Journal of Universal Computer Science, vol. 11, no. 6 (2005), 1011-1039
submitted: 20/6/04, accepted: 14/5/05, appeared: 28/6/05 © *J.UCS*

Exploiting Agent Ontologies in B2C Virtual Marketplaces

Domenico Rosaci
(University Mediterranea of Reggio Calabria, Italy
domenico.rosaci@unirc.it)

Abstract: In the last few years, an overwhelming amount of agent-based systems for supporting business-to-customer (B2C) e-commerce activities have been proposed. In this context, the use of agent ontologies for modelling the realities of both customers and sellers may play an important role. This paper deals with a formal model of agent ontologies, capable of describing the entities involved in the above realities (products, product features, product categories) as well as the behaviour of customers and sellers in performing their activities. Furthermore, we present some techniques that exploit the proposed ontology model for supporting the various B2C e-commerce stages represented in the Consumer Buying Behaviour (CBB) model. Finally, we briefly describe the OBA-B2C multi-agent architecture that implements in a JADE-based environment all the proposed techniques.
Key Words: intelligent agent technology, agent ontology, b2c e-commerce
Category: H.3.4

1 Introduction

1.1 Motivation

Agents operating in e-commerce activities can be seen as *mediators* between the actual involved subjects, i.e. customers and businesses.

It is useful to consider the behaviour of an agent in the context of a uniform framework representing the various phases of its activity. Traditional marketing research has developed many descriptive theories and models that attempt to capture the consumer behaviour in finding, buying and using goods and services.

There are several models that were developed in the CBB context. They differ in various aspects; however, as noticed in [Guttman et al.1998], all of them propose six relevant steps in the consumer buying behaviour.

1. *Need Identification*. In this stage the consumer is stimulated to become aware of some unmet need. For instance, consider the case of a customer that is interested in a certain category of books. Agents can continuously monitor the Web and advert the customer when a new book of that category is available.

2. *Product Brokering*. Once a consumer has identified a need to satisfy, he has to find *what* to buy through a careful evaluation of the various products possibly satisfying that need. This requires a comparison of product alternatives based on some consumer-provided criteria. At the end of this step,

a set of products, usually called the *consideration set*, capable of satisfying the consumer desires, has been identified.

3. *Merchant Brokering*. In this step, the consideration set is combined with merchant-specific alternatives based on consumer-selected criteria (e.g, availability, price, delivery time, warranty, reputation, etc.) for helping the consumer to determine *whom* to buy from.

4. *Negotiation*. During this step, the various terms of the transaction as, for instance, the price, are determined. The benefit of negotiating the price of a product instead of fixing it is that the merchant does not need to determine the value of the good a priori [Maes et al. 1999]. Rather, the price is dynamically determined by the marketplace. The duration and complexity of the negotiation strictly depend on the market.

5. *Purchase and Delivery*. This step can either represent the termination of the negotiation phase or can occur sometimes afterwards.

6. *Service and Evaluation*. This step involves the final activities of the buying process, such as product service, customer service, evaluation of the overall process and so on.

In [He et al. 2003], an extension of the traditional CBB model described above is proposed, in order to also cover the *buyer coalition formation* behaviour. Indeed, this extended model takes into account that customers, after having chosen the product to buy in the product brokering stage, before choosing the most suitable merchant in the merchant brokering stage, may interact with other (similar) buyers to form a *buyer coalition*. If each customer has an associated software agent, the buyer coalition formation can be viewed as a set of software agents that communicate with each other for performing a common task, and that try to approach the merchant with a large order in order to obtain a leverage. This model does not include the two last stages of the traditional CBB model (purchase and delivery, product service and evaluation), because these stages do not present aspects requiring the use of the agent technology.

After determining the most suitable merchant for buying the desired products, before beginning negotiation, the customer often visits the merchant site, in order to know more details about the product offer or to find other offers related to products of interest. Agents can suitably mediate such a stage. Indeed, from the customer's viewpoint, an agent can retrieve, on behalf of its owner, some information about products of interest and give advice to the customer about their presence in some merchant sites. From the seller's viewpoint, an agent can be present on the merchant site for providing the visitor with the most suitable presentation, w.r.t his profile. For instance, a seller agent may recognize that the

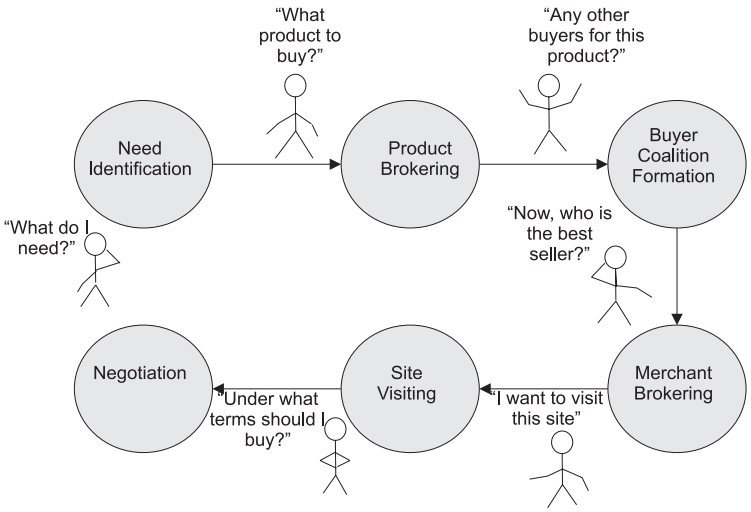

Figure 1: The CBB Consumer Buying Behaviour Model

customer is not interested in multimedia presentations and, consequently, may decide to propose a *light* presentation, without animations, applets, etc.

None of the aforementioned CBB models include such a *site visiting* stage; in the following of this paper, we will use an extension of the past CBB model, covering also the site visiting stage, as the reference model of the Consumer Buying Behaviour. It presents the following six stages (see Figure 1): (a) need identification; (b) product brokering; (c) buyer coalition formation; (d) merchant brokering; (e) site visiting; (f) negotiation. The meanings of each of these stages is that specified in the above discussion.

Several agent-based systems have been proposed for mediating the above activities. For a detailed survey of these systems, see [He et al. 2003].

In [Rosaci, 2004], we have presented a model of personal agent ontologies, capable of representing, in a unique and integrated framework, the single realities of both customers and sellers in a B2C electronic commerce context. As a first contribution, in this paper we extend this model in order to represent the ontology of a whole virtual marketplace. Furthermore, as a second contribution, we study how this model can be exploited for supporting, in a unified manner, all the CBB stages, allowing customers and sellers to effectively interact in a virtual marketplace.

1.2 Related Work

The necessity of representing, in a customer profile, not only concepts of his interest but also his behaviour in accessing those concepts, has been considered in

some works as, for instance, in [Bergamaschi et al. 2001, Calvanese et al. 1998, Terracina and Ursino 2000, Buccafurri et al. 2002a]. As for as the e-commerce research is concerned, MOMIS (Mediator envirOnment for Multiple Information Sources) [Bergamaschi et al. 2001] handles both integration and querying of multiple, heterogeneous information sources, storing both structured and semi-structured data. Data source integration is carried out by following a semantic approach based on Description Logics, clustering and a common data model capable of representing all involved data sources. Instead of manually performing a lot of queries on several Web pages for retrieving information about a product, MOMIS first retrieves information from various interesting data sources and, then, integrates it in a unique homogeneous view called *Virtual Catalogue* [Benetti et al. 2002] that is, therefore, a tool presenting, in a unified manner, product information derived from heterogeneous catalogues. So doing, customers do not need to interact with multiple heterogeneous catalogues but with a unique one uniformly representing all of them. After a Virtual Catalogue has been constructed, MOMIS extracts a list of products fulfilling customer requirements from it. The B-SDR Agent, proposed in [Buccafurri et al. 2002b, Rosaci et al. 2002], is a multi-agent system for representing and handling e-commerce activities. In such an approach, an agent is present in each e-commerce site, handling the information stored therein. Furthermore, an agent is associated with each customer, handling his profile. The information associated with both sites and customer profiles is represented and handled using a particular conceptual model called *B-SDR network*. This latter allows to uniformly manage heterogeneous data sources and to construct and maintain a profile storing information about the visits the customer carries out to the various e-commerce sites. Whenever a customer accesses an e-commerce site, the associated agent updates his profile. This operation is carried out by deriving similarities between concepts present in different e-commerce sites; each group of similar concepts is represented by a unique concept in the profile. When a customer accesses an e-commerce site, the site agent sends its B-SDR network to the customer agent. This latter activates a function for computing semantic similarities between portions of the B-SDR network associated with the site and portions of the B-SDR network representing the customer profile. Each of these similarities represents an issue of interest for the customer which is present in the e-commerce site. For each interesting product thus selected, the site and the customer agents cooperate for presenting to the customer a B-SDR network illustrating the offers of the site for that issue. MORPHEUS [Yang et al. 2001] is a comparison-shopping agent that automatically collects product descriptions from a group of on-line stores on user's behalf. Since the stores are heterogeneous, a wrapper must be built and maintained for each store. In particular, MORPHEUS comprises: (a) a *wrapper generator*, that is a learning module that constructs a wrapper for each

store; (b) a *wrapper interpreter*, that is a module that executes the wrapper for collecting product information from the corresponding store; (c) an *output generator*, that integrates search results from several on-line stores and produces a unified output. The MORPHEUS approach is based on two important assumptions, namely *(i)* a proper keyword is provided for each test query; *(ii)* each product description contains the price attribute which plays a key role in product search. Ontologies, i.e. intensional descriptions of product characteristics and customer and seller behaviour, have been already exploited in B2C e-commerce context [Omelayenko 2001] and also many ontology-based approaches have been proposed in multi-agent systems field [OAS 2002].

In particular, in [Peng et al. 2002], an approach on resolving semantic differences for multi-agent systems in electronic commerce is presented, where agents may have their own specified ontologies defined on top of a shared base ontology. The representations of concepts in this ontology is based on DAML+OIL, a language for ontology definition, manipulation and reasoning. In [Corcho et al. 2003], an ontology based mediation framework for electronic commerce applications is presented. Ontologies are defined as knowledge structures to enable sharing and reuse of information. They provide a consensual representation of the electronic commerce field allowing the exchanges independently of the language of the end user, the service, or the content provider. Ontologies are used for classifying and indexing catalogues, for filtering users query, for facilitating man-machine dialogues between users and software agents, and for inferring information that is relevant to the users request. In [Tamma et al. 2002], an approach to automated negotiation among agents that exploits a shared ontology for describing the negotiation protocol, is presented.

1.3 Contribution

Differently from the aforementioned work, our approach proposes an ontology model able to cover all the stages of the CBB model, in a unified manner. All the previously described approaches try to solve the heterogeneity by adopting techniques based on the integration of the various sources or on the use of wrappers. They are very interesting from the viewpoint of a user that wants to consider the whole Web for searching goods of his interest. However, the price to pay for obtaining the integration of a potentially overwhelming amount of Web sources is often high. Indeed, integration techniques are onerous in terms of time to spend for constructing the integrated global representation of the various sources and such a global representation is often very difficult to handle, since it has large dimensions and needs a continuous pruning of the less important concepts.

On the contrary, the approach we propose in this paper is based on the construction of virtual marketplaces whose actors (sellers and customers) are represented in a uniform manner, due to the use of agents that operate as assistant

and translate the user interests and preferences in ontologies having a standard format. The main contribution of this paper consists in studying how the above ontologies can be effectively exploited for managing all the CBB stages.

This standard is represented by the model of agent ontology that we introduce.

1.4 Plan of the paper

The paper is structured as follows: in the Section 2, we give a formal description of our ontology model. In the Section 3, techniques that exploit the proposed ontology model for supporting the various e-commerce stages are proposed. In the Section 4, a multi-agent system based on the techniques above, is briefly described. Finally, in the Section 5 we draw our conclusions.

2 Agent Ontologies

The term "ontology" is borrowed from philosophy, which it refers to the subject of Existence. This term is widely used in Artificial Intelligence (AI), for generally denoting a description of concepts and relationships among concepts in a domain, since for AI systems what exists is that which can be represented. For instance, in [Gruber 1993], the term ontology is used to mean a *specification of a conceptualization*. This definition is applied for representing a domain in a program by a declarative formalism. In this context, the set of objects, as well as the relationships existing among objects, are denoted by terms in a "vocabulary" with which the program represents knowledge. In [Studer et al. 1998], an ontology is defined as a "formal specification of a shared conceptualization". This means that: (i) each concept and relationship has to be explicitely defined; (ii) the ontology should be machine readable; the ontology should capture consensual, no private, knowledge, accepted by a community of agents. From this point of view, ontology is not a simple computer representation of a domain, but it should imply a certain rate of consensus about the represented knowledge.

The definition of ontology we give in this section tries to satisfy both the exigence of representing declarative knowledge, as in [Gruber 1993], and the necessity to represent a shared conceptualization, as in [Studer et al. 1998]. Thus, we begin with defining the ontology model of a single agent and, in the next section, we will use such a model for constructing a knowledge representation of the whole marketplace, where each term of the personal ontology of each agent is shared with the other agents in the community.

It is worth pointing out that our definition of ontology, conforming to many other existing ontology models, is very different from the classic definition of database (schema). Our model does not represent only data structures, but also logic knowledge and functions, in order to express user behaviour in terms of

causal implications among events, on the one hand, and agent actions, on the other hand.

The ontology of an agent is a representation of the reality of the agent's owner. In a B2C e-commerce context, such a owner can be either a customer or a seller. In the first case, the agent ontology has to store the customer interests and preferences w.r.t. the virtual marketplace and the customer behaviour in purchasing goods (e.g., the way of visiting e-commerce sites, the strategies in negotiating, etc.). In the second case, the agent ontology has to represent the product categories of the seller, the product characteristics (price, availability, etc.) and the service features (warranties, time delivery, etc.).

In this section, we propose a formal definition of agent ontology. In order to better explain the various concepts we introduce below, we present a simple situation of a customer and a seller that interact in a virtual marketplace. This situation will serve as a leading example along this paper.

Example 1. Let John be a customer interested in purchasing books and CDs. His interest in books mainly concerns with narrative and poetry. In the past, he purchased on the Internet the books Anna Karenina*,* The Buddenbrok*,* Les Fleurs du mal *and* La Divina Commedia*. He also purchased the CD* The Ghost of Tom Joad*. In the case of a fixed, non negotiable, price, John usually behaves as follows: (i) when he purchases a book, he also purchases a CD; (ii) moreover, he considers a book (resp., a CD) as an* ***interesting book*** *(resp., an* ***interesting CD****) if the* ***price*** *is smaller than 20 US$ and the* ***delivery time*** *is smaller then 3 days.*

In case of a negotiable price, suppose John negotiates with a seller for a good that has a base ***proposed price*** *equal to* $p(0)$ *at the step 0 of the negotiation. The John's behaviour is as follows. At the step 0, John makes an* ***offer*** $o(0) = 0.8 * p(0)$*. At the step* i *of the negotiation* ($i = 1, 2...$)*, in response to a new proposed price* $p(i)$ *of the seller, John offers a value* $o(i) = (p(i) + o(i-1))/2$*.*

Let $Word\&Music$ *be a seller of books and CDs. It deals with three categories of books, namely* narrative*,* essay *and* poetry*. This seller, in a bilateral negotiation with a customer, behaves as follows. Let* $p(0)$ *be the base* ***proposed price*** *of a book, that it proposes at the step 0 of a negotiation with a customer. Suppose it receives an* ***offer*** $o(i)$*, at the step* i *of the negotiation* ($i = 0, 1, ...$)*, by the customer. If* $o(i)$ *is smaller than* $0.7 * p(i)$*, the seller aborts the negotiation. Otherwise, it proposes a new price equal to* $p(i+1) = (o(i) + p(i))/2$*.*

From this simple situation, we can observe that different kinds of knowledge need to be represented in the ontologies of the customer and the seller. Namely:

- **Entities**. In both the cases of the customer and the seller in the example above, the agent ontology has to represent some products relative to the agent's owner. Each product must have an *ID* that identifies it, and a set

of associated *features* that gives some information about it. For instance, a book may have a feature *title*, a feature *author*, a feature *delivery_time* and a feature *price*. Such a representation of book is an intensional entity (a metadata) and each actual book can be viewed as an instance of this entity (a data). For example, the book *Anna Karenina* may be represented by an entity instance with ID=1, *title* = *"Anna Karenina"*, *author* = *"Tolstoj"*, *delivery time* =2 and *price* =17. More formally:

Definition 1. *Let T be a set {String,Integer,Real...} of data types. We define the* feature set *, denoted by F, as a set of variable names, each variable having a type that belongs to T. A* feature *is an element $f \in F$. Let D be a function mapping F into T, that associates each feature $f \in F$ with its data type $D(f) \in T$. An* instance *of the feature f is a value $v \in D(f)$.*

Definition 2. *Let $f, g \in F$ be two features with $D(f) = D(g)$ and let $k \in D(f)$ be a constant. With the notation $f = g$ we mean that the value of f is set equal to the value of g, and with the notation $f = k$ we mean that the value f is set equal to the value k.*

We give below a formal definition of entity.

Definition 3. *The* entity domain *$\mathcal{E}$ is the set of all the tuples $\langle ID, f_1, f_2, ..., f_n \rangle$, where ID is an Integer variable and $f_1, f_2, ..., f_n \in F$. An* entity *is an element of $\mathcal{E}$.*

Definition 4. *Let $e = \langle ID, f_1, f_2, ..., f_n \rangle$ be an entity. An* instance *of e is a tuple $i = \langle idv, fv_1, fv_2, ..., fv_n \rangle$, where $idv \in Integer$ and $fv_1 \in D(f_1), fv_2 \in D(f_2), ..., fv_n \in D(f_n)$ are feature values.*

Finally, we define the notion of *entity list*, for representing a list of elements that have the same entity schema:

Definition 5. *An entity list $L(e)$ on the entity e is a list $\{e_1, e_2, ...\}$, where $e_1, e_2, ...$ are entities of e.*

- **Categories**. A category is a collection of entities. For example, in the ontology of the seller $Word\&Music$, the category $BOOKS$ may group all the books available to be purchased. However, a category can be organized in some sub-categories. For instance, in our leading example, the category $BOOKS$ of $John$ contains the sub-categories $POETRY$ and $NARRATIVE$. Thus, generally, if we consider an entity list as a limit-case of category (i.e., a category that does not contains any sub-categories), we can inductively say that a category is either an entity list or a set of sub-categories. More formally:

JOHN=⟨ ID, **BOOKS**, **CDS**, **BEHAVIOUR** ⟩; **BOOKS**=⟨ ID, **NARRATIVE**, **POETRY** ⟩;
NARRATIVE=⟨ ID, **L(Book)** ⟩;
POETRY=⟨ ID, **L(Book)** ⟩;
CDS=⟨ ID, **L(Cd)** ⟩;
BEHAVIOUR=⟨ ID, **Negotiation** ⟩;
Negotiation=⟨ ID, **proposed_price**,**offer** ⟩;
Book=⟨ ID, **title**, **author**, **delivery_time**, **price** ⟩;
Cd=⟨ ID, **title**, **author**, **delivery_time**, **price** ⟩;
title, author=*String*;
delivery_time, price, proposed_price,offer=Integer;

Figure 2: The *John*'s categories, entities and features

Definition 6. *The* category domain $\mathcal{C}$ *is the set of all the tuples* $\langle ID, L(e)\rangle$*, where* ID *is an Integer variable and* $L(e)$ *is an entity list on* e*, and of all the tuples* $\langle ID, c_1, c_2, ..., c_n\rangle$*, where* ID *is an integer and* $c1, c2, ...,$ $c_n \in \mathcal{C}$*. A* category *is an element of* $\mathcal{C}$*.*

In the Figure 2, all the categories, entities and features involved in the $John's$ reality are described.

Categories are intensional information. An instance of a category is a set of instances of all the entities belonging to it. We can inductively define a category instance as follows:

Definition 7. *Let* $c = \langle ID, L(e)\rangle$ *be a category, where* ID *is an Integer variable and* $L(e)$ *is an entity list on* e*. An instance of* c *is a tuple* $i = \langle idv, ei_1, ei_2, ...,\rangle$*, where* $idv \in Integer$ *and* $ei_1, ei_2, ... \in Integer$ *are entity instances. Let* $c = \langle ID, c_1, c_2, ..., c_n\rangle$*, where* ID *is an integer and* $c1, c2, ...,$ $c_n \in \mathcal{C}$*. An instance of* c *is a tuple* $i = \langle idv, ci_1, ci_2, ..., ci_n\rangle$*, where* $idv \in Integer$ *and* $ci_1, ci_2, ..., ci_n \in Integer$ *are category instances.*

- **Knowledge patterns**. An ontology has to store, besides information about the involved entities, also other information relative to the behaviour of the customer (resp.,the seller) in purchasing (resp., in selling) the products. In our leading example, the negotiation behaviour of the customer *John* and the seller $Word\&Music$ are some examples of this kind of knowledge that has to be represented.

  Such a knowledge can be stored into an ontology by considering a set of *events*, that represents actions belonging to the activity of the customer (resp., the seller), that may happen or not. For examples, in the leading

example's situations, the concepts of **interesting book, interestingCD, buy a book, buy a CD, make an offer, propose a new price** can be modelled as events and represented by boolean variables as $interestingBook$, $interestingCD$, $buyBook$, $buyCD$, $makeO$, $proposeP$, respectively. Moreover, also a relational expression involving features is an event as, for instance, $(deliveryTime < t)$ or $(price < p)$. An event can be thus represented by a boolean expression.

Definition 8. *An* event *is either: (*i *) a boolean variable (*ii *) or an expression of the form $a\theta b$, where $a, b \in F$, $D(a) = D(b)$ and $\theta \in \{=, <, \leq, >, \geq\}$ is a relational operator (*iii *) or an expression of the form $a\theta c$, where $a \in F$ and $c \in D(a)$ and θ is a relational operator. The events of the type (*ii *) and (*iii *), that involve only features and do not involve any boolean variable, are called* feature-events.

Note that an event does not represent here a temporal variable, but simply a logic propositional predicate. Indeed, in our ontology, we do not aim at representing time series, but only the existence of situations (the events) that may happen or not. Time series will be implicitly represented (see below) by the sequence of changes produced in the ontology instances with the time.

The existing relationships between events can be represented by *propositional rules* as, for instance, relative to the situations described in our leading example for the $John$ customer:

$$k_1 : buyBook \Rightarrow buyCD \tag{1}$$

$$k_2 : (price < 20), (time < 3) \Rightarrow interestingBook \tag{2}$$

$$k_3 : (price < 20), (time < 3) \Rightarrow interestingCD \tag{3}$$

$$k_4 : interestingBook, proposeP \Rightarrow makeO \tag{4}$$

or, for the $Word\&Music$ seller:

$$k_5 : makeO \Rightarrow proposeP \tag{5}$$

These rules, that we call *knowledge patterns*, affirm that some events happen when other related events happen. For instance, the rule k_1 affirms if the

event *buyBook* has the value *true*, also the event $BuyCD$ has the value *true*.

More formally:

Definition 9. *A* knowledge pattern k *is a propositional rule of the form* $a_1, a_2, ..., a_n, \bar{a}_{n+1}, \bar{a}_{n+2}, ..., \bar{a}_m \Rightarrow b$ *, where* a_1*,* a_2*,* a_n*,* a_{n+1}*,* a_{n+2}*, ...,* a_m*,* b *are events. This notation means that, if it happens both* $a_1, a_2, ..., a_n$ *assume at the same time the value true, and* $\bar{a}_{n+1}, \bar{a}_{n+2}, ..., \bar{a}_m$ *assume at the same time the value* $false$*, then* b *assumes the value true. Let* $fs = \{f_1, f_2, ..., f_o\}$ *be a set of features. We say that* k *is a knowledge pattern on* fs*, if both each feature of* fs *appears in at least one of the events* $a_1, a_2, ..., a_m$ *contained in* k*, and all the events* $a_1, a_2, ...a_m$ *are feature-events.*

- **Actions**. Often, when an event happens in the world of a customer or a seller, an action is consequently produced. For instance, in our leading example relative to the customer $John$, when $John$ decides to make an offer in a negotiation phase, the value of the offer is equal to the mean between his previous offer and the price proposed at the present by the seller. We can thus say that, when the event $makeO$ is true, a program, that we denote by of, is called that sets the value of the feature $offer$ equal to $(proposed\_price + offer)/2$, where *proposed_price* is the feature representing the price proposed by the seller. Similarly, in the case of the $Word\&Music$ seller, we can say that, when the event $proposeP$ is true, a program pf is activated that behaves as follows: if $offer < 0.7 * proposed\_price$, it sets the event *end* to the value *true* and then terminates, otherwise it sets the value of *proposed_price* equal to $(proposed\_price + offer)/2$. In this latter case, we observe that the program modifies both a feature value and an event value.

  We call *action* a 5-tuple composed by an event, as $makeO$, a program, as *co*, that is activated by the event, a set of features, as $\{proposed\_price\}$, that are the arguments of the program, ,and another two sets of features and events, as $\{offer\}$ and $\{end\}$, respectively, whose value is modified by the program. More formally:

  Definition 10. *An* action *is a tuple* $\langle e, P, fs1, es, fs2 \rangle$ *where* $e \in \mathcal{E}$*,* P *is a program,* $fs1, fs2 \in 2^F$*,* $es \in 2^{\mathcal{E}}$*, such that the program* P *is activated if* $e = true$ *by passing it as input arguments the features belonging to the set* $fs1$ *and* P *modifies both the value of the features belonging to the set* $fs2$ *and the events belonging to the set* es*.*

  Note that our definition of *action* is very general, and can represent any negotiation process (by using a suitable program and some suitable events

$\mathbf{E}$={Book,Cd,Negotiation}

$\mathbf{C}$={JOHN,BOOKS, NARRATIVE,POETRY,BEHAVIOUR}

$\mathbf{K}$={k_1, k_2, k_3, k_4}

$\mathbf{A}$={a_{John}}

Figure 3: The *John*'s ontology

and features), as well as any other process performed by the agent as, for example, the formation of a buyer coalition.

In our leading example, we can define the actions

$a_{John} = \langle makeO, of, \{proposed\_price\}, \{\}, \{offer\}\rangle$

$a_{Word\&Music} = \langle proposeP, pf, \{offer\}, \{end\}, \{proposed\_price\}\rangle$.

Now, we can give our definition of agent ontology. This is a collection of four sets: a set of entities, representing all the products which the customer is interested in (resp., which are in the seller's catalog) and all the other entities belonging to the customer (resp., the seller) activities , a set of categories, describing the hierarchical structure of the entities, a set of knowledge patterns, describing the rules followed by the customer (resp. seller) in purchasing (resp. selling) products and a set of actions, specifying what actions the customer (resp., seller) performs when the environment changes. Since all this sets contain only intensional information, such an ontology is purely intensional and can be viewed as a schema. We also define the *instance* of an ontology, that contains, for each entity (resp., category) of the ontology, also a set of instances of that entity (resp., category). More formally:

Definition 11. *An* agent ontology schema *is a 4-tuple* $\langle E, C, K, A\rangle$*, where: (*i*) E is a set of entities; (*ii*) C is a set of categories; (*iii*) K is a set of knowledge patterns; (*iv*) A is a set of actions.*

Definition 12. *An* ontology instance *of an ontology* O=$\langle E, C, K, A\rangle$ *is a 4-tuple* $\langle EP, CP, K, A\rangle$*, where: (*i*) EP is a set of of pairs* (e, ei)*, such that* $e \in E$ *is an entity and ei is a set of instances of e; (*ii*) CP is a set of of pairs* (c, ci)*, where* $c \in C$ *is a category and ci is a set of instances of c; (*iii*) K is a set of knowledge patterns; (*iv*) A is a set of actions.*

As an example, the ontology of the customer *John* is shown in the Figure 3.

It is necessary to use an ontology instance when we need to access information about the various products (*author, title, price*, etc.) However, there are cases

in which we need to access only intensional information (e.g., the hierarchy of the categories or the knowledge patterns). In this case, it is sufficient to handle an ontology schema.

Note that our definition of ontology schema is according to traditional definitions, that represent ontology only an intensional information. Our definition of ontology instance, on the contrary, is not a classical ontology definition, but it also aims to represent the intentional data having a structure represented by the corresponding ontology schema.

3 Exploiting agent ontologies in B2C e-commerce

In this section, we study how the ontology model presented above can be exploited for supporting the six stages of the CBB model described in Section 1.1. Firstly, we describe a formal framework that can be used for representing an agent community.

3.1 The virtual marketplace

We assume our agents operate in a virtual marketplace where each customer and each seller are supported by an agent, univocally identified by an ID, that stores and uses the ontology of its owner and that behaves following a particular program. Moreover, we also assume that an agent can communicate with another agent by exploiting two communication primitives called *send* and *receive*. The former allows an agent to send another agent a message, which some objects (e.g., features, entities, categories, knowledge patterns, actions, ontologies) can be attached to, by exploiting the ID of the agent to be contacted as an address. The latter allows an agent to receive a message by another agent. Finally, we suppose each agent has an *address-book* of known customer agents that have interacted with it in the past or whose addresses it has found in some a way (e.g., that have been transmitted to it by another agent). The address book contains, for each agent stored in it, a *profile* of the agent composed by the following two data structures :

- an *interesting category set*, representing several information about the past agent behaviour in performing the various e-commerce activities i.e, what categories and products it has accessed, what products it has purchased and at what time the last purchase has been made;
- a set of knowledge patterns, representing the agent preferences.

Formally:

Definition 13. *Let an* object *be either a* feature *or an* entity *or a* category *or a* knowledge pattern *or an* action *or an* ontology*. A* message *is a pair* $(text, os)$*, where text is a String and os is a set of objects.*

Now, we can define an agent as follows:

Definition 14. *An* agent *is a tuple* $\langle ID, A, O, B\rangle$*, where:*

*(*i*)* ID *is an Integer that identifies the agent in the virtual marketplace.*

*(*ii*)* $A = \langle a_1, a_2, ..., a_n\rangle$*, called* address-book*, is an array of pairs* (id, p) *such that:*

- *id represents the identifier of a known customer agent a;*
- *p is the* profile *of a, that is, a pair (ic,kp) such that:*
  - *ic, called* interesting category-set *is an array of tuples*

    $\langle id, accesses, purchases, last\_access, last\_purchase, ip\rangle$

    where id is the identifier of a category x which a deals with, accesses (resp., purchases) is an Integer values that represents the number of times a accesses (resp., purchases) products belonging to x, last_access (resp., last_purchase) is a Date value representing the last date customer accesses (resp., purchases) products belonging to x; ip is an array of tuples

    $\langle id, accesses, purchases, last\_access, last\_purchase\rangle$

    where id is the identifier of a product p, belonging to x, which a deals with, accesses (resp., purchases) is an Integer value that represents the number of times a accesses (resp., purchases) p, last_access (resp., last_purchase) is a Date value representing the last date customer accesses (resp., purchases) p;
  - *kp is a set of knowledge patterns.*

*(*iii*)* O *is an ontology instance.*

*(*iv*)* B *is a program, codifying the overall behaviour of the agent.*

Definition 15. send *is a function that yelds as input an Integer id and a message m and returns a boolean value. send returns true if m has been received by the agent having the* $ID = id$*,* $false$ *otherwise.*

receive *is a void function that yields as input an Integer id and a message m and, when it is called, waits for a possible message coming from another agent. If a message* $m1$ *arrives, coming from an agent with ID equal to i, the function sets* $m = m1$ *and* $id = i$*.*

Definition 16. *A* virtual marketplace *of m agents and n sellers is a pair $VM = (C, S)$ where C is a set of agents $\{c_1, c_2, ..., c_m\}$, such that each agent of C is associated with a customer, and S is another set of agents $\{s_1, s_2, ..., s_n\}$, such that each agent of S is associated with a seller.*

3.2 The Agency

As pointed out in the Section 1, the existing Web stores are heterogeneous in their data formats. A product, as well as a product category, may have a certain name in a Web store, and a different name in another one. In our approach, the agents populating a virtual marketplace use a unique code for identifying categories and products. This is possible by exploiting a particular tool, called *agency*, that provides both each seller of the marketplace with an automatic engine for creating its own product-catalog and each customer of the marketplace with another engine for searching goods of interest in the marketplace. In details, the agency operates as follows:

- When a seller desires to register himself to the marketplace, he can use the utility *register* of the agency. The agency assigns an ID to him, and no two identical IDs exist in the marketplace.
- The creation of the product catalog is performed by the assistance of the agency. The seller can insert in his catalog categories already available (because they have been inserted by other sellers), each of them having an own ID, otherwise he can insert a new category. In the latter case, the agency assigns a new ID to this category. Then, the seller can organize each of his category in sub-categories, and populate them with specified products. The products can be chosen by the products already existing in the marketplace (i.e., already specified by other sellers), and already having their own ID and features. Otherwise, new products can be defined by the seller, with personalized features, and a new ID is assigned by the agency to each of these products. Note that a seller that desires to insert in his catalog a category (resp., product) as, for example, the category of books, has the possibility to examine all the existing categories (resp. products) and to decide if one of these categories (resp. products) represents just the one he deals with, on the basis both of the category's (resp. product's) name and the category's (resp. product's) structure (e.g., he might observe that there exists a category *book* that contains books, thus he might choose to insert it in his ontology). If none of the existing categories (resp. products) satisfies the seller, he can choose to create a new category (product). This approach tries to avoid the presence of synonymies by stimulating the seller not to create a new category (resp. product) when a suitable one already exists but, obviously, this does

not assure that synonymies will be really absent, since some seller may (either erroneously or intentionally) ignore the presence of an already existing category (resp. product) suitable for him and create another (synonym) category (resp. product). This inconvenient could be avoided by introducing the usage of a widespread electronic catalog standard, like BMEcat [BMEcat], (that is often used in connection with the product feature description standard eCl@ass [eclass]) or UNSPSC [UNSPSC], by imposing that each seller has to choose the name of their categories /resp. products) by the catalog standard. This latter approach would avoid completely the existence of synonymy categories (resp. products), by paying the price of a lesser flexibility than the method previously proposed, since the use of a standard catalog makes impossible to create new categories (resp. products), while in real situations this possibility is often required. The research of a good compromise between the exigence of eliminating synonymies and allowing a certain degree of flexibility is one of the object of our ongoing research, as we point out in Section 5.

– When a customer visits a seller site of the marketplaces, and accesses the various products and product categories, his agent builds its own ontology based on the information contained in the site, that are homogeneous w.r.t. the other sites of the marketplaces. This assures an uniform representation of products and categories in the whole marketplace, and makes effective the exploitation of the agent ontologies.

3.3 Supporting the CBB stages

Below, for each of the CBB phases, we describe how the agents of a virtual marketplace $VM = (C, S)$ exploit their ontologies for supporting the involved taks.

- **Need identification**. In this stage, the seller that has included a new product in his catalog, desires to contact each customer of the virtual marketplace for notifying it the product offer. The seller agent can support this stage since it is provided with a list of all the customers having interact with the seller in the past and, furthermore, for each customer the seller agent stores the customer profile. Thus, the seller agent can use customer profiles for detecting those customers that has in their profile the category of the new proposed product and that are thus probably interested in the new offer. It is important for the seller not to contact all the customers in the community, for avoiding to disturb customers that do not have interest in the category of the offered product. Indeed, our approach tries to realize a promoting activity that does not include spamming or annoying adverts, in order to

produce, by such a non intrusive approach, a better effectiveness for the seller. More formally, suppose that $s \in S$ is a seller agent. Let A the address book of s. Now suppose that a new product is inserted in the category c of the seller associated with s. Agent s inspects each element $a_k = (id, p) \in A$, in order to check either (i) if c belongs to the interesting category-set $p.ic$ of a_k or (ii) for each category $x \in p.ic$, if c is a subcategory of x. If this happens, s sends a message to the customer agent identified by id for notifying it the existence of a new product p in the category c which it is interested in.

- **Product brokering**. Suppose that $c \in C$ is a customer agent and that the owner of c is looking for products that both belong to a category x and some of their features have values satisfying specified customer preferences. Note that it is possible to express such preferences by means of knowledge patterns. For instance, the knowledge patterns k_2, k_3 of our leading example represent requests of this kind.

  Automatically, c sends a message to each seller agent $s \in S$ of the virtual marketplace, containing: (i) the interesting category x; (ii) a set of knowledge patterns representing the customer preferences.

  Then, s checks if the category x exists in its ontology. We have various alternatives for performing the matching between x and the categories of s: A first, naive approach, may consider only the possibility of an absolute matching between the categories' names. A more sophisticated approach may exploits a structural matching, that evaluated similarities between the category's structures, as proposed in [Bergamaschi et al. 2001, Buccafurri et al. 2002a, Terracina and Ursino 2000]. Finally, it is possible to use search engines for e-commerce applications as, for example, those described in [Kießling et al. 2004] or the multi-objective bargaining process described in [Fisher et al. 2002]. In these latter work, authors present the personalized search *Preference XPath* , the *Preference Presenter* implementing a sales psychology based presentation of search results, the *Preference Repository* responsible for the management of situated long-term preferences, the flexible *Personalized Price Offer* and the multi-objective *Preference Bargainer*. All these components can be customized for different application scenarios.

  We have the possibility to use in our product brokering technique any of the approaches above, obviously with different results, whose analysis is beyond of the scope of this paper. In the case x exists in the s's ontology, s firstly updates its address book as follows: if c is not present in the address book, c is inserted in the address book as a new element with an interesting category set that contains only x. Otherwise, if c is already present in the address book, but the interesting category set of c does not contain x, x is inserted

in the interesting category set of c.

After this, s sends to c a message containing information about the available products that both belong to the category x and that satisfy all the customer preferences expressed by the knowledge patterns. Below we formally define when a product satisfies, on the basis of its features, a knowledge pattern:

Definition 17. *Let $p = \langle id, p_1, p_2, ..., p_n \rangle$ be a* product*, that is, an instance of an entity $\langle ID, f_1, f_2, ..., f_n \rangle$ and let k be a knowledge pattern on the features $\{f_1, f_2, ..., f_n\}$. We say that p satisfies k if the values $p_1, p_2, .., p_n$, assigned respectively to $f_1, f_2, ..., f_n$, makes* true *the body of the knowledge pattern.*

For instance, in our leading example, the product

$p = \langle 1, \text{“}AnnaKarenina\text{”}, \text{“}Tolstoj\text{”}, 17, 2 \rangle$ satisfies the knowledge pattern k_2.

Finally, if x does not exist at all in the ontology, s sends to c a message saying it does not deal with category x.

- **Buyer coalition formation**. In this stage, a buyer tries to form a coalition with other buyers having similar interests, in order to approach merchants with large orders and thus possibly obtain a leverage. In order to form such a coalition, the buyer has to detect those buyers in the virtual marketplace that are the most similar to him. Customer ontologies can support this operation, because it is possible to compare different ontologies and computing a similarity degree. However, the extraction of these similarity degrees is a very hard task. Ontologies are schemas for the agent realities, thus any of the technique for extracting inter-schema similarities, proposed in the information system literature as, for instance, those proposed in [Buccafurri et al. 2002b, Rosaci et al. 2003], might be profitably applied for solving the problem. Unfortunately, these techniques do not deal with semantic representations that include logic knowledge as that expressed by the knowledge patterns of our model. Moreover, the aforementioned approaches are quite computationally complex, and thus no suitable for online operations as those involved for implementing the formation of buyer coalitions. Here, we propose a simple technique, that computes the similarity between ontologies only based on schema similarities, without considering knowledge patterns. Our ongoing research, as we will note in Section 5, is dealing with the problem of realizing more sophisticated techniques that take also into account knowledge patterns. Suppose that $c \in C$ is a customer agent. In order to find suitable partners for forming a buyer coalition, c sends a message to each customer agent $a \in C$, attaching to it that portion S of its

ontology (entities and categories) that it want to share with a. In its turn, a computes the similarity degree between S and its own ontology O, applying the following technique:

1. a computes both the overall number n_c of distinct categories belonging either to S or to O, i.e., $n_c = |O.C \bigcup S.C|$, and the overall number n_e of distinct entities belonging either to S or to O, i.e., $n_e = |O.E \bigcup S.E|$;
2. a computes the number x_c of categories belonging both to S and to O, i.e., $x_c = |O.C \bigcap S.C|$, and also it computes the number x_e of entities belonging both to S and to O, i.e., $n_e = |O.E \bigcap S.E|$
3. a computes the ratios $f_1 = \frac{x_c}{n_c}$ and $f_2 = \frac{x_e}{n_e}$;
4. a computes the similarity degree as a weighted mean $sd = \frac{W_c * f_1 + W_e * f_2}{2}$, where W_c and W_e are two coefficients, whose value is defined by the a's owner, that weights the importance of the categories and the entities, respectively, in defining the similarity between the two agents.

If this degree is higher than a certain threshold, a sends a message to c saying it agrees to form a coalition in order to buy goods together with c. Possibly, before accepting the c's invitation, a may analyze other parameters that c has to provide, as the coalition duration, the coalition costs, etc. Furthermore, a may also consider c only as a candidate for the coalition, reserving to its human owner the final decision.

- **Merchant brokering**. Suppose that $c \in C$ is a customer agent and that the owner of c is looking for a product p belonging to the category x. Automatically, c sends a message to each seller agent s of the virtual marketplace, for informing s that c is interested in the product p belonging to the category x. Then, s checks if the category x exists in its ontology. In the affirmative case, similarly to the product brokering stage, s firstly updates its address book as follows: if c is not present in the address book, c is inserted in the address book as a new element with an interesting category set that contains only x. Otherwise, if c is already present in the address book, but the interesting category set of c does not contain x, x is inserted in the interesting category set of c.

  After this, s checks also if p is present in the category x and, if this happens, s sends a message m to c for notifying it that it is possible to purchase p in its Web site, and sends it all the available information about p.

  If x exists in the ontology of s but p does not exist in x, then s sends to c a message advising it that the product is not available, and containing information about the other possible available products belonging to the category x.

Finally, if x does not exist at all in the ontology, s sends to c a message saying it does not deal with category x.

The customer agent c collects all the messages coming from the various sellers, in order to determine the best offer.

- **Site visiting**. Suppose that a customer c accesses the site of a seller s. There are two possible cases: (i) c is a new visitor; (ii) c is a known visitor, already existing in the s's address book. In the first case, the c's identifier is inserted in the address-book of s, and a new profile is created for him; in the latter case, the existing profile of c is updated. Recall that the profile of c consists of the sets ic and kp. The set ic represents the various categories and the various products of s which c is interested in. The set kp represents the various preferences of c in buying. Two basic operations are performed by s on the c'profile:
  - Update profile: The profile is updated when the customer c accesses, at a time t, a product y belonging to a category x. Suppose that in the profile p of c, the category x exists (if it does not exist, it is inserted), and the element corresponding to x is $p.ic[i]$. Suppose that the product y exists in p (if it does not exist, it is inserted), contained in $p.ic[i].ip[j]$. Both the counters $p.ic[i].accesses$ and $p.ic[i].ip[j].accesses$ are increased of one unit and both $p.ic[i].last\_access$ and $p.ic[i].ip[j].last\_access$ are set equal to t. Furthermore, if c buys the product p, both the counters $p.ic[i].purchases$ and $p.ic[i].ip[j].purchases$ are increased of one unit and both $p.ic[i].last\_purchase$ and $p.ic[i].ip[j].last\_purchase$ are set equal to t. Finally, the agent of the seller s is able to detect some characteristics of the customer behaviour as, for instance, his preferences about the presentation style. As an example, if the customer prefers a site presentation with a small presence of graphical objects, the seller agent stores this preference in a knowledge pattern, and then inserts this pattern in the knowledge pattern set. $p.kp[i]$.
  - Exploit profile: when the customer c, already known, accesses the seller's site, the seller agent examines the c's profile, in order to retrieve customer preferences. Firstly, it determines the most accessed categories and proposes to the customer the most accessed products of these categories, highlighting those categories which customer mostly purchased from in the past. Then, it uses the knowledge patterns of the knowledge pattern-set for selecting the most suitable layout for product presentation and for taking into account other possible customer preferences.

Note that in this work we do not deal with the techniques for deriving knowledge patterns by monitoring the customer behaviour. As we point out

in the Section 5, this is a subject of our ongoing research. However, we want to explain by a simple example what are our basic ideas about this issue.

Example 2. Suppose that the seller agent s is able to generate three different layouts for site presentation, by using three different programs named $complete\_layout()$, $medium\_layout()$ and $soft\_layout()$, respectively. The $complete\_layout$ layout creates a presentation with all the available graphical and textual objects. The $soft\_layout$ generates a layout that is only textual, and the $medium\_layout$ generates a layout with a small content of graphics.

Suppose now that in the ontology of s there are three events called $complete$, $medium$ and $soft$, and the following actions:

$a1 = \langle complete, complete\_layout, \emptyset, \emptyset, \emptyset \rangle$,

$a2 = \langle medium, medium\_layout, \emptyset, \emptyset, \emptyset \rangle$

$a3 = \langle soft, soft\_layout, \emptyset, \emptyset, \emptyset \rangle$

that means the program $complete\_layout$ is executed when the event $complete$ is true, the program $medium\_layout$ is executed when the event $medium$ is true and the program $soft\_layout$ is executed when the event $soft$ is true. Finally, suppose that in the ontology of s there are three events called PC, WAP and $PALMTOP$, that become true when the customer currently served is using PC, a WAP or a palmtop, respectively.

Now suppose that a customer c, that is using a palmtop, is accessing for the first time the seller site. Since s does not yet know the preferences of c, it proposes to c three modalities of presentation before accessing the site, namely full graphical, medium graphical *and* textual *. Suppose that c choices the* textual modality *. The agent s can derive the knowledge pattern $\rightarrow soft$, for representing the fact c prefers in all the cases a textual layout. In the future, by observing other accesses of c, that sometimes uses a palmtop and sometimes exploits a PC, the seller agent s may derives that when c uses the palmtop generally he choices the* textual *layout, whereas when he uses a PC, he generally selects the* medium graphical *option. Then, s can store such derived information in the two knowledge patterns $PALMTOP \rightarrow soft$ and $PC \rightarrow medium$, and insert them in the knowledge pattern-set of the c's profile, deleting the old (and too general) pattern $\rightarrow soft$.*

- **Negotiation**. Consider a customer agent c that interacts with a seller agent s in a negotiation task relative to a product p. Suppose that the ontology of c contains an event $makeO$ that becomes $true$ when c makes an offer for the product p, and suppose that the ontology of s contains an entity $proposeP$ that becomes $true$ when s decides to propose a new price for p. Furthermore, suppose that when $makeO$ becomes $true$ (this happens when the customer

decides to make a new offer for p), a program $make\_offer$ is activated, by passing it a set of features $f1$ belonging to the ontology of c and describing the terms of the transaction (e.g, $proposed\_price$,$avalaibility\_time$ etc.). The program $make\_offer$ contains the strategy of the customer, and produces the value of the new customer offer on the basis of the values of the features belonging to $f1$. The value produced by $make\_offer$ is stored in a feature $offer$ of the c's ontology, i.e., the action

$a_c = \langle makeO, make\_offer, \{proposed\_price..\}, \{\}, \{offer\}\rangle$

is contained in the customer ontology. The feature $proposed\_price$ is sent to the seller agent s, that stores it in a feature $offer$, belonging to its own ontology, and sets the event $proposeP$ to the value $true$. When $proposeP$ becomes $true$, a program $propose\_price$ is activated by the agent s by passing it the feature $offer$ and a set of other features $f2$. This program contains the strategy of the seller, and generates the new proposed price on the basis of both the $offer$ value and the values of the features belonging to $f2$, that generally represent other variables taken into account by the seller agent as, for instance, the time available for completing the negotiation and other similar seller preferences. The program also sets an entity end, belonging to the seller's ontology, to the value $true$ if it decides the negotiation must terminate. This behaviour is represented by the action

$a_s = \langle proposeP, propose\_price, \{offer\}, \{end\}, \{proposed\_price\}\rangle$

defined in the ontology of s.

4 The OBA-B2C System: An ontology-based Agent System

The OBA-B2C is a multi-agent system based on the ontology model presented in the Section 2. It supports the six stages of the CBB model, by using the techniques described in the Section 3.3.

Due to space reasons, it would not be possible to describe the system into detail, and in all cases the purpose of this paper is not to deal with system implementation. We give here a brief overview of the tool in order to clarify how the techniques previously described can be practically realized.

The OBA-B2C MAS is able to realize a virtual community, by following an approach that has been adopted by various existing e-commerce marketplaces as, for example, *ebay* [ebay], that allow customers and sellers to participate to e-commerce activities only after having completed a mandatory registration phase. This is due to the necessity of assuring the respect of the community's rules that customers and sellers have to accepted in such a registration phase. This implies, besides other things, that each seller has to register with the agency before it is able to provide services to customers, and this is motivated by considering that

all the community actors share a common ontology and they desire to provide with the access to this ontology only people that subscribe the community's policies. Furthermore, in the OBA-B2C environment, each customer and seller uses a client software in order to perform the various e-commerce operations, instead of exploiting the Web pages of a centralized site (as in the case of ebay). This latter approach is derived from the analogous one adopted by many file-sharing communities (e.g. e-donkey, e-mule), and presents the advantage, w.r.t. the ebay's solution, of performing the various e-commerce activities in a client side manner, instead of having to connect with a central server that might have to face a very onerous work. The (reasonable) price to pay for this solution is that customers and sellers have to download and install the client programs.

The OBA-B2C prototype is entirely realized in JAVA, and exploits the JADE libraries for implementing the agents of the virtual marketplace. JADE (Java Agent Development Framework) is a software development framework aimed at developing multi-agent systems and applications conforming to FIPA standards for intelligent agents.

The standard model of an agent platform, as defined by FIPA, is constituted by the following modules:

- an Agent Management System (AMS), that is the agent who manages the Agent Platform. The AMS provides white-page and life-cycle service and provides a directory of agent identifiers (AID) and agent state. Each agent, in order to get a valid AID and thus to join with the agent community, must register with the AMS.
- The Directory Facilitator (DF) agent provides a yellow page service in the platform.
- The Agent Communication Channel (ACC), is the software that controls all the exchange of messages within the platform.

JADE fully complies with this reference architecture and when a JADE platform runs, AMS and DF agents are immediately created. Furthermore, the ACC module is set to allow message communication. A JADE platform can be split on several hosts and only one Java Virtual Machine (JVM), is executed on each host. Each JVM is a basic container of agents and allows several agents to concurrently execute on the same host. AMS and DF agents live in a main-container, while other agent containers are connected to the main container and provide a complete run-time environment for the execution of any set of JADE agents.

In our implementation, the AMS implements the agency described in the section 3.2.

A customer, after having registered himself at the agency site, can enter to the marketplace by using a client software, realized in Java and available

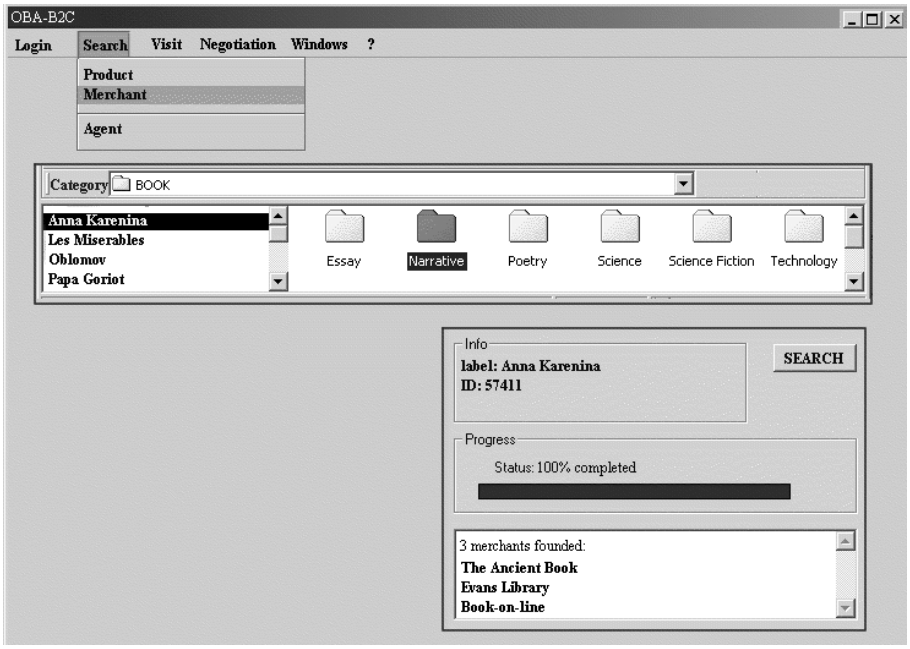

Figure 4: The OBA-B2C Customer Agent

as a stand-alone application, whose interface is showed in Figure 4 and that is able to communicate, by using the ACC, with all the other agents existing in the marketplace via the Internet. The client uses JADE functionalities for joining with the AMS of the virtual community. By choosing *Login* from the main menu, the customer accesses the marketplace specifying his user ID and password. At this moment, an agent is created by the client software and it is joined with the AMS that represents the agency. Then, the customer may choose from the menu Search the search type he wants to perform, that can have as objective a product, a merchant or an agent. In the first case, a software tool is activated that helps the customer to find a product by specifying some features and personal preferences, and the OBA-B2C agent uses the ontology of the customer for supporting the task as described in the product brokering technique. In the second case, the customer can specify a product, belonging to a desired category, and the agent searches for this product, always exploiting the customer ontology, among the various merchants, presently connected, that participate to the marketplace. This situation is depicted in Figure 4, where the customer selected the product labelled *Anna Karenina* from the sub-category *Narrative* of the category *Book*. The agent returns three merchant sites that deal with the specified product, sorted by the convenience of their offers. In the

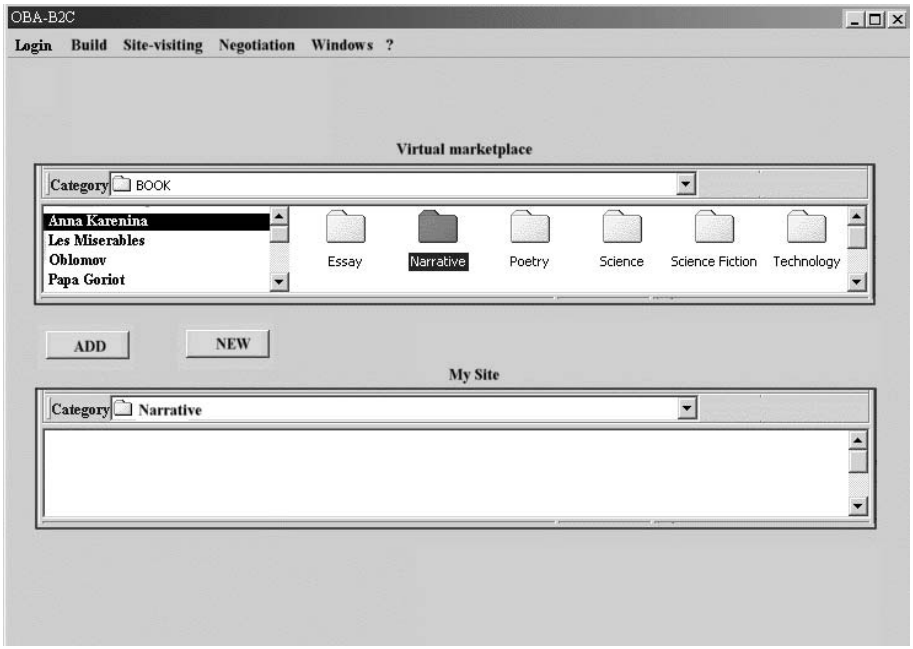

Figure 5: The Seller Agent

third case, the customer can choose to make public a portion of its ontology and to specify a product that he would like to purchase. Automatically, the agent searches in the marketplace for determining customers that are similar to its owner, for allowing to form an agent coalition.

The item *Visit* in the main menu requires the customer to specify a merchant of the marketplace; then, the agent visualizes the site of the merchant. This latter, by exploiting the information about the visitor possibly contained in its ontology, sends to the customer agent a presentation suitable to the customer preferences, by exploiting the previously described site visiting technique.

Finally, the customer agent helps the customer to negotiate with a merchant, by activating the function *Negotiation* of the main menu, that implements the negotiation technique described in the previous section.

Each seller can use a software client, whose interface is shown in Figure 5, that, on the one hand, helps him to construct his personal product catalog and, on the other hand, performs on his behalf the various e-commerce activities represented in the CBB model. As it is shown in Figure 5, the seller can build his catalog by choosing the categories he deals with in a list box that contains all the categories already present in the catalogs of the other sellers in the whole marketplace. If the seller thinks that a certain category he deals with does not

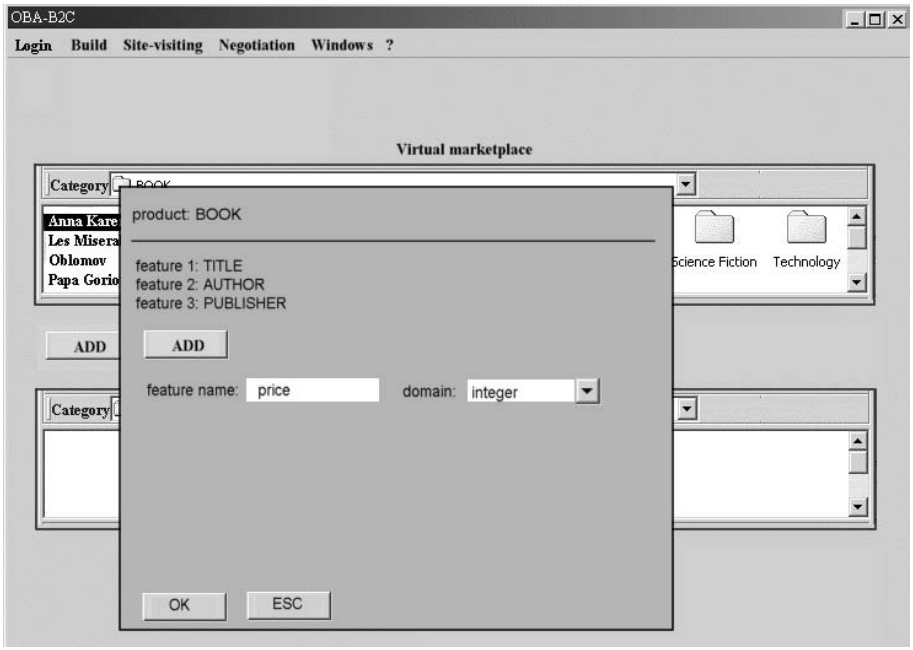

Figure 6: The Seller Agent: Creation of product features

exist in the list, he can create a new category with a new name and a new ID. The seller can also create sub-categories in a category, with a structure similar to the directory tree in a file system. After having created a category, the seller can populate it with products. The creation of a product allows the seller to assigns some desired features to it as, for instance, *name*, *price*, *availability*, etc. A product is represented in the graphical interface as a directory without subdirectories. The seller can create in a product directory the product instances, that are represented in the graphical interface as files in the directory. The features of a product can be create by right-clicking on the relative product directory. As an example, by right-clicking on the directory BOOK of Figure 5, we can associate to the product BOOK the features *title*, *author*, *publisher* and *price*, by choosing for each feature a suitable domain, as shown in Figure 6. If the seller tries to create a category (resp., a product) that has the same structure of another category (resp., product) already existing in the marketplace, the seller agent recognizes this fact, and asks the seller about the possible synonymy between his category (resp., product) and the founded one. This approach for detecting synonymies gives the system to efficiently realize a semantic homogeneity in the whole marketplace. Note that the seller can update his catalog (by adding/deleting either a product category or a product in an existing category)

whenever he wishes, by using its client software, that automatically informs the agency about the performed changes.

We are presently realizing some practical experiments with OBA-B2C system, by building B2C virtual marketplaces composed by real customers and sellers supported by the agents described above. Preliminary results clearly show that the performances obtained by the involved customers and sellers in their activities are drastically improved, compared with the situation in which OBA-B2C system is not present. Improvements mainly concern the realization of more efficient and qualitatively better, product-brokering and merchant-brokering operations, as well as the automatization of negotiation transactions, that leads customers and sellers to save significant amount of times on the Web.

5 Conclusions

This paper describes how to exploit agent ontologies in a B2C e-commerce scenario, for supporting the various stages of the Consumer Buyer Behaviour (CBB) model. The proposed agent ontology model is capable of representing both the concepts involving in customers and sellers realities as well as the behaviour of customers and sellers in performing their activities. For each stage of the CBB model and with respect to a virtual marketplace populated by agents provided with this kind of ontology, we describe how ontologies can be used by agents for efficiently operating on behalf of their owners. We have briefly show a JADE-based implementation of the above techniques, represented by the OBA-B2C System.

We note that the main limitation of OBA-B2C is that it uses, in the agent ontologies, knowledge patterns that have to be directly specified by the agent's owner. This is due to the fact that this work does not face the very important problem of extracting logical rules by observing the agent behaviour. Our ongoing research just deals with the definition of machine learning techniques for efficiently extracting the knowledge patterns of the agent ontologies. In particular, we are developing a neural network-based approach able to induce knowledge patterns by simply observing the agent actions. This approach will lead to build ontologies that will be dynamics regarding the knowledge patterns, since the induction process is continuously repeated and new rules could be add to or substitute the existing ones, following the possible changes in the agent behaviour.

Another subject of our ongoing research is the definition of a good trade-off between the exigence of eliminating the synonymies and allowing a certain degree of flexibility in the agent ontologies. Moreover, for the future, we plan to define more sophisticated techniques for detecting similarities between agents, in order to make more effective the buyer coalition formation phase.

References

[Benetti et al. 2002] Benetti, I., Beneventano, D., Bergamaschi, S., Guerra, F., and Vincini, M.:"An Information Integration Framework for E-Commerce"; IEEE Intelligent Systems Magazine 17(1),(2002), 17–25.

[Bergamaschi et al. 2001] Bergamaschi, S., Castano, S., Vincini, M., and Beneventano, D.:"Semantic integration and query of heterogeneous information sources"; Data & Knowledge Engineering 36(3),(2001), 215–249.

[Buccafurri et al. 2002a] Buccafurri, F., Lax, G., Rosaci, D., and Ursino, D.: "A User Behavior-Based Agent for Improving Web Usage";Proc. Int. Conference on Ontologies, Databases and Applications of Semantics Conference, Irvine, CA, USA (2002), pp. 1168–1185.

[BMEcat] bmecat, www.BMEcat.org.

[Buccafurri et al. 2002b] Buccafurri, F., Rosaci, D., Sarné, G., and Ursino, D.:"An Agent-Based Hierarchical Clustering Approach for E-Commerce Environments";Proc. of the 3th E-Commerce and web Technologies, Aix-en-Provence, France (2002), pp. 115–118.

[Calvanese et al. 1998] Calvanese, D., Di Giacomo, D., Lenzerini, M., Nardi, D., and Rosati, R.: "Description Logic Framework for Information Integration";Proc. of International Conference on Principles of Knowledge Representation and Reasoning, Trento, Italy (1998), pp. 2–13.

[Corcho et al. 2003] Corcho, O., Gmez-Prez, A., Leger, A., Rey, C., and Tourmani, F.:"An Ontology-based Mediation Architecture for e-commerce applications";Proc. of the International Intelligent Information Processing and Web Mining Conference, Zakopane, Poland (2003), pp. 477–488.

[ebay] ebay, www.ebay.com.

[eclass] eclass, www.eclass-online.com.

[Fisher et al. 2002] Fisher, S., Kießling, W., Holland, S., Fleder, M.:"The COSIMA Prototype for Multi-Objective Bargaining";Proc. of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems, Bologna, Italy (2002), pp. 1364-1371.

[Gruber 1993] Gruber, T.R.:"A translation approach to portable ontologies"; Knowledge Acquisition 5(2) (1993), pp. 199–220.

[Guttman et al.1998] Guttman, R., Moukas, A., and Maes, P.:"Agent-Mediated Electronic Commerce: A Survey";Knowledge Engineering Review 13(2) (1998), pp. 147–159.

[He et al. 2003] He, M., Jennings, N., and Leung, H.: "On Agent-Mediated Electronic Commerce";IEEE Transactions on Knowledge and Data Engineering 15(4) (2003), pp. 985-1003.

[Kießling et al. 2004] Kießling, W., Fisher, S., Döring, S.: "COSIMA B2B - Sales Automation for E-Procurement";Proc. of the 6th IEEE Conference on E-Commerce Technology (CEC04), San Diego, CA, USA (2004).

[Maes et al. 1999] Maes, P., Guttman, R., and Moukas, A.: "Agents that buy and sell". Communications of the ACM 42(3) (1999), pp. 81–91.

[OAS 2002] : "Workshop on Ontologies in Agent Systems, first International Joint Conference on Autonomous Agents and Multi-Agent Systems (2002), available at http://autonomousagents.org/oas/2002/proceedings.pdf.

[Omelayenko 2001] Omelayenko, B.: "Syntactic-Level Ontology Integration Rules for E-commerce"; Proc. of the 14th International FLAIRS Conference, Key West, Florida, USA (2001), pp. 324–328.

[Peng et al. 2002] Peng, Y., Zou, Y., Luan, X., Ivezic, N., Gruninger, M., and Jones, A.: "Semantic resolution for e-commerce"; Proceedings of the first international joint conference on Autonomous agents and multiagent systems, Bologna, Italy (2002), pp. 1037–1038.

[Rosaci, 2004] Rosaci, D.: "A Model of Agent Ontologies for B2C E-Commerce";Proc. of the 6th International Conference on Enterprise Information Systems, Porto, Portugal (2004), pp. 3–9.

[Rosaci et al. 2002] Rosaci, D., Sarné, G., and Ursino, D.: "A multi-agent model for handling e-commerce activities"; Proc.of the International Database Engineering and Applications Symposium, Edmonton, Canada (2002), pp. 202–211.

[Rosaci et al. 2003] Rosaci, D., Terracina, G., and Ursino, D.: "A Technique for Extracting Sub-Source Similarities from Information Sources Having Different Formats"; World Wide Web Journal 6(4) (2003), pp. 375–399.

[Studer et al. 1998] Studer, R., Benjamins, V.R., and Fensel, D.: "Knowledge engineering, principles and methods"; Data and Knowledge Engineering 25(1-2) (1998), 161–197.

[Tamma et al. 2002] Tamma, V., Wooldridge, M., Blacoe, I., and Dickinson, I.:"An Ontology Based Approach to Automated Negotiation"; Proc. of the Workshop on Agent Mediated Electronic Commerce in the Autonomous Agents and Multi-Agent Systems Conference, Bologna, Italy (2002).

[Terracina and Ursino 2000] Terracina, G. and Ursino, D.: "Deriving Synonymies and Homonymies of object classes in semi-structured information sources"; Proc. of International Conference on Management of Data, Pune, India (2000), pp. 21–32.

[UNSPSC] UNSPC: www.unspsc.org.

[Yang et al. 2001] Yang, J., Seo, H., and Choi, J.: "MORPHEUS: a more scalable comparison-shopping agent";Proc. of the 5th International Conference on Autonomous Agents, Montreal, Quebec, Canada (2001), pp. 63–64.