agent to place new offers.

Deactivate():

Deactivates a previously active CL-agent, setting its state to stand-by mode.

Abandon():

Instructs a CL-agent to permanently abandon an auction.

C- or S-agent to A-agent

Message sent from the C- or S-agent to the A-agent before any CL-agent has been created.

GetCurrentInfo():

Requests the list of parameters needed for getting informed about the auction's running conditions (for example current maximum offer, participants etc.).

CL-agent to A-agent

Messages sent by CL-agents to the A-agent for interacting in an auction. The message format is always independent of the type of the particular auction. Clones are characterized by their name and a unique ID in an auction session.

RegisterInAuction(AgentName, CLidStr):

Message sent by a clone to register as a bidder in an auction session.

MonitorAuction(AgentName, SCLidStr):

Message sent by a clone to registering as an observer in an auction session.

MakeNewBid(AgentName, newmaxbid, quantity):

Placement of a new bid.

AbandonAuction(AgentName, CLidStr):

Withdrawal from an auction session.

CL-agent to C-agent or S-agent

Messages sent by CL-agents to their master agents (C- or S-agent) to inform them about the progress of the auction that they participate in. They always enclose their current state in the auction (for example, holding maximum offer, outbidded, reached offer limit etc.).

ArrivalConfirmation(CLname, CLstate):

Informs the master agent that the migration to the remote place, where the auction is conducted, was successful along with its current state.

InformAboutStartOfAuction(CLname, StartingBid, state):

Informs the master agent about the initiation of the session that the clone participates in.

InformAboutNewBid(CLname, MaxBidAgentName, NewBid, state):

A new offer has been accepted in the corresponding auction.

InformAboutEndOfAuction(CLname, Winner, WinningBid, state):

Informs about the termination of the corresponding auction session.

6.2 Implementation of multi-auction bidding logic

An important component of any intelligent software agent is its inference engine. In SeMPHoNIA, the most challenging task that agents are called to accomplish is that of bidding across multiple auctions with varying start and end times and varying protocols, attempting to ensure that at most one of the desired items will be purchased, thus procuring the best deal for the customer. Although this task is decomposed and distributed to multiple agents, the one that maintains knowledge concerning all the auctions and instructs the others according to a well-defined dynamic plan is the C-agent. Therefore, the implementation of a sophisticated reasoning behavior for the C-agent is of vital importance for the success of a profitable auction participation.

Our current approach is indicative and can be used as a reference point for implementing more advanced strategies. The C-agent is designed to manage multiple CL-agents and decide which of them should bid and which should wait, according to the progress of their auction. Four are the basic constraints that the C-agent's logic mechanism is designed to comply with:

- Singularity Constraint. Among all clone agents, only one is allowed to be active at any particular moment and, thus, permitted to place bids in its auction. The rest of them are set at a stand-by mode, waiting for the C-agent to activate them.

- Exclusiveness Constraint. While the active clone holds the maximum bid in its auction, no other clone can be set active. Only after the first has been outbidded, another one is eligible to become active.

- Optimum Path Constraint. The intention is to always select as active clone the one that stays on the optimum path among all bidding sessions, meaning that it is the clone which maximizes the C-agent's utility function. Whenever this does not hold true, due to changes occurring in its auction or in the world (i.e. new auction initiation), the C-agent activates another one.

- Eligibility Constraint. The purpose of this constraint is to prompt active agents to adopt an "agreement-oriented" behavior and prevent them from sitting out during a session when no purchase has been made. The active clone is always considered to be the highest bidder in its auction, in order to ensure the acquisition of the item. As a result, the clone, while active, persistently pursues to hold the maximum bid of its auction, until reaching the offer limit.

These constraints are the core principles of the agent's logic reasoning mechanism that ensure single auction winning and potentially best deal agreement for the customer. They guarantee that no surplus items would be bought or purchases exceeding the user's offer limit would be made.

Figure 35 summarizes the general framework of the algorithm that implements the aforementioned C-agent logic. This algorithm is triggered either by events occurring in the active auction or by a random initiation of a new auction session. No other events affect the progress of the session and, therefore, necessitate no alteration at the state of the CL-agents.

```
IF a change occurs in the active auction
    CHECK the state of the active clone
        CASE the active clone has won the auction
                THEN abandon all other auctions
        CASE the active clone does not hold the maximum bid
            OR the active clone has reached its offer limit
            OR the active clone has lost the auction
                THEN among all clones activate the one whose
                    auction has the minimum value of maximum
                    bid
        CASE the active clone holds the maximum bid
                THEN do nothing
ELSE IF a new auction has opened
    CHECK the state of the active clone
        CASE the active clone has won the auction
            OR the active clone holds the maximum bid
                THEN do nothing
        CASE the active clone does not hold the maximum bid
            OR the active clone has reached its offer limit
            OR the active clone has lost the auction
                THEN among all clones activate the one whose
                    auction has the minimum value of maximum
                    bid, taking into account the new auction's
                    opening price
```

Figure 35 The C-agent's general inference logic.

A separate component of the agent's reasoning mechanism handles the start-up decisions that determine the course of action during the initiation of a session. These decisions aim at specifying the most appropriate auction for bidding first and are evolving in a two-step selection that is consistent with the previous constraints. Figure 36 depicts the C-agent's decision approach at start-up.

```
Step 1
        From those auctions running, select the one that has the
            minimum Maximum Offer among them.
Step 2
        If none auction is currently running, select the one that
            opens first.
```

Figure 36 The C-agent's logic during start-up.

It should be noted that many other tactics can be easily created from this approach, by enhancing the reasoning mechanism with new constraints, that would consider factors such as remaining time, remaining auctions, level of desperateness, bargaining level etc. Since no optimal solution can be traced due to the variety of auction start and end times, the performance of each tactic is a variable that can change depending on the conditions of the environment encountered and can be greatly influenced by the personal style of its user.

6.3 Auction Implementation

The implementation of the auction protocols follows formalisms of traditional auction regulations and principles. Concerning the English and Peer-to-Peer auction, the auctioneer accepts offers greater than the running Maximum Offer and broadcasts the new bid to all participants. After the termination of the auction, only one of the participants wins at a price equal to the value of the proposed maximum bid. The Vickrey auction on the other hand is different, because all participants place their bids, but none of them is publicly announced. Only the auctioneer knows the candidate offers and informs the participants whenever a new offer has been accepted. After the end of the session, the participant that has placed the highest offer wins and pays the price of the second highest bid. No other information is broadly announced.

There are certain policies, though, that the auctioneer follows, which are independent of the type of the auction. One such policy regards the disjunction between monitoring and bidding agents. The auctioneer maintains two distinct groups to separate these two roles that agents can play during an auction session and associates each of them with the appropriate groups upon registration. Even though in most cases the exact same messages are broadcasted to both groups, this tactic improves the performance of message propagation to bidders that are very much concerned about rapid and accurate event notification.

Another issue that applies to all SeMPHoNIA auctions, regardless of their type, is the regulations concerning bid acceptance and withdrawal. All bidders are allowed to withdraw their offers taking into account an important detail. A withdrawal of the winning bid during the progress of an auction will provisionally not be accepted, until another bidder matches the price of that offer and releases it. This policy has been implemented in other auction applications as well ([24], [23]).

Moreover, upon registration, the auctioneer is requested to provide information concerning the current status of a session. A list of arguments is constructed that contains data about the number of participants, the current maximum offer etc. In cases where the information is private (for example, offers in Vickrey auctions), the corresponding slots are filled with the *null* value. Thus, all registration data required by agents to begin interacting in an auction are provided by an auction-type-independent element promoting the uniformity of the platform.

After explaining, in this chapter, certain implementation aspects of SeMPHoNIA platform, we going to present results of performance evaluation carried out through numerous real-condition test cases. We compare the performance of different auctions and, also, attempt an interpretation of the findings obtained.

Chapter 7

PERFORMANCE ANALYSIS

This section focuses on the performance study of SeMPHoNIA's agent architecture design. The hypothesis that we seek to evaluate is that our proposed negotiation scheme is scalable and performs efficiently in a wide range of conditions. We measure the performance of our platform in terms of negotiation rate and utilization factor (messages processed per second). We present the results obtained by running several test cases, including simultaneous progression of multiple auctions. We performed our evaluations under realistic executions of the platform, instead of running simulations.

7.1 Experimental Setup

We carried out our performance analysis on a local Ethernet network with nodes connected at speeds of approximately 100 Mbit/sec. Since we were mostly concerned about the environment, where the auctions were conducted, we activated one operator peer on a Pentium M 1,6 GHz workstation for dealing with all the processing activity required.

To retain a common base line for our evaluations, each C-agent was given a random maximum asking price ranging from 1 to 500 units and all English and Peer-to-Peer auctions were evolving using an incremental step of 5 units. As a result, if only one auction were to be carried out, it would converge, at the worst case, after 100 bidding rounds at an equilibrium price of 500 (we refer to such auctions with the abbreviation 500/5). All participants join the auctions before its initialization. In addition, we chose to implement the most demanding scenario, where CL-agents inform their C-agents at every new bid notification, reducing their performance, but increasing their reliability. Nevertheless, it is possible to consider other similar approaches, with CL-agents informing about their progress every 2 bids or every frequent time intervals.

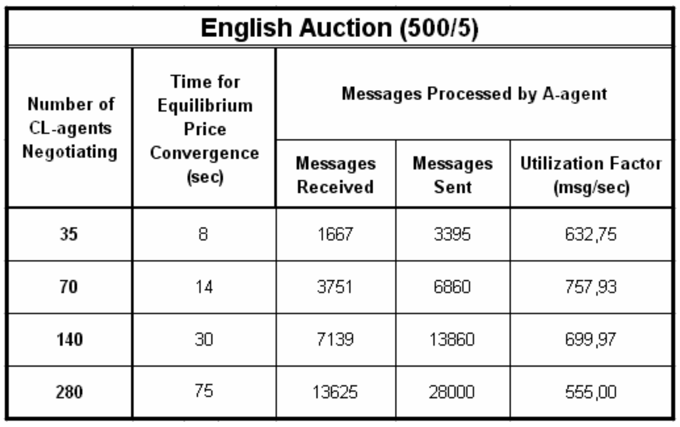
For a typical program execution, the Java Virtual Machine occupies by default a maximum setting of approximately 64Mb of the host's memory. We found that a single operator peer can handle up to 600 agents, before running out of memory at such conditions. Our experiments measured the performance of the system close to this limit as well (280 CL-agents negotiating, 280 C-agents participating, and 4 A-agents offering auction houses). If we consider the fact that the majority of auctions held on eBay, the largest consumer-to-consumer auction site, have an average of 20 to 40 bidders [107], then a single operator peer can host up to 15 parallel auctions (or, at the other side, one auction with 600 participants, which is only applicable for experimental purposes, but does not reflect real life situation). This we believe is a very satisfactory number for a distributed system, such as SeMPHoNIA, which is designed to allocate services at multiple peers across the network. Although it is also possible to extend the maximum amount of memory that Java Virtual Machine utilizes during a single execution, depending on the host machine specifications, we chose to conduct our experiments at the default limit of the 64Mb.

7.2 Performance Evaluation

The first set of test cases investigate how the behavior of the system changes as we increase the number of agents participating, during a single auction. We consider the English 500/5, the Peer-to-Peer 500/5 and the Vickrey auctions and compare the first two, since they present similar behavior. The second set of test cases considers multiple English 500/5 auctions and discuses how agents handle the situation.

7.2.1 Single Auction

Table 1 presents data taken during a single execution of an English 500/5 auction by increasing the number of participating agents. Column 2 displays how much time was required for the auction to settle to the equilibrium price. In addition, columns 3 to 5 present the computational needs of the system in notions of messages exchanged. Since all participants communicate only with the A-agent, all processing is concentrated there. The A-agent receives messages from several bidders, but only the most recent is considered valid. At each round, the auctioneer notifies all



English Auction (500/5)				
Number of CL-agents Negotiating	Time for Equilibrium Price Convergence (sec)	Messages Processed by A-agent		
		Messages Received	Messages Sent	Utilization Factor (msg/sec)
35	8	1667	3395	632,75
70	14	3751	6860	757,93
140	30	7139	13860	699,97
280	75	13625	28000	555,00

Table 1 Evaluation using English auction with 35, 70, 140 and 280 negotiating agents.

participants about the current maximum offer accepted. As a result, the number of messages sent depends on the number of negotiation round, as well. We express the utilization factor in terms of messages processed by the A-agent per second.

The corresponding values for the Peer-to-Peer auction are displayed in table 2. Since no centralized figure is implemented in such auction, messages are exchanged between bidders and are distributed among all participant agents. Therefore, in this case, the utilization factor expresses the performance of the system as a whole, instead of the single point of congestion (namely the A-agent in the English auction). One would expect messages received and messages sent to be equal in number, but this is not true, because every time the auction’s current maximum offer is updated, all negotiating agents receive the new value and continue negotiating from that value.

Peer-to-Peer Auction (500/5)				
Number of CL-agents Negotiating	Time for Equilibrium Price Convergence (sec)	Messages Exchanged by All Agents		
		Messages Received	Messages Sent	Utilization Factor (msg/sec)
35	54	1236	1161	44,39
70	57	2296	2145	77,91
140	78	5834	5563	146,12
280	192	11412	10981	116,63

Table 2 Evaluation using peer-to-peer auction with 35, 70, 140 and 280 negotiating agents.

Figures 37 and 38 compare the performance readings of those two auctions and reveal their strong and weak points. From figure 37 it is obvious that the rate of convergence in the English auction is significantly faster in comparison to that of the peer-to-peer auction. In fact, when the number of agents negotiating becomes very high, the peer-to-peer auction delays even more. One possible solution for improving the performance of peer-to-peer auctions it to group agents with high offer limits together. This will cause the auction's maximum offer to increase rapidly during the first rounds and, as a result, a large number of participants will not qualify to the following rounds.

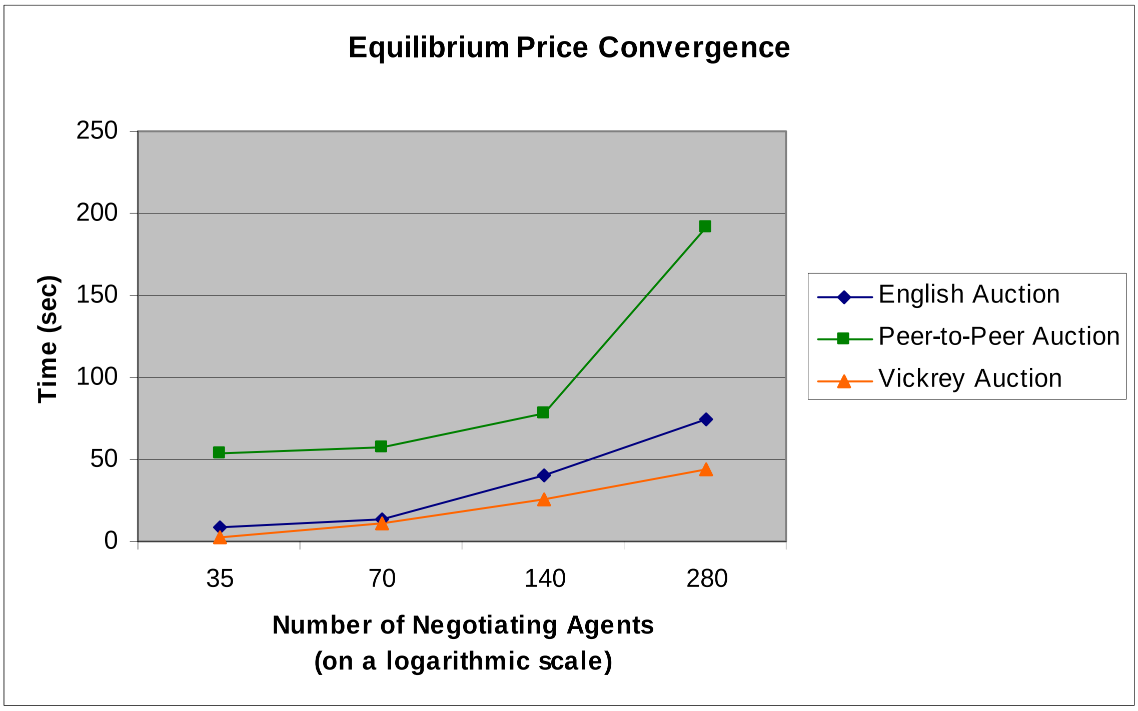


Figure 37 Comparison of performance using English, Vickrey and peer-to-peer auctions in terms of convergence rate to equilibrium price.

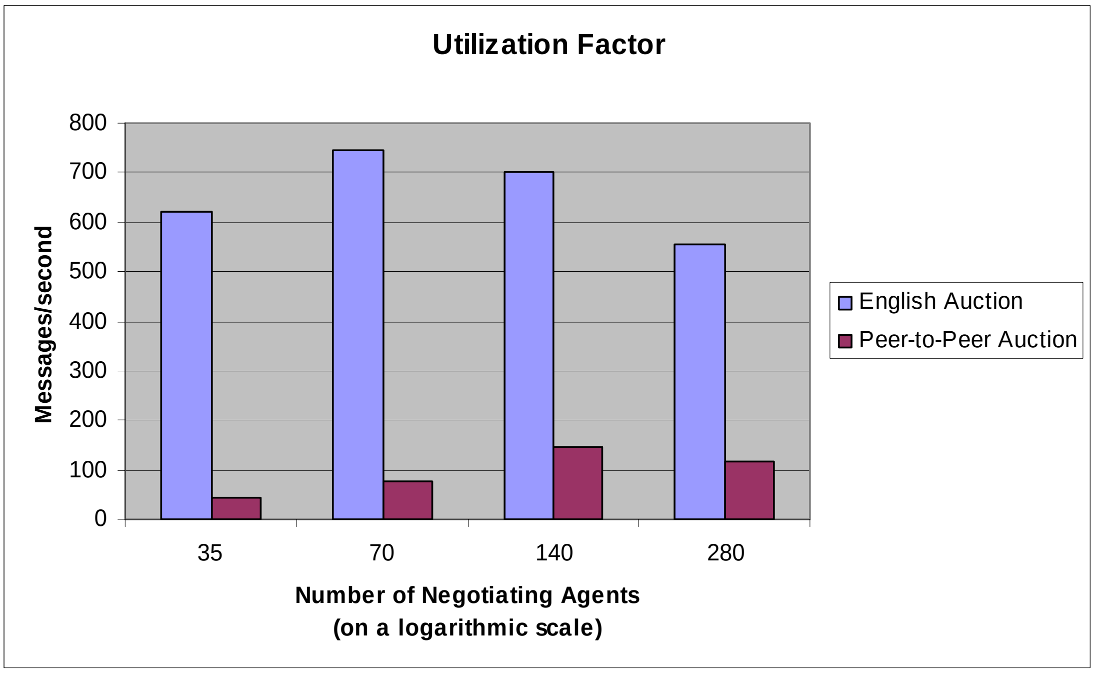


Figure 38 Comparison of performance using English and peer-to-peer auctions in terms of utilization factor.

On the other hand, the English auction suffers broadly in terms of efficiency. Figure 38 shows a considerable difference in the number of messages exchanged per second, during an English and a peer-to-peer auction. It is clear that the English auction is more costly. Besides that, it is also less efficient. The figure shows that for both auctions the utilization factor increases up to a certain upper limit and then starts diminishing. This upper limit is the point at which the maximum number of messages can be processed, maximizing the performance gain, and below that point, the computational needs are so great that no resources are available for handling all messages arriving or departing. We can see that for the English auction the utilization factor's upper limit is reached almost two times faster compared to the peer-to-peer auction (for our system the limit is at around 90 agents for the English auction and 200 agents for the peer-to-peer auction). This finding implies that the peer-to-peer auction is suitable for greater numbers of negotiating agents. (We note that our findings agree with observations made in [70] simulating negotiations with 2,500 to 160,000 agents).

Except of the English and peer-to-peer auctions, we also evaluated the system's performance running Vickrey auctions. This type of auctions evolve in a single negotiation round, therefore we were only interested in how fast the auctioneer can handle new offers. Table 3 displays the time required for the auctioneer to validated incoming offers of increasing numbers of bidders. All bids were placed almost simultaneously, forcing the A-agent to preserve a stuck of incoming bid. Comparing these times with those of the other auctions (figure 37), it is quite apparent that the Vickrey auction outperforms the others when a great number of bidders are participating, if time is the most important metric.

Vickrey Auction	
Number of CL-agents Negotiating	**Time for Validating All Offers (sec)**
35	2,053
70	11,126
140	25,857
280	44,032

Table 3 Evaluation using Vickrey auction.

7.2.2 Multiple English Auctions

Our next set of experiments involved the parallel execution of multiple English 500/5 auctions that start almost simultaneously. The C-agents participated in all of them, creating the respective number of CL-agents to bid in each. At any random moment, C-agents activated only one of the CL-agent they manage. At the start of the

	36 CL-agents Negotiating			72 CL-agents Negotiating			144 CL-agents Negotiating			288 CL-agents Negotiating		
Number of Parallel English Auctions	**Number of Sessions (C-agents)**	**Time for Equilibrium Price Convergence (sec)**	**Average Number of Auctions Changed During a Session**	**Number of Sessions (C-agents)**	**Time for Equilibrium Price Convergence (sec)**	**Average Number of Auctions Changed During a Session**	**Number of Sessions (C-agents)**	**Time for Equilibrium Price Convergence (sec)**	**Average Number of Auctions Changed During a Session**	**Number of Sessions (C-agents)**	**Time for Equilibrium Price Convergence (sec)**	**Average Number of Auctions Changed During a Session**
1*	*35*	*8*	*0,00*	*70*	*14*	*0,00*	*140*	*30*	*0,00*	*280*	*75*	*0,00*
2	18	8	17,05	36	14	12,11	72	26	7,54	144	64	2,15
4	9	5	40,00	18	11	35,94	36	23	19,41	72	90	19,30

* For comparison purposes, we cite the previously mentioned findings of a single English auction execution.

Table 4 Evaluation with multiple English auctions.

auctions, we distributed the active CL-agents among all auctions. Table 4 summarizes the data aqcuired from these tests. Expect of the time required for the auctions to converge to the equilibrium price, we also measured the average number that a C-agent has changed auctions, before reaching its offer limit. We found that while some sessions devoted in one auction, bidding persistently in it, other sessions repeatedly rotated auction rooms. We attribute this behavior in computations occurring in C-agents' inference engines; when an agent does not place bids in time in one auction for a while it is expected to change interest in another auction, until becoming an active participant in one.

Figure 39 compares the convergence rate in 1, 2 and 4 parallel auctions. No significant difference is observed when few bidders are present. When this number increases, though, bidding in only two auctions is less efficient than bidding in four. To understand why this holds true, we need to investigate the behavior of participants in each case. Figure 40 explains how agents behave in parallel auctions. When only two auctions are running, C-agents tend to focus on only one of them, resulting in a scheme, where bidders are split between the two and behave as if the auctions were not interrelated. This causes the auctions to delay more for converging to equilibrium. Increasing the number of contemporary auctions, parallelism is improved. The reason why parallelism in not promoted with two auctions is because bids are placed faster in

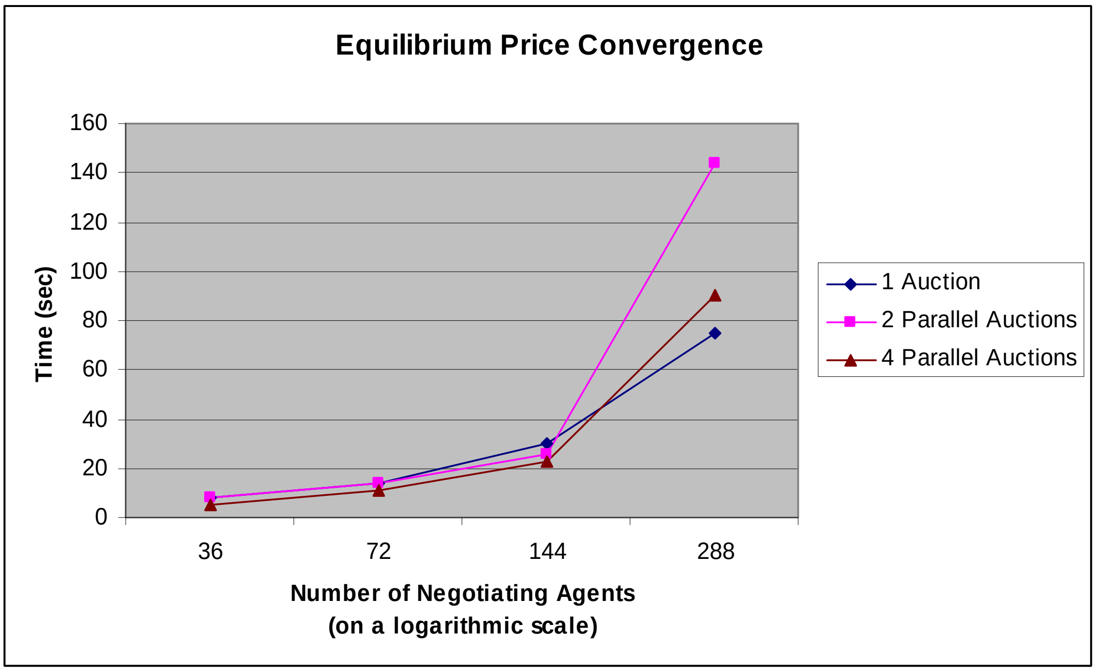


Figure 39 Comparison of performance using 1, 2, and 4 English auctions in terms of convergence rate to equilibrium price.

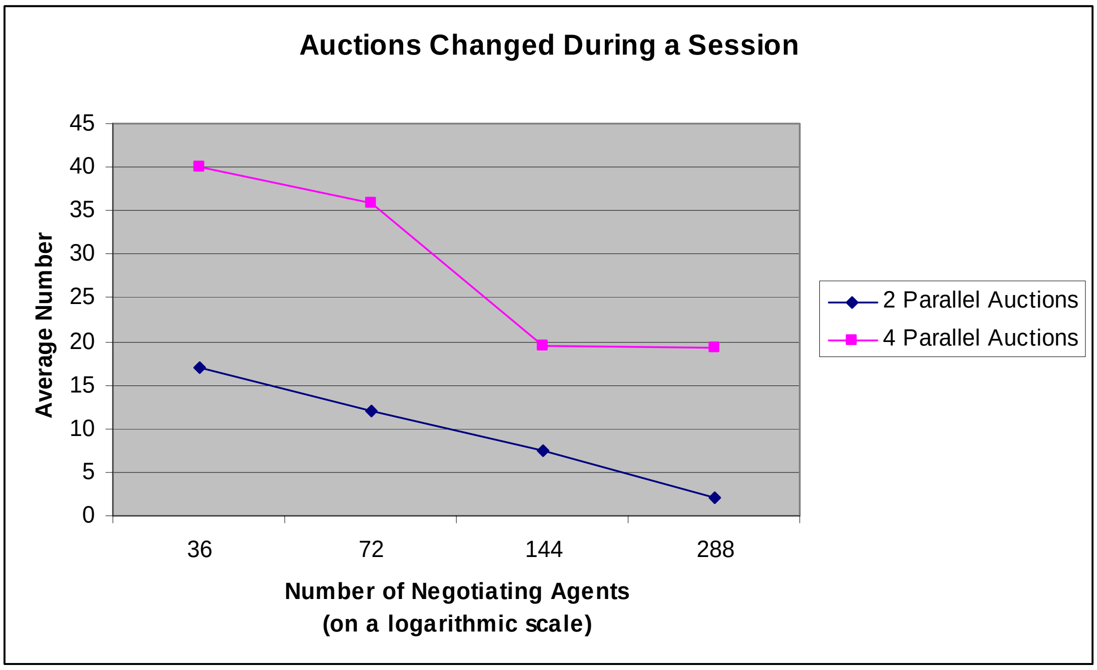


Figure 40 Comparison of performance using 1, 2, and 4 English auctions in terms of auction sessions changed.

an auction than required for an agent to decide whether to change its current auction or not, invalidating its decision (especially if the number of participants is high, increasing the system's computational needs and, as a result, increasing proportionately the time it takes for an agent to make an inference).

This chapter evaluated the system performance under different workloads and conditions. In the next chapter we will propose suggestions for extending the current architecture with new approaches, allowing researchers to study and experiment with novel techniques.

Chapter 8 EXTENSIBILITY SUGGESTIONS

Although the implemented version of the SeMPHoNIA platform is currently fully operational, its scalable architectural design allows us to reengineer several of its aspects and experiment new approaches. We can identify possible extension alternatives at different levels of abstraction intended to enhance its capabilities and allow the platform to adapt to various user needs and broader operational conditions.

8.1 Organizing Operator Peers into Semantic Consortia

There is as yet no concerted classification of peer-to-peer topologies, despite the fact that many attempts have been made to provide peer-to-peer network definitions and models ([83], [1]). The implemented peer-to-peer network architecture of SeMPHoNIA, already described in section 3.2.3, could be characterized as a *hybrid, resource-sharing, direct peer-to-peer model*, when examined from the levels of decentralization and communication point of view. In general, *hybrid* models imply to networks that not all peers have equal roles, but rather some of the resources or services are centralized. In SeMPHoNIA, operator peers originate such a level of centralization, because they are the only peers to provide agency services to the network. On the other hand, among operator peers computational resources are distributed, due to the ability of agents to migrate between agencies, and, thus, achieving a certain degree of *resource sharing*. Moreover, the term *direct model* refers to the message propagation technique applied, according to which no broker service or central indexing is implemented, but instead, information is propagated from one peer to all its physical neighbors or subsets of them.

In this context, we consider an alternative peer-to-peer architectural setting in accordance with the domain distribution on the network. Currently, simple peers connect with at least one operator peer in an ad-hoc manner. Instead, according to the improve approach, consortia of operator peers can be formed characterized by the domain of the products that they are responsible for. Simple peers, either auctioneers or customers, can connect with the appropriate operator peer consortium that manages auctions concerning the domain they wish to be involved with (for example tickets, cars, art etc). As a result, a backbone of semantically related peer clusters can be created, within which all members refer to the same domain and all broadcasted advertisements or queries contain relevant metadata information. This infrastructure

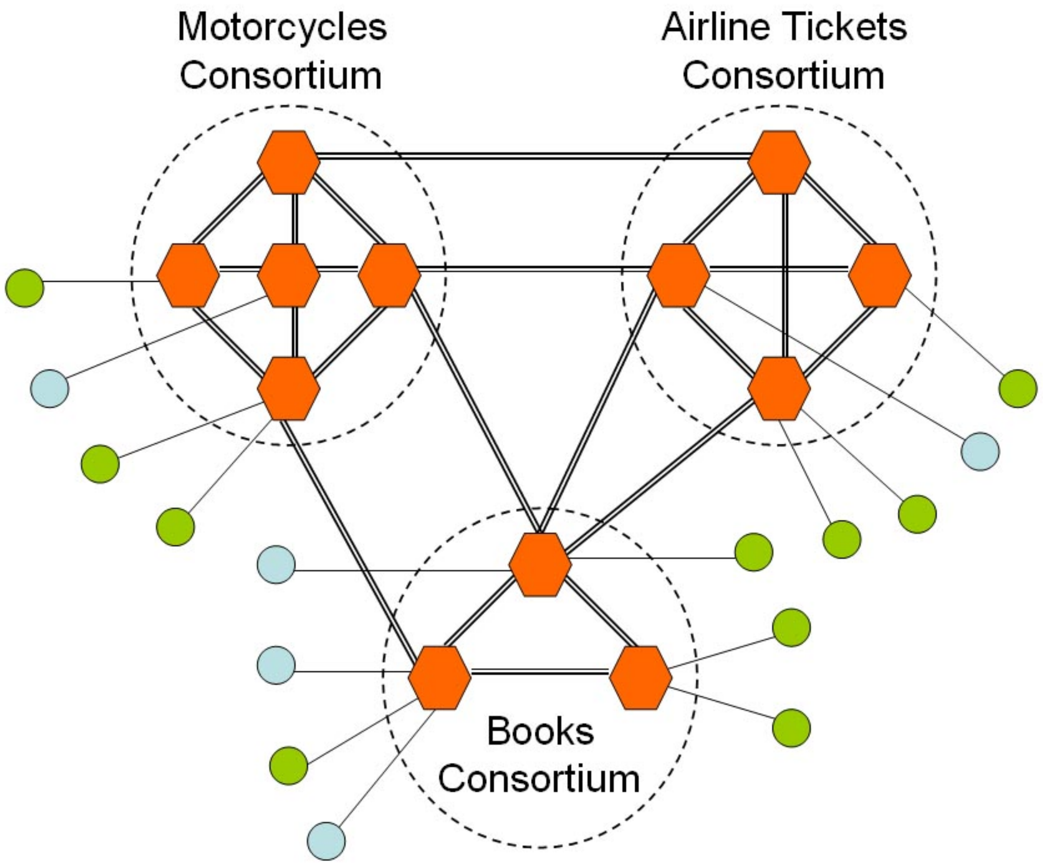


Figure 41 A simple peer-to-peer topology with domain consortia.

accomplishes a more efficient message routing technique and ensures scalability to a large number of nodes.

We distinguish two different topologies for organizing operator peers into domain specific consortia. The one is rather straightforward, but performs well, while maintaining low complexity levels. As it is shown in figure 41, each operator peer belongs to only one domain consortium, but holds knowledge about the location of others. Thus, when a simple peer first joins the system, seeking for an operator to establish connection with, it identifies the appropriate consortium by sending a request, containing the domain it is interested in, and waiting for responses with the appropriate directions. The responses could arrive from peers of the corresponding consortium or from other peers that posses this information in their cache. The consortia interconnect in a random manner, while a public index containing the existing communities is necessary, in order for status information about running operator peers to be available.

A more sophisticated approach, based on the work conducted in the HyperCuP project [82], is to group operator peers into ontology-based hypercube graphs. According to this approach, product domains are organized in a global ontology, which defines the relationships between their concepts and categorizes them according to their relevance. Referencing this ontology, consortia are not randomly connected any more, as described previously, but instead they are organized into a hypercube topology in such a way that consortia whose domains differ in only one logical minterm (conjunction of structuring concepts) are direct neighbors of each other (all but one of their dimension numbers are equal). Operator peers themselves, too, can be constructed in hypercubes following inter-domain concept similarities, allowing messages to be sent to exactly those peers that are explicitly interested in them. Figure 42 shows a potential snapshot of a 4-dimensional operator peer hypercube for one domain (one consortium). Broadcast messages are forwarded only into the appropriate dimensions of the hypercube, based on concept relevance, rather than into all of its dimensions. Such a topology presents certain promising features; its connectivity is optimal, which means that the minimum number of nodes to be removed in order to partition the graph is equal to node degree - 1. Furthermore, it is less vulnerable to denial of service attacks in comparison to ad-hoc networks, such as Gnutella [41] or Freenet [36]. But most importantly, more efficient broadcast and search algorithms can be applied, which guarantee that all nodes receive a message

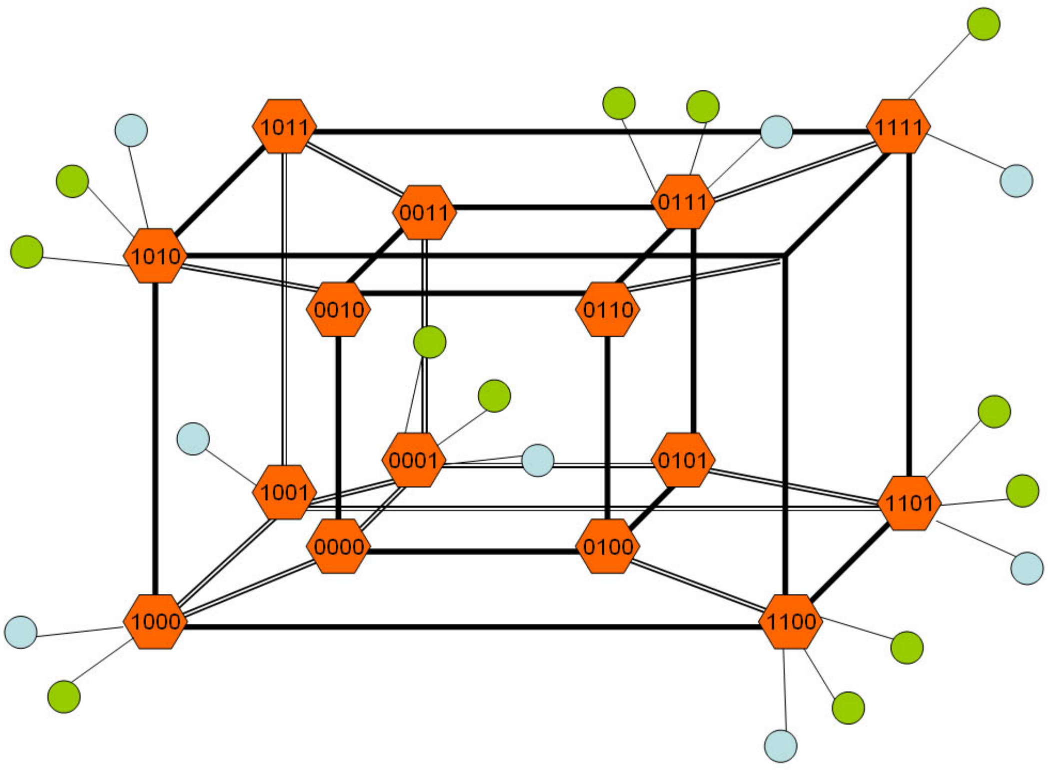


Figure 42 A consortium of operator peers organized in a 4-dimensional peer hypercube.

exactly once during broadcast, exactly N - 1 messages are required to reach all nodes and that the last nodes are reached after $\log_b N$ steps, where N depicts the number of nodes and b the base of the graph (extensive details about the performance of those algorithms, as well as how to construct hypercube peer-to-peer networks and keep them balanced, are presented in [82]).

8.2 Supporting Multiple RDF Query Languages

The current implementation of the SeMPHoNIA platform utilizes RQL as a language for querying the underlying RDF ontologies. The reasons for our choice have been explained in section 3.1.2. Nevertheless, we recognize the fact that no standard for RDF query language has yet emerged and that several other languages have been proposed with different philosophies and features referring to a variety of domains and personal styles (such languages are RDQL, SeRQL, Triple, Versa, N3 etc.). Since querying is a fundamental functionality in our system, special care has been taken in its design so as to be flexible enough to potentially support multiple query languages.

Extending the platform with such a capability is a relatively easy task demanding low human effort. In SeMPHoNIA, all auctioneers store metadata about their products in local ontologies, as it has been described in chapter 3. In order for other users to access those data, a custom RDFServer is invoked that runs at the auctioneer's side and is responsible for all query activity at the underlying database. Members of the SeMPHoNIA network that desire to issue queries to an ontology need only to be aware of the corresponding server's address without being required to understand any of the local database's specifications. For example, the current RDFServer implementation, the RDFSuiteServer, is specialized to support RQL queries, since it is dedicated to collaborate with the ICS-FORTH RDFSuite. Users forward their queries to that server, still ignoring the fact that ICS-FORTH RDFSuite's tools are utilized to produce the answer or which techniques are applied by the server to communicate with the database.

In this context, new RDFServers could be created or the same RDFServer could be extended so as to support different query languages, depending on the associated storage system (for example ICS-FORTH RDFSuite, Sesame, Jena etc.) and the client's language preference. Such an extension would demand no further modification at the system's architecture or the end-user applications, since the communication protocol between clients and RDFServers would remain unaltered. Currently, no parser is available for converting one query language to another, allowing multi-platform interoperability, due to the diversity in expressiveness and feature implementation of those languages. Therefore, one possible approach towards integrating multiple RDF systems and query languages in SeMPHoNIA would be to include the supported features of an RDFServer into the Ontology Advertisement that auctioneers publish to inform others about the domain of their products and the address of that server. Thus, a client could determine which retailers are able to interpret the query languages that s/he prefers and forward the queries to them.

8.3 Additional Suggestions

There are other functionalities of the platform, as well, that could be extended to improve the quality of services and confirm its characterization as a scalable testbed for a variety of experimental scenarios. Intimations of such upgrades have been made in previous sections, where a description of the system's components was attempted.

In section 4.2, the novel design of the agent subsystem operation was illustrated together with an example for demonstrating its potentials (peer-to-peer auction). We argued that such a flexible design is suitable not only for auction scenarios, but also for other models of negotiation, accommodating agent developers in focusing on the aspects they desire (i.e., negotiation algorithms, agent inference logic etc), disembarrassing them from other, more general, issues, concerning interoperability, communication protocol design etc. Considerations for organizing and developing a set of primitive methods (agent API) for describing standard agent communication and functionality parameters are examined, to simplify the programmer's task.

Moreover, in section 3.2.3, the usefulness of network space segmentation into multiple peer groups has been elicited. Each agent auction is associated with a peer group, which the user must join to become eligible to participate and bid in the auction. Certain authorization mechanisms are utilized to control the process of joining. As a result, various alterations of auction sessions can be generated by this pattern; private auctions or secure auction sessions could be implemented by extending the authorization process for a peer to join the corresponding peer group.

Chapter 9 CONCLUSIONS AND FUTURE WORK

We have presented the design and implementation of SeMPHoNIA, a system that integrates three emerging technologies; intelligent software agents, peer-to-peer networking and the Semantic Web. SeMPHoNIA is an architecture for an agent-based virtual marketplace and is intended as a platform for research in multi-agent systems, ontologies, peer-to-peer networks and automatic negotiation, as well as an application for experimentation with these technologies. Our primary motivation in creating this platform has been to demonstrate the power of combining the three technologies in facilitating the participation of human users in next-generation e-Commerce transactions. The system realizes a number of auction scenarios as a general negotiation framework among users, while maintaining a flexible design that allows it to be easily extended with new scenarios and techniques.

SeMPHoNIA project deals with many issues concerning most of e-Commerce phases and proposes novel and feasible solutions. Semantic languages and formalisms are being used for conceptualizing negotiation protocols and providing product domain characterization. They are also used for semantically publishing and discovering available resources on the network. Local ontologies capture concepts and relations of products and allow retailers to add metadata to their descriptions, following a common schema definition, while customers query those repositories using highly expressive languages.

Software agent skills are exploited to automate human user's participation in multiple, possibly interrelated, auctions and to provide advanced means of e-

Commerce transactions. The multi-agent architecture is flexible enough to allow experimentation with various negotiation protocols, bidding algorithms and job allocation techniques. We have shown how SeMPHoNIA agents segment the job of participating into multiple auctions into subtasks and cooperate appropriately to accomplish them in a manner consistent with their owner's preferences. Intelligent software agents perform various functions, such as auction bidding and monitoring, statistical analysis of auction data, background execution and more.

We have also described the peer-to-peer infrastructure that is utilized to provide the medium for remote users to contact and negotiate, while at the same time distributes computational and knowledge resources. The architecture of SeMPHoNIA's peer-to-peer network is scalable, permitting various alternative models to be implemented, enhancing its capabilities. We have presented two possible alternatives that could be applied according to the conditions encountered.

In addition, three auction protocols have been implemented that cover many issues of multi-party negotiation; the English auction, which is a multi-round open model, the Vickrey auction, which is single-round and sealed-bid, and, finally, a hybrid peer-to-peer auction, which is a multi-round and non-centralized auction. We have shown that extending the platform with new negotiation protocols is a non-trivial task, affecting only specific components of the system.

Nevertheless, the approach we have presented is open to further improvements and modifications. Our first concern is to use a standardized communication language to allow heterogeneous agents to exchange information and knowledge. Languages, such as FIPA ACL [33] or KQML [31], provide the means for agents to share a common understanding of the possible message types and terms used in the communication represented in a declarative format. The content of the messages can be represented by an ontology developed for the application domain. Another alternative could be SOAP ([86],[87]), which provides a way to communicate between applications running on different operating systems, with different technologies and programming languages, so as to exchange information in a decentralized, distributed environment.

Another interesting research direction is to further enhance the auctioneer's task by integrating the RDFSuite UpdateAPI, which permits users to dynamically modify data stored in their ontologies. A possible approach could be to create a fill-in form that would allow auctioneers to provide standard values for the parameters of their auction sessions and leave the task of developing the corresponding ontologies to the SeMPHoNIA platform.

Finally, an important aspect of our future objectives is to enhance the graphical user interface with graphics and diagrams both for following the progress of auctions and for viewing the statistical information gathered by S-agents.

Chapter
10
REFERENCES

[1] Abecker A., Tellmann R.: Analysis of Interaction between Semantic Web Languages, P2P architectures, and Agents. EU-IST Project SWWS – Semantic Web Enabled Web Services project IST-2002-37134, 2003.

[2] Alexaki Sofia, Christophides Vassilis, Karvounarakis Grigoris, Plexousakis Dimitris, Tolle K.: The ICS-FORTH RDFSuite: Managing Voluminous RDF Description Bases. Proc. of the 2nd International Workshop on the Semantic Web (SemWeb'01) in conjunction with Tenth International World Wide Web Conference (WWW10), pp. 1-13, Hong-Kong, May 1st, 2001.

[3] Andreasen A.: Attitudes and Customer Behavior: A Decision Model. In L. Preston (ed.), New Research in Marketing. California Institute of Business and Economics Research, 1965.

[4] Anthony Patricia, Jennings Nicholas R.: Developing a Bidding Agent for Multiple Heterogeneous Auctions. ACM Transactions on Internet Technology, Vol. 3, No. 3, Pages 185–217, 2003.

[5] Antoniou Grigoris, Frank van Harmelen: A Semantic Web Primer. The MIT Press, 2004.

[6] Arumugam Madhan, Sheth Amit, Arpinar Budak I.: Towards Peer-to-Peer Semantic Web: A Distributed Environment for Sharing Semantic Knowledge on the Web. In Proc. of WWW-02 Workshop on Real World RDF and Semantic Web Applications, Honolulu, Hawaii, 2002.

[7] Baader Franz, Horrocks Ian, Sattler Ulrike.: Description Logics as Ontology Languages for the Semantic Web. In: D. Hutter and W. Stephan (eds.), Festschrift in honor of Jorg Siekmann, LNAI, Springer, 2003.

[8] Bapna R., Goes P., Gupta A.: Insights and Analyses of Online Auctions. Comm. ACM Vol. 44, No. 11, pp. 42-50, 2001.

[9] Bartolini Claudion, Preist Chris, Jennings Nicholas R.: A Generic Software Framework for Automated Negotiation, , AAMAS '02, Bologna, Italy, July 15-9, 2002.

[10] Bayardo R. J. Jr., Bohrer W., Brice R., Cichocki A., Fowler J., Helal A., Kashyap V., Ksiezyk T., Martin G., Nodine M., Rashid M., Rusinkiewicz M., Shea R., Unnikrishnan C., Unruh A., Woelk D.: InfoSleuth: Agent-Based Semantic Integration of Information in Open and Dynamic Environments. In Proceedings of SIGMOD '97, 1997.

[11] Bejar Javier, Cortes Ulises: Agent Strategies on DPB Auction Tournaments. F. Dignum and U. Cortes (Eds.): AMEC 2000, LNAI 2003, pp. 155-172, 2001.

[12] Ben-Ameur H., Chaib-draa B., Kropf P.: Multi-item auctions for automatic negotiation. Journal of Information and Software Technology, 44 pp. 291-301, 2002.

[13] Beneventano D., Bergamashi S., Fergnani A., Guerra F., Vincini M., Montanari D.: A Peer-to-Peer Agent-Based Semantic Search Engine. Eleventh Italian Symposium on Advanced Database Systems Cetraro (CS), Italy, June 24-27, 2003.

[14] Berners-Lee T., Hendler J., Lassila O.: The semantic web. Scientific American, 284(5) pp. 34-43, May 2001.

[15] Bettman J.: An Information Processing Theory to Consumer Choice. Addison-Wesley, 1979.

[16] Bichler Martin, Field Simon, Werthner Hannes: Introduction: Theory and Application of Electronic Market Design. Electronic Commerce Research Journal, 1:215-220, 2001.

[17] Branco F.: The design of multidimensional auctions. RAND Journal of Economics, Vol 28, pp. 63-81, 1997.

[18] Brickley D., Guha R. V.: Resource Description Framework Schema Specification 1.0, W3C Candidate Recommendation, Mar 2000, available at http://www.w3.org/TR/rdf-schema/

[19] Byde Andrew: A Comparison among Bidding Algorithms for Multiple Auctions. J. Padget et al (Eds.): AMEC 2002, LNAI 2531, pp. 1–16, 2002.

[20] Byde A., Preist C., Jennings N. R.: Decision Procedures for Multiple Auctions. AAMAS'02 Bologna, Italy, July 1519, 2002.

[21] Cabac Lawrence, Moldt Daniel, Rolke Heiko.: A Proposal for Structuring Petri Net-Based Agent Interaction Protocols. W.M.P. van der Aalst and E. Best (Eds.): ICATPN 2003, LNCS 2679, pp. 102–120, 2003.

[22] Comparison of RDF Query Languages website: http://www.aifb.uni-karlsruhe.de/WBS/pha/rdf-query/

[23] Cramton Peter C.: The FCC spectrum auctions: An early assessment. Journal of Economics and Management Strategy, 6(3):431-495, 1997.

[24] Csirik Janos A., Littman Michael L., Singh Satinder, Stone Peter: FAucS: An FCC Spectrum Auction Simulator for Autonomous Bidding Agents. WELCOM 2001, LNCS 2232, pp. 139-151, 2001.

[25] Cultural Portal, http://139.91.183.30:8999/RQLdemo/

[26] DAML+OIL Reference (Mar. 2001). http://www.w3.org/TR/daml+oil-reference/

[27] Duvigneau Michael, Moldt Daniel, Rolke Heiko.: Concurrent Architecture for a Multi-agent Platform. F. Giunchiglia et al. (Eds.): AOSE 2002, LNCS 2585, pp. 59–72, 2003.

[28] Engel J., Blackwell R.: Consumer Behavior. 4th ed. CBS College Publishing, 1982.

[29] Esther David, Azoulay-Schwartz Rina, Kraus Sarit.: An English Auction Protocol for Multi-attribute Items. J. Padget et al. (Eds.): AMEC 2002, LNAI 2531, pp. 52-68, 2002.

[30] Fernandes Joao M., Belo Orlando.: Modeling Multi-Agent Systems Activities Through Colored Petri Nets. Proc. 16th IASTED Int. Conf. on Applied Informatics, 23-25 February 1998, Anaheim, CA, pp. 17-20, 1998.

[31] Finin T., Labrou Y., Mayfield J.: KQML as an agent communication language. In J. Bradshaw (ed.), Software Agents AAAI/MIT Press, 1997.

[32] FIPA 2001, Communicative act library specification (XC00037H). http://www.fipa.org/spec

[33] FIPA Specifications, http://www.fipa.org/specifications/

[34] Fischer, G., Lemke A., Mastaglio T., Morch A. I.: Critics: An Emerging Approach to Knowledge Based Human Computer Interaction. International Journal of Man-Machine Studies, p.695-721, 35(5), November, 1991.

[35] Fornara Nicoletta, Gambardella Luca Maria. An Autonomous Bidding Agent for Simultaneous Auctions. M. Klusch and F. Zambonelli (Eds): CIA 2001, LNAI 2182, pp. 130–141, 2001.

[36] Freenet website, http://www.freenetproject.org

[37] Giang Nguyen T., Tung Dang T.: Agent Platform Evaluation and Comparison, Pellucid 5FP IST-2001-34519.

[38] Grasshopper web site, http://www.grasshopper.de

[39] Greenwald Amy, Stone Peter.: Autonomous Bidding Agents in the Trading Agent Competition. IEEE Internet Computing, vol. 5, no. 2, pp. 52-60, 2001.

[40] Grosof Benjamin N., Poon Terence C.: SweetDeal: Representing Agent Contracts with Exceptions using XML Rules, Ontologies and Process Descriptions. In Proc. of WWW 2003, Budapest, Hungary, May 20-24, 2003.

[41] Gnutella website, http://gnutella.com

[42] Guth Werner, Ivanova-Stenzel Radosveta, Konigstein Manfred, Strobel Martin.: Learning to Bid – An Experimental Stud of Bid Function Adjustments in Auctions and Fair Division Games. The Economic Journal, 113 (April), pp. 477-494, 2003.

[43] Haase Peter, Broekstra Jeen, Eberhart Andreas, Volz Raphael: A Comparison of RDF Query Languages. S.A. McIlraith et al. (Eds.): ISWC 2004, LNCS 3298, pp. 502–517, 2004.

[44] He Minghua, Jennings Nicholas R.: SouthamptonTAC: An Adaptive Autonomous Trading Agent. ACM Transactions on Internet Technology, Vol. 3, pp. 218-235, 2003.

[45] He Minghua, Jennings Nicholas R., Leung Ho-Fung: On Agent-Mediated Electronic Commerce. IEEE Transactions on Knowledge and Data Engineering, Vol. 15, No. 4, July/August, 2003.

[46] Homayounfar H., Wang F., Areibi S.: Advanced P2P Architecture Using Autonomous Agents. The ISCA 15th International Conference on Computer Applications in Industry and Engineering (CAINE 2002), San Diego, California, p115-118, November, 2002.

[47] Howard J., Sheth J.: The Theory of Buyer Behavior. John Wiley and Sons, 1969.

[48] ICS-FORTH, The ICS-Forth RDFSuite web site, http://139.91.183.30:9090/RDF, 2001.

[49] Java website, http://www.java.sun.com

[50] Jennings N. R.: On agent-based software engineering. Artificial Intelligence, 117 (2)277-296, 2000.

[51] Jennings N. R., Anthony P.: Evolving Bidding Strategies for Multiple Auctions. Proc. 15th European Conf. Artificial Intelligence, F. van Harmelen, ed., pp. 178-182, Amsterdam: IOS Press 2002.

[52] Joseph Sam: NeuroGrid: Semantically Routing Queries in Peer-to-Peer Networks. Proc. International Workshop on P2P Computing, 2002.

[53] Joshi Nicky, Thakore Kushal, Su Stanley Y. W.: IntelliBid: An Event-Trigger-Rule-Based Auction System over the Internet.World Wide Web: Internet and Web Information Systems, 7, pp.181-210, 2004.

[54] JXTA configuration help: http://platform.jxta.org/java/confighelp.html

[55] JXTA Project, http://www.jxta.org

[56] Kagal L., Perich F., Chen H., Tolia S., Zou Y., Finin T., Joshi A., Peng Y., Cost R. S., Nicholas C.: Agents Making Sense of the Semantic Web. Proceedings of the First GSFC/JPL Workshop on Radical Agent Concepts, 2003 (WRAC).

[57] Karvounarakis G., Alexaki S., Christophides V., Plexousakis D., M. Schol.: RQL: A Declarative Query Language for RDF. In Proceedings of the Eleventh International World Wide Web Conference (WWW'02) Honolulu, Hawaii, USA, May 7-11, 2002.

[58] Klemperer P.: Auction theory: a guide to the literature. Journal of Economic Surveys Vol. 13, No. 3, pp 227-286, 1999.

[59] Klusch M.: Intelligent Information Agents: Agent-based Information Discovery and Management on the Internet. (eds) Springer-Verlag Berlin Heidelberg, 1999.

[60] Kurt Jensen: An Introduction to the Practical Use of Coloured Petri Nets. To appear in the course material from the Advanced Course on Petri Nets, Dagstuhl Germany, Springer-Verlag, 1996.

[61] Lassila O., Swick R.: Resource Description Framework Model and Syntax Specification, W3C Recommendation, Feb 1999, available at http://www.w3.org/TR/REC-rdf-syntax/

[62] Lee Raymond S.T., Liu James N.K.: iJADE eMiner – A Web-Based Mining Agent Based on Intelligent Java Agent Development Environment (iJADE) on Internet Shopping, Hong Kong Polytechnic University. D. Cheung, G.J. Williams, and Q. Li (Eds.): PAKDD 2001, LNAI 2035, pp. 28-40, 2001.

[63] Liu Duen-Ren, Hwang Tzyy-Feng: An agent-based approach to flexible commerce in intermediary-centric electronic markets. Journal of Network and Computer Applications, 27 pp. 33-48, 2004.

[64] Maes P.: Agents that reduce work and information overload. Communication of the ACM, Vol. 37 No. 7, pp. 31-40, 1994.

[65] Maudet N., Chaib-Draa B.: Commitment-based and dialogue-game-based protocols: new trends in agent communication languages. The Knowledge Engineering Review, Vol. 17, No. 2, pp. 157-179, Cambridge University Press, 2002.

[66] McAfee Preston R., McMillan John: Auctions and bidding. Journal of Economic Literature, Vol. 25, pp. 699-738, 1987.

[67] Mota Luis, Botelho Luis, Mendes Hugo, Lopes Antonio: O_3F: an Object Oriented Ontology Framework., AAMAS 2003, Melbourne, Australia, July 14-18, 2003.

[68] Napster website, http://napster.com

[69] Nejdl Wolfgang, Wolf Boris, Qu Changtao, Decker Stefan, Sintek Michael, Naeve Ambjorn, Nilsson Mikael, Palmer Matthias, Risch Tore.: EDUTELLA: A P2P Networking Infrastructure Based on RDF. WWW2002, Honolulu, Hawaii, USA. May 7-11, 2002.

[70] Ogston Elth, Vassiliadis Stamatis: A Peer-to-Peer Agent Auction. AAMAS’02, Bologna, Italy, July 15-19, 2002.

[71] Ono Chihiro, Nishiyama Satoshi, Horiuchi Hiroki.: Reducing complexity in winner determination for combinatorial ascending auctions. Electronic Commerce Research and Applications, Vol. 2, pp. 176–186, 2003.

[72] OWL (Ontology Working Language), from W3C WebOntologies (WebOnt) Working Group, http://www.w3.org/2001/sw/webont . Working draft of July 29, 2002.

[73] Paolucci Massimo, Sycara Katia, Nishimura Takuya, Srinivasan Naveen: Using DAML-S for P2P Discovery. In Proceedings of the First International Conference on Web Services (ICWS'03), Las Vegas, Nevada, USA, pp 203- 207, June, 2003.

[74] Preist C., Bartolini C., Phillips I., Dignum F., Cortes U.: Algorithm Design for Agents Which Participate in Multiple Simultaneous Auctions. (Eds.): AMEC 2000, LNAI 2003 pp. 139-154, Springer-Verlag, 2001.

[75] Preist C., Byde A., Bartolini C.: Economic Dynamics of Agents in Multiple Auctions. Proc. Fifth International Conference on Autonomous Agents, pp. 545-551, 2001.

[76] Purvis Martin K., Nowostawski Mariusz, Cranefield Stephen, Oliveira Marcos: Multi-agent Interaction Technology for Peer-to-Peer Computing in Electronic Trading Environments. C. Zhang, H.W. Guesgen, W.K. Yeap (Eds.): PRICAI 2004, LNAI 3157, pp. 625–634, 2004 and Second International Workshop on Agents and Peer-to-Peer Computing, Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2003), Moro, G., Sartori, C., and Singh, M. (eds.), Melbourne Australia pp. 103-114, 2003.

[77] Reeves Daniel M., Wellman Michael P., Grosof Benjamin N.: Automated Negotiation from Declarative Contract Descriptions. AGENTS 2001, Montreal, Quebec, Canada, May 28-June 1, 2001.

[78] Ren Z., Anumba C.J.: Learning in multi-agent systems: a case study of construction claims negotiation. Advanced Engineering Informatics Vol. 16, No. 3, pp. 227-284, Elsevier 2002.

[79] Resource Description Framework (RDF) http://www.w3.org/RDF

[80] Rosenschein J., Zlotkin G.: Rules of Encounter: Designing Conversations for Automated Negotiation Among Computers. Artificial Intelligence, MIT Press, Cambridge, MA, 1994.

[81] Sandholm T.: Distributed Rational Decision Making. Multi-agent Systems, G. Weiss, ed., pp. 201-258, The MIT Press, 1999.

[82] Schlosser Mario, Sintek Michael, Decker Stefan, Nejdl Wolfgang: A Scalable and Ontology-Based P2P Infrastructure for Semantic Web Services. Proceedings of the Second International Conference on Peer-to-Peer Computing, 2002.

[83] Schollmeier R.: A Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Applications. Proceedings of the First International Conference on Peer-to-Peer Computing, 2001.

[84] Semantic Web and Peer-to-Peer, http://swap.semanticweb.org

[85] Semantic Web Enabled Web Service, http://swws.semanticweb.org

[86] SOAP, http://www.w3.org/2000/xp/Group/

[87] SOAP Version 1.2 specification, http://www.w3.org/TR/soap12

[88] Stone Peter, Littman Michael L., Singh Satinder, Kearns Michael.: ATTac-2000: An Adaptive Autonomous Bidding Agent. Journal of Artificial Intelligence Research 15:189-206 2001, Fifth International Conference on Autonomous Agents, 2001.

[89] Stone Peter, Schapire Robert E., Csirik Janos A., Littman Michael L., McAllester David.: ATTac-2001: A Learning, Autonomous Bidding Agent. In Agent Mediated Electronic Commerce IV, LNCS Vol. 2531, Springer Verlag, 2002.

[90] TAGA website, http://taga.umbc.edu

[91] Tamma Valentina, Wooldridge Michael, Blacoe Ian, Dickinson Ian.: An Ontology based Approach to Automated Negotiation. In Proceedings of the Fourth International Workshop on Agent-Mediated Electronic Commerce (AMEC-2002), Bologna, Italy, July 2002.

[92] Terziyan Vagan, Zharko Andriy: Semantic Web and Peer-to-Peer: Integration and Interoperability in Industry. International Journal of Computers, Systems and Signals. In: International Journal of Computers, Systems and Signals, IAAMSAD, ISSN 1608-5655, Vol. 4, No. 2, pp. 33-46, 2003.

[93] The Trading Agent Competition website. http://www.sics.se/tac/

[94] Trastour David, Bartolini Claudio, Preist Chris.: Semantic Web Support for the Business-to-Business e-Commerce Lifecycle, WWW2002, Honolulu, Hawaii, USA, May 7-11, 2002.

[95] UDDI: The UDDI Technical White Paper. http://www.uddi.org/ 2000.

[96] Verma Kunal, Sivashanmugam Kaarthik, Sheth Amit, Abhijit Patil, Oundhakar Swapna, Miller John: METEOR-S WSDI: A Scalable P2P Infrastructure of Registries for Semantic Publication and Discovery of Web Services. Journal of Information Technology and Management 2004 (to appear).

[97] Web Ontology Language (OWL) Guide Version 1.0. W3C Working Draft 4 November 2002. http://www.w3.org/TR/2002/WD-owl-guide-20021104/

[98] Weiss G.: Adaptation and learning in multi-agent systems: some remarks and a bibliography. In: Weiss G, Sen S., Adaptation and learning in multi-agent systems. Lecture notes in artificial intelligence. Vol. 1042 pp. 1-21 Berlin, Springer, 1996.

[99] Wellman Michael P., Greenwald Amy, Stone Peter, Wurman Peter R.: The 2001 Trading Agent Competition, Fourteenth Innovative Applications of Artificial Intelligence Conference (IAAI-2002), Edmonon, August 2002.

[100] Weyns Danny, Holvoet Tom.: A Colored Petri Net for a Multi-Agent Application. Proceedings of MOCA’02 (Moldt, D., ed), Vol. 561, DAIMI PB, pp. 121-141, 2002.

[101] Wooldridge Michael, Dickinson Ian.: Towards Practical Reasoning Agents for the Semantic Web. AAMAS 2003, Melbourne, Australia, July 14-18, 2003.

[102] Wooldridge M., Jennings N.: Intelligent Agents: Theory and Practice. Knowledge Engineering Review Vol. 10, no 2, pp. 115-152, 1995.

[103] Wooldridge M., Jennings N.: Reasoning about rational agents. MIT Press, 2000.

[104] Wurman P. R.: Dynamic pricing in the virtual marketplace. IEEE Internet Computing, Vol. 5, pp. 36-42, March/April, 2001.

[105] Wurman Peter R., Wellman Michael P., Walsh William E.: A Parametrization of the Auction Design Space, Games and Economic Behavior, vol. 35, issue 1-2, pages 304-338, 2001.

[106] Wurman Peter R., Wellman Michael P., Walsh William E.: The Michigan Internet AuctionBot: a configurable auction server for human and software agents, Proceedings of the second international conference on Autonomous agents, p.301-308, Minneapolis, Minnesota, United States, May 10-13, 1998.

[107] Yang I., Jeong H., Kahng B., Barabasi A. L. Emerging behavior in electronic bidding. Physical Review E 68, 016102, The American Physical Society, 2003.

[108] Zou Youyong, Finin Tim, Ding Li, Chen Harry.: Using Semantic Web Technologies in Multi-Agent Systems: a case study in the TAGA trading agent environment. Proceedings of the 5th international conference on Electronic commerce, Pittsburgh, Pennsylvania, pp. 95-101, September 30 - October 03, 2003.

[109] Zou Youyong, Finin Tim, Ding Li, Chen Harry, Pan Rong.: TAGA: Trading agent competition in Agentcities. IJCAI-03 Workshop on Trading Agent Desing and Analysis, Acapulco, 2003.

Appendix A APPENDIX A – ONTOLOGIES

The appendix contains RDF ontologies of the MetaSchema, Auction Schema, Artifacts Domain Schema, as well as, an example RDF Resource Description, concerning descriptions of one auctioneer's auction sessions.

MetaSchema

```
  <?xml version="1.0" ?>
      <!--
       This is a metaschema. Here are defined meta-classes of classes
       (e.g., RealWorldObject), meta-classes of properties (i.e.,
      SchemaProperty)
       as well properties that are applied to classes (e.g., related)
       and properties that are applied  to properties (e.g.,
      maxCardinality)
    -->
<rdf:RDF xml:lang="en" xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
    syntax-ns#" xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#" xmlns="">
<rdfs:Class rdf:ID="RealWorldObject">
  <rdfs:subClassOf rdf:resource="http://www.w3.org/2000/01/rdf-
    schema#Class" />
    </rdfs:Class>
<rdfs:Class rdf:ID="WebResource">
  <rdfs:subClassOf rdf:resource="http://www.w3.org/2000/01/rdf-
    schema#Class" />
    </rdfs:Class>
<rdfs:Class rdf:ID="SchemaProperty">
  <rdfs:subClassOf rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-
    ns#Property" />
    </rdfs:Class>
<SchemaProperty rdf:ID="related">
```

```
  <rdfs:domain rdf:resource="http://www.w3.org/2000/01/rdf-
    schema#Class" />
  <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"
    />
    </SchemaProperty>
<SchemaProperty rdf:ID="maxCardinality">
  <rdfs:domain rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-
    ns#Property" />
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#integer"/>
    </SchemaProperty>
    </rdf:RDF>
```

Auction Ontology

```
    <?xml version="1.0" ?>
<rdf:RDF xml:lang="en" xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
    syntax-ns#" xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:ms="file:C:\AuctionRDFSchema\metaschema.rdf#"
    xmlns:rdfsuite="http://139.91.183.30:9090/RDF/rdfsuite.rdfs#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#">
  <ms:AuctionObject rdf:ID="Auction" />
  <ms:AuctionObject rdf:ID="Broker" />
  <ms:AuctionObject rdf:ID="ItemForSale" />
  <ms:AuctionObject rdf:ID="AuctionType" />
  <ms:AuctionObject rdf:ID="Shipping" />
  <ms:AuctionObject rdf:ID="Payment" />
<ms:AuctionObject rdf:ID="English">
  <rdfs:subClassOf rdf:resource="#AuctionType" />
    </ms:AuctionObject>
<ms:AuctionObject rdf:ID="Dutch">
  <rdfs:subClassOf rdf:resource="#AuctionType" />
    </ms:AuctionObject>
<ms:AuctionObject rdf:ID="Vickrey">
  <rdfs:subClassOf rdf:resource="#AuctionType" />
    </ms:AuctionObject>
<ms:AuctionObject rdf:ID="Peer-to-Peer">
  <rdfs:subClassOf rdf:resource="#AuctionType" />
    </ms:AuctionObject>
<ms:AuctionObject rdf:ID="First_Price_Sealed">
  <rdfs:subClassOf rdf:resource="#AuctionType" />
    </ms:AuctionObject>
<ms:AuctionObject rdf:ID="Credit_Card">
  <rdfs:subClassOf rdf:resource="#Payment" />
    </ms:AuctionObject>
<ms:AuctionObject rdf:ID="Cash">
  <rdfs:subClassOf rdf:resource="#Payment" />
    </ms:AuctionObject>
<ms:AuctionObject rdf:ID="Personal_Check">
  <rdfs:subClassOf rdf:resource="#Payment" />
```

```
    </ms:AuctionObject>
<ms:AuctionObject rdf:ID="Bank_to_Bank">
  <rdfs:subClassOf rdf:resource="#Payment" />
    </ms:AuctionObject>
<ms:AuctionObject rdf:ID="US_Airmail">
  <rdfs:subClassOf rdf:resource="#Shipping" />
    </ms:AuctionObject>
<ms:AuctionObject rdf:ID="Europe_Airmail">
  <rdfs:subClassOf rdf:resource="#Shipping" />
    </ms:AuctionObject>
<ms:AuctionObject rdf:ID="Door_to_Door_Delivery">
  <rdfs:subClassOf rdf:resource="#Shipping" />
    </ms:AuctionObject>
<rdf:Property rdf:ID="ending_time">
  <rdfs:domain rdf:resource="#Auction" />
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#dateTime"
    />
  <ms:maxCardinality>1</ms:maxCardinality>
    </rdf:Property>
<rdf:Property rdf:ID="starting_time">
  <rdfs:domain rdf:resource="#Auction" />
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#dateTime"
    />
  <ms:maxCardinality>1</ms:maxCardinality>
    </rdf:Property>
<rdf:Property rdf:ID="name">
  <rdfs:domain rdf:resource="#Auction" />
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
    </rdf:Property>
<rdf:Property rdf:ID="reserve_price">
  <rdfs:domain rdf:resource="#Auction" />
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#integer"/>
  <ms:maxCardinality>1</ms:maxCardinality>
    </rdf:Property>
<rdf:Property rdf:ID="bid_step">
  <rdfs:domain rdf:resource="#Auction" />
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#integer"/>
    </rdf:Property>
<rdf:Property rdf:ID="number_of_products">
  <rdfs:domain rdf:resource="#Auction" />
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#integer"/>
    </rdf:Property>
<rdf:Property rdf:ID="sells">
  <rdfs:domain rdf:resource="#Auction" />
  <rdfs:range rdf:resource="#ItemForSale" />
    </rdf:Property>
<rdf:Property rdf:ID="controled_by">
  <rdfs:domain rdf:resource="#Auction" />
  <rdfs:range rdf:resource="#Broker" />
    </rdf:Property>
<rdf:Property rdf:ID="has_type">
  <rdfs:domain rdf:resource="#Auction" />
  <rdfs:range rdf:resource="#AuctionType" />
    </rdf:Property>
```

```
<rdf:Property rdf:ID="accepts">
  <rdfs:domain rdf:resource="#Broker" />
  <rdfs:range rdf:resource="#Payment" />
     </rdf:Property>
<rdf:Property rdf:ID="delivers_by">
  <rdfs:domain rdf:resource="#Broker" />
  <rdfs:range rdf:resource="#Shipping" />
     </rdf:Property>
     </rdf:RDF>
```

Artifacts Domain Schema

```
     <?xml version="1.0" ?>
<rdf:RDF xml:lang="en" xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
     syntax-ns#" xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:ms="file:C:\AuctionRDFSchema\metaschema.rdf#"
     xmlns:rdfsuite="http://139.91.183.30:9090/RDF/rdfsuite.rdfs#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#">
  <ms:RealWorldObject rdf:ID="Artist" />
  <ms:RealWorldObject rdf:ID="Artifact" />
  <ms:RealWorldObject rdf:ID="Museum" />
<ms:RealWorldObject rdf:ID="Sculptor">
  <rdfs:subClassOf rdf:resource="#Artist" />
     </ms:RealWorldObject>
<ms:RealWorldObject rdf:ID="Painter">
  <rdfs:subClassOf rdf:resource="#Artist" />
     </ms:RealWorldObject>
<ms:RealWorldObject rdf:ID="Cubist">
  <rdfs:subClassOf rdf:resource="#Painter" />
     </ms:RealWorldObject>
<ms:RealWorldObject rdf:ID="Flemish">
  <rdfs:subClassOf rdf:resource="#Painter" />
     </ms:RealWorldObject>
<ms:RealWorldObject rdf:ID="Sculpture">
  <rdfs:subClassOf rdf:resource="#Artifact" />
  <ms:related rdf:resource="#Sculptor" />
     </ms:RealWorldObject>
<ms:RealWorldObject rdf:ID="Painting">
  <rdfs:subClassOf rdf:resource="#Artifact" />
  <ms:related rdf:resource="#Painter" />
     </ms:RealWorldObject>
<rdf:Property rdf:ID="creates">
  <rdfs:domain rdf:resource="#Artist" />
  <rdfs:range rdf:resource="#Artifact" />
     </rdf:Property>
<rdf:Property rdf:ID="paints">
  <rdfs:domain rdf:resource="#Painter" />
  <rdfs:range rdf:resource="#Painting" />
  <rdfs:subPropertyOf rdf:resource="#creates" />
```

```
   </rdf:Property>
<rdf:Property rdf:ID="sculpts">
  <rdfs:domain rdf:resource="#Sculptor" />
  <rdfs:range rdf:resource="#Sculpture" />
  <rdfs:subPropertyOf rdf:resource="#creates" />
   </rdf:Property>
<rdf:Property rdf:ID="technique">
  <rdfs:domain rdf:resource="#Painting" />
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
   </rdf:Property>
<rdf:Property rdf:ID="exhibited">
  <rdfs:domain rdf:resource="#Artifact" />
  <rdfs:range rdf:resource="#Museum" />
   </rdf:Property>
<rdf:Property rdf:ID="first_name">
  <rdfs:domain rdf:resource="#Artist" />
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
   </rdf:Property>
<rdf:Property rdf:ID="last_name">
  <rdfs:domain rdf:resource="#Artist" />
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
   </rdf:Property>
  - <!--
  The property working_hours has as range an Enumeration
   -->
<rdf:Property rdf:ID="working_hours">
  <rdfs:domain rdf:resource="#Museum" />
<rdfs:range>
<rdfsuite:Enumeration>
  <rdfsuite:enum_elem>9-1, 5-8</rdfsuite:enum_elem>
  <rdfsuite:enum_elem>9-4</rdfsuite:enum_elem>
   </rdfsuite:Enumeration>
   </rdfs:range>
   </rdf:Property>
   </rdf:RDF>
```

Example of domain ontology

```
   <?xml version="1.0" ?>
<rdf:RDF xml:lang="en" xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
   syntax-ns#" xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:cult="file:C:\AuctionRDFSchema\culture.rdf#"
   xmlns:adm="file:C:\AuctionRDFSchema\admin.rdf#"
   xmlns:auc="file:C:\AuctionRDFSchema\auction.rdf#">
<cult:Cubist rdf:about="http://www.culture.net/picasso132">
<cult:paints>
<cult:Painting rdf:about="http://www.museum.es/guernica.jpg">
  <cult:technique>oil on canvas</cult:technique>
<cult:exhibited>
```

```
  <cult:Museum rdf:about="http://www.museum.es" />
    </cult:exhibited>
    </cult:Painting>
    </cult:paints>
<cult:paints>
<cult:Painting rdf:about="http://www.museum.es/woman.qti">
  <cult:technique>oil on canvas</cult:technique>
    </cult:Painting>
    </cult:paints>
  <cult:first_name>Pablo</cult:first_name>
  <cult:last_name>Picasso</cult:last_name>
    </cult:Cubist>
<adm:ExtResource rdf:about="http://www.museum.es">
  <adm:title>Reina Sofia Museum</adm:title>
  <adm:last_modified>2000-06-09T12:30:34Z</adm:last_modified>
    </adm:ExtResource>
<auc:Auction rdf:about="http://www.auctions.com/guernica_auction"
     auc:name="GuernicaAuction">
<auc:sells>
  <auc:ItemForSale rdf:about="http://www.museum.es/guernica.jpg" />
    </auc:sells>
<auc:has_type>
  <auc:English
     rdf:about="http://www.auctions.com/guernica_english_auction" />
    </auc:has_type>
  <auc:starting_time>2004-07-01T12:00:00Z</auc:starting_time>
  <auc:ending_time>2004-07-03T12:00:00Z</auc:ending_time>
  <auc:reserve_price>0</auc:reserve_price>
  <auc:bid_step>2</auc:bid_step>
  <auc:number_of_products>1</auc:number_of_products>
<auc:controled_by>
<auc:Broker rdf:about="http://www.auctions.com/guernica_broker">
<auc:accepts>
  <auc:Credit_Card
     rdf:about="http://www.auctions.com/guernica_broker_Credit_Card" />
    </auc:accepts>
<auc:accepts>
  <auc:Personal_Check
     rdf:about="http://www.auctions.com/guernica_broker_Personal_Check"
     />
    </auc:accepts>
<auc:delivers_by>
  <auc:US_Airmail
     rdf:about="http://www.auctions.com/guernica_broker_US_Airmail" />
    </auc:delivers_by>
    </auc:Broker>
    </auc:controled_by>
    </auc:Auction>
<auc:Auction rdf:about="http://www.auctions.com/woman_auction"
     auc:name="WomanAuction">
<auc:sells>
  <auc:ItemForSale rdf:about="http://www.museum.es/woman.qti" />
    </auc:sells>
<auc:has_type>
```

```
  <auc:English rdf:about="http://www.auctions.com/woman_english_auction"
     />
    </auc:has_type>
  <auc:starting_time>2004-07-01T12:00:00Z</auc:starting_time>
  <auc:ending_time>2004-07-03T12:00:00Z</auc:ending_time>
  <auc:reserve_price>0</auc:reserve_price>
  <auc:bid_step>2</auc:bid_step>
  <auc:number_of_products>1</auc:number_of_products>
<auc:controled_by>
<auc:Broker rdf:about="http://www.auctions.com/woman_broker">
<auc:accepts>
  <auc:Credit_Card
     rdf:about="http://www.auctions.com/woman_broker_Credit_Card" />
    </auc:accepts>
<auc:accepts>
  <auc:Personal_Check
     rdf:about="http://www.auctions.com/woman_broker_Personal_Check"/>
    </auc:accepts>
<auc:delivers_by>
  <auc:US_Airmail
     rdf:about="http://www.auctions.com/woman_broker_US_Airmail" />
    </auc:delivers_by>
    </auc:Broker>
    </auc:controled_by>
    </auc:Auction>
    </rdf:RDF>
```