ELSEVIER

Expert Systems with Applications 28 (2005) 223–235

Expert Systems with Applications

www.elsevier.com/locate/eswa

An integrated approach for developing e-commerce applications

Francisco García-Sánchez, Rafael Valencia-García, Rodrigo Martínez-Béjar*

Departamento de Ingeniería de la Información y las Comunicaciones, Facultad de Informática, Universidad de Murcia, 30071 Espinardo (Murcia), Spain

Abstract

This paper presents a framework that merges various advanced information technologies for developing electronic commerce (e-commerce) applications. The use of e-commerce utilities provides several advantages to businesses. Intelligent agents can be used to facilitate some tasks from those that take place in a commercial transaction moving to a second generation of e-commerce applications. We present a prototype which integrates a multiagent system (composed by buyer and seller agents) with a Web application.

Keywords: Electronic commerce; Intelligent agents; Natural language processing; Ontologies

1. Introduction

Several definitions have been given to the term 'e-commerce' or 'electronic commerce'. The one given by the Electronic Commerce Association[1] is: 'electronic commerce covers any form of business or administrative transaction or information exchange that is executed using any information and communication technology'. We limit our approach to covering commercial activities conducted on the Internet. E-commerce offers opportunities to dramatically improve the way that businesses interact with both their customers and their suppliers, that is, to make business negotiations faster, cheaper, more personalized, and/or more agile.

The number of web users who shop for or buy products online is continuously increasing (Silverman, Bachann, & Al-Akharas, 2001). However, searching and buying products via on-line can be frustrating due to the lack of help or decision support given to the user. Nowadays e-commerce applications are being improved from a first generation stage where buyers are humans who browse through a catalog of commodities (e.g. books, computer components, films) and make purchases, often by means of a credit card transaction, to a second generation with a greater degree of automation on both the buyer's and the seller's side (Wooldridge, 2002).

The aim of this work was to develop a technology for facilitating 'Business-to-Consumer' (B2C) and 'Business-to-Business' (B2B) processes. For it, various methodologies including (multi)agents (which allow for the interaction among several manufacturers), decision support (used for rating and differentiation processes between the products found) and natural language processing (NLP) have been used.

The previous objective can be decomposed into the following sub-objectives: (1) generation of a (computer) product ontology, (2) design of a database containing all relevant information about the products in question, (3) design and implementation of two agent models, one playing a buyer's role and another playing a seller's role, (4) integrating the agents into the Foundations for Intelligent Physical Agents-Open Source (FIPA-OS) framework so that communication among the agents registered at the platform may take place, and (5) design and implementation of a Web application that makes use of agents to offer e-commerce services (this will be the prototype).

The solution described in this work is built on top of an integrated system elaborated on topics belonging to a number of disciplines, including knowledge management, NLP, decision support theory and multiagent technology. The experiments done with the system up to now make us to believe that its future is promising in helping solve some problems we can find in actual e-commerce applications.

* Corresponding author. Tel.: +34 968 364634; fax: +34 968 364151.
E-mail addresses: fgs2@alu.um.es (F. García-Sánchez), valencia@um.es (R. Valencia-García), rodrigo@dif.um.es (R. Martínez-Béjar).

[1] http://www.theeca.org/.

0957-4174/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2004.10.004

The structure of this paper is as follows. Section 2 offers a brief description of the methodology applied. In Section 3, the e-commerce framework used in this work is introduced. Section 4 describes the system prototype developed making use of the framework presented. The validation of the system prototype is shown in Section 5. Finally, some conclusions are put forward in Section 6.

2. Methodology

In this work, we have applied a methodology that integrates different advanced information technologies with the aim of building a framework for developing e-commerce applications. These are: intelligent agents, which have been used to represent the different entities we find in a transaction process (buyers/sellers); decision support systems (DSSs) theory, applied to help buyers in some decision processes; ontologies that establish a reusable, shareable semantics for the terminology, so that different types of agents are able to communicate among themselves; and NLP, which permits to better determine the users desires (since the users can use their own terminology). In the following lines, a brief description of each of them is realised.

2.1. Multiagent systems

A multiagent system is seen as a system that consists of a group of agents that can potentially interact with each other (Vlassis, 2003). The term *agent* is defined as follows (Wooldridge, 2002): "an agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives". For this author, an agent is intelligent if it fulfils the following properties: reactivity (it responds in a timely fashion when needed), pro-activeness (it satisfies internal goals; it takes actions when it seems it will be useful) and social ability (it interacts with other agents in order to satisfy goals). In order to design and construct agents it will be necessary to take the following points into account: (i) *Agents theory* (formal specification that describes what properties the agents must fulfil), (ii) *Agent languages* (tools for the design and construction of agent-based systems, e.g.: agent-oriented programming—Agent0—, concurrent METATEM) and, (iii) *Agent architectures* (agents' internal structure which can be logic-based, reactive, Belief-Desire-Intention-based or layered).

Agents can be useful as stand-alone entities that are delegated particular tasks on behalf of a user. However, in the majority of cases agents exist in environments that contain other agents. In these multiagent systems (MASs), the global behaviour derives from the interaction among the constituent agents. There are two main classes of MASs (Wooldridge, 2002): *distributed problem solving systems*, in which the component agents are explicitly designed to cooperatively achieve a given goal; and *open systems*, whose agents are not co-designed to share a common goal, and have been possibly developed by different people to achieve possibly different objectives. Moreover, the composition of the system can dynamically vary as agents enter and leave the system.

When a group of individual agents form a MAS, the presence of a mechanism to coordinate such a group as well as a communication language becomes necessary. Among the coordination mechanisms the cases of agents having common objectives (and, therefore, they cooperate) can be distinguished from those where agents are self-interested and have conflictive objectives with other agents. For the latter case we will need negotiation. The most usual cooperation mechanisms are: organizational structures, multiagent planning—centralized and distributed—, contract nets and functionally exact cooperation. Among the ones belonging to negotiation we may highlight coalitions formation, market mechanism, the bargaining theory, vote, auctions and task assignment between two agents.

Agent communication languages allow agents to interact each other while they hide their internal work details by exchanging information and knowledge. Examples of those ones are KIF (Knowledge Interchange Format), KQML (Knowledge Query and Manipulation Language) and FIPA-ACL (Foundations for Intelligent Physical Agents-Agents Communication Language).

2.1.1. Multiagent platforms

For this project, we have employed available implementations of agent platforms which conform to FIPA (The Foundations for Intelligent Physical Agents) Specifications, namely, a group of normative rules that permit an agent society to operate among themselves. This model identifies some necessary agent's roles for the platform management: the AMS (Agent Management System), the DF (Directory Facilitator), the ACC (Agent Communication Channel), the IPMT (Internal Platform Message Transport) and the IIOP (Internet Interface Object Protocol) (FIPA-OS Developers Guide, 2001).

The communication language among agents that FIPA proposes is FIPA-ACL, which specifies a standard message passing language, establishing codification, semantics and message pragmatics.

Finally, FIPA Standard includes the non-agent software integration specification, the agents mobility and security, the ontology service, and the human-agent interaction.

Amongst the frameworks that satisfy the FIPA Standard, the more popular are ZEUS, JADE and FIPA-OS. We have selected this last one in this work. FIPA-OS is an agent framework developed with the aim of building heterogeneous agent platforms, agents and services that adjust to FIPA Standard. In addition to the advantage to be an open system in terms of license of use and extensibility, like

ZEUS and JADE, FIPA-OS presents some additional advantages:

- ☐ Interoperability with other FIPA agents platforms not developed with FIPA-OS.
- ☐ It allows for agents heterogeneity, that is, agents implemented with different programming languages.
- ☐ It supports FIPA specification for agents management.
- ☐ It provides abstractions and application program interfaces (APIs) that permit extending and integrating an agents's platform with other already existent software platforms.

2.1.2. Agents for electronic commerce

According to Wooldridge (2002) agents make it possible the second-generation e-commerce systems, in which many aspects of a consumer's buying behaviour is automated.

One phased model that attempts to describe consumer-buying behaviour is the following (Guttman, Moukas, & Maes, 1998):

1. *Need identification*. This stage characterizes the consumer becoming aware of some need that is not satisfied.
2. *Product brokering*. A would-be consumer obtains information related to available products in order to determine what product to buy.
3. *Merchant brokering*. The consumer selects the supplier. This stage will typically involve examining offers from a range of different merchants.
4. *Negotiation*. The terms of the transaction are agreed between the would-be consumer and the would-be merchant. In some markets, the negotiation stage is empty—the terms of agreement are fixed and not negotiable.
5. *Purchase and delivery*. The transaction is actually carried out, and the good delivered.
6. *Product service and evaluation*. The post-purchase stage involves product service, customer service, etc.

Agents have been widely promoted as being able to (partially) automate some of these stages.

2.2. Decision support systems (DSS)

"In the rush to open their website, e-commerce sites too often fail to support buyer decision making and search, resulting in a loss of sale and the customer's repeat business" (Silverman et al., 2001). DSSs cover a wide variety of systems, tools and technologies that support decision-making in semi-structured and unstructured situations where no one knows exactly how the decision should be made. It provides information, models, and data manipulation tools to solve the structured parts of the problem and help isolating places where both judgment and experience are required.

As Silverman remarks, in online shopping sites buyers have decisions to work out and tasks to perform that can be facilitated by means of DSS software. We help clients with part of this process using a natural language system (based on a morphological analysis) that allows the system to find out the user's desires. Besides, a sorting process helps buyers to determine the most convenient product for them.

2.3. Ontologies

Defining a common vocabulary (apart from the communication language selected) is necessary for the agents to communicate with one another. Thus, the communication is facilitated through one ontology that defines the concepts and the relations between the concepts of a particular domain.

With all, an ontology is viewed in this work as a specification of a domain knowledge conceptualisation (Van Heijst, Schreiber, & Wielinga, 1997), and is represented through a set of concepts, while concepts are defined through sets of both attributes and interconceptual relations. So, the main ontological entity in the system developed is the concept but the use of the other ontological entities such as attributes is also possible in the model in order to provide the system with powerful representational capabilities.

2.4. Natural language processing (NLP)

In the approach presented in this work, the methodology presented in Valencia-García, Ruiz-Sánchez, Vivancos-Vicente, Fernández-Breis, and Martínez-Béjar (2004) has been used to get knowledge from text. This methodology, which uses ontologies and one incremental knowledge acquisition technique termed MCRDR (Kang, 1996), is based on the idea that relationships between concepts are usually associated to verbs in natural language. This methodology uses the mentioned technique and the grammar category of the words in the current sentence to infer other knowledge entities (e.g. concept, attributes and values) in order to create an ontology from a text fragment.

3. A framework for developing e-commerce applications

3.1. Overview of the system

The system architecture, which is shown in Fig. 1, has four principal interconnected components. The most important component is the multiagent platform, where two agent types have been implemented, namely a buyer and a seller. These agents communicate with one another by means of ACL (Agent Communication Language) messages. These agents must register in the Agent Platform (AP), which specifies several types of agents that can facilitate the multiagent communication. In particular, the two Agent Platform types that will be necessary to get registered in are: the *Directory Facilitator* (DF), which provides a yellow pages service to

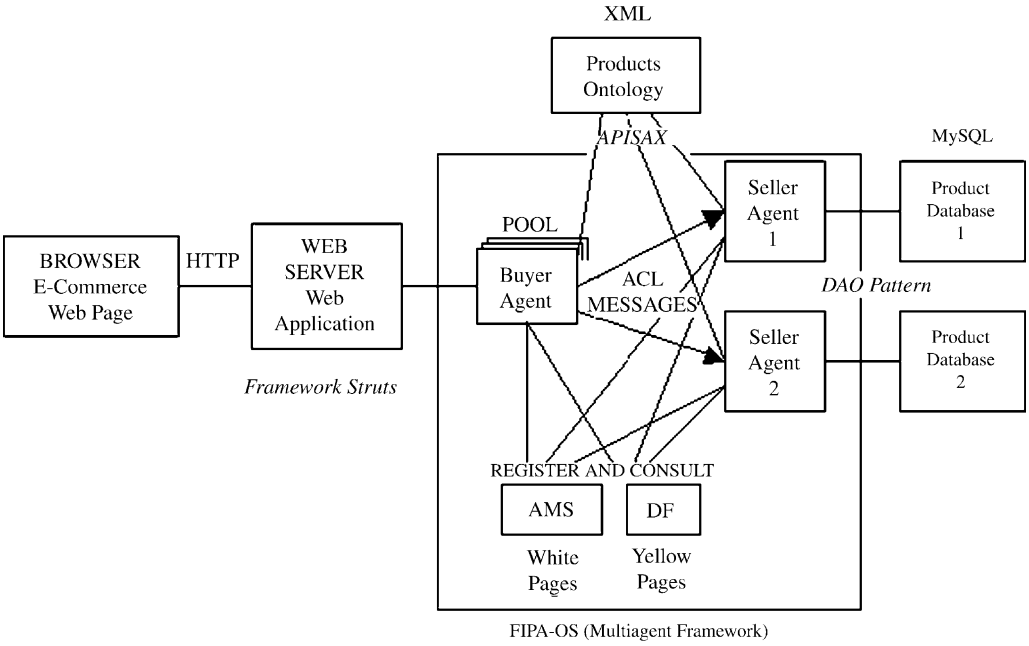

Fig. 1. Framework general structure.

the other agents, and the *Agent Management System* (AMS), which provides platform-typical management functions (such as life cycle monitoring, checking of all the entities' correct behaviour and 'white pages'). The buyer agent functions include to get external petitions, process them and return the obtained results. The petition processing may include NLP, product ontology matching, planning, plan execution and answer composition. Regarding the seller agent, this has been provided only with communication services with the buyer agents, so that agent is able to answer for two petitions types: questions about a particular type of product and orders. It can also communicate with a product database, which contains the product catalog of a manufacturer.

The second system component is the database (one for each seller agent), which can be shared by the agents. The database can be acceded by the seller agents using the Data Access Object (DAO) pattern. The database is composed by the products a manufacturer wants to sell, so that the database contents are ontology-dependent.

Another important component of the system is the product ontology. The ontology is saved in an XML file and represents the specific relevant domain knowledge we are dealing with, establishing a common set of words that allows different types of agents to communicate one another.

The last main component is the Web application, which employs the FIPA-OS platform together with the above-referred agents. For the application design, we have made use of the Struts framework and Java technology.

In the following paragraphs, all these components are detailed more precisely.

3.2. FIPA-OS and intelligent agents

According to FIPA-OS Developers Guide (2001), FIPA-OS is a component-orientated toolkit for constructing FIPA compliant Agents using mandatory components (i.e. components required by all FIPA-OS Agents to execute), components with switchable implementations, and optional components (i.e. components that a FIPA-OS Agent can optionally use) (see Fig. 2).

Agent shell (FIPAOSAgent). It provides a shell for Agent implementation to use by simply extending the FIPAOSAgent class. The FIPAOSAgent shell is responsible for loading an agent's profile, and initialising the other components which the Agent is composed by.

Task manager. It provides the ability to split the functionality of an agent into smaller, disjoint units of work known as tasks. These can be defined as self-contained pieces of code that carry out some task and (optionally) return a result, have the ability to send and receive messages, and have little or preferably no dependence on the agent which they are executed within.

Conversation manager. It provides the ability to keep track of the conversation state at the performative level, as well as a mechanism for grouping messages of the same conversation together.

Message transport service. It provides the ability to send/receive messages to/from an agent implementation.

3.2.1. The buyer agent

The buyer agent's target is to look for the products that a customer requests for (starting from a sentence) as well as asking for a concrete product.

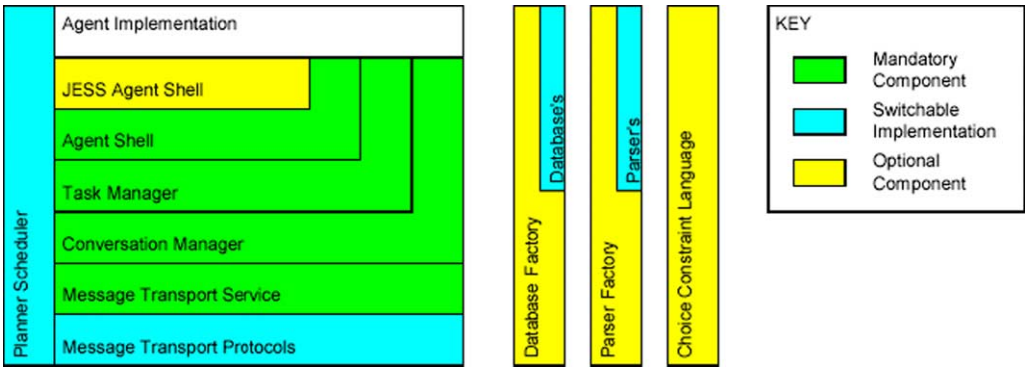

Fig. 2. Components within FIPA-OS.

Agent initialisation. It involves four main tasks. First, it is needed to initialise the NLP tool. Second, the agent must be registered in the FIPA Agent Platform (AMS and DF Agents). Third, messages buffers must be activated before conversations take place. Finally, the agent has to get all products categories from the ontology (main concepts, synonymous, attributes, relationships,…); the agent will use this information when that one communicates with other agents.

Products search. This process begins when a customer (i.e. the application user) writes a sentence denoting the product that (s)he wants to get, and finishes when the system returns the products matching what the user wants. In order to do this, it is necessary a planner that determines the steps that the agent must follow. For it, we have used an STRIPS-type planner and some operators (see Section 3.2.3 for details).

When the system has the plan, it is necessary to execute it, so that the possible steps then are the following:

- *Sentence analysis*. Through this step, the system obtains the concepts underlying the sentence introduced by the customer. For it, the sentences analyser referred to before in this paper is used.
- *Category obtaining*. The system determines the product categories that the customer is looking for. For this, the system compares the concepts found in the sentence with the categories extracted from the product ontology. If the system does not find any category that matches any of the extracted concepts, then the plan execution is stopped.
- *Manufacturer finding*. This is executed thanks to a task included in the agent's implementation. The task target is to look for seller agents in the 'yellow pages'. For it, the system incorporates the Directory Facilitator Agent—DF—, which returns a collection with all the identifiers (ID) of the agents found. In order to match this ID with the agent transport address, the system uses the Agent Management System (AMS), i.e. white pages. If the DF does not find any agent, then the plan execution stops.
- *Information request*. For each category and seller agent found, a task is created that starts a conversation with the found agent and returns all products found grouped by seller agent. The information returned has the following format (Fig. 3).Where 'Cat-*i*' represents the *i*th category, 'Man-*j*' the *j*th manufacturer and 'Prod-*ij*' is a product of the *i*th category offered by the *j*th manufacturer.For each category, the system will return all the products (and its respective manufacturers) found with respect to that category.It is important to emphasize that each seller agent works in parallel to find the products of each category. The buyer agent saves this information then and asks for the next category.
- *Offer selection*. Once a buyer agent has the information described before, the only remaining thing it has to do is to sort the products of each category.The result of this process is a collection with the following format (Fig. 4).Where 'BestPr' represents the best product for the category under question, 'ManBP' represents the manufacturer for such a product, '2ndPr' represent the second best product for the category, 'Man2P' is the manufacturer for the mentioned product, and so on.This will be returned to the customer.

Asking for a product. In order to execute a product petition, a search must have taken place previously, because it is necessary to indicate the seller agent ID, in addition to other product-related features such as the amount of products, the product type and the product code.

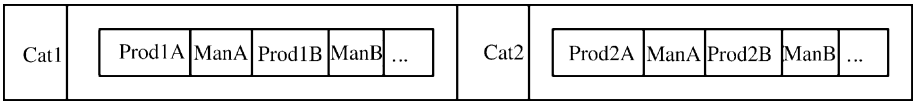

Fig. 3. Message format 1 for information request.

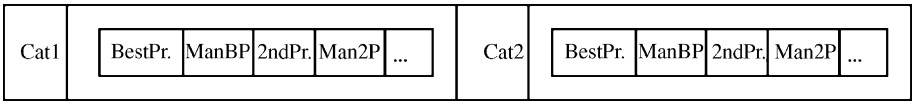

Fig. 4. Message format 2 for offer selection.

The task to attend the request hence needs the product code (this must agree with the one that the seller agent has registered), the product type or category, the AgentID of the seller agent that offers the product and the number of units the customer requests. All this information is then sent to the seller through an ACL message.

3.2.2. The seller agent

The seller agent is in charge of performing three tasks: one consists of attending petitions and determining which task to execute; the second is to keep the communication with the buyer agent which requests some products, and the last carries out the petition record of a given product. Therefore, this agent's main objective is to receive petitions and questions from different buyer agents.

Agent initialisation. This process is similar to the agent initialisation process in the buyer agent. This agent does not use NLP neither planning.

Search task. This task is executed when a 'Query'-type message is received. The aim of this task is to return all the products found in the database with the category indicated in the query.

Petition task. This task is called when a message 'Petition' is received. The function of this task is to insert a record of the product sale into the database. This task can check whether the product is available in stock or it is necessary to make a petition to the providers.

3.2.3. The planner

STRIPS (Fikes & Nilsson, 1971) is a linear planner that attempts to find a sequence of operators in a space of world models to transform a given initial world model into a model in which a given goal formula can be proven true. It represents a world model as an arbitrary collection of first-order predicate calculus formulae and works with models consisting of a large number of formulae. It uses a resolution theorem prover to answer questions of particular models and mean-ends analysis as a guide towards the desired goal-satisfying model.

For every world model, we have a set of applicable operators, which transform the world model into another world model. The problem solver finds some composition of operators that transforms a given initial world model into one that satisfies a stated goal condition.

In STRIPS, a world model is represented by a set of well formed first-order predicate calculus formulae. Each operator in a solution corresponds to an action routine whose execution causes a robot to take certain actions.

Each operator in STRIPS is composed by:

- A set of preconditions. To execute the action related to the operator it is necessary that preconditions are true before the operator can be applied.
- Delete list, which is a set of formulae that will not be true after the operator has been applied (so that the planner has to delete them from the current world model).
- Add list, which is a set of formulae that will be true after the operator has been applied (so that the planner has to add them to the current world model).

For example, an operator for the Blocks World problem is shown in Table 1.

In Table 1, *clear(block)*, *on-table (block)*, *arm-empty* and *holding(block)* are well formed formulae.

STRIPS, like most of planners, has been applied to the blocks world problem as an effective benchmark (Slaney & Thiébaux, 2001). The blocks world problem consists of a finite number of blocks stacked into towers on a table large enough to hold them all. The towers' positioning on the table is irrelevant. The Blocks World planning problem is to turn a blocks initial state into a goal state, by moving one block at a time from the top of a tower onto another tower or to the table. The optimal Block World planning problem consist in doing so in a minimal number of moves.

Most of planners adopt the STRIPS representation and search forward the state space. The GRT planner (Refanidis & Vlahavas, 2001) is a domain-independent heuristic planner, which solves planning problems calculating in a first, phase the distances between the facts and the goals of the problem. The second phase consists of a simple best first search strategy using the distances calculated in the previous phase. This planner has been validated in several domains like blocks-world domain, in which GRT can easily solve problems with more than 20 blocks.

Another planner, which adopts the STRIPS representation is the Fast-Forward Planning System (Hoffmann, 2001), which attacks the planning problems by forward

Table 1
An operator for the blocks world problem

Pick-up(block)	
Preconditions	Clear(block)
	On-table(block)
	Arm-empty
Add list	Holding(block)
Delete list	On-table(block)
	Clear(block)
	Arm-empty

Table 2
The set of well formed formulae

Formula	Description
PREFERENCE(S)	The customer's desires contained in the sentence S
CONCEPT(CO, S)	The concept CO in S
CATEGORY(C, CO, S)	The category C is one of the customer's preferences related to CO in S
MANUFACTURER(MA)	The manufacturer MA is registered in the system, therefore, it is available for possible petitions
INFO(OF, MA, C)	The manufacturer MA presents the offer OF for C
SORT(OF)	The offer's group is sorted out attending to the relation quality-price

searching in the state space, guided by a heuristic function. This function is extracted from the domain description relaxing the planning problem by ignoring parts of its specification, concretely the delete lists of operators.

In this work, we have used a STRIPS planner for allowing agents to determine dynamically the actions to be performed. More precisely, this planner has been implemented into the buyer agent to accomplish the product search task. The well formed formulae of first-order predicate calculus used in this planner are showed in Table 2.

The initial world model is formed by the formulae which represent the user's preferences. The formulae of the final world model represent an ordered set containing the products that the customer may be interested in (according to his/her likes and dislikes).

The operators which have been designed can be viewed in Table 3.

3.2.4. Natural language processing

The NLP application used in this approach is based on the work presented in Valencia-García et al. (2004), where the authors described a software tool for ontology building from natural language texts. The principal idea sustaining our approach is that in natural language relationships between knowledge entities are usually associated to verbs.

The knowledge acquisition process is divided into three sequential phases, which have been implemented into three separate modules: POS-tagging, Concept search and inference.

POS-tagging. The main objective of this process is to obtain the grammatical category of each word in the current sentence. For this purpose, the POS-tagger described in Ruiz-Sánchez, Valencia-García, Fernández-Breis, Martínez-Béjar, and Compton (2003) is used. This is the first subprocess to be carried out.

Concept search. Through this process, linguistic expressions representing concepts are identified. The associations between linguistic expressions and concepts have been stored in the conceptual knowledge base in a previous training of the system. This process is quite simple and as a result we obtain all the expressions of the fragment which are already in the conceptual knowledge base.

Inference. In natural language, relationships between concepts are usually associated to verbs. Although the subprocess in which MCRDR acts is mainly concerned with obtaining relationships between concepts, it can also be used to get other knowledge categories like concepts, attributes, or values. This MCRDR component is formed by a knowledge-base that contains linguistic expressions representing generic conceptual relationships, and by an MCRDR subsystem that infers the participants in these relationships. We describe next the modus operandi of this process. Firstly, the verb in the current sentence is identified. Then, the user (of the system if the knowledge base is not empty) searches for the type of semantic relationship associated to that verb. Once the type of relation associated to the main verb in the current sentence has been found, the MCRDR sub-system is applied to extract knowledge by

Table 3
The STRIPS operators

Operator			Action
FINDMANUFAC-TURER( )	Preconditions		Return all the manufacturer registered
	Delete list		
	Add list	MANUFACTURER(MA)	
ANALYZE(F)	Preconditions	PREFERENCE(S)	Obtain the relevant concepts within a sentence
		MANUFACTURER(MA)	
	Delete list	MANUFACTURER(MA)	
	Add list	CONCEPT(CO, S)	
GETCATEGORY(CO)	Preconditions	CONCEPT(CO,S)	Try to match a concept with the set of product categories
	Delete list	CONCEPT(CO,S)	
	Add list	CATEGORY(C, CO, S)	
REQUESTINFO(C)	Preconditions	MANUFACTURER(MA)	Make a request to a manufacturer about a category
		CATEGORY(C, CO, S)	
	Delete list	CATEGORY(C, CO, S)	
	Add list	INFO(OF, MA, C)	
SELECTOFFER(OF)	Preconditions	INFO(OF, MA, C)	Sort a set of offers related to the same category
	Delete list	INFO(OF, MA, C)	
	Add list	SORT(OF)	

means of the grammatical category of the words, their position in the current sentence, and the type of relation associated to the verb, if any.

The inputs of this tool are the product queries, in natural language, introduced by the customers. An example of these queries could be: "I need a 20 GB Hard Disk". Then, the system would obtain, for instance, an ontology formed by the concept 'HARD DISK', so that the attribute 'Capacity' of this concept is '20 GB'.

As it has been indicated before, the NLP system needs a previous training in the domain. This training is based on the study of a set of texts in natural language related to the domain. In our case, this domain has been a hardware domain.

3.3. Product ontology

In order to maintain a conversation, agents must have a common language. This is established by means of an ontology, which contains the main concepts owning to the domain we are dealing with. In addition to this information, the ontology also includes attributes, values, relations between concepts and axioms so that consistency checking and inferences are done.

These concepts are the ones the buyer agent utilizes to indicate the customer's preferences to the seller agent and the ones the seller agent utilizes to access the database.

An (incomplete) example of a product ontology related to the computer products domain is shown in Fig. 5.

In this example, ontology shows how a computer is composed by several elements: monitor, keyboard, mouse, motherboard, processor, etc.

3.3.1. Ontology editor

To specify the product ontology, a graphic tool prototype has been implemented that allows to introduce concepts, attributes and semantic relations in a intuitive way. It also generates an XML document with the designed ontology's description.

The XML file has the following characteristics:

- Concept:

```
<concept comment="" name="COMPUTER">
<alternative-names>
<name>WORKSTATION</name>
<name>PC</name>
<name>LAPTOP<name>
</alternative-names>
<specific-attributes>
<attribute comment="" name="TRADE-
MARK" type="STRING"></attribute>
<attribute comment="" name="MODEL"
type="STRING"></attribute>
<attribute comment="" name="DESCRIP-
TION" type="STRING"></attribute>
<attribute comment="" name="GUAR-
ANTY" type="INTEGER"></attribute>
<attribute comment="" name="PRICE"
type="INTEGER"></attribute>
</specific-attributes>
</concept>
```

It can be noticed that every concept, besides the 'main name', has also alternative names (that we will use like synonyms), in addition to its attributes.

- Relation:

```
<relation name="COMPONENT_OBJECT">
<concept_name>PROCESSOR<
/concept_name>
<concept_name>COMPUTER<
/concept_name>
<relation_type>
<property>NonSymmetry</property>
</relation_type>
</relation>
```

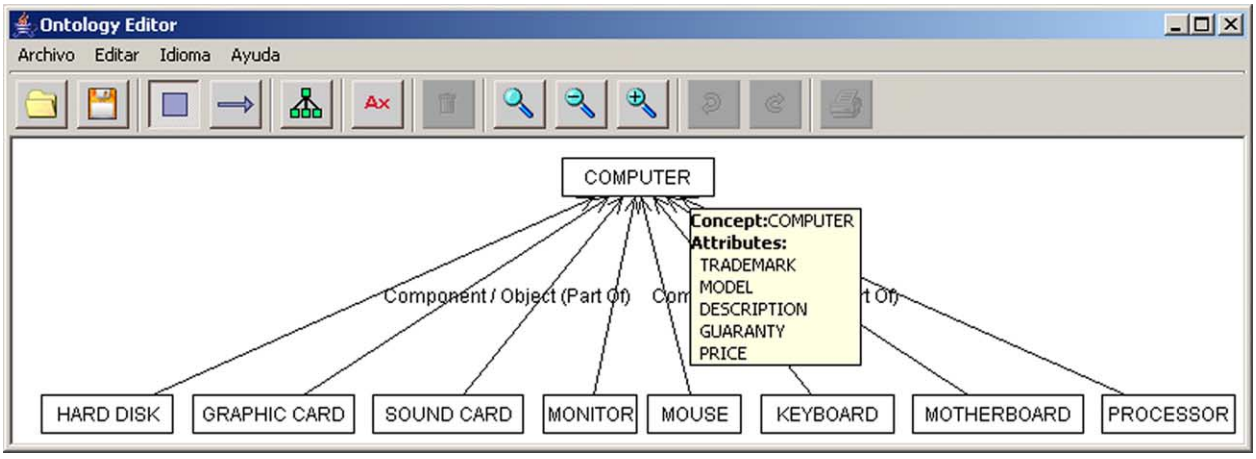

Fig. 5. Computer products ontology.

In this case, in addition to the concepts taking part in the semantic relation under question, the relation will have a name with the relation type and eventually some other properties associated to that relation.

3.3.2. The analyser

There are two possible parsers. The API SAX accomplishes a sequential access to the document and provides an event based programming model (callbacks). On the other hand, the API DOM makes a tree structure from the document. The main advantages of API SAX are its simplicity and that it consumes less time of processing. So, we have chosen this option because we do not need to update the document and the API SAX is a more simple solution to do the queries in the ontology. Thus, every agent only needs to do one document analysis.

The buyer agent, once the concepts of the customer sentence have been found, should compare these concepts with that found in the ontology in such a way that, if they match, we understand that it is a product the customer wants to buy.

3.4. Product database

The product each manufacturer possesses has been stored in a database. For it, MySQL Database Server has been chosen because, apart from being Open Source (uses the GPL—GNU General Public License), it is very fast, reliable, and easy to use MySQL Reference Manual (2002). Besides, the MySQL Database Software is a client/server system that consists of a multi-threaded SQL server that supports different backends, several different client programs and libraries, administrative tools and a wide range of application programming interfaces (APIs).

A JDBC–ODBC bridge driver has been utilized to access the database from Java due to its efficiently using native database client code. Besides, the drive may be changed easily in the Web application description file.

In order to add flexibility to the database management the DAO pattern has been used, so that persistent information from different sources (relational databases, LDAP, XML, etc.) can be stored and recovered. The benefits from utilizing this solution are the following: transparency-oriented, easier component migration, complexity reduction and centralization of the access to data through a single layer.

We also use a connection pool to improve the performance (see Fig. 6).

The general structure of the DAO pattern is shown in Fig. 7.

3.5. Web application

With the purpose of applying the designed agents in a real environment, the FIPA-OS framework has been integrated into a Web application.

3.5.1. Agents integration

We have to distinguish three subjects: the *AgentLoader* creation, the buyer agent creation and the seller agent creation.

AgentLoader creation. The environment initialisation is needed in order to start working with FIPA-OS. This is done by 'Loader', which reads configuration files and creates the AMS and DF agents (those that form the platform kernel).

The *AgentLoader* must be created by the *FrontController Servlet* (see Section 3.5.2) at the application initialisation time.

The buyer agent creation. Each customer has a buyer agent associated to him/her. When a customer starts to use the application, a buyer agent is created. The problem observed then was that the agent creation process is very slowly.

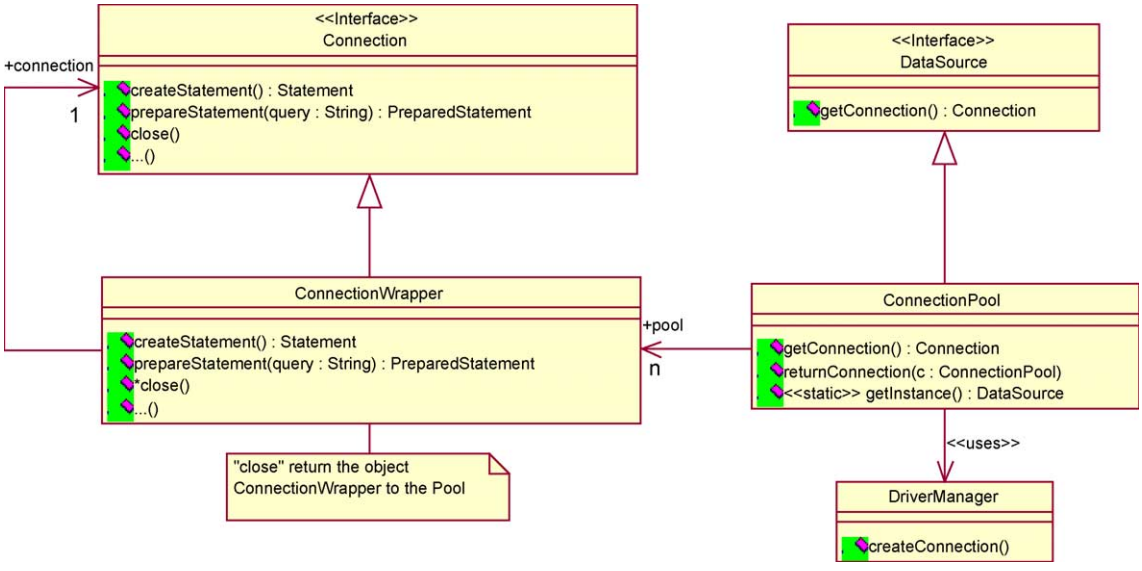

Fig. 6. Connection pool implementation.

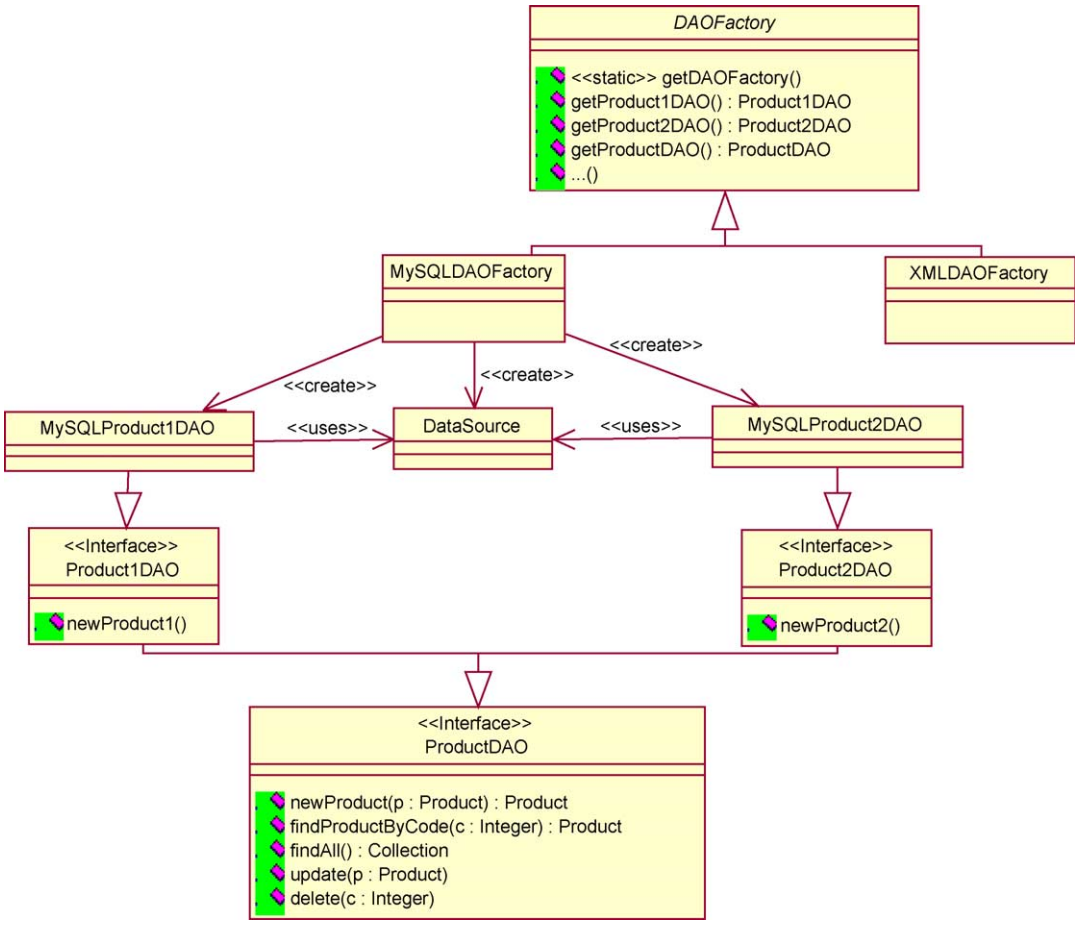

Fig. 7. DAO pattern implementation.

The solution provided for that problem was to use the connection pool's idea commented before for the database, so that when there are not enough agents to associate to customers new agents are created.

The seller agent creation. For each manufacturer, we will have previously set up a database containing the stock together with a seller agent associated to it and registered in the platform.

In this way, the customer through his buyer agent will have access to the stock of a large number of manufacturers being able to choose a better product.

3.5.2. Application design

The design is based on the *Apache Struts Web Application Framework*. In this way, we only have one Servlet (*FrontController*) where all petitions go to. This Servlet studies every petition and executes the action that corresponds to the petition.

4. The system prototype

A software tool based on the approach described in this work has been designed and implemented for facilitating e-commerce. This is a Web application implemented with Java technology and designed following Struts recommendations. The application domain on which it has been applied is computer hardware.

The user has been provided with two ways to introduce a query. On the one hand, it is possible to write free text with the user's desires and, on the other hand, the user can select specific product categories from those indicated in Fig. 8.

We have trained the system with some sentences a user can write to ask for a hardware component. In this process, when we introduce a sentence to the tool, we have to establish the concepts, attributes, values and relations this sentence contains. For example, if we say *I want a 60 GB hard disk*, we will have to indicate that there is a concept, 'hard disk', with an attribute, Capacity, which value in this example is '60 GB'. Once the tool have learnt this, if we introduce a new sentence like *I want a 256 MB memory*, the tool will automatically determine that there is a concept 'memory', with the attribute Capacity which value is '256 MB'.

Once the user has entered information into the system about the wanted products, the system creates a set with the product categories that match the user wishes. Once the communication with the seller agents is established,

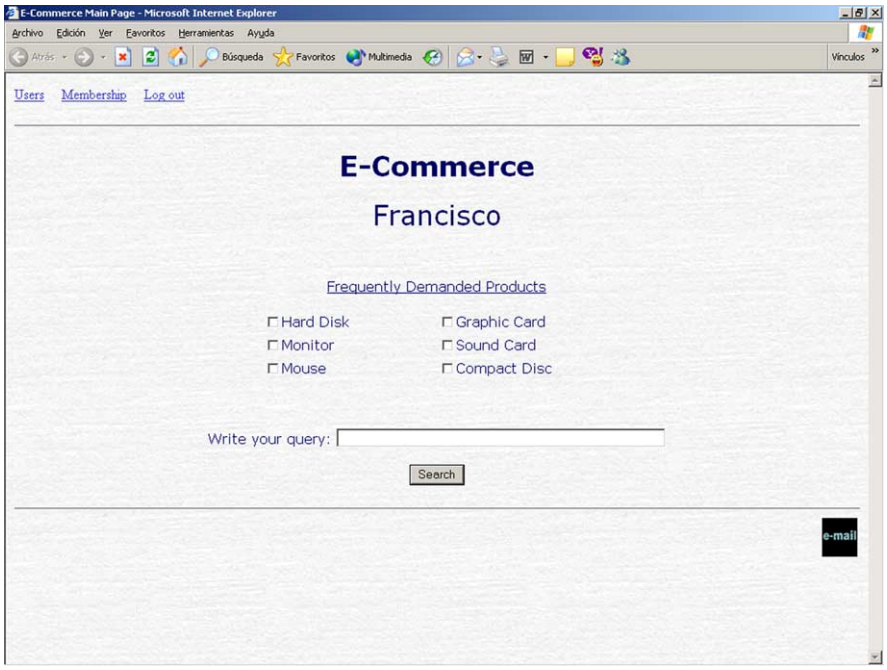

Fig. 8. Screenshot showing product categories.

the application shows a page with the best products of each category. Then, the user is shown a list of all the products related to a found category sorted by an internal relation quality-prize.

Finally, for each product the user is interested in (according to the information supplied by her/him to the system), the user will be able to ask for the amount (s)he wants (see Fig. 9). Internally, this request will be addressed to the seller agent that represents the manufacturer which has the product under question in its catalog.

5. Validation of the system

The system described in previous sections has been employed by eight users, who were given two hardware

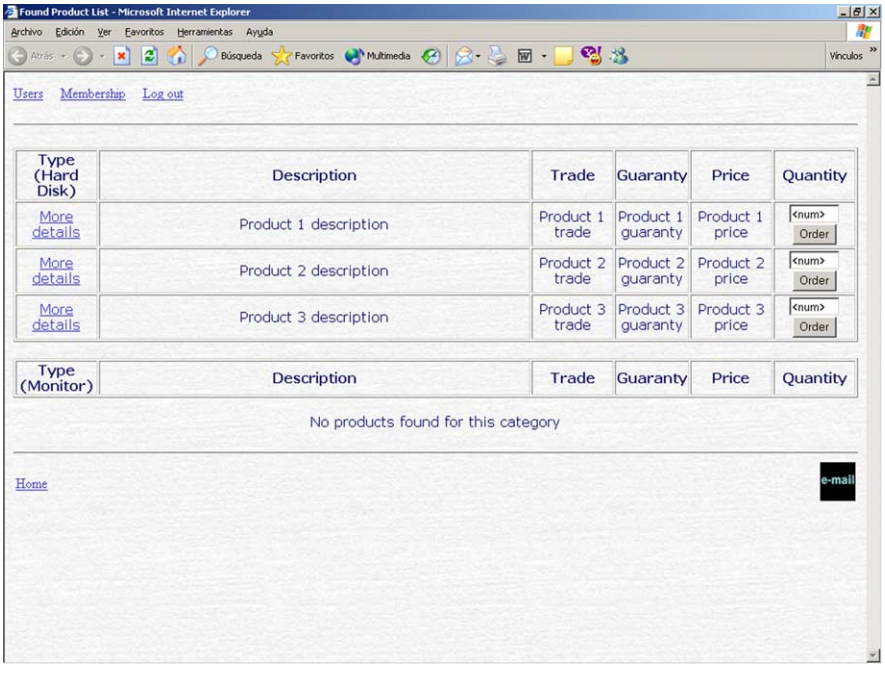

Fig. 9. List of products sorted by categories.

Table 4
System scoring

Question	Score (by user)							
	U1	*U2*	*U3*	*U4*	*U5*	*U6*	*U7*	*U8*
Previous experience in suchlike application	0	5	8	6	7	4	1	3
Interface	7	6	6	3	5	5	6	5
Performance	–	6	5	3	4	8	6	5
Application utility	8	8	–	6	5	8	7	8
Easy use	9	7	8	6	7	9	9	8
Agree with solution	7	6	6	5	5	8	7	6
Utility (Do they think new characteristics are important (DSS, NLP,…?))	9	6	7	7	–	10	9	8

product catalogs. Each catalog had products belonging to seven categories: hard disk, mouse, graphic card, sound card, monitor, motherboard and CD player. Moreover, each catalog possessed three items.

The mode the system was deployed is the following. In one computer, we installed the Web application and one of the two seller agents (the one which accesses to its catalog). On the other hand, in a distinct computer we set up then the seller agent to make queries on the another available catalog.

The users were asked to use the system during half an hour and to make at least 10 requests to the system. Utilities like search by all product categories to browse between products in an easy way were assessed as important for them.

The main problem found by them was the performance (caused by the time consumed by the NLP module). They all think that the interface is intuitive but too simple and agree in the fact that the functionalities added to the e-commerce application are positive.

Finally, the goodness of solution was well assed for most of them: they think that it is necessary to take additional attributes into account (price, brand, product characteristics, etc.) in order to minimize the space of search. These results are summarized in Table 4.

6. Discussion and conclusion

The results presented in this paper contribute to the development of intelligent electronic-commerce applications that are capable of simulating real world transactions. Concerning related work, a similar solution to ours can be found in Silverman et al. (2001). However, while in our solution we split the functionality into two different agent roles (buyer and seller), the quoted author makes use of an unique search agent that integrates all the functionality.

Using intelligent agent technologies, NLP, decision support systems, ontologies and web technologies we have made a system that facilitates B2C (business-to-consumer) and B2B (business-to-business) transactions.

The agents have been designed and implemented within a multiagent framework, FIPA-OS, that follow a broad standard such as FIPA (Foundation for Intelligent Physical Agents). It allows to develop a distributed application in a very simple way. All this makes the development process more realistic, simulating the interaction between a store customer and the seller, which attempts to help the buyer to find the product that (s)he is looking for. At the same time, the solution presented in this paper also gives the user an object pre-classification that supports the user decision.

The system developed has made use of advanced technologies to build an intelligent application but there are some aspects that we have not taken into account such as security, auctions, efficiency, etc. In particular, a few interesting issues should be explored and improved in future system improvements. First, a more complete ontology (i.e. with more knowledge) must be created. The application must be able to take into account and to use all the knowledge contained in the ontology (axioms, attributes, etc.). Second, it is important to better train the NLP tool allowing more free text, so that the application be able to understand the customer's wishes and to do a better search. Third, in the current approach the agents must follow a rigid plan created by an STRIPS-like planner. We can use a more flexible planner, which uses more knowledge about the environments and creates plans that take into account the possible contingencies that could take place. Fourth, we can make the application totally product-independent. This implies the intensive use of the ontology in all application fields. Fifth, both the buyer user and the seller user might be given the possibility of configuring their respective agents. So, a buyer agent could be configured with the buyer user's preferences (e.g. the cheapest product, or the one with more quality,…) and something similar with the seller agent. Finally, it could be interesting to use XML messages for the agents to communicate with one another, as a number of manufacturers currently do.

Acknowledgements

We thank the Spanish Ministry for Science and Technology for its support for the development of

the system through projects TIC2002-03879, FIT-110100-2003-73 and FIT-150500-2003-503; the Regional Government of Murcia (Spain) through project 2I03SIU0039; and Seneca Foundation through project PI-16/0085/FS/01. We also thank the European Commission for its support under project ALFA II0092FA.

References

Fikes, R. E., & Nilsson, N. J. (1971). Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, *2*, 189–208.

FIPA-OS Developers Guide (2001). http://fipa-os.sourceforge.net/, February.

Guttman, R. H., Moukas, A. G., & Maes, P. (1998). Agent-mediated electronic commerce: A survey. *The Knowledge Engineering Review*, *13*(2), 147–159.

Hoffmann, J. (2001). The fast-forward planning system. *AI Magazine,* 57–61.

Kang, B. (1996). *Multiple classification ripple down rules*. PhD Thesis, University of New South Wales.

MySQL Reference Manual (2002). http://www.mysql.com/.

Refanidis, I., & Vlahavas, I. (2001). The GRT planner. *AI Magazine,* 63–65.

Ruiz-Sánchez, J. M., Valencia-García, R., Fernández-Breis, J. T., Martínez Béjar, R., & Compton, P. (2003). An approach for incremental knowledge acquisition from text. *Expert Systems with Applications*, *25*, 77–86.

Silverman, B. G., Bachann, M., & Al-Akharas, K. (2001). Implications of buyer decisión theory for design of e-commerce websites. *International Journal of Human Computer Studies*, *55*, 815–844.

Slaney, J., & Thiébaux, S. (2001). Blocks World revisited. *Artificial Intelligence*, *125*, 119–153.

Valencia-García, R., Ruiz-Sánchez, J. M., Vivancos-Vicente, P. J., Fernández-Breis, J. T., & Martínez-Béjar, R. (2004). An incremental approach for discovering medical knowledge from texts. *Expert Systems with Applications*, *26*(3), 291–299.

Van Heijst, G., Schreiber, A. T., & Wielinga, B. J. (1997). Using explicit ontologies in KBS development. *International Journal of Human Computer Studies*, *45*, 183–292.

Vlassis, N. (2003). *A concise introduction to multiagent systems and distributed AI*. http://carol.science.uva.nl/~vlassis/cimasdai/.

Wooldridge, M. (2002). *An introduction to MultiAgent systems*. New York: Wiley.