MatchMaking Shell

A Shell for Supporting Service Negotiation

Simone A. Ludwig, Omer F. Rana, Daniel J. Veit

Cardiff University – Distributed Collaborative Computing
5 The Parade, Roath, Cardiff CF24 3AA, UK
{simone.ludwig,o.f.rana}@cs.cardiff.ac.uk

University of Karlsruhe – Information Management and Systems
Englerstrasse 14, D-76131 Karlsruhe, Germany
veit@iw.uka.de

ABSTRACT. *Matchmaking is a key issue in electronic negotiation scenarios, and is often defined as the automatic ranking of a set of offers with reference to a set of requests. Once such an initial ranking has been achieved, individual negotiations between participants making offers and requests can be launched. Matchmaking is often an important phase in any e-negotiation system, as it enables the selection of suitable participants between which negotation should take place. We describe a generic MatchMaking Shell, which enables service independent matchmaking, and one which is able to support matching via user defined algorithms. Moreover, this shell can cater for domain specific data models (defined in XML). A definition of this shell and a prototype implementation are described. Finally, an evaluation to determine its effectiveness by measuring the performance and the matchmaking quality within a on-line bookshop is presented.*

RÉSUMÉ. *La génération des correspondances est une question clé dans les scénarios électroniques de négociation et est souvent définie comme le classement automatique d'un ensemble d'offres correspondant à un ensemble de demandes. Une fois ce classement réalisé, les négociations individuelles entre les offrants et les demandeurs peuvent être lancées. La génération des correspondances est généralement une phase importante pour n'importe quel système de négociations électroniques, car il permet la sélection des participants appropriés entre lesquels la négociation doit avoir lieu. Nous décrivons ici un support générique de génération des correspondances qui permet à celles-ci d'être indépendantes du service et fonction d'algorithmes définis par les utilisateurs. De plus, ce support accepte différents modèles de données (définis en XML) spécifiques au domaine. Une définition de ce support et l'implémentation d'un prototype sont décrits. Enfin, une évaluation de l'efficacité en mesurant la performance et la qualité de la génération des correspondances pour une librairie en ligne est présentée.*

KEYWORDS: *Service Negotiation, Service Matchmaking, Matchmaking Performance Analysis*

MOTS-CLÉS : *Négociation de service, génération de correspondance de service, analyse des performances de la génération de correspondance*

3e soumission à *JDS SI: E-negotiations: Models, Systems, and Processes*, le May 18th 2005.

1. Motivation for the Work

Software systems have been used as the basis for decision support within recent years. Such support can range from the provision of collaboration tools (sometimes also referred to as *Groupware*), to data analysis/machine learning tools that enable some quantitative analysis to support decision making. The availability of ubiquitous connectivity through the Internet, and increasing availability of XML-based data models, make such systems particularly effective – as a decision maker can now integrate results from a variety of different sources with less effort than before. Such decision support function may not be a 'one-shot' process – i.e. there is only a single interaction between the data source provider and the user. Although in the past such decision support tools were often based on such single, often one-way interaction. Data about a particular process would be captured, and subsequently analysed to discover trends that could facilitate decision support. Increasingly, such decision support is now being undertaken through a 'multi-shot' process – where each interaction is intended to allow convergence to take place towards some commonly-accepted outcome.

A key benefit provided by such Groupware tools is the support for negotiation between partners involved in a consortium. For instance, support for contract negotiation [SCH 03] provides a mechanism to allow a group of businesses to reach some consensus about the contents of a document that outlines their joint intentions. Increasingly, such Groupware tools have led to the emerging area of dynamic collaborations, generally referred to as "Social Networking" – whereby like-minded individuals can come together and negotiate common interests and themes. The increasing use of Internet technologies to support the formation of such social networks – such as `LinkedIn.com`, `Orkut.com` (created by search engine company `Google.com`) and `Friendster.com` – suggests how important such networks can be in structuring social interactions. According to `TrendIQ.com` [1] – a market analysis company specialising in social networking – the use of *LinkedIn* has increased by 400% from February to April 2004. In this instance, there is no direct negotiation provided between individuals or organisations, but tools are provided to allow individuals to negotiate with each other – thereby forming "chains" of interactions on common topics of interest. Many such systems utilise collaboration tools to support interaction between people – however, automated interactions between electronic services remains an important research area. Facilitating such automated interactions between services is a key theme of this work.

We use the term "E-Negotiation System" (ENS) to describe software that utilises Internet technologies, is capable of being deployed on the Web, and capable of supporting, aiding or replacing one or more negotiators, mediators or facilitators [KER 04]. A key requirement in such ENSs is to determine suitable partners to interact with prior to undertaking negotiation. Our focus is particularly on the discovery of such partners in the context of electronic services – whereby services can automatically try to discover other suitable services of interest. Often this process is undertaken through

1. See news story at `http://www.corante.com/getreal/archives/002993.html`

a registry service – whereby a client queries a registry to find partners/services that match a particular set of criteria. However, it is often the case that such queries need to be very precisely defined, and do not allow discovery of services which do not exactly match the required criteria specified in the query, but which "may" be of interest to the requester. We view e-negotiation as a multi-stage process which involves (partner and service are used interchangeably here):

1) finding suitable partners of interest with reference to a set of criteria. Partners that may match *some* of the criteria should also be flagged – with some indication of their "degree of relevance". Software that helps support such a function is often referred to as a "Broker" or "Facilitator" – one example of which is a MatchMaker;

2) interacting with these partners to reach some overall consensus via a variety of mechanisms – these include auctions, argumentation, etc;

3) agreement on terms reached as a consequence of (2). Perhaps, this agreement can also update search criteria for (1), to support subsequent discovery. This agreement is often referred to as a contract or a Service Level Agreement (in the context of electronic services).

The key contribution made here is the description of a generic MatchMaker shell, that can allow multiple types of functions to discover suitable partners for interaction. The shell can be customised with specialist matching criteria, and means to calculate "similarity". Performance issues in the use of such a shell are described with reference to an on-line bookshop application example.

This paper is organised as follows. Section 2 gives background information and shows related efforts in the area of service negotiation and matchmaking. In Section 3 the MatchMaking Shell is described in detail, with a prototype implementation and a matchmaking example of a book shop business case. Section 4 evaluates the prototype to determine its effectiveness by measuring the execution/messaging performance and the accuracy of the matchmaking process. Section 5 summarises and concludes this paper.

2. Background and Related Work

Matchmaking is one of the crucial tasks in service and negotiation oriented fields of computation. Locating and identifying the most suitable services among all available services is the key issue of matchmaking. This is also sometimes referred to as the "connection problem" – whereby one is required to identify suitable partners to interact within a collaboration. Therefore, several approaches have been proposed as likely solutions. Within most application scenarios, three different roles of agents can be identified: (i) agents which offer services (provider agents), (ii) agents which request services (requester agents) and, (iii) agents which mediate between providers and requesters, so called middle agents. Sycara et al. define a taxonomy of middle agents which include several types [WON 00]. However, they identify matchmaking agents as the most relevant middle agent. Matchmaking agents collect data and information from providers, and compute a relevance value among each incoming request and all known offers. Thereafter, a ranked list with the most relevant offers is transferred to the requester, allowing the requester to choose between them.

2.1. *Matchmaker*

The most relevant matchmaking approaches are those of Sycara et al. with their LARKS (Language for Advertisement and Request for Knowledge Sharing) approach (see [SYC 02]) and Veit et al. with their GRAPPA (Generic Request Architecture for Passive Provider Agents) approach (see [VEI 03]). Within the LARKS approach, four different kinds of matches are distinguished: (i) the exact match, (ii) the plug-in match, (iii) the relaxed match, and the (iv) profile match. In order to compute these matching mechanisms, different methods have been applied: (i) context, (ii) profile, (iii) similarity, (iv) signature, and the (v) semantical matching. However, these steps are processed successively in order to obtain best possible matching results. Hence, a signature match does not preclude the use of a semantic match – although the semantic match is the most complex form of matching mechanism supported. However, the matchmaking functions in the LARKS approach are static.

Within the GRAPPA approach, a multidimensional matchmaking mechanism is used, which takes into account arbitrary offer and request structures. Each offer and request consist of multiple criteria. Here, the semantics of the offer and the request play the key role. Normally, a matchmaking designer supports the dimensions for a concrete matchmaking application within a certain domain. Now, a concrete offer and request are matched by applying distance functions which compute the similarities between the individual dimensions of an offer and a request. Using certain aggregate functions on top, the similarities are condensed to an overall value, which represents the relevance of the concrete offer towards the concrete request. In Section 2.2 and [VEI 03], the distance functions are provided in more detail.

Another worthwhile matchmaking solution has been contributed by Subrahmanian et al. within their IMPACT (Interactive Maryland Platform for Agents Collaborating Together) platform (see [SUB 00]). The authors present a complete open multi-agent society based on matchmaking via a yellow-pages server. Matchmaking is performed via a term called `verbnounterm`. Each agent describes its services using one English verb and two nouns. These are matched under application of a so called Σ-hierarchy using a range- or a k-nearest-neighbour algorithm.

Ströbel and co-authors developed a matchmaking component for the discovery of agreement and negotiation spaces in electronic markets (see [STR 01]). The architecture and idea of this component is quite simple. In each specified attribute of a negotiable item it identifies negotiation spaces between an offer and a request. If there are positive values in one offer-request pair, a negotiation can be initiated.

2.2. *Distance Functions*

Distance functions take as input two parameters from a value domain and provide as output a positive real number. A distance function which fulfils a set of pre-defined mathematical preferences and constraints is referred to as a “metric" here. The properties of such functions are specified using mathematical bounds [VEI 03]. However, it is not trivial to obtain distance functions which provide a good fit of human judgement. It is therefore necessary to obtain functions which compare values from a domain (e.g. free text) and provide a judgement of the distance between values. For several types this is simple (e.g. numbers, dates, intervals). In other cases such as for the comparison of values such as free text, an entire research domain exists (such as distance functions used in Information Retrieval [SAL 89] – an example includes counting Term Frequency in both the query and the service description).

2.3. *Negotiation Support*

Research on matchmaking can be dated back to work on vague query answering, where the need to overcome limitations of relational databases was addressed, with the aid of weights attributed to several search variables [NOI 03]. Matchmaking plays a key role in supporting negotiation – (i) to find suitable partners to interact with, and (ii) to mark particular attributes in the requester query or the provider capability to determine which attributes can be further negotiated. By utilising such a *negotiable tag* – it is possible to restrict discovery of service providers (or service users), and thereby make the process more efficient. Such a tag does not specify to what extent a constraint, on a particular attribute, may be relaxed – only that there is a willingness on the part of the provider to support this. The SILKROAD framework provides the basis for this [STR 03] – and can provide both single-sided (only originator of a constraint is flexible towards the issue) and double-sided tagging of attributes. Once again, we can support such a tagging approach in the proposed MatchMaker Shell – allowing these tags to be included as annotations within the XML schema.

3. MatchMaker Shell

The MatchMaker Shell (MMS) supports a number of different matchmaking algorithms in the same architecture. Therefore, the key difference between the MatchMaker Shell and the LARKS approach is the following: the MatchMaker Shell allows different matchmaking algorithms to be plugged-in whereby for each algorithm a similarity value is calculated. LARKS in contrast only uses the set of filters (context, profile, similarity, signature and semantical) to progressively restrict the number of advertisements which are then candidates for a match. The Shell therefore can be modified through additional algorithms provided by a user, some of which can be used in conjunction with each other. Additional differences from LARKS include the ability to support both simple and complex data types, and reference these to an ontology description provided by a user. The Shell also acts as a hosting environment allowing a different matchmaking mechanisms to share data with each other. The architecture of the Shell comprises of a service requester interface, a service provider interface, a configuration interface, different matchmaking algorithms, an XML-based schema for complex data types, a reasoning engine, an ontology and a MatchMaker. Figure 1 outlines the architecture. Each of these interfaces allow external tools to either utilize data generated by the Shell, or provided data to it, based on these exposed interfaces.

The content maintained by the MatchMaker Shell is similar to that contained within a directory service in the Service-Oriented Architecture (SOA) approach [COL 04]. Generally, three architectural components are contained within a SOA: a service provider, a service consumer, and a directory service. The directory generally contains a list of services (with a limited set of properties associated with each service) that can be accessed by a consumer. In some instances, only details about the provider are made available (and not the list of services they provide). Providers register with the directory service and consumers query the directory service to find service providers. Most directory services typically organize services based on criteria and categorize them. Consumers can then use the directory services' search capabilities to find providers. Web Service technologies provide an implementation of the SOA approach, and make use of a registry service referred to as "Universal Description, Discovery and Integration". UDDI allows HTTP-enabled business services to be published, and subsequently searched, based on their interface. UDDI consists of three components: "white pages" to hold basic contact information and identifiers for a company, "yellow pages" to enable companies to be listed based on their industry categories (using standard taxonomies), and "green pages" to record interface details of how a Web service is to be invoked. Providers may register services with one or more registries (using the same identifier) – and may be discovered by search over one or more registries. Current UDDI implementations are however limited in scope – in that they only allow search to be carried out on limited attributes of services – namely on `service name` (which may be selected by the service provider), `key Reference` (which must be unique for a service), or based on a `categoryBag` (which lists all the business categories within which a service is listed). Furthermore, public UDDI registries may contain a lot of listings of business that do no longer exist, or sites that are no longer active. The interaction between UDDI registries is also an important concern – and there is no consensus on the question who should own the root UDDI registries. A key difference between UDDI and our approach is that UDDI only undertakes matching on `service name` and `categoryBag` – whereas we allow an arbitrary set of algorithms to be used as part of the "plug-in" match. Furthermore, we also support arbitrary XML Schemas to be used to describe services, allowing for matching on service properties (in addition to service name), contained within the Schema.

The MatchMaking Shell (MMS) can be driven in four different matchmaking modes which are: (1) Attribute Matching Mode (AMM): attributes of the service requester and the service provider are matched; (2) Value Matching Mode (VMM): value of the attributes of the service requester and the service provider are matched

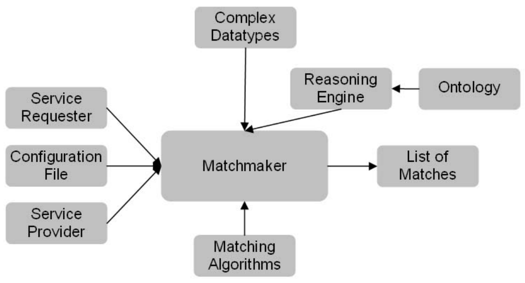

Figure 1. *Matchmaker Architecture*

using a configuration file; (3) Plug-in Matching Mode (PMM): performed by matching the profile of the service requester with a service provider using a specific algorithm provided by the user; (4) Semantic Matching Mode (SMM): attributes of the service requester are matched with an ontology description and a reasoning engine. The ontology is domain specific – and needs to be provided by the user. The MMS can also explicitly match parts of the service description which are service names, service attributes and service and metadata descriptions. The result of a match is either an exact or a partial match, and measured by a similarity value.

3.1. *Shell Concept*

The concept of the MMS is to provide different matchmaking algorithms to be used depending on the requirements of the user and the structure of the service data. The five main components of the MatchMaker shell are:

– Service Requester and Service Provider Interface: These two interfaces contain the service descriptions of the requester and the provider. If driven in the AMM, the attribute descriptions of the service are compared and if driven in the VMM, the values of the attributes are compared using an additional configuration file.

– Matchmaker: The MatchMaker is responsible for matching the service request with the service provider depending on the category chosen and the criteria selected.

– Matching Algorithms: Depending on the category chosen and the criteria defined, the matching algorithms are called by the MatchMaker accordingly. These matching algorithms contain e.g. distance functions for different data types.

– Schema for Complex Data Types: Complex data types are defined with respect to elements in an XML document. Such types may either include the basic elements in the schema (number and string), or may include user defined types. Definitions of complex data types need to be stored in a namespace, and referenced in the service description.

Comparison of a service request with a service provider description is achieved by applying different matching algorithms. This results in a value which indicates how similar the service requester and provider queries are. The similarity measure for AMM and VMM must be looked at separately. For AMM, the service attribute descriptions are compared syntactically. The similarity measure S_A is calculated as follows: $S_A = \frac{\sum_{i=1}^{n} sa_i}{n}$, $(0 \leq S_A \leq 1)$, where sa represents service attributes and n represents the number of service attributes. In the case of VMM, a configuration file needs to be provided, which specifies the data type of each service attribute, and the chosen distance function with a weight value. The weight value represents the quality of a match for that particular service attribute. Distance functions calculate the distance between basic and complex data type values.

The distance functions for the basic data types are `boolean`, `keyword`, `free text`, `number`, `date` and `time`. The simplest distance functions are the ones for `boolean` data types and `keywords`. The comparison of these attributes only results in the distance value of either 0 (when both values are the same) or 1 (when both values are different). The distance for the data types `number`, `date` and `time` are calculated as follows. For instance, consider the case of a number – the first step is to identify the smallest difference that can exist based on the particular representation scheme being used to encode the numerical value. Consider two numbers: α and β, which are represented to an accuracy of 2 decimal places. The smallest possible difference in this instance is 0.01. Therefore, the distance is calculated as: $\frac{|\alpha - \beta|}{0.01}$, where the numerator corresponds to the absolute value of the difference between the two numbers. For example, if we have two numbers 5.2 and 4.1, then the smallest unit is 0.1. The distance value is then calculated by $\frac{|5.2 - 4.1|}{0.1}$. For `free text`, the input is divided into token, then the stop words are removed. Stop words generally include articles such as 'the', 'an' etc – which do not represent useful information. In addition, each word is checked to see if it is a valid word (i.e. one which can be found in a dictionary). No word stems are calculated currently. This process is undertaken for both the service request, and also the description from the service provider. Both queries are converted into a word vector, and elements of the vector are compared with each other. The distance between the vectors is calculated as the number of common words in both the query and the service provider description.

The similarity S_V for VMM is calculated as: $S_V = 1 - \frac{\sum_{i=1}^{n} df_i}{n}$, $(0 \leq S_V \leq 1)$, where df_i represents the distance functions and n represents the number of distance functions. The PMM matches the profile provided by a service requester with that of a service provider. This implies that data types of attributes are compared in order to identify whether the signature of a service requester matches with the service provider. Plug-in matching is used if the user has some knowledge about the input/outputs to a service, data types associated with these input/outputs, and any associated conditions (such as pre- and post-conditions associated with these). For instance, in the case of the on-line book shop, data types of the elements contained in `serviceattributes` (see Figure 2) must be specified in a configuration file. When a user query is received, these data types are compared with those in the configuration file. Prior to defining a query, if the user has some knowledge of the data types, these may be used in the context of a PMM. If multiple matches are found to the user query, all of the matched service descriptions are returned to the user. Using PMM, the system therefore does not distinguish between these matched descriptions, and treats them interchangeably.

On the other hand, a SMM uses an ontology in case no syntax match is found. Such an ontology can be formulated by an exhaustive and rigorous conceptual schema within a given domain, a typically hierarchical data structure containing all the relevant entities and their relationships and rules (theorems, regulations) within that domain. This semantic description enables one to define services which can be used to enrich the matchmaking process. For instance, consider the attribute `price` in Figure 2. As prices can be defined in different currencies, the value associated with a price can be tagged. Hence, if a different currency was used, an automatic translation could be undertaken using a "currency" ontology. Relationships and rules can be defined on elements contained within such an ontology, e.g. different exchange rates.

3.2. *Prototype Implementation*

The current implementation of the MMS consists of the AMM and the VMM. A partial and exact match in both modes can be achieved. The service requester, the service provider and the configuration file are implemented as XML documents. The service requester and service provider files are parsed and the attributes and values are extracted, then the algorithm chosen depending on the matching mode is applied. If driven in VMM, distance functions with weight values are obtained from the configuration file and are applied to the values. The final step is the calculation of the similarity measure which is returned to the user.

On-line Book shop Example
An online book shop is described, where the user can browse, search for and purchase a book. To demonstrate how the AMM and VMM operates the search for a book is considered. The service description is split into 4 categories representing the service name, service attributes, service description and metadata information. Figure 2 shows the service requester file.

The service provider file for AMM has a similar structure to the service requester file, however this specific service provider file has the following changes: `title` tag was exchanged with `name` tag and `category` with `class` (compare Figure 3). In the service provider file for VMM the values for `author`, `isbn` and `price` are changed (see Figure 4). The configuration file shown in Figure 5 defines the distance functions for the VMM mode, for instance, the tag for the book title is specified in the service requester file as `name` and in the service provider file as `title`. The distance function in this case is `keyword`.

For the AMM, the MatchMaker parses the service requester description and the service provider description and compares all attributes of the service depending the

```
<?xml version="1.0" encoding="UTF-8"?> <service>
    <servicename>
        <name>SearchBook</name>
    </servicename>
    <serviceattributes>
        <title>Information Retrieval: Data Structures and Algorithms</title>
        <author>William B. Frakes, Ricardo Baeza-Yates</author>
        <isbn>0134638379</isbn>
        <category>computer</category>
         <price value="dollars">69.67</price>
        <publisher>Prentice Hall</publisher>
        <pages>464</pages>
    </serviceattributes>
    <servicedescription>
        <description>This guide discusses Information Retrieval data structures and algorithms.</description>
    </servicedescription>
    <metadata>
        <attr1>http://www.amazon.com/exec/obidos/tg/detail/-/0134638379/102-2173778-1724124?v=glance</attr1>
        <attr2>http://www.amazon.com/gp/reader/0134638379/ref=sib_dp_pt/102-2173778-1724124#reader-link</attr2>
    </metadata>
</service>
```

Figure 2. *Service Requester File*

```
<?xml version="1.0" encoding="UTF-8"?> <service>
    <servicename>
        <name>SearchBook</name>
    </servicename>
    <serviceattributes>
        <name>Information Retrieval: Data Structures and Algorithms</name>
        <author>William B. Frakes, Ricardo Baeza-Yates</author>
        <isbn>0134638379</isbn>
        <class>computer</class>
        <price value="dollars">69.67</price>
        <publisher>Prentice Hall</publisher>
        <pages>464</pages>
    </serviceattributes>
    <servicedescription>
        <description>This guide discusses Information Retrieval data structures and algorithms.</description>
    </servicedescription>
    <metadata>
        <attr1>http://www.amazon.com/exec/obidos/tg/detail/-/0134638379/102-2173778-1724124?v=glance</attr1>
        <attr2>http://www.amazon.com/gp/reader/0134638379/ref=sib_dp_pt/102-2173778-1724124#reader-link</attr2>
    </metadata>
</service>
```

Figure 3. *Service Provider File for AMM*

```
<?xml version="1.0" encoding="UTF-8"?> <service>
    <servicename>
        <name>SearchBook</name>
    </servicename>
    <serviceattributes>
        <name>Information Retrieval: Data Structures and Algorithms</name>
        <author>Ricardo Baeza-Yates</author>
        <isbn>0134634444</isbn>
        <class>computer</class>
        <price value="dollars">55.55</price>
        <publisher>Prentice Hall</publisher>
        <pages>464</pages>
    </serviceattributes>
    <servicedescription>
        <description>This guide discusses Information Retrieval data structures and algorithms.</description>
    </servicedescription>
    <metadata>
        <attr1>http://www.amazon.com/exec/obidos/tg/detail/-/0134638379/102-2173778-1724124?v=glance</attr1>
        <attr2>http://www.amazon.com/gp/reader/0134638379/ref=sib_dp_pt/102-2173778-1724124#reader-link</attr2>
    </metadata>
</service>
```

Figure 4. *Service Provider File for VMM*

```
<?xml version="1.0" encoding="UTF-8"?> <Configuration>
    <SRA>ServiceRequesterAttribute</SRA>
    <SPA>ServiceProviderAttribute</SPA>
    <ServiceName>
        <attribute>
            <SRA>name</SRA>
            <SPA>title</SPA>
            <DistanceFunction>KeyWord</DistanceFunction>
            <weightValue>0.45</weightValue>
        </attribute>
    </ServiceName>
    ...
    <ServiceMetaData>
        <attribute>
            <SRA>price</SRA>
            <SPA>price</SPA>
            <DistanceFunction>Number</DistanceFunction>
            <weightValue>0.1</weightValue>
        </attribute>
        ...
    </ServiceMetaData>
</Configuration>
```

Figure 5. *Configuration File for VMM*

attribute label tags and calculates the similarity value. For this example, 9 out of 11 attributes match – which results in a similarity value of 0.81818. For the VMM, the service requester and the service provider descriptions are compared and the similarity value is calculated. This evaluation makes use of configuration information, specifying which attribute tags of the service requester relate to the ones for the service provider. It also specifies the distance function and a weight value. The distance functions of all the attributes of the service requester and the service provider are calculated and the overall similarity value is determined. In this example the similarity results in a value of 0.99091.

4. Performance Evaluation

Matchmaking performance is often a key constraint in e-negotiation systems. Enabling a quick discovery of suitable partners for interaction can improve the overall negotiation process. To determine the effectiveness of the shell, two types of measurements are taken. The first set are performance measurements to investigate match execution time and messaging delays, and the second set are precision and recall measurements investigating the accuracy of the matchmaking process. Both of these criteria are important to evaluate the matchmaking process – as often the time to reach a suitable match needs to be compared with the accuracy of the match.

For the measurement the same domain of an online book shop was considered. The first measurements were conducted to see how the MMS performs. Eight set of measurements were taken, whereby the performance was measured having the client and the MMS locally running on one machine, compared to having the MMS on a remote machine. The second scenario is the most useful – when the MMS is being accessed over the Internet. The measurements were always taken for the AMM and the VMM. The communication between client machine and the remote MMS was implemented using Java RMI (Remote Method Invocation). The MMS creates some

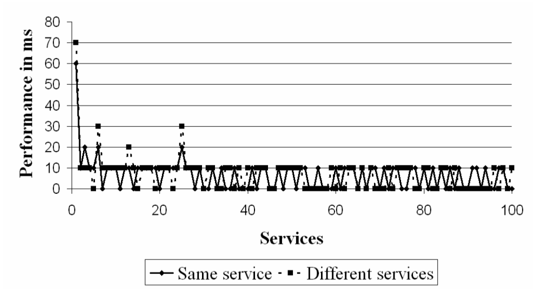

Figure 6. *Local AMM Performance Measurements*

remote objects, generates references to them, and waits for clients to invoke methods on these remote objects. The client application gets a remote reference to one or more remote objects in the server and then invokes methods on them. For the local measurements the same RMI setup was used – only having the client and the MMS running on the same machine. The conditions for the measurements were as follows:

– The service requester description was stored in a file, which represented the query, and was always the same.

– For the service provider, 10 different files for each mode (AMM and VMM) were replicated 10 times in order to have 100 services overall for the measurements. The service requester and service provider files follow the structure given in Section 3.2 consisting of 1 service name, 7 service attributes, 1 service description and 2 metadata descriptions.

– For each set of measurements the registry and the server were stopped and started in order to guarantee the same conditions.

– For each measurement two XML documents, one for the service requester and one for the service provider were compared with each other and the similarity value was calculated. For the VMM, an additional configuration file was used for the match-making process.

In Figure 6 the measurements for the AMM are shown comparing 100 service matches of the same type with 100 different service matches. The distribution shows an average value between 0 and 10 ms. The variation is due to the time precision of calculating execution time in Java. The two plots showing the same service matches, and those showing different service matches give a similar distribution. We can see the caching mechanism within the JVM (Java Virtual Machine) showing the first measurement greater than subsequent ones. This seems to suggest that the result from the first invocation is cached at the service requester. Figure 7 shows the measurements for VMM. It shows a similar distribution than the one shown in Figure 6.

In Figure 8, the remote measurements for the AMM are shown – with an average value of approximately 10ms. There are some peaks within the distribution which are

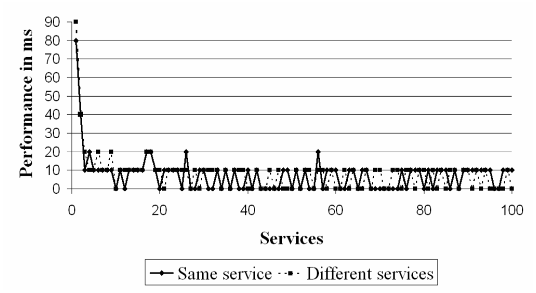

Figure 7. *Local VMM Performance Measurements*

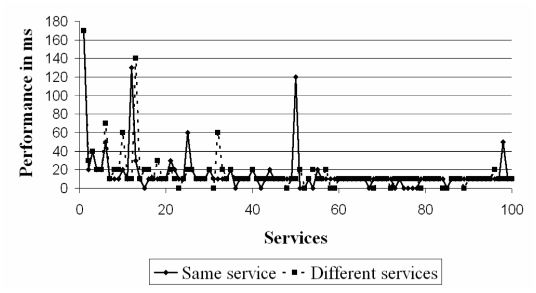

Figure 8. *Remote AMM Performance Measurements*

due to network load. The first measurement point is much higher, compared to values seen in the plot of local measurements. However, the consecutive measurement points give a similar result compared to the plot of the local measurements. The caching of objects within the JVM, once again, results in an average value of 10ms.

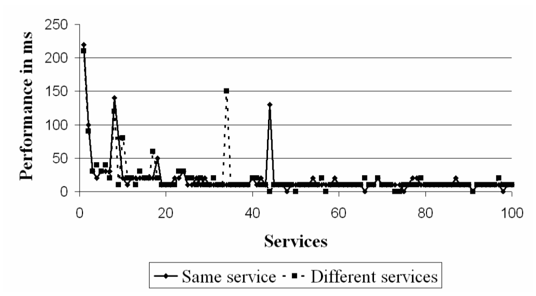

Figure 9. *Remote VMM Performance Measurements*

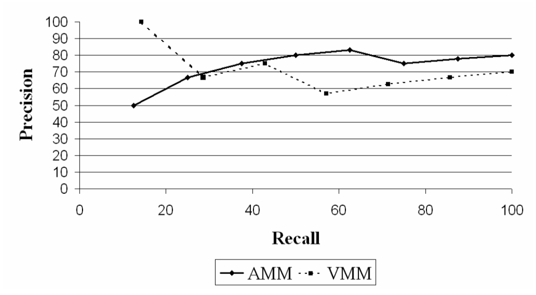

Figure 10. *Precision and Recall Diagram for AMM and VMM*

The measurements for the VMM are shown in Figure 9. The distribution shows a similar behaviour when compared to Figure 8. However, the first value measured is higher than the one measured for the AMM. This is due to the additional processing of the configuration file and the calculation of the different distance functions. This behaviour can also be seen comparing Figure 6 with Figure 7.

To calculate the accuracy of the match process we use Precision and Recall measures. Recall represents the fraction of total services found, compared to all the services that are relevant (for the given service request query) in the registry. Precision is the number of services that are relevant, compared to all those that have been found. Recall requires an ‘expert’ to specify which are the relevant services in the registry. To calculate these metrics, the set of all service providers are divided into two, those that are the relevant providers, and those that are not, with reference to a service request. Precision is calculated as “the ratio of the number of correct target predictions to the number of target predictions", and Recall as “number of correct target predictions to the number of target examples". The same service requester file and service provider files were used for these measurements. The set of service providers were taken and a predicted set was determined. For the measurement set, a border value was set to a similarity value of 0.6, which represents 60% of the service matches were allowed to be taken as real matches. The following precision and recall curves for AMM and VMM, shown in Figure 10 were obtained. The average recall values are 56.25% for AMM and 57.14% for VMM. The average precision values result in 73.48% for AMM and 71.14% for VMM.

5. Conclusions

The matchmaking process plays a key role in e-negotiation systems. It is often seen as the process of finding a set of suitable participants prior to the negotiation process. Matchmaking is particularly important in the context of electronic services, where the discovery process needs to be automated. Of significant concern within the matchmaking process is the time required to (i) find suitable services, and (ii) the accuracy of the result. Both of these aspects taken together can influence the efficiency of the negotiation process.

We outlined a generic MatchMaker Shell (MMS) that can be customised to support user-defined algorithms – and evaluated a prototype implementation of this using a numerical service. Within the evaluation we showed the performance of the MatchMaking Shell and provided first insights in the Precision and Recall values. It can be seen that an adequate quality of matchmaking is achieved, which enables the application of the MatchMaking Shell in electronic negotiation scenarios.

However, several issues have to be addressed in future research. The MatchMaking Shell needs to be evaluated in the wider context of specific negotiation scenarios. For these, additional distance functions have to be provided. Within these scenarios Precision and Recall measurements will be preformed in order to proof the quality of matchmaking of the shell in these different domains.

6. References

[COL 04] COLAN M., "Service-Oriented Architecture expands the vision of Web services", report , 2004, IBM Corporation, http://www-106.ibm.com/developerworks/webservices/library/ws-soaintro.html.

[KER 04] KERSTEN G. E., STRECKER S., LAW K. P., "Protocols for Electronic Negotiation Systems: Theoretical Foundations and Design Issues", *Electronic Commerce and Web Technologies (EC-Web 2004)*, 2004, forthcoming.

[NOI 03] NOIA T. D., SCIASCIO E. D., DONINI F. M., MONGIELLO M., "A System for Principled Matchmaking in an Electronic Marketplace", *Proceedings of World Wide Web Conference, May 20-24, Budapest, Hungary*, , 2003.

[SAL 89] SALTON G., *Automatic Text Processing*, Addison-Wesley, 1989.

[SCH 03] SCHOOP M., JERTILA A., LIST T., "Negoisst: a Negotiation Support System for Electronic Business-to-Business Negotiations in E-commerce", *Data and Knowledge Engineering*, vol. 47, num. 3, 2003, p. 371–401.

[STR 01] STRÖBEL M., STOLZE M., "A Matchmaking Component for the Discovery of Agreement and Negotiation Spaces in Electronic Markets", *Proceedings of the Group Decision and Negotiation Conference, La Rochelle France*, 2001, p. 61-75.

[STR 03] STRÖBEL M., WEINHARDT C., "The Montreal Taxonomy for Electronic Negotiations", *Group Decision and Negotiation*, vol. 12, num. 2, 2003, p. 143–164.

[SUB 00] SUBRAHMANIAN V. S., BONATTI P., DIX J., EITER T., KRAUS S., OZCAN F., ROSS R., *Heterogenous Agent Systems*, MIT Press, June 2000, ISBN: 0262194368.

[SYC 02] SYCARA K., WIDOFF S., KLUSCH M., LU J., "Larks: Dynamic Matchmaking Among Heterogeneous Software Agents in Cyberspace", *Autonomous Agents and Multi-Agent Systems*, vol. 5, 2002, p. 173-203.

[VEI 03] VEIT D., *Matchmaking in Electronic Markets*, vol. 2882 of *LNCS*, Springer, 2003, Hot Topics.

[WON 00] WONG H., SYCARA K., "A Taxonomy of Middle-Agents for the Internet", *Fourth International Conference on MultiAgent Systems (ICMAS-2000)*, 2000.