A Semantic Matching Method for Clustering Traders in B2B Systems

Ricardo Ferraz Tomaz, Sofiane Labidi and Bernardo Wanghon
Intelligent System Lab.
Federal University of Maranhão
Campus of Bacanga, São Luís – MA, Brazil.
rferrer@ma.trf1.gov.br
labidi@uol.com.br
bernardojr@ma.trf1.gov.br

Abstract

This Work is part of the ICS (Intelligent Commerce System) project whose aim is to design and implement an effective Intelligent B2B Commerce System. This paper presents a practical application on Semantic Web and Web Services concepts where a matching procedure to associate potential buyers and suppliers in a B2B system is shown and the use of patterns and tools in the development of Semantic Web’s structure is proposed as an alternative for its implementation.

1. Introduction

One of the most important concepts of e-commerce is the matchmaking service. Matchmaking is the process in which agents, representing traders that are interested in having exchange of economic value are put in contact with potential counterparts to negotiate.

Unfortunately, actual proposed techniques for services’ matchmaking don’t provide a semantic automatic crossing between customer’s requests and suppliers’ advertisements.

We believe that the Ontologies and the Semantic Web technology could support advanced semantic interoperability in the next generation of e-commerce systems.

This paper presents succinctly the ICS [8], a B2B architecture that uses intelligent mobile agents to provide automatic negotiation with less cost and time within open and heterogeneous environments like the Internet. We focus on the matchmaking phase and propose a new approach to make trader clusters.

We suggest the web services’ approach as assembly blocks for the Matchmaker Agent. Though, as Web Service’s approach suffers a lack of semantics, we apply the Semantic Web perspective to overcome this problem.

In the next section, we describe the ICS architecture. Section 3 presents the ICS matchmaker agent. We emphasize our approach and its contributions based on the integration of the Semantic Web concepts and the Web Services technology. We detail the functional requirements and architecture of our matching agent, the algorithms used to perform matching between customer’s requests and supplier’s advertisements and the tools used to the matchmaker agent’s implementation. Section 4 presents a simple use case of the matchmaking process used in the ICS Matchmaker Agent. Finally, section 5 draws conclusions and the future works.

2. The Intelligent Commerce System – ICS

The ICS is a distributed Agent-Oriented implementation of Business-to-Business E-Commerce, where the Agents work in an open environment as Internet (or Intranet), moving through the network to meet at common negotiation areas. The use of mobile agents paradigm is very adequate to applications of the ICS nature because after the Matchmaker agent localizes the suppliers’ agents offering a service that satisfies customers’ agents, all the agents can move themselves to a region where they will interact locally. In this way, the transactions can be effected quickly.

The main ICS components are: Virtual Marketplaces (VMP), Regions, Matchmaker Agent, Mediator Agent, Modeling Agent, Ontologies Repository and Stereotype database as it shows figure 1.
Modeling Agent: The Modeling Agent objectives to inform the Mediator Agent of the preferences of each Trader Agent so that the Mediator Agent can interact in a particular way with each Trader Agent. To infer the profile of each Trader Agent the Modeling Agent gathers information from the interface, from domain (Ontologies) and from the database Stereotypes.

This section presented a high level vision of the ICS architecture. In the next section we concentrate in the implementation of the Matchmaker Agent, one of the main components of the ICS Architecture.

3. The matchmaking agent design

Traditionally, B2B has been accomplished through bilateral agreements. The EDI (Electronic Data Interchange) requires that each commercial partner negotiate individually with one another through a “one-to-one” model. The Semantic Web’s perspective provides the evolution to a “many-to-many” model, where each buyer has the possibility to negotiate their products or services to many customers.

In a marketplace, where cognitive software agents are used to represent the interests of customers and suppliers, an intermediary agent is normally used to accomplish the discovery of negotiation’s opportunity among them. This is done comparing the customer’s requests to the published advertisements by the suppliers.

We propose a Web Service's vision as assembly blocks for the Matchmaker Agent allied to the Semantic Web perspective, as an alternative to make possible the semantic crossing between the buyers' requests and the suppliers' advertisements.

3.1. Contributions of the semantic Web's perspective

The Semantic Web aims to transform the available information in the Web from “human-readable” to “machine understandable” [20][21]. Many researches have been accomplished in this area. Nowadays, the search of patterns and the development of tools to support their development has been the focus of works in many research centers [20][21].

We suggest the use of patterns and tools in the development of Semantic Web’s structure as an alternative for the development of a semantic matchmaking. On the web, the many service providers need to be able to advertise their services to an extremely wide and varied audience of service users. Without agreed upon content, many different mappings between diverse user communities need to be expressed. The expressive power of the Semantic Web languages is a novel and interesting alternative to remedy this problem.

The DAML-OIL (DARPA Agent Markup Language Ontology Interchange Language) [17] is, now the most
expressive ontological language. DAML-OIL has a formal semantic capable to be understood by humans and also, through an automatic way, by software agents. We suggest the DAML-OIL use as language for description of the customers and supplier’s requests and advertisements respectively.

The Semantic Web technologies enhance the utility of Web Services. The union of the Web Services technology with the Semantic Web view originates the Semantic Web Services [23].

3.2. Contribution of the Web services technology

The Web Services are being defended by industry as an advantageous approach for integrating existing enterprise systems. Such technology supports famous Architectures like e-Speak from Hewlett-Packard, ONE from Sun Microsystems and .NET from Microsoft.

The World Wide Web is changing. While initially conceived and implemented as a collection of static pages for browsing, it is now promising to become a web of services i.e. a dynamic aggregation of interactive, automated, and intelligent services that interoperate via the Internet. Multiple web services will interoperate to perform tasks, provide information, transact business, and generally take action for users, dynamically and on demand [19].

The Web Services offer new models for Web applications where sites change information dynamically on demand. These characteristics are particularly relevant to the e-business community since they provide the chance to do business through a fast and effective way.

A marketplace, in the ICS architecture, is seen as a Web Service that groups buyers and suppliers. For the buyers, the marketplace offers the opportunity to find a supplier with the best price or product. The suppliers benefit themselves with a greater number of buyers in potential.

While modeling the marketplace over the view of the Web Services architecture, we identified three roles:

- **Service Provider.** They are the suppliers; they must decide what to offer, how to describe their products and where their advertisements may be published.

- **Service Customer.** They are the buyers; they must express their requests.

- **Matchmaker Agent.** They have the goal to help the buyers and suppliers crossing.

The key to approach suppliers and buyers consists of making an automatic crossing of customer’s requests and supplier’s advertisements, seeking a semantic connection among them. DAML-OIL allows reasoning of classification on taxonomy’s concepts, as well as it allows connections among concepts. For instance, we may create sentences like “X” is part of “Y” to show that there is a connection among the concepts “X” and “Y” (“Y” contains “X”). In such case, if a buyer “A” details in his customer’s requests the “X” concept, the suppliers who offer the “Y” concept in its advertisement will be pleasing with some degree of satisfiability. The buyer’s request “A”, since “Y” contains “X”.

Even though DAML-OIL allows the description of requests and advertisements, which may be crossed automatically, each supplier may express his advertisements freely, which causes discovery problems, inter-operation and composition of Web Services. DAML-S [10] is an high level ontology for services to describe properties and capacity of Web Services and agents. It provides standardization on how to describe the Web Services functionalities (Service Profile), how it performs the task (Service Model) and how we can access it (Service Grounding).

Among the benefits supplied by DAML-S we emphasize the Web Service’s automation through the use of agents and the possibility of reasoning about the Web Service’s properties and capacity. We have particular interest in information presented at the Service Profile. The Service Profile provides the information needed by the matchmaker agent to discover a Web Service. The Service Profile class provides the functional description of the service, which is specified in terms of input, output, pre-conditions and effects.

- An **input** is what the Web Service requires to produce the expected output.

- An **output** is what the Web Service supplies as answer to a request. It’s the confirmation that a request was received and duly processed.

- The **Pre-conditions** represent the conditions of the real world that must be true so that the service may be performed (for example, to have a valid credit card and not overdrawn).

The **effects** are some change or action in the world, such as the sale of a product.

3.3. Functional and design requirements

The Matchmaker agent restricts itself to compute the degree of similarity (distance) among customers’ requests and suppliers’ advertisements. As result, it returns a binary match-pairs list (or record-match list) which contains the possible matching among customers and suppliers (a set of clusters).

When the Matchmaker agent is asked to make a search for a customer agent, this will be done inside an advertisement’s database. The matchmaker looks for possible supplier agents that satisfy the search’s requests and returns the possible supplier agents’ partners.

An important requirement for the Matchmaker agent is the search flexibility. The search for partners should not be restricted to the syntax aspect only. This is because equivalent terms incorporated in more generic concepts...
(subsumption), as well as terms having some relationships (such as aggregation, synonymy, antinomy, etc.), must be considered in the matchmaking process.

The figure 2 displays the ICS Matchmaking process. Customers and suppliers use DAML-S to express requests and advertisements respectively. The DAML-S parser extracts inputs, outputs and pre-conditions from DAML-S requests and Advertisements.

The Matchmaker Agent, made by the semantic matching engine and DAML-S Parser, classifies and crosses the requests inputs, outputs and pre-conditions to the advertisements inputs, outputs and pre-conditions. The ICS reasoning server, implement by a Description Logic reasoner, supports the classification of taxonomy of the terms found in the inputs, outputs and pre-conditions based on the application domain ontology.

There are several approaches for the development of the matchmaking algorithms. Nevertheless, DAML-OIL is greatly influenced by the description logics languages. That’s why, we proposed, in our work, the use of algorithms based on description logics (DL) languages.

### 3.4. Matching algorithms

The algorithms used in the ICS to implement the Matchmaker Agent are an extension of those proposed in [9]. The main idea of such algorithms is that an advertisement combines with a request when all the requests’ outputs combine with all the advertisements’ outputs as well as all the advertisements’ inputs combine with all the requests’ inputs. It guarantees that the supplier satisfies the criteria of customer’s purchase so that the customer offers the supplier all the inputs he needs to operate correctly.

We extend these algorithms so that they may consider the inputs, outputs and pre-conditions. The pre-conditions are very important to the negotiation phase because they turn possible the agents negotiate details like: payment mode, credit card type, number of stallments, delivery date, etc.

The main algorithm proposed in [9] is shown in figure 3.

```plaintext
Match(request)
{
    recordMatch = empty list
    forall adv in advertisement do
    {
        if match(request, adv) then
            recordMatch.append(request,adv)
    } return sort(recordMatch);
}
```

Figure 3: Main Loop

Observe that each advertisement (adv) stored in the advertisement’s database (advertisement) will be submitted as parameter to the match function, which will return true value when the request offers any degree of satisfiability on the customer’s request. In this case, customers’ identifiers and suppliers’ identifiers will be included in the record-match list.

The match function is agreed in two phases. The first phase compares the requirement’s output with the outputs of each advertisement stored in the advertisement’s database. The second phase compares the advertisements’ inputs with the requirements’ inputs. The first phase of the match function is shown in figure 4. The second part of the match function is quite similar to the first one. In the second the parameters order must be alternated, once that it will compare the advertisements’ inputs and requirements’ inputs.

```plaintext
OutputMatch(outputRequest, outputAdvertisement)
{
    globalDegreeMatch=Exact
    forall outR in outputRequest do
    {
        find outA in outputAdvertisement such as that
        degreeMatch=maxDegreeMatch(outR,outA)
        if (degreeMatch=fail) return fail
        if (degreeMatch<globalDegreeMatch)
            globalDegreeMatch=degreeMatch
    }
}
```

Figure 4: Output Matching
combinations, “exact”, “plugin”, “subsumes” and “fail”,
as shown in figure 5.

```
DegreeOfMatch(outR,outA)
  If outA=outR then return exact
  If outR subclassOf outA then return exact
  If outA subsumes outR then return plugin
  If outR subsumes outA then return subsumes
  Otherwise fail
```

**Figure 5. Degree of Combination**

If the advertisement’s output (outA) is equal or it is a subclass of the request’s output (outR), we say that the combination is “exact”. If outA is a group that includes outR, consequently outA may be plugged instead of outR we say that the combination is “plugin”. However, if outR is a group that contains outA, it means that the advertisement partially satisfies the request, then we say that the combination is “subsumes”. The “Fail” case occurs when there isn’t any connection among advertisement and request.

The algorithm makes the classification of the results of matching. The rules used to make the classification are shown in figure 6.

```
SortRule(match1,match2) {
  If match1.output>match2.output then
    Match1>match2
  If (match1.output =match2.output) and (match1.input>match2.input) then
    match1>match2
  If (match1.output =match2.output) and (match1.input=match2.input) then
    match1=match2
```

**Figure 6. Classification of the Matching List**

As we can observe, the criterion for the classification of the matching is firstly based on the outputs. The inputs are just considered as break ties’ criterion.

We still propose a matching on the pre-conditions similar to the one accomplished with the inputs and outputs as a second criterion of break tie.

### 3.5. Implementation tools

The ICS agents are implemented using the SOMA Platform (Secure and Open Mobile Agents) [3]. SOMA assures the mobility through the network, the security and the communication between agents.

The ICS Matchmaker Agent uses the JENA API [18] to load requests and advertisements using the DAML-S format. JENA is a toolkit developed at HP Bristol’s laboratory. The DAML-S parsers, implemented in JAVA language, extract inputs, outputs and pre-conditions from requests and advertisements. These parameters are sent to the reasoner DL RACER (Renamed ABox and Concept Expression Reasoner) [15] to be classified. Based on the obtained classification, the semantic matching engine defines the degree of combination present between customer’s request and supplier’s advertisements and send back to the customer a binary match-pair list containing the cluster of agents.

In the figure 7, we show an example of crossing concepts from a DAML-OIL ontology in the domain of Automobile Sales (Auto_Sales ontology). The auto_sales ontology is created using the OILED [24] ontology editor. The specification of such ontology is given in Appendix (cf. Page 8). To submit a query to the RACER, we used the graphical client interface RICE (RACER Interactive Client Environment) [25]. In this example, the query is “is the truck a vehicle?” and the answer is “T” (True). This is because the Truck concept subsumes the Vehicle concept in the auto_sales ontology, as we can observe in the treeview.

```
In the figure 7, we show an example of crossing concepts from a DAML-OIL ontology in the domain of Automobile Sales (Auto_Sales ontology). The auto_sales ontology is created using the OILED [24] ontology editor. The specification of such ontology is given in Appendix (cf. Page 8). To submit a query to the RACER, we used the graphical client interface RICE (RACER Interactive Client Environment) [25]. In this example, the query is “is the truck a vehicle?” and the answer is “T” (True). This is because the Truck concept subsumes the Vehicle concept in the auto_sales ontology, as we can observe in the treeview.
```

**Figure 7. Taxonomy’s tree of Auto-Sales Ontology**

In this context, consider that a customer request’s output specifies “truck”, and the advertisement’s output of an Automobile Sales specifies “vehicle”. The search engine must match the Automobile Sales to the customer, once that truck concept is subsumed by the vehicle concept.

The semi-structured database used to store domains’ ontologies and users profile in the ICS environment is Tamino XML Server developed by SoftwareAG. Our choice is justified by the fact that Tamino, in the current version, implements a XQuery [27] Engine and contains numerous improvements in interface support like Microsoft .NET, Java, EJB, SOAP, UDDI, XML:DB, C, JavaScript and HTTP Client API for ActiveX.

### 4. A Simple use case

Seeking to clarify the matchmaking process used in the ICS Matchmaker Agent, let’s consider the following simple use case.
Initially the domain ontology must be stored in the Tamino database through the ICS Web interface shown in figure 8.

![Figure 8 - Ontology definition form](image)

Before any matchmaking is made possible, the Matchmaker Agent must load from Tamino’s databases the domain ontology, insert it into a standard DIG [26] message and send it to the RACER reasoner.

When the Matchmaker Agent receives a request from a buyer, it will try to pair the request with the advertisements stored in the advertisements database based on the domain ontology previously loaded in the RACER reasoner.

Let’s consider that a sales agent produces the advertisement as shown in figure 9.

```xml
<profile:Profile rdf:ID="GM">
  <profile:ServiceName>GM</profile:ServiceName>
  <profile:PersonalContact>Dr. Gutemberg Farias</profile:PersonalContact>
  <profile:EMAIL>Gutemberg@gm.com.br</profile:EMAIL>
  <input>
    <profile:ParameterDescription rdf:ID="Price_Input">Price</profile:ParameterDescription>
  </input>
</profile:Profile>
```

![Figure 9 - Advertisement in DAML-S format](image)

Consider also a buyer agent specifies the request shown in figure 10.

```xml
<profile:Profile rdf:ID="DALCAR">
  <profile:ServiceName>DALCAR</profile:ServiceName>
  <profile:PersonalContact>Alessandro Martins</profile:PersonalContact>
  <profile:EMAIL>martins@dalcar.com.br</profile:EMAIL>
  <input>
    <profile:ParameterDescription rdf:ID="Price_Input">Price</profile:ParameterDescription>
  </input>
</profile:Profile>
```

![Figure 10 - Request in DAML-S format](image)

The Matchmaker Agent, following the algorithms presented in section 3.4, will first try to match the outputs (outR, outA). In this case, the request output has only one concept: “Sedam”. The advertisement output also has only one concept: “Car”. The degreeOfMatch function (see figure 5) returns “exact”, since the “Sedam” concept is subsumed from “Car” concept according to domain ontology Auto-Sales. Both agents will be invited to move to a virtual marketplace (VMP) where they can negotiate locally.

5. Conclusions

The Semantic Web perspective amplified the possibility of the e-commerce development tools. The use of multi-agent systems representing customers and suppliers at a marketplace, with autonomy to interact on behalf its users, offers the opportunity of a great increase of efficiency, reducing costs on business-oriented process. This benefits the final consumer as well as the whole productive chain.

The ICS architecture proposes the use of mobile agents. The mobile agent’s paradigm reveals adequate to applications of the ICS nature. After the Matchmaker agent localizes the suppliers’ agents offering a service that satisfies customers’ agents, all the agents can move themselves to a region where they will interact locally. In this way, the transactions can be effected quickly.

We are now investigating the improvement of our matching engine with better semantic matching techniques originated from research in the Information Retrieval (IR).

An interesting proposal for future works is to extend our matchmaking model to build a generic framework for matching agents i.e. independent of the application domain. The programs such as agents have no way to locate one that will perform a specific function. This process is named service discovery. The service discovery problem is a recurrent problem on many others applications domains such as, personal assistant, e-learning, meeting planning, etc. A framework implemented as a matchmaking engine API could be used to solve the discovery problem in applications that make use of agents in diverse domains, reducing the complexity and the cost of development of these applications.
6. References


[22] CMU Intelligent Software Agent – Semantic Web Services http://www2.cs.cmu.edu/~softagents/web_services_int o.html


7. Appendix: The Auto-Sales ontology. DAML-OIL domain ontology presents at the section 3.5