Chapter II
Challenges on Semantic Web Services

Maria Vargas-Vera
The Open University, UK

Miklos Nagy
The Open University, UK

Dominik Zyskowski
Poznan University of Economics, Poland

Konstanty Haniewicz
Poznan University of Economics, Poland

Witold Abramowicz
Poznan University of Economics, Poland

Monika Kaczmarek
Poznan University of Economics, Poland

ABSTRACT

The promise of being able to support Business-to-Customer applications with a rapidly growing number of heterogeneous services available on the Semantic Web has generated considerable interest in different research communities (e.g., Semantic Web, knowledge representation, software agents). However, in order to overcome the challenges of the current Web services, new level of functionalities is required in order to integrate distributed software components using existing Semantic Web standards. In this chapter, the authors discuss and suggest insights into new solutions to the main challenges in the area of Semantic Web services: composition, discovery and trust. For the first problem they suggest to use program transformation coupled with services’ descriptions. For the second problem (discovery of Web services) a solution based on the authors’ mapping algorithm between ontologies is suggested. While, for the last problem a solution based on fuzzy voting model is outlined. Through the chapter, the authors work with an investing scenario, in order to illustrate our suggested solutions to these three challenges.
INTRODUCTION

Web services technology has greatly advanced since its first emergence. Although, it has been adopted worldwide and is successfully used in the industry, it is still in the focus of attention of many research communities. The most active research is to automate interactions with and between Web services. One of the methods that may be used to achieve this is taking advantage of the semantic annotation of services and application of Semantic Web technologies, thus using Semantic Web services. A Semantic Web service (SWS) is defined as an extension of Web service description through the Semantic Web annotations, created in order to facilitate the automation of service interactions (McIlraith et al., 2001). These annotations are usually expressed using ontologies.

Ontologies are explicit formal specifications of the terms in the domain and the relations among them (Gruber, 1993). They provide the mechanism to support interoperability at a conceptual level. In a nutshell, the idea of interoperating Semantic Web services, being able to exchange information and carry out complex problem-solving on the Web, is based on the assumption that they share common, explicitly-defined, generic conceptualizations. These are typically models of a particular area, such as product catalogues or taxonomies of medical conditions. However, ontologies can also be used to support the specification of reasoning services (McIlraith et al., 2001; Fensel & Motta, 2001), thus allowing not only “static” interoperability through shared domain conceptualizations, but also “dynamic” interoperability through the explicit publication of competence specifications.

The promise of being able to support Business-to-Customer applications with a rapidly growing number of heterogeneous services available on the Semantic Web has induced a lot of interest within different research communities e.g. Semantic Web, knowledge representation, software agents. However, in order to overcome the challenges of the current Web services, new level of functionalities is required. In addition, although Semantic Web services are perceived as a very promising technology, as in case of any new technology, the problem of its maturity and its impact on the society and the business interactions arises. When it comes to business-to-customer interactions, the very important issues are social aspects connected with automatic transactions, especially the issue of trust within service discovery and composition.

Therefore, within this chapter, we discuss and suggest insights to new solutions to the main challenges in the area of Semantic Web services, namely composition and discovery as well as the issue of trust of Semantic Web services within those interactions. For the first problem we suggest to use program transformation coupled with services’ descriptions. For the second problem (discovery of Web services) a solution based on our mapping algorithm between ontologies is suggested. While, for the last problem a solution based on fuzzy voting model that may be used both within discovery as well as composition is outlined. The fuzzy voting model reflects the typical social situation in which one has to decide the opinion of which expert should be finally considered. Through the chapter, we work with an investing scenario, in order to exemplify our suggested solutions to these three challenges.

The main aim of this chapter is to delve into the issue of Semantic Web services and familiarize a reader with the main challenges lying ahead of them. We focus mainly on discovery, composition and trust and then, present possible solutions. In order to fulfil its aims, the chapter is structured as follows. First, we present the idea of Semantic Web services, their interactions as well as the main challenges in this area. Then, the issue of trust of Semantic Web services is discussed and we show the fuzzy voting model and its possible usage. In the next section, a solution to services discovery using Dempster-Shafer theory of evidence is pre-
sented. Then, we suggest an alternative solution
to the problem of composition of Semantic Web
services. Finally, the chapter concludes with a
summary and outlook on our future work.

SEMANTIC WEB SERVICES: AN
OVERVIEW

A Semantic Web service is defined as an extension
of Web service description through the Semantic
Web annotations, created in order to facilitate
the automation of service interactions (McIlraith et
al., 2001). Therefore, from the perspective of the
functionality offered, Semantic Web services are
still Web services. The only difference lays in
their description and the consequent benefits that
follow, namely the reduction of human involve-
ment in the performed interactions.

The contribution of the Semantic Web annota-
tions in case of Semantic Web services is twofold.
First, it provides ontologies acting like a shared
knowledge base and thus providing universal
dictionaries. In this way all Web services share
the same interpretation of the terms contained in
the exchanged messages and provide the bases
for the description of capabilities of Web services
(Paolucci et al. 2003). Secondly, it provides a logic
that allows performing reasoning. Only these two
mechanisms (ontologies and reasoning infra-
structure) allow to introduce the required level
of automation to performed interactions.

Interactions between Web services, and in
consequence also Semantic Web services, typi-
cally involve three or more parties: a user, one or
more providers and a registry that supports the
Web services during the transaction and possi-
bly mediates between the user and the provider
(Paolucci, Sycara, & Kawamura, 2003). In case
of Semantic Web services, a user is represented
by an intelligent agent or system acting on the
behalf of the user.

Let us consider the typical interactions with and
between Web services. First, a provider in order
to make his services available needs to advertise
their description with the registry (O’Sullivan
et al. 2002). Then the process of the provider
discovery i.e. Web services discovery follows.
This process is composed of at least three main
stages. First a user has to compile a request for a
service and send it to the registry. Within the next
step the registry is to locate an advertisement that
matches a request (Paolucci et al. 2003). Matching
process is not an easy task, depending on the ap-
proach taken. Needless to say, that it should take
into account the fact, that different parties with
different perspectives may (and in most cases do)
provide different description of the same service
(not only when it comes to a different terminology
used, but also a different level of the granularity
of the description). This step should at the end
guarantee that the selected Web service produces
the effects that the user desires (also when it comes
to the social and trust related aspects). The result
should be a list (preferably a kind of a ranking)
of potential providers among which the requester
has to make a selection. It is stated that there is
no general rule for the selection of the provider
and it is in fact domain specific decision as well
as it depends on the trust the user assigned to
a particular service or its provider. In the most
basic scenario, the provider with the highest score
among the matches returned is selected (service
that maximizes some utility function) or further
negotiations may follow. In general, this step ends
with signing the contract that defines the rights
and obligations of both parties. Invocation is in
fact the call for the execution of a service. Invoca-
tion of the contracted service usually entails
service execution and finally, the assessment of
the delivered results may follow.

Of course, individual services offer only
limited capabilities. The real power of Service-
Oriented Architecture (SOA) paradigm lies in the
possibility to easily create composite applications
and reconfigure them on demand. Therefore, in
fact two types of Web services may be identified,
namely atomic and composite. Atomic services
are Internet-based applications that do not rely on other Web services to fulfil consumer requests. A composite service may be defined as an ordered set of outsourced services (atomic as well as composite) working together to offer a value-added service. The composite services may be used as basic services in further compositions or may be utilized directly by end clients.

To summarize, the following interactions may be distinguished:

1. **Publication**: Making the Web service (description) available for use by others.
2. **Discovery**: Web service discovery is the process of locating Web services that can be used to request a service that fulfils some user needs.
3. **Filtering**: Identifying relevant services from the stream of new ones.
4. **Composition**: Creation of applications/business processes out of atomic and composite services.
5. **Negotiation**: Negotiating the terms of service provisioning and its characteristics.
6. **Contracting**: The process of contracting, via a discovered Web service, a concrete service fulfilling such needs.
7. **Invocation**: Call for service execution.
8. **Enactment and monitoring**: The actual provisioning of a service and monitoring its behaviour.

Ideally, we would like Web services to act autonomously and require minimal human intervention. Therefore, software agents or intelligent systems should be able to autonomously register with infrastructure registries and use the registries to discover services that they (or their users) need, and finally, create service compositions. Such automation is of interest of providers as well as customers and the automation of Web service interactions is the ultimate goal of many research projects.

The research conducted in the field of Semantic Web services may be divided into two main areas – SWS description methods and SWS interactions. First of these gathers groups working on languages and formalisms used to describe Web services and their capabilities. A Semantic Web Services representation is currently shaped by three main initiatives – OWL-S (Web Ontology Language for Web Services) (Martin et al. 2004), WSMO (Web Services Modelling Ontology) (Roman et al. 2005) and SAWSDL (Semantic Annotations for WSDL) (Farrell et al. 2007). The two first share a common attitude to a service (in order to automate service interactions define the description of a service with use of standardized language, concepts from the ontology build with the use of this language and providing the ability of being enhanced by its users) and in some sense are interoperable due to the activities of the research communities.

The second group focuses on a very wide universe of interactions with and between Web services. The ultimate goal is to automate and simplify the whole process of SWS management. This process includes following activities: SWS description creation and its publication, discovery and filtering, negotiation and contracting, composition, execution and monitoring as well as profiling. Some research projects like ASG (Fahringer et al., 2007), DIP (DIP 2007) or INFRAWEBS (Nern 2006) provided sophisticated platforms that support and automate all abovementioned Semantic Web services interactions. Some of the solutions proposed by these projects were even submitted to standardization bodies as industry standards proposals.

The researchers heavily investigate each of the SWS interactions. And in fact, each interaction is supported in most cases by a number of applicable mechanisms. There are, however, several research problems which still need to be addressed such as a composition and Semantic Web services discovery. Other issues are the social aspects of the desired automation. As the discovery and
composition are performed automatically, it is the system or an agent that selects the service, not the user. If an agent is to act on a user’s behalf it has to be equipped with the appropriate information on user’s preferences and desires. In the real world scenario, users do not only select and use services that are functionally meeting their requirements. Usually they select trusted providers, those were recommended by people their trust or those that were recommended by a big number of other users. This element is somehow ignored within most of current approaches. Within the next sections we present the possible solutions to composition and discovery of services and also show how the problem of trust may be tackled.

An investment scenario which is presented in the section presenting SWS composition is a detailed version of the outlined SWS usage scenario.

TRUST

In the context of Semantic Web services the requestors interact with unknown service providers, therefore they have to determine if they can trust such services or not.

In the context of the Semantic Web services trust can have different meaning therefore before we describe the problem let us define the basic notions of our argument.

- **Trust**: One mapping agent’s measurable belief in the competence of the other agents’ belief over the established similarities.
- **Content-related trust**: Dynamic trust measure that is dependent on the actual vocabulary of the mappings, which has been extracted from the ontologies and can change from mapping to mapping.
- **Belief**: The state in which a software agent holds a proposition or premise over a possible mapping of selected concept pair combination to be true.

Detecting and managing conflicting evidence over the possible meaning of the Semantic Web data is not always obvious. In practice the degree of conflict can differ considerably from case to case.

The problem of trust in this environment has different aspects, which have been investigated by different research communities. Most of the research is inspired by the traditional security problem where in order to secure a communication between two parties, the two parties must exchange security credentials (either directly or indirectly). However, each party needs to determine if they can “trust” the asserted credentials of the other party.
A peer to peer policy based trust approach is presented for establishing trust through iterative negotiation (Olmedilla et al., 2004) where both the service provider and the requestor possess a trust policy which is published either in a centralised registry or in distributed P2P environment. The negotiation is carried out by a matching algorithm which increases trust between the requestor and provider during the iteration revealing more and more information between the parties. Other security related aspects of trust can be represented through access control (Agrawal et al., 2004), which means the users must fulfil certain conditions in order to access certain services which are based on issued trust credentials by each Web service provider. Additionally these credentials are independent of other service providers. A Semantic Web service provider, acting as a verifier, can locally and autonomously decide whether access to his service should be granted or not. These solutions range from authorization based access control to authentication based access control of Web services which can be based either on public keys or identities. Other aspects of trust that focus on the provided answers that was requested from a service are based on the idea that trust in a Semantic Web service should take into account not only the functional suitability of the services but also their prospective quality offered to the end-users. This quality can be expressed directly as a proof for the provided answer using a specific language (Silva et al., 2004) or calculated with ranking algorithms (Vu et al., 2005), trust and reputation evaluation techniques e.g. compare the announced and actual service performance.

After analysing the related work on trust for Semantic Web services the main effort seems to be in security, encryption and authentication. Therefore, in this section we will introduce our proposal to trust.

During the Semantic Web services discovery we assume that different agents considered as different experts from the Dempster-Shafer point of view assess similarities between different services and the request. The application of agent is required because in the context of Semantic Web services it is hardly imaginable that isolated applications will be able to serve successfully the users’ ever growing requirements since the information normally available to human decision makers continues to grow beyond human cognitive capabilities. In such an environment a single agent or application limited by its knowledge, perspective and its computational resources cannot cope with the before mentioned scenarios effectively. Further, the combination of individual beliefs into a coherent view works in most of the cases and turns out that an aggregated belief function can provide more reliable mapping than several individual ones. However in certain situations the belief combination may produce an incorrect result even though before the combination a correct mapping could have been derived for the particular case based on individual beliefs. The problem occurs when the different experts’ similarity assessments produces conflicting beliefs over the correctness of a particular mapping. A conflict between two beliefs in Dempster-Shafer theory can be interpreted qualitatively as one source strongly supports one hypothesis and the other strongly supports another hypothesis, where the two hypotheses are not compatible. In this scenario applying Dempster’s combination rule to conflicting beliefs can lead to an almost impossible choice because the combination rule strongly emphasizes the agreement between multiple sources and ignores all the conflicting evidence through a normalization factor.

The counter-intuitive results that can occur with Dempster’s rule of combination are well known and have generated a great deal of debate within the uncertainty reasoning community. Different variants of the combination rule (Sentz, 2002) have been proposed to achieve more realistic combined belief. Instead of proposing an additional combination rule we turned our attention to the root cause of the conflict itself namely how
the uncertain information was produced in our model (Nagy et al. 2008).

In order to illustrate the problem let’s take the before mentioned scenario where Maria wants to invest money through on-line technologies. Our system has found a Web service which might be useful for converting the investment currency:

- ServiceDescriptor{interchange, foreign exchange, IBAN number}

However once the different experts assess whether the service is relevant or not the following situation occurs before belief combination:

- Expert 1: belief(Request-ServiceDescriptor)=0.05
- Expert 2: belief(Request-ServiceDescriptor)=0.90
- Expert 3: belief(Request-ServiceDescriptor)=0.10
- Expert 4: belief(Request-ServiceDescriptor)=0.92
- Expert n: belief(Request-ServiceDescriptor)=0.93

In this scenario the belief of Expert 1 and Expert 3 is in conflict with the other experts’ beliefs and due to the normalization factor the combination would result in an incorrect mapping therefore incorrect service selection. In our mapping framework the belief functions are considered as a method to model an expert’s beliefs, therefore the belief function defined by an expert can also be viewed as a way of expressing the expert’s preferences over choices, with respect to masses assigned to different hypotheses. The larger the mass assigned to a hypothesis is the more preferred the hypothesis will be. In this context the problem is how we handle the experts’ conflicting individual preferences that need to be aggregated in order to form a collective preference. Therefore, instead of modifying the combination rule we need to revise the conflicting information itself since this is what poses the problem and not the combination rule. Additionally, the similarity algorithm hence expert which contributes the conflicting belief can vary from mapping to mapping. In order to resolve the problem of choosing between different propositions for a selected service we propose as a method for assessing trust in Semantic Web services the fuzzy voting model developed by Baldwin (Baldwin, 1999). Fuzzy voting model has been used in fuzzy logic applications. However, to our knowledge has not been introduced in the context of Semantic Web services. In this section, we introduce briefly the fuzzy voting model theory with a simple example of 10 voters (agents) voting against or in favour of a Semantic Web service.

According to Baldwin (Baldwin, 1999) a linguistic variable is a quintuple \((L, T(L), U, G, M)\) in which \(L\) is the name of the variable, \(T(L)\) is the term set of labels or words (ie. the linguistic values), \(U\) is a universe of discourse, \(G\) is a syntactic rule and \(M\) is a semantic rule or membership function.

We also assume for this work that \(G\) corresponds to a null syntactic rule so that \(T(L)\) consists of a finite set of words. A formalization of the fuzzy voting model can be found in (Lawry 1998).

Let us consider the set of words \{Low_trust, Medium_trust, High_trust\} as labels of a linguistic variable trust with values in \(U=[0,1]\). Given a set “m” of voters where each voter is asked to provide the subset of words from the finite set \(T(L)\) which are appropriate as labels for the value \(u\). The membership value:

\[ X_{M(w|u)} \]

is taking the proportion of voters who include \(w\) in their set of labels which is represented by \(u\).

Let us start illustrating previous ideas with a small example - let us define our linguistic variable \(L\) as TRUST and \(T(L)\) the set of linguistic values as \(T(L)=(\text{Low}_\text{trust}, \text{Medium}_\text{trust}, \text{High}_\text{trust})\). The universe of discourse is \(U\)
which is defined as $U=[0,1]$. Then, we define the fuzzy sets $M(\text{Low\_trust})$, $M(\text{Medium\_trust})$ and $M(\text{High\_trust})$. Consider the following fuzzy sets for “Expert 1”:

- $M(\text{Low\_trust}) = [10:0, 20:1, 30:1, 40:0]$
- $M(\text{Medium\_trust}) = [45:0, 50:1, 60:1, 65:0]$
- $M(\text{High\_trust}) = [65.5:0, 70:1, 80:1, 90:1, 100:1]$

We need to introduce more opinions to the system i.e. we need to add the opinion of the other agents in order to vote for the best possible outcome.

Let us assume for the purpose of our example that we have 10 voters (agents). Formally, let us define $V=\{A1, A2, A3, A4, A5, A6, A7, A8, A9, A10\}$,

$T(L) = \{L_t, M_t, H_t\}$ where $V$ represents the voters and $T(L)$ describes the trust levels. The number of voters can differ however assuming 10 voters can ensure that:

1. The overlap between the membership functions can proportionally be distributed on the possible scale of the belief difference [0..1]
2. The work load of the voters does not slow the mapping process down

The random set $L=\text{TRUST}$ is defined by the table below. Note that in the table we use a short notation $L_t$ means Low\_trust, $M_t$ means Medium\_trust and $H_t$ means High\_trust.

We need to evaluate the different voting on the selected expert’s assessment e.g. for the assessment of “Expert 1” we get the results in Table 1. Then, we compute the membership value for each of the elements on set $T(L)$:

$X_{M(Low\_trust)} = 1$
$X_{M(Medium\_trust)} = 0.6$
$X_{M(High\_trust)} = 0.3$

$T(L) = \frac{\text{Low\_trust}}{1} + \frac{\text{Medium\_trust}}{0.6} + \frac{\text{High\_trust}}{0.3}$

A value $x$ is presented and voters pick exactly one word from a finite set to label $x$ (Table 2).

This gives a probability distribution on words:

$\sum \text{Pr}(L = \text{Low\_trust}|x) = 0.3$
$\sum \text{Pr}(L = \text{Medium\_trust}|x) = 0.6$
$\sum \text{Pr}(L = \text{High\_trust}|x) = 0.1$

Taken as a function of $x$ these probabilities form probability functions. They should therefore satisfy:

$\sum \text{Pr}(L = w|x) = 1$

$w \in TL$

At the end of the process we will have the trust evaluated by different voters. In our example

---

**Table 1.**

<table>
<thead>
<tr>
<th>Voters</th>
<th>A1</th>
<th>A2</th>
<th>A3</th>
<th>A4</th>
<th>A5</th>
<th>A6</th>
<th>A7</th>
<th>A8</th>
<th>A9</th>
<th>A10</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>L_t</td>
<td>L_t</td>
<td>L_t</td>
<td>L_t</td>
<td>L_t</td>
<td>L_t</td>
<td>L_t</td>
<td>L_t</td>
<td>L_t</td>
<td>L_t</td>
</tr>
<tr>
<td></td>
<td>M_t</td>
<td>M_t</td>
<td>M_t</td>
<td>M_t</td>
<td>M_t</td>
<td>M_t</td>
<td>M_t</td>
<td>M_t</td>
<td>M_t</td>
<td>M_t</td>
</tr>
<tr>
<td></td>
<td>H_t</td>
<td>H_t</td>
<td>H_t</td>
<td>H_t</td>
<td>H_t</td>
<td>H_t</td>
<td>H_t</td>
<td>H_t</td>
<td>H_t</td>
<td>H_t</td>
</tr>
</tbody>
</table>
the voters have produced different probability distributions on the words. In our case the voters have assigned low level of probability to the High_trust quantitatively 0.1 and 0.2. From the other side all other experts have received high level of probability for the High_trust quantitatively 0.7, 0.8, 0.9 for Expert 2, 4, n. Therefore as a result of the voting we can conclude that Expert 1 and Expert 3 assessments cannot be trusted and the service should be selected for the currency conversion task.

As we have seen in the previous sections, the main challenges and our proposed solutions for the fundamental components of the Semantic Web services can be applicable for capturing the data and metadata associated with a service together with specifications of its properties which we believe is a prerequisite for widespread adoption of Semantic Web services. Our solution provides means for agents to populate their local knowledge bases so that they can reason on Web services to perform automatic Web service discovery, and composition even if certain information is missing or ambiguously defined. This allows computer agents to automatically construct and execute Semantic Web service request and potentially respond to the service request.

**Validation of the Service Discovery and Trust**

For validating our service discovery and trust we have simulated mapping scenarios on the scientific publications domain. The evaluation was measured with recall and precision, which are useful measures that have a fixed range and meaningful from the mapping point of view.

Before we present our evaluation let us discuss what improvements one can expect considering the mapping precision or recall. Most people would expect that if the results can be doubled i.e. increased by 100% then this is a remarkable achievement. This might be the case for anything but ontology mapping. In reality researchers are trying to push the limits of the existing matching algorithms and anything between 10% and 30% is considered a good improvement. The objective is always to make improvement in preferably both in precision and recall.

We have carried out evaluation with the benchmark ontologies of the Ontology Alignment Evaluation Initiative (OAEI)\(^2\), which is an international initiative that has been set up for evaluating ontology matching algorithms. The experiments were carried out to assess how trust management influences results of our mapping algorithm. Our main objective was to evaluate the impact of establishing trust before combining beliefs in similarities between concepts and properties in the ontology. The OAEI benchmark contains tests, which were systematically generated starting from some reference ontology and discarding a number of information in order to evaluate how the algorithm behave when this information is lacking. The bibliographic reference ontology (different classifications of publications) contained 33 named classes, 24 object properties, 40 data properties. Further each generated ontology was aligned with the reference ontology.

The benchmark tests were created and grouped by the following criteria:

- **Group 1xx**: Simple tests such as comparing the reference ontology with itself, with

---

**Table 2.**

<table>
<thead>
<tr>
<th>Voters</th>
<th>A1</th>
<th>A2</th>
<th>A3</th>
<th>A4</th>
<th>A5</th>
<th>A6</th>
<th>A7</th>
<th>A8</th>
<th>A9</th>
<th>A10</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>L₃</td>
<td>M₁</td>
<td>L₃</td>
<td>M₁</td>
<td>M₁</td>
<td>M₁</td>
<td>H₁</td>
<td>M₁</td>
<td>M₁</td>
<td>L₃</td>
</tr>
</tbody>
</table>
another irrelevant ontology or the same ontology in its restriction to OWL-Lite.

- **Group 2xx:** Systematic tests that were obtained by discarding some features from some reference ontology e.g. name of entities replaced by random strings or synonyms.

- **Group 3xx:** Four real-life ontologies of bibliographic references that were found on the web e.g. BibTeX/MIT, BibTeX/UMBC.

As a basic comparison we have modified our algorithm (without trust), which does not evaluate trust before conflicting belief combination just combine them using Dempster’s combination rule. The recall and precision graphs for the algorithm with trust and without trust over the whole benchmarks are depicted on Figure 1. Experiments have proved that with establishing trust one can reach higher average precision and recall rate.

Figure 1 shows the improvement in recall and precision that we have achieved by applying our trust model for combining contradictory evidences. From the precision point of view the increased recall values have not impacted the results significantly, which is good because the objective is always the improvement of both recall and precision together. We have measured the average improvement for the whole benchmark test set that contains 51 ontologies.

Based on the experiments the average recall has increased by 12% and the precision is by 16%. The relative high increase in precision compared to recall is attributed to the fact that in some cases the precision has been increased by 100% as a consequence of a small recall increase of 1%. This is perfectly normal because if the recall increases from 0 to 1% and the returned mappings are all correct (which is possible since the number of mappings are small) then the precision is increases from 0 to 100%. Further the increase in recall and precision greatly varies from test to test. Surprisingly the precisions have decreased in some cases (5 out of 51). The maximum decrease in precision was 7% and maximum increase was 100%. The recalls have never decreased in any of the tests and the minimum increase was 0.02% whereas the maximum increase was 37%.

**WEB SERVICES DISCOVERY**

Another, yet equally important challenge, which needs to be addressed in the Web services area is the discovery of services. In order to use SWS or compose new applications, the appropriate SWS need to be first discovered. A Semantic Web service discovery may be defined as an automated process of finding appropriate SWS.

---

*Figure 1. Recall and precision*
Challenges on Semantic Web Services

(and in consequence also service providers) for the needs of a service requestor through a semantic service matchmaking. The semantic matchmaking process operates on the similarity measure that is defined adequately to the available representation of a service and a user goal definition.

There is a number of different semantic-based algorithms that are used to perform matchmaking between a request and an offer for each of the description initiatives already mentioned. They all were successfully implemented in the various research projects. For example the entire group of OWL-S based matchmaking algorithms may be distinguished. Algorithms falling under this category reason on the OWL-S ontology and OWL domain ontologies used to provide semantics of a service. Four levels of similarity are distinguished: exact, plug-in and subsumes and fail e.g. (Paolucci et al. 2002). Some more sophisticated methods were also defined that e.g. propose a matchmaking technique that goes beyond simple subsumption comparisons between a service request and service advertisements offering even more flexible matching by identifying not exact match but the best cover of the request etc.

From a number of approaches the one which got our attention relates to the case-base reasoning. Case-based reasoning (CBR) is an AI method that relies on a case-base of prior experiences and similarity criteria for comparing two situations and retrieving the cases more relevant to the new situation. Therefore, the problem of discovery of Web services can be seen as a problem of matching two cases in a CBR.

The matching of two Semantic Web services can be done either at syntactic or semantic level. As the syntactic matching is a text processing technique, we concentrate on the semantic matching. If we imagine that each of our services is subscribed to an ontology, then we can use the concepts and properties of the ontology in our algorithm for matching. Therefore, we suggest to use our Dempster-Shafer based Similarity (DS-Sim) framework for the discovery of services, especially for matching concepts and properties between two ontologies. Then, our proposal for service discovery is to use one of the algorithms for matching concepts and properties between two ontologies as described in (Vargas-Vera & Motta, 2004; Nagy et al., 2005, 2006a, 2006b, 2007).

In order to describe and share Semantic Web services, different Semantic Web Markup languages like OWL-S or WSMO have been defined as standard ontologies, consisting of a set of basic classes and properties. Usually a service request which can be considered as a query which needs to be answered from different service descriptions. As these service descriptions are expressed using different concepts from ontology, the problem of matching the query and description concepts, properties and constraints has been in fact in the focus of different research communities for several years.

The matching problem was first investigated by databases community where the issue of how different database schemas can be aligned during database integration was the primary interest of the community. A good survey of automatic schema matching approaches is described in (Rahm & Bernstein, 2001). With the emergence of the Word Wide Web and Semantic Web the matching problem remained the same, although the metadata i.e. ontologies that describes the concepts and properties has evolved considerably. Today these ontologies can easily be converted into logical formulas which can be used for reasoning purposes. In spite of the fact that Semantic Web services and data can be described with rich semantics which is based on sound logic formulas, the problem of matching these ontologies (Kalfoglou & Schorlemmer, 2003) still remains unsolved.

Further, the problem of mapping two ontologies effectively and efficiently is a necessary precondition to discover the Semantic Web services. During recent years different research communities have proposed (Choi et. al, 2006) a wide range of methods for creating ontology
mappings on the Semantic Web. The proposed methods usually combine syntactic and semantic measures by introducing different techniques ranging from heuristics to machine learning. While these methods perform well for certain ontologies, the quality of the produced mappings can differ from domain to domain depending on the specific parameters defined in the methods e.g. tuning similarity threshold. These limitations are particularly important for Semantic Web services discovery, since in order to find the correct set of services which corresponds to our needs one usually have to browse through multiple domains or sub-domains. Naturally, one may not expect that any mapping method will provide perfect mappings or perform equally well in each and every context, but we can think of a solution which would perform “well enough” for the users in such situations.

Therefore our objective is to come up with a method that:

1. Performs equally “well enough” in case of a domain changes.
2. Does not depend on any fine tuned internal parameters.

Additionally, in the context of Semantic Web services where the service and its parameters are specifically defined one cannot assume having large amount of data samples a’priori for machine learning. Our hypothesis is that the correctness of different similarity mapping algorithms is always heavily dependent on the actual content and conceptual structure of these ontologies which are different even if two ontologies have been created for the same domain but with different purposes. Therefore, from the mapping point of view these ontologies will always contain inconsistencies, missing or overlapping elements and different conceptualisation of the same terms which introduces a considerable amount of uncertainty into the mapping process. Further, the ontology mapping process can definitely be improved by applying background knowledge. The form of this background knowledge can differ depending on the mapping need and varies from WordNet (Giunchiglia et.al, 2006) based solutions to Semantic Web based (Sabou et. al., 2006) knowledge extraction. The use of any kind of background knowledge, however, does not necessarily increase our certainty of the ontology mapping, but it can also introduce further possibilities that can in fact be irrelevant from the mapping point of view. The question is how one can decide what alternatives for the mapping are worth investigating or select as a possible Web service that would help achieving the users’ task. This selection between possible alternatives is therefore based on subjective and uncertain information which is extremely dependent on the context of the mapping. We believe that if we can handle this uncertainty properly during the mapping the service discovery process can provide relatively good alternatives for the Semantic Web service composition.

**Our Solution to Mapping Problem**

In our ontology mapping framework (Nagy et al., 2005, 2006a, 2006b, 2007) we assume that one have only partial knowledge of the domain and can observe it from its own perspective where available prior knowledge is generally uncertain. Our main argument is that knowledge cannot be viewed as a simple conceptualization of the world, but it has to represent some degree of interpretation. Such interpretation depends on the context of the entities involved in the process. This idea is rooted in the fact the different entities’ interpretations are always subjective, since they occur according to an individual schema, which is then communicated to other individuals by a particular language. In order to represent these subjective probabilities in our system we use the Dempster-Shafer theory of evidence (Shafer, 1976), which provides a mechanism for modelling and reasoning uncertain information in a numerical way, particularly when it is not
possible to assign belief to a single element of a set of variables. Consequently, the theory allows the user to represent uncertainty for knowledge representation, because the interval between support and plausibility can be easily assessed for a set of hypotheses. Missing data (ignorance) can also be modelled by Dempster-Shafer approach and additionally evidences from two or more sources can be combined using Dempster’s rule of combination. The combined support, disbelief and uncertainty can each be separately evaluated. The main advantage of the Dempster-Shafer theory is that it provides a method for combining the effect of different learned evidences to establish a new belief by using Dempster’s combination rule. In order to illustrate our algorithm and the elements of the Dempster-Shafer theory take the before mentioned example where the requested service is exchanging currency:

\[
\text{convert_currency}(2500, \text{EUR}, \text{USD}, \text{marias_account_number}, \text{USD_account_number})
\]

Consulting background knowledge we augment our query for the terms convert, currency and account number. Once our extended query is finalized considering concepts and properties from the query we start extracting Semantic Web service descriptions from the available domain. Based on these descriptions we determine that there is a service which interchanges foreign exchange. Our primary objective is to create hypotheses for the possible mappings. In order to demonstrate our algorithm we show the main steps and associated values for the belief functions. The demonstration of the whole state space is not feasible since the number of subsets for a hypothesis with 8 variables is \(2^8\).

**Definition:** Frame of Discernment (\(\Theta\)) is a finite set representing the space of hypotheses. It contains all possible mutually exclusive context events of the same kind:

\[
\Theta = \{H_1, \ldots, H_n, \ldots, H_N\}
\]

In our system this corresponds to the possible mappings between concepts and properties i.e. the base entities that describe the domain e.g. conversion, exchange, currency, foreign exchange etc. Our algorithm iterates through all items in the Semantic Web services descriptors and creates several hypotheses that must be verified with finding evidences e.g.:

\[
\begin{align*}
H_1 (\text{mapping}) & \text{ Query}\{\text{convert}, \text{currency}, \text{account number}\} \\
& \leftrightarrow \text{ServiceDescriptors}\{\text{interchange}, \text{foreign exchange}, \text{IBAN number}\}
\end{align*}
\]

\[
\begin{align*}
H_2 (\text{mapping}) & \text{ Query}\{\text{convert}, \text{currency}, \text{account number}\} \\
& \leftrightarrow \text{ServiceDescriptors}\{\text{exchange}, \text{money}, \text{bank account}\}
\end{align*}
\]

Further we try to find supporting evidences for the hypotheses.

**Definition:** Evidence is available certain fact and is usually a result of observation. Used during the reasoning process to choose the best hypothesis in \(\Theta\). For finding evidences different syntactic and semantic similarity measures (Nagy et al. 2005) are considered as different experts are used which will determine belief mass functions for the hypothesis. We encounter certain evidence e.g. when we find similarity between term “convert” and exchange.

**Definition:** Belief mass function \((m)\) is a finite amount of support assigned to the subset of \(\Theta\). It represents the strength of some evidence and:

\[
\sum_{A \in \Theta} m(A) = 1
\]

where \(m(A)\) is our exact belief in a proposition represented by \(A\). The similarity algorithms itself produce these assignment based on different similarity measures. As an example let us consider the query that contains the concept “convert”. Based on background knowledge like WordNet we identify that the concept “exchange” is one
of the inherited hypernyms of the “convert” so after similarity assessment our variables will have the following belief mass value for the first hypothesis:

- \( m_1(Q, convert, currency, account number, ServiceDescriptor{ interchange, foreign exchange, IBAN number}) = 0.85 \)
- \( m_2(Q, convert, currency, account number, ServiceDescriptor{ interchange, foreign exchange, IBAN number}) = 0.91 \)

In practice we assess up to 8 inherited hypernyms similarities using different similarity measures e.g. Jaccard, Jaro-Winkler (considered as experts) which can be combined based on the combination rule in order to create a more reliable mapping. In practice we evaluate similarity with different experts for the whole hypothesis set with the size of \( 2^8 \). For our example the main associated belief masses are shown in Table 3.

**Definition:** Dempster’s rule of combination. Suppose we have two mass functions \( m_i(E_k) \) and \( m_j(E_k') \) and we want to combine them into a global \( m_{ij}(A) \). Following Dempster’s combination rule:

\[
m_{ij}(A) = m_i \oplus m_j = \sum_{E_k \cap E_k' \neq \emptyset} m_i(E_k) * m_j(E_k')
\]

An important aspect of the mapping is how one can make a decision over how different similarity measures can be combined and which nodes should be retained as best possible candidates for the match. The combined belief mass function which is based on both qualitative similarity and semantic measures and is a coherent view of different expert’s assessment of a hypothesis. Once the combined belief mass functions have been assigned the belief can be derived from the available information. The belief combination is computationnally really expensive operations since in our case the number of combination operations is \( 2^8 \).

**Definition:** Belief is the amount of justified support to \( A \) that is the lower probability function of Dempster, which accounts for all evidence \( E_k \) that supports the given proposition \( A \).

\[
belief(A) = \sum_{E_k \subseteq A} m_i(E_k)
\]

As last step we need to select the hypothesis in which we believe in most i.e. we need to select a Semantic Web service which we believe corresponds to our need. In our example the final beliefs for the different hypotheses are:

\[
H_1(\text{convert, interchange})=0.95
H_2(\text{convert, exchange})=0.45
\]

Therefore, we select the service where the descriptor contained:

\[
ServiceDescriptor{\text{interchange, foreign exchange, IBAN number}}
\]

The belief value for both \( H_1 \) and \( H_2 \) has been calculated as a sum of variable subsets where the hypothesis and subset contain common variables. Therefore the demonstrated belief values are derived from the available state spaces and are just values to demonstrate this example.

Based on preliminary experiments we have verified (see section validation of the service discovery and trust) that our mapping approach for Semantic Web service can improve the quality.

<table>
<thead>
<tr>
<th>Table 3.</th>
<th>Expert1(m₁)</th>
<th>Expert2(m₂)</th>
<th>Expert3(m₃)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( H_1 )</td>
<td>0.85</td>
<td>0.91</td>
<td>0.99</td>
</tr>
<tr>
<td>( H_2 )</td>
<td>0.33</td>
<td>0.43</td>
<td>0.45</td>
</tr>
</tbody>
</table>
of the discovery on the Semantic Web. However, it poses numerous challenges as well e.g. if the service is described only with natural language text or the background knowledge cannot describe properly the domain.

Further a problem could arise when two semantic services have the same belief value of $H$ then the system should use a heuristic to decide which one can be used. This can be seen as the old problem of conflict resolution in rule based systems. In our view, heuristics is necessary in this case since there is no easy way to create an another method that would provide the better choice then the first one, especially considering the fact that using one kind of calculation we have reached equality between two services. The heuristics has to consider different aspects of the Web service e.g.:

- Who is the publisher?
- When was the service published?
- How well the service is described in its repository?
- What is the difference between some reference rates and the provided service?
- How much do we need to pay for the service offered by the service provider?

Additionally, this heuristics can also be ranked according to their importance. As an example assume that we have found two services for the exchange rate conversion. The first is offered by a bank and the second is by a financial services company. Based on the publisher information we would normally choose the service of the bank because we think is more reliable, however it is really important for us how much we can lose during the conversion. Based on the European Central Bank’s reference rate we realise that choosing the financial services company we would be better off after conversion therefore we choose the second service.

Further there is also the possibility that using our matching algorithm we do not find any service that would match our need. This can happen if the mapping algorithm assigns all belief masses to the empty set therefore, indicating ignorance on the correctness of the mapping. In this case the mapping cannot be carried out and the user needs to indicate his/her choice between the available services.

An algorithm for matching Semantic Web services called Matching-SWS (Figure 2) is outlined below.

Algorithm

Match($S_1, S_2$) where $S_1$ and $S_2$ are Semantic Web service:

![Figure 2. Matching SWS](image)
1. Compute masses $m_i$.
2. If there is conflict between the assessed hypotheses use the voting model to assess trust.
3. Combine masses using Dempster-Shafer theory of combination and obtain $H_i$.
4. If mappings $(H_i = H_{i+1})$ use heuristic to decide for one $H_i$.
5. If no matching was found i.e. all variables has been assigned to the ignorance provide the user all the available services so the user can decide which one to choose.
6. Use the selected services as bases and assess trust based on the voting model in order to select the one which needs to be used for the composition.

Finally, to our knowledge Dempster-Shafer has not been used in Semantic Web services as a way to discover Semantic Web services. It has been investigated in the context of information fusion (Yu et al., 2005) where several successful applications have been published. For the ontology mapping problem it has started to be evaluated in different context, but to date Bayesian approaches are dominant for this problem. The main disadvantage of the Dempster Shafer theory is it’s the scalability for large domains since the belief combination is computationally expensive operation. Our ongoing research is devoted to this problem since both for ontology and Semantic Web services’ mapping this is the key obstacle that needs to be resolved for further applications.

**COMPOSITION OF SERVICES**

The process of developing a composite service may operate on composite as well as on atomic services. Within the typical process of service composition a composer tries to solve the user requirements by composing the services advertised by the providers. Usually, the composer takes the functionality of service to be composed as an input and as outputs returns the process model describing the composite service. The process model contains a set of selected atomic services, as well as definition of a control and data flow among them.

Although the composition process is quite well supported by a number of different initiatives, the achieved results still leave a space for improvement. The algorithms used to perform automatic composition range from those using workflow techniques, to those taking advantage of the achievements from the AI area such as: situation calculus (e.g. (McIraith & Son, 2002)), rules (Ponnenkanti & Fox, 2002), theorem proving, HTN (Wu et al. 2003) or STRIPS based solutions (Rao et al. 2004). The algorithms used to perform SWS composition differ in terms of their precision and quality of the returned results, efficiency, complexity, performance as well as the form (e.g. BPEL process description) and the scope of the returned result (whether the generated plan is ready to be deployed on the execution engine and then executed, consists only out of service specifications and service implementations still need to be selected to the plan generated, the control flow is specified but the data flow still needs to be defined etc.).

In our view an elegant solution to the composition of Web services is the one which sees the problem of composition as a planning problem (Drumm et al., 2007; Meyer, 2006). However, it has some important limitations i.e., the complexity of algorithm and there is not guarantee that the result might be found as the approach uses enforced hill climbing algorithm. The second problem of this method is that it is not possible to create intermediate variables.

Therefore, although a number of different approaches exist, the problem of composition has not been solved and more research work on composition of Semantic Web services is needed. Some inspiration may be taken from the program transformation area as is shown further in this section.
The presented idea of composition comes from early days of AI where researchers were interested in combining programs (flows of control) in an integrated manner for functional programming (Burstall & Darlington, 1977) and also for logic programs (Vargas-Vera, 1995; Vargas-Vera and Robertson 1994). Let us define a composition of Semantic Web services in the context of program transformation as a tight integration of the specification/code of services. The composition operator is a binary operator that takes two services at the time and without a lost of generality the composition can be performed number of times.

Web services can be seen as functions in Functional Programming Languages. Complex services can be obtained by combining simple services. In the simplest case, composition can be reduced to compose functions like in mathematics. If we take this perspective, then a semantic service is a function with parameters, preconditions & effects, inputs and outputs. However, the composition of services can be more complex. Semantic services can be described as logic statements. Then the composition problem can be seen as merging logic statements with constraints. The work reported in (Vargas-Vera, 1995) describes an automatic system which combines logic programs using program histories. This approach could be adapted to the composition problem since each service can be seen as a logic program and we also have histories for each service describing its functionality and restrictions imposed by the service creator. The composition problem is explained by means of an example in the following section.

We envision the composition of Semantic Web services which will be generated on the fly using either combining specifications (Fuchs, 1994) or combining the code associated to each service using services’ descriptions which guide synchronisation of the flows of control. Our vision of composition system is having a library of simple services and to produce more complex services by program composition or by composing specifications of the services. Then, the synchronization of flows of control is defined in a Joint Specification.

**Composition Scenario**

Maria has just found a new hobby and decided to start investing money through online channels. During her research on available platforms that could be used to do that she discovered that there is a solution that allows for service composition. In addition, there exists a set of services that support financial domain. These services provide a wide range of granularity and functionality, from the simplest ones such as currency converters and stock quotes tickers to the more advanced such as services providing trend for user investments. Of course, Maria wants to use only trusted services.

Let us consider an investment scenario where Maria wants to prepare a custom application made out of the available Web services. She is aware that not all of the functionalities she wants to employ are readily available. Nevertheless, services are prepared in a manner that allows for their composition.

Maria tries this solution and starts with a simple task of creating a service that will allow her to invest a spare sum of money. The workflow is composed of the following tasks:

1. Checking the account balance in order to assess whether there is superfluous amount of money available.
2. Checking the currency exchange rates between euro and US dollar in order to establish whether investment in arbitration game is feasible and brings profits.
3. Assessment of the trend of euro and dollar exchange rates.
4. Risk analysis of the investment.
5. Confirmation of the transaction.
This is a simple scenario, which can be split by a broker into several simple Semantic Web services such as check-account-balance, currency-exchange-rate, trend-analysis, risk-analysis, convert-currency, and investment-trigger. A formal specification for Maria’s request is shown below. The request is written in Prolog language.

```
request :- check_account_balance(marias_pin_num, marias_account_number),
currency_exchange_rate(EUR, USD, 02-10-2007), trend_analysis(USD, 02-08-2007, 02-10-2007),
risk_analysis(USD),
convert_currency(2500, EUR, USD, marias_account_number, USD_account_number),
confirm_investment(marias_activation_key)
```

Imagine the idealistic scenario where our library of services contains already two services which can be used to build a new service on the fly. The initial library of services is not meant to be complete.

```
investment_situation(currency, start_date, end_date) :-
trend_analysis(currency, start_date, end_date),
risk_analysis(currency).
```

The second service is perform_arbitration_game defined as follows:

```
perform_arbitration_game(PIN, account_number1, account_number2, currency1, currency2, date, key) :-
```

Please note that variables currency from the first service and currency2 from the second service relate to the same concept.

### Combined Service

The composition can be specified by means of a joint specification where the synchronisation of the two flows of control is requested:

\[ JS \rightarrow S1 \circ S2 \]

where S1 and S2 are services:

```
proceed_with_arbitration(currency1, currency2, start_date, end_date, date, PIN, account_number1, account_number2, key) :-
investment_situation(currency, start_date, end_date) and perform_arbitration_game(PIN, account_number1, account_number2, currency1, currency2, date, key).
```

The combined program produced by using the above JS is shown as follows:

```
proceed_with_arbitration(currency1, currency2, start_date, end_date, date, PIN, account_number1, account_number2, key) :-
trend_analysis(currency2, start_date, end_date),
risk_analysis(currency2),
check_account_balance(PIN, account_number1),
currency_exchange_rate(currency1, currency2, date),
convert_currency(amount, currency1, currency2, account_number1, account_number2),
confirm_investment(key).
```
currency_exchange_rate(currency1, currency2, date),
convert_currency(amount, currency1, currency2, account_number1, account_number2),
confirm_investment(key).

In our working example, we can observe that we are dealing with a hybrid definition of Semantic Web services. Each service appears to have logic and a numerical component that need to be satisfied. Therefore, we could think in dividing our HKB (Hybrid Knowledge Base) in two components, the first one will contain logical statements and axioms associated to them. In turn, the second one should contain numerical constrains which need to be solved using a constraint solver. A general framework using a several distributed reasoners which merge their results together (Knowledge fusion), like the one presented by Bo Hu (Bo, 2003) on Fusion knowledge, is to be used. Bo Hu’s approach divides a HKB into smaller components each containing the homogenous knowledge that can be processed by a different specialized reasoning system. Results of the inferences are then consolidated. This solution should alleviate the problem that reasoning is a time consuming task.

Coming back to our working example, it is worth to note that the description of each service is needed in order to synchronise the control flows. If information in the descriptions of services states that the flows of control cannot be synchronised as requested the composition cannot be performed. A mathematical property that can be guaranteed after the composition is that the meaning is preserved i.e. the solutions generated for each individual service are given to the resulting composed service. This is guaranteed if during the composition process only operations that preserve the meaning are used. Examples of such operations that preserve meaning are for instance fold/unfold (replacement of a procedure call by its definition) operations (Tamaki & Sato, 1984).

Validation of the Composition Services System

Our composition system has been tested on composing logic programs, refers to the work reported in (Vargas-Vera 1995). This composition system was written in Sictus Prolog running in a windows and Unix operating system. The properties observed in the composition system are the following:

The combined services generated automatically are different from the ones that programmers produced. This is because there is an automatic optimization of variables in the code generated automatically and also by transforming the services using the fold/unfold operation the code of the original services looks rather different. However, the solutions generated for each individual service are given to the resulting composed service answers. The latest as we already mentioned is guarantee by the use of operations which preserve meaning for instance, fold/unfold operations.

The specification defined by users is validated automatically by the system using the history of each service in order to ensure that the composition is feasible. If the composition is not feasible then a message is displayed to the user telling that the services cannot be combined.

In our first implementation the join specification needs to be provided in pseudo language (like the one shown in example) and then internally is transformed into First Order Logic expressions. However, in future the request (of the investment scenario) could be submitted by a user in natural language. It could then be processed by a natural language parser that would map it into first order logic predicates.

Finally, a comprehensive use of the composition system and uses of “program history” in applications such reverse engineering and Prolog Explanation system can be found in (Bowles et al. 1994). Furthermore, a program generator for simulation models in Ecology using our composition system is reported in (Castro 1994).
FUTURE WORK

In the future research we would like to investigate how our proposed approach can be applied in the context of Enterprise Application Integration (EAI) which still requires human interaction to a large extent e.g. the human programmers have to manually search for appropriate Web services in order to combine them in a useful manner. This is important since the majority of IT development efforts of different organisations is still focusing on how to find, extract, and interpret information from highly heterogeneous systems. Our long term objective is to provide the possibility of helping people to develop and to manage services more efficiently and effectively on the Semantic Web. Our proposed agent based solution for service discovery is especially promising considering the fact that the Semantic Web is a web of distributed knowledge bases, and agents can read and reason about published knowledge with the guidance of ontologies. Further the reasoning capabilities of these agents are limited by the “lack” of proper information which is an obstacle for creating applications that fulfill their missions autonomously and intelligently.

CONCLUSION

The main contribution of this chapter is to provide new insights to three challenging problems that in our view they still not completely solved in the area of Semantic Web services. Firstly, we suggest a solution to the composition problem. Our method is based on program transformation of specification/code of each of the two services participating in the composition. The composition makes use of services descriptions to guide the whole process. This solution is not expensive in time complexity since we do not need to build the planning graph as in solutions based on planning algorithms. For the discovery of Semantic Web services, a solution based in Dempster-Shafer theory of evidence is suggested. We also, outline a model to assess trust based on fuzzy voting model. To our knowledge the fuzzy voting model has not been used to solve the problem on trust on Semantic Web services. Furthermore, Theory of Evidence has not been applied to the problem of discovery of Web services in the way presented in this chapter. Therefore, we believe that an agent framework (like the one suggested in this chapter) fits nicely with two of the problems in Semantic Web services namely discovery and trust assessment.

ACKNOWLEDGMENT

This work has been supported partially by a Marie Curie Transfer of Knowledge Fellowship of the European Community’s Sixth Framework Programme under the contract number MTKD-CT-2004-509766 (enIRaF).

REFERENCES


Challenges on Semantic Web Services


**KEY TERMS AND DEFINITIONS**

**Dempster-Shafer Theory of Evidence:** A statistical uncertain reasoning model which uses belief functions for combining separate pieces of information (evidence) to calculate the probability of an event.

**Fuzzy Voting Model:** A model which used different voters for fuzzy sets in order to determine the membership value.

**Service Discovery:** The capability of automatically identifying a software service in Internet which matches the service request criteria.

**Services Composition:** In a Service Oriented Architecture (SOA) the operation which ag-
aggregates or combines small services into larger services

Trust: The ability to assess the credibility of source information based on different criteria

Unfold and Fold Operations: Techniques for source level program transformation. Those operations allow to transform clear but inefficient programs into more efficient equivalent programs.

ENDNOTES

1. http://www.daml.org/services/owl-s/1.1/related.html#wsmo