Semantic Web Support for the Business-to-Business E-Commerce Lifecycle

David Trastour
david_trastour@hp.com

Claudio Bartolini
claudio_bartolini@hp.com

Chris Preist
chris_preist@hp.com

Hewlett-Packard Laboratories
Filton Road, Stoke Gifford
Bristol, BS34 8QZ, UK

ABSTRACT

If an e-services approach to electronic commerce is to become widespread, standardisation of ontologies, message content and message protocols will be necessary. In this paper, we present a lifecycle of a business-to-business e-commerce interaction, and show how the Semantic Web can support a service description language that can be used throughout this lifecycle. By using DAML+OIL, we develop a service description language sufficiently expressive and flexible to be used not only in advertisements, but also in matchmaking queries, negotiation proposals and agreements. We also identify which operations must be carried out on this description language if the B2B lifecycle is to be fully supported. We do not propose specific standard protocols, but instead argue that our operators are able to support a wide variety of interaction protocols, and so will be fundamental irrespective of which protocols are finally adopted.

Categories and Subject Descriptors

K.4.4 [**Computing Milieux**]: Computers and Society—*Electronic Commerce*; I.2.4 [**Computing Methodologies**]: Artifitial Intelligence—*Knowledge Representation Formalisms and Methods*

General Terms

Languages, Standardization

Keywords

Semantic Web, E-Commerce, Matchmaking, Automated Negotiation, DAML+OIL, Service Description

1. INTRODUCTION

Electronic commerce is having a revolutionary effect on business. It is changing the way businesses interact with consumers, as well as the way they interact with each other. Electronic interactions are increasing the efficiency of purchasing, and are allowing increased reach across a global market.

E-commerce is not a static field, but is constantly evolving. Initially, e-commerce involved the use of EDI and intranets to set up long-term relationships between suppliers and purchasers. This increased the efficiency and speed of purchasing, but resulted in lock-in in the relationships. Both suppliers and purchasers had to invest significantly up-front in the relationship, so were not easily able to move their business elsewhere. The technological relationship between the parties was a friction factor, preventing free competition in the longer term.

The second phase of e-commerce aimed to address this problem. With the increasing availability of the web, a more open e-commerce environment is developing, allowing businesses to trade more flexibly with each other. Some of this openness is achieved by competition between web portals, while some competition occurs within a single web portal, acting as a marketplace for buyers and sellers to meet. Some of the efficiencies of EDI can now be achieved in a more open environment, where relationships no longer need to be long-term.

However, there is a benefit of the EDI approach that is often lost in this new phase. Price negotiation was carried out in advance in the EDI world, so purchasing can be entirely automated. When a manufacturing planning and forecast system identifies the need for a purchase, it can initiate it automatically without any human involvement, increasing speed and efficiency. In phase two, each purchase may involve interaction with a new supplier, and so may involve new negotiation of terms. As a result of this, many of these purchases cannot be made automatically, and instead require human interaction, mediated by the web.

The third phase of e-commerce is just beginning. It aims to address this issue, allowing automated business interactions to take place in a fluid environment. Technology will no longer be a friction factor to changing supplier or customer. Long-term relationships will still play an important role, but they will persist because of the choice of both parties rather than technological lock-in. The key building blocks of this new world, e-services, will be able to interact dynamically with each other to create short-term or long-term trusted trading relationships.

The remainder of this paper is structured as follows. In section 2 we describe a framework for modelling the lifecycle of B2B e-commerce interactions. In section 3 we explore the phases of matchmaking and negotiation in detail, with particular attention to the operations that are carried out on the messages that participants exchange. In section 4, we identify the need for a declarative language for service descriptions, derive requirements for it and show that DAML+OIL satisfies them. We also present a set of exam-

WWW2002, May 7–11, 2002, Honolulu, Hawaii, USA.
ACM 1-58113-449-5/02/0005.

ple service descriptions used at various stages in the B2B e-commerce interaction lifecycle. In section 5, we specify the operations that are required during the B2B lifecycle, and demonstrate that they can be straightforwardly be implemented on a description logics reasoner. We then discuss related work (section 7) and we conclude presenting our future work intentions (section 8).

2. E-SERVICES FRAMEWORK

We have developed a lifecycle model to help us understand the interactions which take place between businesses engaged in e-commerce. This model (based on that in [19]) follows the lifecycle of an interaction between two (or more) parties and has the following stages:

Matchmaking: A trader locates other traders that it could potentially do business with. This is done by some traders placing *advertisements*, and others making queries over these advertisements.

Negotiation: The trader enters into negotiation with one or more of these potential business partners, to see if they can agree mutually acceptable terms of business. This is done through an interchange of *negotiation proposals* describing constraints on an acceptable deal. The outcome of this is an *agreement,* specifying the terms that both parties consider acceptable. These terms could include a definition of the good or service being traded, price, delivery date, etc.

Contract Formation: The *agreement* is transformed into a legally binding *contract*.

Contract Fulfillment: The parties carry out the agreed transaction, within the parameters specified in the contract. The transaction may be automatically monitored, and parties would be warned if any behaviour outside the agreed terms of the contract takes place.

If the third phase of e-commerce is to become pervasive, interactions throughout this lifecycle must be standardised by the industries using it. Standardisation must take place at three levels:

1. Standards for business-specific ontologies which describe goods, services and contracts being traded. These ensure that when one trader uses a set of terms to describe a given good, another trader will be able to interpret them accurately.

2. Standards for specifying the format of advertisements, proposals, contracts and other constructs which are used during B2B interactions. These standards would specify the syntax of these constructs, with the semantics being defined by the ontologies. Hence, these standards need not be business-specific.

3. Standards that specify the protocols which traders use to interact with each other during different phases of the B2B lifecycle. These determine the messages that are sent back and forth containing the standard constructs described above.

The ARPA knowledge-sharing project [25] was the first to tackle these standardisation issues, albeit in the domain of information exchange rather than e-commerce. Ontolingua [15] provides a tool for defining standard ontologies, KIF [12] a language for representing information and KQML [10] a set of messages for exchanging this information. The FIPA [11] agent standardisation effort has defined a messaging language, and protocols for conducting B2B interchanges such as auctions. While some of the ideas developed in these efforts are clearly important (such as the notion of advertising and facilitators), they do not provide appropriate primitives for defining the constructs used in e-commerce.

The focus of this paper is primarily on standardisation of B2B constructs (point 2). As a result of this, we assume the existence of appropriate domain-specific ontologies. We believe that the semantic web provides an opportunity to develop a service description language that can be used throughout the lifecycle of interaction, rather than just at the advertising phase.

In the industry standard arena, initiatives such as UDDI [33], ebXML [26] and WSDL [7] are gaining momentum. However, they offer limited support for ontologies, semi-structured data and constraints which are essential when modelling B2B constructs as we shall discuss in section 4, and have argued more fully in [32].

In this paper, we demonstrate that DAML+OIL [20] can be used to specify a highly expressive service description language. Such a service description language is sufficiently expressive and flexible that it can be used not only in advertisements, but also in matchmaking queries, negotiation proposals and agreements. We also identify which operations must be carried out on this description language if the B2B lifecycle is to be fully supported. We do not propose specific standard protocols, but instead argue that our operators are able to support a wide variety of interaction protocols, and so will be fundamental irrespective of which protocols are finally adopted.

3. MATCHMAKING AND NEGOTIATION

In this paper, we focus on the first two stages of the e-commerce lifecycle presented above: Matchmaking and Negotiation. We now describe these phases in detail, showing the different interaction protocols that can be used. We identify commonalities between the different protocols, to allow us to develop a service description language and set of operators able to support them all. We also identify commonalities between the different stages, to allow the same service description language and operator set to support both stages in the lifecycle.

3.1 Matchmaking

Matchmaking is the process whereby potential trading partners become aware of each other's existence [8]. A buyer wishing to purchase access to a service must locate potential service providers able to meet its needs. The buyer's requirements may initially be not fully specified, and the service providers may be able to offer a range of services. The process of matchmaking should not result in the service becoming fully specified: this is the purpose of the negotiation phase which follows. Instead, the matchmaking phase should result in a buyer (or service provider) having a list of potential trade partners, each with an associated partially specified service description. This description defines the set of possible services the provider can offer which are of interest to the buyer.

There are many different protocols which can be used to accomplish this. Work on information brokerage using KQML identified the majority of these [10]. Here, we list some canonical examples. For more, see [10] or the FIPA proposed standards [11].

1. A buyer agent broadcasts its requirements to all agents in the system, irrespective of their abilities. Those agents able to meet the buyer's needs reply with information about what they are able to offer. This protocol is used at the start of the Contract Net negotiation protocol [30].

2. Whenever a service provider joins the agent community, or alters its capabilities, it broadcasts a specification of the service it offers to all agents. When an agent wishes to use a service, it contacts just those agents able to meet its needs.

3. An agent community has a centralised facilitator agent, which provides a yellow-pages service. Service providers send advertisements, consisting of descriptions of the service they offer, to the facilitator. Buyers send queries, and receive lists of providers potentially able to satisfy their requirements in response. UDDI provides a simple example of such a service. There are several variants on this protocol: buyers may be able to submit persistent queries, allowing them to be informed of new descriptions as they arrive. Buyers may be permitted to advertise alongside or instead of providers, and providers may be permitted to make queries. For use-case analyses of these and other variants, see [32].

4. An agent community may have several facilitator agents, each specialising in information about a given class of service. A buyer can either contact the appropriate facilitator, if they know which, or contact a single 'meta-facilitator', which will direct their query appropriately.

Despite these different agent architectures and communication protocols that can be used to achieve the matchmaking process, we can identify clear roles which are common to all of them. We have a *repository* of information about services or service requirements, which is maintained by the *repository host*. Agents adopting *advertiser* role are willing to *advertise* descriptions of services in the repository. These are usually, though not always, service providers. (They may be buyers, advertising a service request, or may be marketplaces offering environments where such services can be traded.) Similarly, agents adopting the *seeker* role wish to locate appropriate advertisers. Seekers can *query* a repository, via the repository host, and may be able to *browse* the repository.

In the first matchmaking protocol described above, the buyer adopts the seeker role, and all service providers adopt the role of advertiser and repository host. However, the repository is local to the service provider, and only contains information about their own service offerings. The buyer must broadcast their query. In the second protocol, the buyer is both seeker and repository host. Instead of broadcasting their query, they make it locally. However, advertisements must be broadcast. In the third protocol, the facilitator agent plays the role of repository host. In the final protocol described above, the meta-facilitator is also a repository host, but uses the repository to direct a query rather than to respond immediately.

As [29] demonstrates, different protocols may be appropriate in different situations, depending on the expected message flow. Hence, it is not appropriate to standardise on a unique protocol for all agent systems. Instead, we should allow choice from a variety of such protocols, but standardise aspects of the roles which are common to all of them. Protocol specifications determine where information is stored, and how appropriate messages are passed to access it. Role specifications determine how the information is represented, accessed and used.

For the advertiser to place an advert, it must be able to specify the set of services it is interested in trading. In many cases, these services will not be fully specified immediately. Because of this, it needs a language which is rich enough to allow an abstraction of a service to be advertised, together with constraints over that abstraction. Similarly, for a seeker to make a query, it must be able to specify, as an abstraction together with constraints, the set of services it is interested in. When a query is made, this is treated as a constraint on the acceptable set of services to the seeker. The repository host must identify which advertisements are compatible with the query. An advertisement is compatible with a query if there is at least one instantiation of the advertisement which is also an instantiation of the query. The repository host responds with a list of all such advertisers and their advertisements. Ideally, it would also return an abstraction of a service, together with constraints, specifying the most general solution acceptable to both the seeker and the advertiser.

3.2 Negotiation

The negotiation stage of the e-commerce interaction lifecycle refines the abstract service specification from the matchmaking phase to a concrete agreement between two parties. Negotiation can be one-to-one, one-to-many or many-to-many, and as a result, many different protocols have been designed to carry this out. Negotiation protocols determine the interchange of messages which take place during negotiation, and the rules by which the negotiators must abide. One-to-one protocols include the shop-front, where a seller simply offers a good at a fixed price, and iterated bargaining, with buyer and seller taking turns to exchange proposed agreements. One-to-many protocols include the English auction, the Dutch auction and the Contract Net. Many-to-many protocols include the Continuous Double Auction and the Call Auction [34]. We now describe some example protocols in more detail.

The Contract Net protocol [30] defines two roles: *manager* and *contractor*. The manager has the responsibility of monitoring the execution of a task that can be split in many sub-tasks whose results are to be aggregated. Each of the contractors can assume the responsibility of executing one or more sub-tasks.

The manager advertises the task via a task *announcement*. Prospective contractors evaluate the announcement and can decide to submit a *bid* to the manager. The manager then sends an *award* message to one or more bidders, indicating that they have been selected to perform the task.

The announcement contains information such as criteria for eligibility of contractors (*eligibility specification*) a de-

scription of the task (*task abstraction*) and a specification of the bid format (*bid specification*).

The award specification contains a complete specification of the task to be executed. The Contract Net protocol also defines what happens in the execution phase, but in the context of our discussion only the negotiation phase is relevant. The Contract Net protocol doesn't specify the format of the information objects (eligibility specification, task abstraction and bid specification). The assumption is that manager and contractors will draw from a common ontology.

A one-to-one iterated bargaining process [28] consists of two *participants* playing the role of buyer and seller exchanging *proposals* between them. The proposals consist of a tentative agreement, with all parameters of the agreement specified. On receipt of a proposal, the recipient may respond by accepting, or may send an alternative proposal with different parameter values.

An English auction [21] defines the role of *auctioneer* (seller) and *bidder* (buyer). The auctioneer makes a description of the good for sale (*good description*) available to all bidders. It communicates to the bidders the minimum bid price that is requested to get the auction going (*starting price*) and the minimum increment over the current highest bid for a new bid to be accepted (*bid increment*). It also records private information about the minimum price that it is prepared to sell the good at (*reservation price*). The auctioneer then solicits progressively higher bids from the bidders until only one bidder is left. The winner claims the item, at the price they last bid.

In the same way we analysed the different protocols for matchmaking in section 3.1, we can analyse the different negotiation protocols and identify roles and behaviours common to all. Because of the rich variety of negotiation protocols available, this is more complex than the matchmaking case. We present our full analysis of the commonalities in [4], and use this analysis to define a general software framework for negotiation. This framework abstracts away from the particular message interchange required for a given protocol, and can be parameterised with rules to implement different protocols. Here we summarise this work, restricting our attention to commonalities of role and role behaviour.

From the analysis of the negotiation protocols presented above, and others, we can identify certain abstract roles, data structures and behaviours common to all. In each case there are at least two *negotiation participants* trying to make a deal with each other. In addition, there is at least one (possibly more) *negotiation host*, responsible for enforcing the rules of the negotiation and ensuring it goes smoothly. Before negotiation can begin, the parties have already agreed roughly what the negotiation is about (usually as a result of the matchmaking process). Hence, this places a restriction on the parameters and values to be negotiated. We call this restriction the *negotiation template*. The negotiation template refers to a common ontology accepted by all participants in the negotiation. It defines a schema for valid *negotiation proposals* that participants submit to each other. The schema declares which fields are admissible and how their values are constrained. A proposal is a further refinement of the negotiation space that represents a configuration of parameters that would be acceptable to the submitter. The result of the negotiation process is an *agreement*. That is a configuration of parameters that is non-ambiguous and can be used during the execution phase to instantiate the service. Therefore we can define the negotiation process as the process through which participants move from a pre-agreed *negotiation template* to an *agreement*, via an exchange of *negotiation proposals*. A single negotiation may involve many parties, resulting in several agreements between different parties and some parties who do not reach agreement. For example, a stock exchange can be viewed as a negotiation where many buyers and many sellers meet to negotiate the price of a given stock. Many agreements are formed between buyers and sellers, and some buyers and sellers fail to trade.

Revisiting the three example protocols described above, we can represent them in terms of our abstract roles and behaviours.

The contract net manager is both negotiation participant (as buyer) and negotiation host. The contractor is a negotiation participant (as seller). The bid specification within the task announcement defines the negotiation template, and the bids made by contractors are negotiation proposals. The award specification is the final agreement, determined by the contract net manager.

During iterated bargaining, there are two negotiation participants. One, possibly both, will also play the role of negotiation host (to ensure that the other party submits proposals at the appropriate time, in an appropriate format.) The negotiation template will be agreed prior to the negotiation, usually as the output of the matchmaking process. Each proposal is a fully instantiated version of the template, and when one party agrees to the proposal of the other, that proposal becomes the agreement.

In an auction, there is one seller participant (who remains silent after specifying the good for sale), many buyer participants, and the auctioneer who acts as the negotiation host. The negotiation template is fully instantiated to define exactly what good/service is for sale in the auction, except for the price which remains undetermined. The seller participant lodges a proposal with the auctioneer which states that they are willing to sell the good, and the minimum price they will accept. Buyers announce proposals, in the form of versions of the negotiation template with the price instantiated to their current bid. When no more proposals arrive, the last proposal is used as an agreement, provided the price is higher than the minimum price in the buyer's original proposal.

We now describe the three key actions which the negotiation host carries out during the abstract negotiation process presented earlier:

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

Protocol Checking: The proposal must be submitted according to the rules of the protocol which governs the way the negotiation takes place. These rules specify (among other things) who can make proposals, when they can be made, and what proposals can be submit-

ted in relation to previous submissions. (For example, auctions often have a 'bid improvement' rule that requires any new proposal to buy to be for a higher price than previous proposals).

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

Much work has gone into standardising the different protocols used in negotiation (for instance [11]) though this (rightly) accepts that many different protocols must be available. However, these standards do not standardise the agreement formation process, or the validation process. We propose that these actions, which are common to all negotiations, should also be standardised. Hence we introduce a language for describing templates, proposals and agreements and operations on this language to carry out proposal validation and agreement formation. Furthermore, we design this language to be sufficiently general and flexible to cover the matchmaking phase[1].

4. DESCRIPTION OF SERVICES

In the previous section we have highlighted the information constructs that are exchanged in messages during matchmaking and negotiation, irrespective of what protocol is used. We have categorized them as advertisements, queries, negotiation proposals, negotiation templates and agreements. We have also identified the operations that are carried out on these constructs: matching, proposal validation, protocol checking and agreement formation. If these constructs and operations are to be standardised, we wish to build the constructs from a declarative language for describing services. Furthermore, we need to show that this declarative language can support the required operations over it. We now identify the requirements on this language, and then show that DAML+OIL satisfy most of these requirements.

4.1 Requirements

A description language for the B2B e-commerce lifecycle should satisfy the following requirements:

- Descriptions should offer a high degree of flexibility and expressiveness. Parties must have total freedom to create the service description. Different advertisers will want to describe their services with different degrees of complexity and completeness, and our language must be adaptable to these needs. Similarly, a negotiation proposal may be very descriptive in some aspects, but leave others less specified and open for further negotiation. Therefore, the ability to express semi-structured data is required.

- Descriptions need to share a common semantics. Moreover descriptions should be able to use vocabularies created by different standard bodies or industry sectors. Therefore support for interroperable ontologies is needed.

- Descriptions should easily lend themselves to performing the operations described in the negotiation and matchmaking sections. In particular, matching of advertisements with queries during matchmaking; validation of negotiation proposals against the negotiation template; and compatibility checking of two negotiation proposals to determine if an agreement can be made.

- Descriptions should express restrictions and constraints. Whether it is an offer or a request, it is often the case that what is expressed is not a single instance of a service but rather a conceptual definition of the acceptable instances. A natural way of describing this is by expressing constraints over the parameters of the service.

4.2 DAML+OIL

DAML is a DARPA programme aiming to provide a language and tools for the semantic web. One the most promising technologies it has produced so far is the DAML+OIL [20] ontology language. DAML+OIL will serve as starting point for the design of the future Web Ontology Language from W3C.

DAML+OIL is a good candidate for the language we are looking for, and meets the requirements introduced in section 4.1:

- It provides a reasonable level of flexibility and extensiveness while keeping a nice balance between expressiveness and decidability. It offers support for types, which greatly enhances the expressiveness and modularity of the descriptions.

- DAML+OIL offers support for ontologies. It is almost integrated with tools such as OilEd [5] and Protégé [14] which make the generation of new ontologies for service descriptions much easier. Both tools are being worked on to support the full DAML+OIL specification.

- DAML+OIL is a good candidate for expressing descriptions that will be subject to the operations of matching, proposal validation and agreement formation described in section 3. As we will see in the next section, all our operations can be expressed in terms of the subsumption operation [18]. DAML+OIL descriptions lend themselves very well to this operation and mature tools exist (such as the $\mathcal{SHIQ}$ reasoners Racer [16] and FaCT [17]) that can perform this operation on DAML+OIL descriptions.

- DAML+OIL offers some support for expressing constraints, while still maintaining decidability. Description logics constructors allow restrictions on objects, and XML schema [6] allows unary constraints on datatypes. It is worth noting that DAML+OIL does not support n-ary datatype constraints, which may be a problem for real e-commerce applications (for instance, the shipping cost is waved when the sum of the length and width of the product is bellow a certain threshold).

Furthermore, because DAML+OIL is expressed in RDF [23] and XML schema, it provides the added advantage that the many resources and toolsets developed for these technologies can be applied to the B2B interaction lifecycle.

[1] In [4] we also standardise some aspects of protocol checking, and parameterise these actions with declarative rules to enact different negotiation protocols. However, this aspect of our work is beyond the scope of this paper, so we omit discussion of it here.

4.3 Modelling

In this section, we explain how we use DAML+OIL to describe the various descriptions that are used in the e-commerce lifecycle. While other more general efforts like [24] already use DAML+OIL in their service descriptions, we show here how DAML+OIL is suitable for e-commerce, and especially automated negotiation.

We recognise that service description ontologies and domain specific ontologies will have an important role to play in order to achieve the semantic level of agreement between the various parties. For the sole purpose of the following examples, we define a simple service description ontology along with an ontology for the sale and delivery of computers. To keep the descriptions concise, we have chosen to use the description logics notation which is equivalent to the RDF DAML+OIL syntax[2].

The description ontology: We use the `Description` class as a common superclass for `Advertisement`, `Query`, `Template` and `Proposal`. As we will see later, an agreement is not modelled as a class but as an instance. More precisely, an agreement is an instance of a particular negotiation template.

$$\begin{aligned} Description &\sqsubseteq \top \sqcap (\geq 1\ hasService.Service) \\ Advertisement &\sqsubseteq Description \\ Query &\sqsubseteq Description \\ Template &\sqsubseteq Description \\ Proposal &\sqsubseteq Description \end{aligned}$$

The service ontology: Two services are defined in this ontology: `Sale` and `Delivery`. A `Sale` describes the sale of one `Product` through the object property, for a unit price and a quantity given by the respective datatype properties.

$$\begin{aligned} Service &\sqsubseteq \top \\ CompositeService &\sqsubseteq Service \sqcap \\ &\quad (\geq 1\ isComposedOf.Service) \\ Sale &\sqsubseteq Service \sqcap \\ &\quad (= 1\ buyer.Participant) \sqcap \\ &\quad (= 1\ seller.Participant) \sqcap \\ &\quad (= 1\ item.Product) \sqcap \\ &\quad (= 1\ quantity.positiveInteger) \sqcap \\ &\quad (= 1\ unitPrice.nonNegInteger) \\ Delivery &\sqsubseteq Service \sqcap \\ &\quad (= 1\ location.Place) \sqcap \\ &\quad (= 1\ date.date) \end{aligned}$$

We use the `CompositeService` class and the `isComposedOf` property to leave a choice to model composite services. When using the `isComposedOf` property to specify component services, the component service can only match if the main service also matches. Alternatively, a composite service can be modelled as a boolean combination of component services. In this case, any single component service can match. For instance, if we consider a service of sale and delivery of a computer, we can model it as a service of sale of computer which contains delivery as a sub-service or as the conjunction of both base services. In the first case, the service is considered as being primarily a service of sale, and would not be matched with delivery services whereas in the second case it would.

In addition, we have chosen to model the service of `Sale` to include the buyer and seller roles as properties. In doing so, we allow the buyer (resp. the seller) to specify who they are and who they would like to do business with.

The PC ontology: The `PC` class is a subclass of `Product` and must have at most one `Processor` and one amount of `memory`.

$$\begin{aligned} PC &\sqsubseteq Product \sqcap \\ &\quad (\leq 1\ hasProcessor.Processor) \sqcap \\ &\quad (\leq 1\ memory.positiveInteger) \\ Processor &\doteq \{PentiumIII, Pentium4, Athlon\} \end{aligned}$$

The Participant ontology: Public information about prospective advertisers and negotiators is organized in an ontology, following the yellow pages model. The ontology is built from information that individuals and/or companies are requested to provide at registration time. Such information is then used at matchmaking and negotiation time to verify compatibility of advertisements and proposals. For instance a buyer requiring service provision from an ISO9001 certified company, will only be matched with advertiser that declare to have ISO9001 certification. For the purpose of the examples, we define some disjoint classes `R1`, `R2`, and `R3` that will represent participant identities.

We now give an example for each description type. These examples will use the ontologies we have just defined.

4.3.1 Advertisement

An advertisement is expressed as a DAML+OIL class defined as the boolean combination of a set of restrictions over abstract properties and datatype properties. In Description Logics terms, advertisements are expressed as T-Boxes.

The following example shows an advertisement where `R1` would like to buy some PCs. More precisely, `R1` is advertising for the `Sale` and `Delivery` service. The restrictions over the `Sale` concept are that:

- items must be PCs with at least 128 Mb of memory;
- quantity of PCs being bought will be less than 200;
- unit price must be less than 700.

Since the advertiser is not interested in getting results of delivery services only, they chose not to describe their advertisement as being a `Sale` service and a `Delivery` service (i.e. by subclassing the intersection of `Sale` and `Delivery`), but rather as being a `Sale` service that *has* a `Delivery` service.

[2] XML Schema classes are not described but it should be clear by their names what they actually mean.

The restrictions on the `Delivery` service are the following:

- goods must be delivered before the 15/12/2001;
- goods must be delivered in Bristol.

In description logics notation, this advertisement can be written as:

$$Advert1 \doteq Advertisement \sqcap$$
$$\exists hasService.(Sale \sqcap \exists buyer.R1 \sqcap$$
$$\exists item.(PC \sqcap \exists memory.over128) \sqcap$$
$$\exists unitPrice.below700 \sqcap \exists quantity.below200 \sqcap$$
$$\exists isComposedOf.(Delivery \sqcap \exists date.before20011215 \sqcap$$
$$\exists location.Bristol))$$

As we can see from the ontology of the `Sale` service, we require both the `buyer` and the `seller` roles to be part of the information that is specified in the agreement. When submitting an advertisement, an advertiser who wants to play the role of a seller (resp. a buyer) should restrict the `seller` (resp. `buyer`) property to be its identifier. As the ontology shows, it is not forced to do so, but it is in its best interest. If it does not, it would be matched with advertisements of other sellers (resp. buyers). The seller (resp. buyer) can leave the `buyer` (resp. `seller`) property unconstrained, or can constrain it to be a certain subset or subcategory if they want to focus business on a certain set, for example, a prequalified set of trusted buyers. `Advert2` above is made by seller $R1$ who wishes to avoid doing business with buyer $R3$.

4.3.2 Query

A Query is similar to an Advertisement. It is also a T-Box. We give an example of a Query where the seeker is looking for all buyers and sellers of PCs with an Athlon processor and who are also requesting or providing delivery.

$$Query1 \doteq Query \sqcap$$
$$\exists hasService.(Sale \sqcap$$
$$\exists item.(PC \sqcap \exists hasProcessor.\{Athlon\}) \sqcap$$
$$\exists isComposedOf.Delivery)$$

4.3.3 Negotiation Template

After matchmaking, some parties can choose to enter into negotiation to determine the exact terms of service delivery. The negotiation template represents what is in common between all parties and is the starting point for negotiation. It also serves as a guide to scope the negotiation: negotiation proposals must comply with this template. In DAML+OIL terms, they would have to be subclass of this template.

$$Template1 \doteq Template \sqcap$$
$$\exists hasService(Sale \sqcap$$
$$\exists item.(PC \sqcap \exists memory.256or512 \sqcap$$
$$\exists hasProcessor.\{Pentium4\}) \sqcap$$
$$\exists unitPrice.below700 \sqcap \exists quantity.between100and200 \sqcap$$
$$\exists isComposedOf.(Delivery \sqcap \exists date.before20011215))$$

4.3.4 Negotiation Proposal

As stated above, a negotiation proposal must be a subclass of the negotiation template associated with the ongoing negotiation. We now give an example of negotiation proposal which satisfies the template `Template1`:

$$Proposal1 \doteq Proposal \sqcap$$
$$\exists hasService(Sale \sqcap$$
$$\exists item.(PC \sqcap \exists memory.512 \sqcap$$
$$\exists hasProcessor.\{Pentium4\}) \sqcap$$
$$\exists unitPrice.bellow500 \sqcap \exists quantity.between150and200 \sqcap$$
$$\exists isComposedOf.(Delivery \sqcap \exists date.before20011215))$$

4.3.5 Agreement

When a negotiation terminates with an agreement acceptable to both parties, this agreement must specify the service that is going to be exchanged in an exact and non-ambiguous manner. Hence, whereas a negotiation proposal is a T-box, an agreement must be a fully-instantiated instance of the negotiation template. For this reason, we model an agreement as an A-Box.

In figure 1 we give an example in RDF syntax of an Agreement reached in a negotiation with `Template1` as its negotiation template.

5. OPERATIONS OVER DESCRIPTIONS

We now return to the operations over descriptions which we identified in section 3 as essential to support a variety of matchmaking and negotiation protocols. In this section, we present specifications of these operations, together with examples of their operation, and identify the core functionality required by a reasoner to execute them.

Matchmaking: Recall from section 3 that the matchmaking process requires a repository host to take a query or advertisement as input, and to return all advertisements which may potentially satisfy the requirements specified in the input query or advertisement. Formally, this can be specified as:

Let α be the set of all advertisements in a given advertisement repository. For a given query or advertisement, Q, the matchmaking algorithm of the repository host returns the set of all advertisements which are compatible, $matches(Q)$:

$$matches(Q) = \{A_i \in \alpha | compatible(A_i, Q)\}$$

A set of descriptions are compatible if their intersection is satisfiable:

$$compatible(D1, \ldots, Dn) \Leftrightarrow \neg(D1 \sqcap \ldots \sqcap Dn \doteq \bot)$$

For example, consider the following advertisement:

$$Advert2 \doteq Advertisement \sqcap$$
$$\exists hasService.(Sale \sqcap \exists seller.R1 \sqcap \forall buyer.\neg R3$$
$$\exists item.(PC \sqcap \exists memory.256or512) \sqcap$$
$$\exists quantity.over100)$$

```
<neg:Template1 rdf:ID="AgreementBetweenR1andR2">
   <svc:hasService>
      <sale:Sale>
         <sale:buyer rdf:resource="#R1"/>
         <sale:seller rdf:resource="#R2"/>
         <sale:item>
            <pc:PC>
               <pc:memory><xsd:integer rdf:value="256"/></pc:memory>
               <pc:hasProcessor rdf:resource="#Pentium4"/>
            </pc:PC>
         </sale:item>
         <sale:quantity><xsd:integer rdf:value="150"/></sale:quantity>
         <sale:unitPrice><xsd:decimal rdf:value="600"/></sale:unitPrice>
         <svc:isComposedOf>
            <delivery:Delivery>
               <delivery:date><xsd:date rdf:value="2001-12-15"/></delivery:date>
            </delivery:Delivery>
         </svc:isComposedOf>
      </sale:Sale>
   </svc:hasService>
</neg:Template1>
```

Figure 1: Agreement example

The intersection of this advertisement with `Advert1` above is satisfiable, as `AgreementBetweenR1andR2` is an instance of both advertisements. Hence,

$$Advert1 \in matches(Advert2)$$

Validation: Recall from section 3 that the negotiation host, on receiving a proposal P, must initially check that it is valid. It is valid if it is a more constrained version of the negotiation template T for this negotiation. In description logic, this means that the negotiation host must check that T subsumes P. Formally, this can be specified as:

$$valid_T(P) \Leftrightarrow P \sqsubseteq T$$

Agreement Formation: Recall from section 3 that agreement formation requires the negotiation host to identify all pairs of proposals which are compatible. Protocol specific rules are then used to determine exactly which of these pairs are used to form an agreement, and how exactly to generate the final agreement. Compatibility can be determined using the compatibility operator defined for matchmaking. Hence, the first stage of agreement formation can be specified as follows:

Let Φ be the set of all valid proposals currently registered with the negotiation host.

$$potentialAgreements(\Phi) = \{(P_i, P_j) | compatible(P_i, P_j) \wedge i \neq j\}$$

Protocol validation and protocol-specific aspects of agreement formation are beyond the scope of this discussion. For a full discussion of these operations, together with a rule-based approach to standardising them, see [4]. When an agreement is formed, it can be verified a posteriori that the agreement subsumes the proposals that were used to form it and therefore the original negotiation template.

Note that only two atomic operations are required to define the operations specified above:

- satisfiability ($\neg(X \doteq \bot)$)
- subsumption ($X \sqsubseteq Y$).

A standard description logics reasoner is able to carry out both of these. Satisfiability lies at the core of such a reasoner, as all other reasoning or inference techniques are transformed into satisfiability checks. The subsumption operator is already defined by the DAML+OIL `subClassOf`, because our service descriptions are expressed as DAML+OIL classes (i.e. description logics concepts). A description logics reasoner can check whether two concepts subsume each other [18]. Hence, a description logics reasoner provides a good platform to implement the operations required in the B2B e-commerce interaction lifecycle.

6. IMPLEMENTATION

We have implemented a prototype matchmaking system, based on the FaCT [17] reasoner, operating on service descriptions in DAML+OIL. A full description of the prototype can be found in [13]. A prototype of our negotiation framework is described in [4]. In the current version, the negotiation framework prototype does not use DAML+OIL as a language for proposals. Work is underway to merge the two efforts in order to have a prototype based on DAML+OIL that supports both matchmaking and negotiation.

In terms of performance requirements, the matchmaking and negotiation phases are very different. To find a match for a particular advertisement, the reasoner needs to check the satisfiability of the intersection of the advertisement with each advertisement that has been previously submitted. With the dataset we have (around 100 advertisements and queries), the time spent by the reasoner is barely noticeable. We need to design a way to automatically generate large amounts of meaningfull data to put the system

under a bigger load and study its beahaviour. For negotiation, the number of descriptions to manipulate is function of the number of participants in the negotiation. Compared to matchmaking, the negotiation phase uses few but more complex descriptions (which current reasoners can handle).

Before putting this system to the test of a real e-commerce problem, we would need a reasoner that supports datatype natively.

7. RELATED WORK

Work on service description for use in matchmaking is an important part of developing open agent-based systems. However, work on developing such description languages [31, 2, 22, 27] has focussed on their application in matchmaking and brokering, ignoring the potential role of negotiation. Matchmaking is not used to locate potential trade partners, but rather to determine the functionality of another agent prior to execution. As a result of this, agents advertise exact specifications of their service (with some small amount of flexibility left to the discretion of the service user). This works for a cooperative community of agents, but will not work for a competitive environment such as in e-commerce. Instead, we treat the advertisement as an invitation to trade, and so it will be less constrained. Additionally, we define appropriate operations to support the negotiation phase, to refine an advertisement to a final agreement, and use the same service description language throughout this process.

DAML-S [1] is a core set of markup language constructs for describing the properties and capabilities of Web services, built on top of DAML+OIL. DAML-S markup of Web services is intended to facilitate the automation of Web service tasks including automated Web service discovery, execution, interoperation, composition and execution monitoring. DAML-S is complementary to our work and there is no reason why our work could not be based on DAML-S. One of the features of DAML-S we would benefit most is the support for process description. The service descriptions we are considering in this paper should be thought as the input and output parameters of a DAML-S Service Profile. The points we made in this paper that DAML+OIL allows to model quite complex descriptions of goods and services and that current reasoners allow to perform the necessary computational operations for the e-commerce lifecycle is hence also valid for DAML-S Service Profiles.

8. CONCLUSIONS AND FUTURE WORK

In this paper, we have presented an analysis of the B2B e-commerce interaction lifecycle in terms of roles, information constructs and operations necessary to carry out the interactions. We have argued that a variety of protocols can be used for matchmaking and negotiation, but that the same information constructs and operations can be used to support them all. For this reason, we advocate standardization of these constructs and operations, as opposed to standardization on a single protocol. We have assessed the requirements on an appropriate service description language for this, and have argued that DAML+OIL meets these requirements. We have shown how DAML+OIL can be used to represent advertisements, queries, negotiation templates, proposals and agreements. Furthermore, and more importantly, we have shown that the key operations necessary to support B2B interactions can be expressed in terms of satisfiability and subsumption - two operations which existing Description Logics reasoners are capable of executing. Hence, the Semantic Web provides an ideal framework for the standardization of the B2B e-commerce interaction lifecycle.

We have argued that constraint evaluation is also an important requirement to support this lifecycle. While some constraints can be expressed in DAML+OIL, existing Description Logics reasoners do not support all of these constructs. The Racer [16] reasoner has been enhanced with limited support for concrete domains (i.e. datatypes) and we plan to integrate it with our prototype.

Research on automation of negotiation requires the ability to assess the likely utility of a given advertisement or negotiation proposal. In our service description language, such proposals and advertisements can be complex structures. Up to now, most work on negotiation has assumed that only one parameter (usually price) is being negotiated. Some work has been carried out on multi-attribute negotiation (e.g. [9]) but this assumes a relatively simple utility model. If we are to be able to assign utilities to complex proposals, then research on tools to help people assess the value of different proposals (preference extraction) will be necessary. It will also be necessary to represent the relative utilities of a space of possible proposals. The application of Multi-Attribute Utility Theory to negotiation [3] is a promising approach to do this. We are currently working on ways of extending this work to assign utilities to complex service descriptions.

In this paper, we have not addressed the operations involved in moving from the matchmaking phase to the negotiation phase. If only one matchmaking query is made, and only one advertisement selected, then this process is straightforward: the negotiation template is taken to be the intersection between the query and the advertisement. However, if many queries are made and many advertisements are matched, then the problem becomes more complex. Clusters of potentially compatible participants must be formed, together with appropriate negotiation templates. We hope to explore this issue in the future.

9. REFERENCES

[1] A. Ankolekar, M. Burstein, J. R. Hobbs, O. Lassila, D. L. Martin, S. A. McIlraith, S. Narayanan, M. Paolucci, T. Payne, K. Sycara, and H. Zeng. DAML-S: Semantic Markup for Web Services. In *Proceedings of the International Semantic Web Working Symposium (SWWS)*, 2001.

[2] Y. Arens, C. Hsu, and C. A. Knoblock. Query processing in the SIMS information mediator. In A. Tate, editor, *Advanced Planning Technology*, pages 61–69. AAAI Press, Menlo Park, California, 1996.

[3] M. Barbuceanu and M. S. Fox. Cool: A language for describing coordination in multiagent systems. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, pages 17–24, San Francisco, CA, 1995.

[4] C. Bartolini, C. Preist, and N. R. Jennings. A generic software framework for automated negotiation. In *Poster Proceedings of the First International Conference on Autonomous Agents and Multi-Agents Systems*, 2002.

[5] S. Bechhofer, I. Horrocks, C. Goble, and R. Stevens. OilEd: a reason-able ontology editor for the semantic web. In *Working Notes of the 2001 Int. Description Logics Workshop (DL-2001)*, pages 1–9, 2001.

[6] P. Biron and A. Malhotra. XML Schema Part 2: Datatypes. W3C Recommendation, 2001.

[7] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. WSDL. Web Services Description Language, 2000.

[8] K. Decker, K. Sycara, and M. Williamson. Middle-agents for the internet. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, Nagoya, Japan, 1997.

[9] P. Faratin, C. Sierra, and N. R. Jennings. Negotiation decision functions for autonomous agents. *Int. Journal of Robotics and Autonomous Systems*, 24(3-4):159–182, 1998.

[10] T. Finin, R. Fritzson, D. McKay, and R. McEntire. KQML as an Agent Communication Language. In *Proceedings of the 3rd International Conference on Information and Knowledge Management (CIKM'94)*, pages 456–463, Gaithersburg, Maryland, 1994. ACM Press.

[11] FIPA. The Foundation for Intelligent Physical Agents. Interaction Protocol Library Specification, 2000.

[12] M. R. Genesereth. Knowledge Interchange Format. Principles of Knowledge Representation and Reasoning. In *Proceedings of the Second International Conference, Cambridge, MA*, pages 599–600. Morgan Kaufmann, 1991.

[13] J. González-Castillo, D. Trastour, and C. Bartolini. Description logics for matchmaking of services. In *Proceedings of the KI-2001 Workshop on Applications of Description Logics*, 2001.

[14] W. Grosso, H. Eriksson, R. Fergerson, J. Gennari, S. Tu, and M. Musen. Knowledge Modeling at the Millennium – The Design and Evolution of Protégé. In *Proceedings of the 12th International Workshop on Knowledge Acquisition, Modeling and Mangement (KAW'99)*, 1999.

[15] T. R. Gruber. Ontolingua: A mechanism to support portable ontologies. Technical report, Knowledge Systems Laboratory, Stanford University, Stanford, United States, 1992.

[16] V. Haarslev and R. Möller. Description of the RACER system and its applications. In *Proceedings International Workshop on Description Logics (DL-2001)*, 2001.

[17] I. Horrocks. FaCT and iFaCT. In P. Lambrix, A. Borgida, M. Lenzerini, R. Möller, and P. Patel-Schneider, editors, *Proceedings of the International Workshop on Description Logics (DL'99)*, pages 133–135, 1999.

[18] I. Horrocks and P. F. Patel-Schneider. Comparing subsumption optimizations. In E. Franconi, G. De Giacomo, R. M. MacGregor, W. Nutt, C. A. Welty, and F. Sebastiani, editors, *Collected Papers from the International Description Logics Workshop (DL'98)*, pages 90–94. CEUR, May 1998.

[19] N. R. Jennings, P. Faratin, M. J. Johnson, T. J. Norman, P. O'Brien, and M. E. Wiegand. Agent-based business process management. *International Journal of Cooperative Information Systems*, 5(2&3):105–130, 1996.

[20] Joint US/EU ad hoc Agent Markup Language Committee. DAML+OIL (March 2001), http://www.daml.org, 2001.

[21] P. Klemperer. Auction theory: a guide to the literature. *Journal of Economic Surveys*, 13(3):227–286, 1999.

[22] D. Kuokka and L. Harada. On using KQML for matchmaking. In V. Lesser, editor, *Proceedings of the First International Conference on Multi–Agent Systems*, pages 239–245, San Francisco, CA, 1995. MIT Press.

[23] O. Lassila and R. Swick. Resource Description Framework (RDF) Model and Syntax Specification. W3C Recommendation, 1999.

[24] S. McIlraith, T. Son, and H. Zeng. Semantic Web Services. *IEEE Intelligent Systems. Special Issue on the Semantic Web*, 16(2):46–53, March/April 2001.

[25] R. Neches, R. Fikes, T. Finin, T. Gruber, R. Patil, T. Senator, and W. R. Swarton. Enabling technology for knowledge sharing. *AI Magazine*, 12(3):36–56, Fall 1991.

[26] D. Nickull and B. Eisenberg. ebXML technical architecture specification. Technical report, UN/CEFACT, 2000.

[27] M. H. Nodine, J. Fowler, and B. Perry. Active information gathering in InfoSleuth. In *Proceedings of the International Symposium on Cooperative Database Systems for Advanced Applications CODAS*, pages 15–26, 1999.

[28] S. Parsons, C. Sierra, and N. Jennings. Agents that reason and negotiate by arguing. *Journal of Logic and Computation*, 8(3):261–292, 1998.

[29] C. Preist and S. Pearson. An adaptive choice of messaging protocol in multi-agent systems. In *Proceedings of the Third International Conference on Multi Agent Systems*, 1998.

[30] R. G. Smith. The Contract Net protocol: High-level communication and control in a distributed problem solver. In *Proceedings of the 1st International Conference on Distributed Computing Systems*, pages 186–192, Washington, DC, 1979. IEEE Computer Society.

[31] K. P. Sycara, M. Klusch, S. Widoff, and J. Lu. Dynamic service matchmaking among agents in open information environments. *SIGMOD Record*, 28(1):47–53, 1999.

[32] D. Trastour, C. Bartolini, and J. González-Castillo. A semantic web approach to service description for matchmaking of services. In *Proceedings of the International Semantic Web Working Symposium (SWWS)*, 2001.

[33] UDDI. Universal Description Discovery and Integration. Technical White Paper, 2000.

[34] P. Wurman, M. Wellman, and W. Walsh. A parametrization of the auction design space. *Games and Economic Behavior*, 2000.