A Cognitive Trust-Based Approach for Web Service Discovery and Selection

Ali Shaikh Ali, Simone A. Ludwig, Omer F. Rana
Department of Computer Science
Cardiff University, UK
{Ali.Shaikhali,Simone.Ludwig,Omer.F.Rana}@cs.cardiff.ac.uk

Abstract

Provision of services within a virtual framework for resource sharing across institutional boundaries has become an active research area. Many such services encode access to computational and data resources, comprising single machines to computational clusters. Such services can also be informational, and integrate different resources within an institution. Consequently, we envision a service rich environment in the future, where service consumers are represented by intelligent agents. If interaction between agents is automated, it is necessary for these agents to be able to automatically discover services and choose between a set of equivalent (or similar) services. In such a scenario trust serves as a benchmark to differentiate between services. In this paper we introduce a novel framework for automated service discovery and selection of Web Services based on a user's trust policy. The framework is validated by a case study of data mining Web Services and is evaluated by an empirical experiment.

1. Introduction

The pervasiveness of online services facilitates a novel form of communication between individuals and institutions, thus supporting new flexible work patterns and making institutional's boundaries more permeable. Upcoming standards for the description and advertisement of, as well as the interaction with and the collaboration between, online services promise a seamless integration of business processes, applications, and online services over the Internet. This seamless integration of business processes leads to the need of having a semantic service discovery process which allows a user to find the "right" service based on semantic descriptions rather than keywords. As a consequence of the rapid growth of Web Services, the issue of trust becomes central for businesses. There are no accepted techniques or tools for specification and reasoning about trust. There is a need for a high-level, abstract way of specifying and managing trust, which can be easily integrated into applications and used on any platform. A typical application requiring a formal trust decision becomes apparent when service consumers are faced with the inevitability of selecting the "right" service. The distributed nature of these services across multiple domains and organizations, not all of which may be trusted to the same extent, makes the decision of selecting the "right" service a demanding concern especially if the selection proves to be automated and performed by an intelligent agent. The above challenges necessitates the introduction of a new framework addressing these challenges and enabling users to:

- Describe and discover Web Services semantically.
- Focus on the conceptual basis of their experiments rather than understand the low level details of locating services.
- Automatically select a trustworthy service that meets the user's trust policy.

In this paper, we introduce a novel framework to enable automated semantic discovery and selection of Web Services based on a user's trust policy. We apply our approach and framework to a data mining Web Services case study.

The reminder of this paper is structured as follows. First we provide an overview of the data mining Web Services case study. Then we provide an overview of related work (section 3). In section 4 an overview of the proposed data mining Web Services with a semantic description of the algorithms is presented. In section 5 the proposed framework is introduced. Finally, in section 6 we evaluate the framework by an empirical experiment.

2. Case Study

In order to exemplify our approach and framework we will apply it to an interesting case study, that is one of the case studies identified within the FAEHIM project [25] [1].

The case study we are going to introduce is related to exposing data mining algorithms as Web Services. In this case study, we will show how our framework is used to discover data mining services semantically and how it is used to select a service from a pool of discovered services based on the user's trust policy. We will also show how the factors that produce the final decision for selecting a service depend on the user's trust disposition.

Data Mining and Knowledge Discovery is one of the emerging technologies in todays corporate world. It has ever increasing number of applications with the increasing use of computers in every walk of life. A few examples of its use are customer profiling, market basket analysis, decision-making, weather forecasting, biomedical and DNA data analysis. Formally, it is the process of obtaining hidden information or knowledge from huge data sets.

The availability of Web Service standards (such as WSDL, SOAP), and their adoption by a number of communities, including the Grid community as part of the Web Services Resource Framework (WSRF) indicates that development of a data mining toolkits based on Web Services is likely to be useful to a significant user community. Providing data mining Web Services offers many benefits to the existing data mining toolkits that are installed on the user's machine. Firstly, these Web Services offer the data mining algorithm based on "pay-per-use" basis. Therefore, users do not require to purchase an expensive toolkit that provides a wide collection of different algorithms at the time they merely need to use one or two algorithms or use an algorithm for a limited period of time (cost benefits). Secondly, these Web Services could be hosted on fast machines, e.g. clusters, which enable users to make use of the fast processing power without the need to purchase such machines (cost and time benefits). Thirdly, these Web Services could be integrated with other third party services, allowing data mining algorithms to be embedded within existing applications. However, exposing data mining Web Services must be supported by a framework to enable users to focus on the conceptual basis of their problems rather than understanding the low level details of locating services, hence semantic discovery must be introduced. The framework must also enable users to identify trustworthy services than they believe will fulfil their requirements.

3. Related Work

To the best of our knowledge, no existing literature directly addresses the topic of semantic discovery and selection of Web Services based on a user's trust policy apart from [22] in which we propose a framework to facilitate reputation-based discovery and matchmaking of services using semantic Web technologies. The framework has three main components: (1) a matchmaker which semantically discovers matching services and (2) a composer which will retrieve a combination of services that provide the same functionality as the requested service in case a service cannot be matched by the matchmaker, and (3) a reputation manager which will select a service based on reputation metrics. The framework is limited to the service reputation and does not take the user's own experience into account in the decision of selecting a service.

3.1. Trust

The significance of quantifying the trustworthiness of agents in distributed systems is evident from on-going research. Broadly speaking, there are two main approaches to trust introduced in literature. Firstly, to allow agents to trust each other, there is a need to endow them with the ability to reason about the reliability, or honesty of their counterparts. This ability is captured through trust models. The latter aim to enable agents to calculate the amount of trust they can place in their interaction partners. A high degree of trust in an agent would mean it is likely to be chosen as an interaction partner. Conversely, a low degree of trust would result in it not being selected (if other, more trusted interaction partners are available). In this way, trust models aim to guide an agents decision making in deciding on how, when, and who to interact with. However, in order to do so, trust models initially require agents to gather some knowledge about their counterparts characteristics. This has been achieved in three ways in the literature:

1. **A presumption drawn from the agent's own experience:** Trust is computed as a rating of the level of performance of the agent. The agent's performance is assessed over multiple interactions checking how good and consistent it is at doing what it says it will. To this end, Witkowski et al. [26] propose a model whereby the trust in an agent is calculated based on its performance in past interactions. Similar to Witkowski et al.'s model, Sabater et al. [19] (through the REGRET system) propose a similar model but do not just limit the overall performance to the agent's direct perception, but they also evaluate its behavior with other agents in the system.

2. **Information gathered from other agents:** Trust in this approach is drawn indirectly from recommendations provided by others. As the recommendations could be unreliable, the agent must be able to reason about the recommendations gathered from the other agents. The latter is achieved in different ways: (1) deploying rules to enable the agents to decide which other agents' recommendation they trust more as introduced by Abdul-Rahman et al. in [5]. (2) Weighting the recommendation by the trust the agent has in the

recommender; EigenTrust [13] and PageRank [17] are examples of this approach.

3. **Socio-Cognitive Trust:** Trust is drawn by characterizing the known motivations of the other agents. This involves forming coherent beliefs about different characteristics of these agents and reasoning about these beliefs in order to decide how much trust should be put in them. An example of this is Castelfranchi's work [7].

Refer to [24] for more details on trust and reputation approaches.

3.2. Semantic Service Discovery

Automation of the service discovery process is essential for the e-Science and e-Business communities. To provide such an automatic location, the discovery process should be based on the semantic match between a declarative description of the service being sought and a description being offered. This problem requires not only an algorithm to match these descriptions, but also a language to declaratively express the capabilities of services [15].

Similar research in this area was addressed by Tangmurarunkit et al. with their resource selector and Deelman et al. with their workflow generator. The ontology-based resource selector [11] exploits ontologies, background knowledge, and rules for solving resource matching in the Grid. In order to make the matchmaking more flexible and also to consider the structure of Virtual Organization (VOs) the framework consists of ontology-based matchmakers, resource providers and resource consumers or requesters. Resource providers periodically advertise their resources and capabilities to one or more matchmakers using advertisement messages. The user can then activate the matchmaker by submitting a query asking for resources that satisfy the request specification. The query is then processed by the TRIPLE/XSB deductive database system [9] using matchmaking rules, in combination with background knowledge and ontologies to find the best match for the request.

Mandel and Sheila [16] automated Web Service discovery by using a semantic translation within a semantic discovery service. The approach uses a recursive backchaining algorithm to determine a sequence of service invocations, or service chain, which takes the input supplied by BPWS4J and produces the output desired by BPWS4J. The translation axioms are encoded into translation programs exposed as Web Services. The algorithm invokes the DQL (DAML Query Language) [18] service to discover services that produce the desired outputs. If the semantic discovery services do not have a required input, the algorithm searches for a translator service that outputs the required input and adds it to the service chain. As the process is recursive it terminates when it successfully constructs a service chain, or the profiles in the knowledge base are exhausted.

4. Data Mining Web Services

Several Web Services are implemented to support the data mining process. The following is a description of these Web Services.

4.1. Classification Web Services

Each classification algorithm available in the WEKA toolkit[1] [2] is converted into a Web Service. For example, a J48 Web Service that implements a decision tree classifier, based on the C4.5 algorithm. The J48 service has two key options: (1) classify and (2) classifyGraph. The classify option is used to apply the J48 algorithm to a data set specified by the user. The data set must be in the Attribute-Relation File Format (ARFF), which essentially involves a description of a list of data instances sharing a set of attributes. The result of invoking the classify operation is a textual output specifying the classification decision tree. The classifyGraph option is similar to the classify option, but the result is a graphical representation of the decision tree created by the J48 algorithm.

4.2. Clustering Web Services

Similar to the classification algorithms, Web Services have been developed and deployed for a variety of different clustering algorithms. For example, a Web Service is implemented for the Cobweb Clustering algorithm. The Web Service has the following operations: (1) cluster and (2) getCobwebGraph. The cluster operation is used to apply the algorithm on a specified data file. The data set must be in the ARFF format. The result of the invocation is a textual output describing the clustering results. The getCobwebGraph operation is similar to the equivalent operation for the clustering algorithm, but the result is a tree created by the cobweb algorithm.

4.3. Semantic Description

Semantic description of services can be used to enrich the discovery process by enabling its usage in a more dynamic way. In order to describe service semantics an ontology can be used. Gruber stated in 1992 "an ontology is a set of definitions of content-specific knowledge representation primitives: classes, relations, functions, and object constants" [10]. With this definition, an ontology is both human

[1] Weka is a collection of machine learning algorithms for data mining tasks

and machine readable. An ontology, together with a syntax and semantics, provides the language by which knowledge-based systems can interoperate at the knowledge-level by exchanging assertions, queries and answers.

The taxonomy of the data mining algorithms is shown in Figure 1. The structure shows two branches: `clusters` and `classifiers`. A further refinement of the algorithms are the classification of the classifiers into the following categories: `Bayes`, `Functions`, `Lazy`, `Meta`, `Trees`, `Rules` and `Misc`. Each of these categories contain various data mining algorithms. For example, in the category `Trees` there are 4 algorithms specified. Each of these data mining algorithms are furthermore described by a functional attribute.

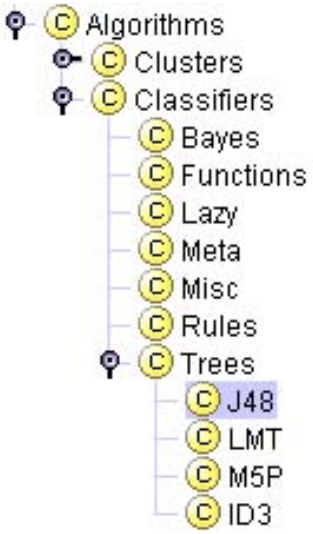

Figure 1. Algorithm Taxonomy

For the discovery of a data mining algorithm we have on one hand the algorithm taxonomy allocating different sets of algorithms to different categories, and on the other hand a functional description of the algorithm. This allows an automatic discovery of the data mining algorithms.

5. Framework

The key feature of our framework is the use of an agent that conceptually resides between the service user and the service itself. The user agent is a software component configured to each service user system. This agent proxies all the activities from the service user, exposing the same tasks but additionally enabling other useful functionality. Instead of communicating directly with the service, the service user now communicates with the agent. In this manner, the agent can add value to the user-service interaction.

- **The service user:** The user performs the same activities as described in the Service Oriented Architecture (SOA), including: service discovery and service binding. No extra functionality is added, however, the service discovery request is extended to include two additional parameters: semantic description and trust policy.
- **The user's agent:** On capturing a service discovery request from the user, it adds extra functionality to discover and select services based on the user's trust policy.

The framework architecture of the user's agent is shown in Figure 2. The key bootstrap operation is the location of the Discovery Manager Service (DMS), which interacts with the two core services: Matchmaker Service (MS) and Selection Service (SS). The DMS accepts an incoming service request and attempts to retrieve a matching service. The DMS will request the Matchmaker Service (MMS) to retrieve matching services semantically. The MMS will attempt to match the service request with the service descriptions in the Service Repository (SR). Once the matching services are retrieved, the DMS will instruct the SS to select a service from the matching services that matches the user's trust policy.

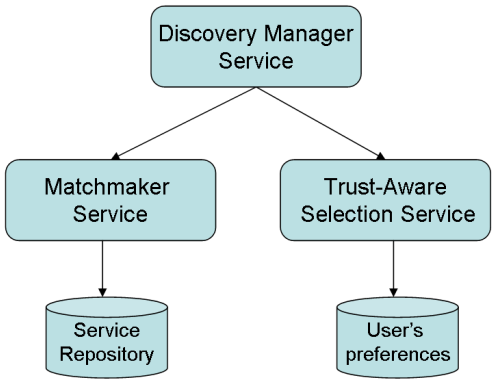

Figure 2. The User's Agent Architecture

5.1. The Matchmaker Service (MS)

The Matchmaker service is responsible to semantically match the service request with service descriptions stored in a repository. The service repository might be made available either residing on the local machine or on a remote machine. The MS uses the data mining algorithms specified in an ontology for the semantic matchmaking process. MS loads and parses the ontology first. Then the ontology is

queried returning the algorithm name. Having the algorithm name allows to lookup service descriptions stored in the service repository. All services representing the requested data mining algorithm are then returned to the DMS.

5.2. The Selection Service (SS)

The selection service is responsible for selecting a trustworthy service. *Trust* is the key concept in this paper. If a consumer A trusts a service s, then this signifies a passive construct that is synonymous to the phrase "A *believes* service s is trustworthy to perform a particular operation". Trust is articulated as an assumption or an expectation that A makes about s. This expectation is based upon more specific beliefs which form the basis or the components of trust [7]. Those beliefs are the beliefs we have for someone we want to trust; the basic components of the mental state of trust. They are the answers for the question when we ask ourselves "What do we have in mind when we trust a service?". For example, we may trust a service because we believe that the service is able to do what we need (competence), and it will actually do it quickly (promptness). Competence and promptness are therefore examples of the basic beliefs and mental state components of trust in this instance. Those beliefs are the analytical account and the components of trust, and we derive the *level of trust* directly from the aggregation of its componential and supporting beliefs. Therefore, in our model, the trust level as defined by Falcone et al [7] is *a function of the subjective certainty of beliefs*. The level of trust is used to formalize a rational basis for the decision of relying and betting on the trustee, i.e. the service.

5.3. Belief

A belief describes a state of the world from the point of view of an agent. It is the state a user has in his mind for a service. Many types of beliefs exist. Below we list the relevant beliefs for our model:

- Competence (or Reliability) Belief: The service's raw ability to accomplish a task, such as providing accurate results or performing a desired action [6].
- Availability Belief: The availability of the service.
- Promptness Belief: The speed at which the service responds to task requests by accomplishing the agreed upon task [6].
- Cost Belief: Cost refers to the monetary value that the user is willing to pay.

5.4. Sources of Beliefs

Since the trust level is articulated as a function of the subjective beliefs, the latter needs to be obtained first to compute the level of trust. Beliefs are obtained from various sources. Several models in literature propose a quantification of the level of trust and make it dynamic. For example in [12] and [21], the authors propose models in which they consider the direct interaction (experience) or reputation as sources. Experience is defined as the personal knowledge derived from participation or observation from direct interactions, whereas reputation is articulated as how the rest of the world feels about the behaviour of a particular service [4]. In [8] Falcone et al., consider two more possible sources: categorization and reasoning. Categorization is the process of grouping things based on prototypes, whereas reasoning is the act of using reason to derive a conclusion from certain premises.

In our model, we will restrict ourselves to the two common sources in literature: experience and reputation for simplicity. The model, however, is flexible to include additional sources in the future. For the purpose of our model, we define the following two sources:

Definition 1 An Experience is the knowledge gained after having a transaction with a service. The experience with a particular service is stored in a private repository as a set of values termed as *Quality of Experience* (QoE). The term (QoE) is defined as how the user feels about how a service was delivered, relative to his expectations and requirements. The QoE of a particular service in obtained from the repository by executing the *getQoE(Service s)* query.

Definition 2 Reputation of a service is defined on how the service has performed with the users who have interacted with the service. In other words, how the service complied in delivering the expected qualities. The reputation of the service is stored in a public repository as a set of values termed as the *Quality of Compliance* (QoC). The term QoC therefore is defined as the set of values which describe how the service complied with delivering the promised quality of service (QoS) as agreed upon with the user beforehand. The QoC of a particular service in obtained from the repository by executing the *getQoC(Service s)* query.

5.5. Trust Management

The trust management problem in SOAs derives from the need to embed trust-based decision making support in service selection and service composition. In our view, a trust management for a SOA, in its simplest form, must have the following functions in its life cycle as illustrated in Figure 3:

1. Capture the user's trust disposition in a policy language.

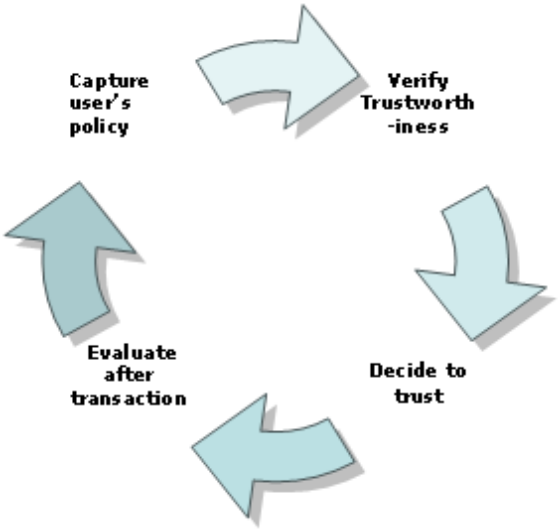

Figure 3. The Trust Management Life Cycle

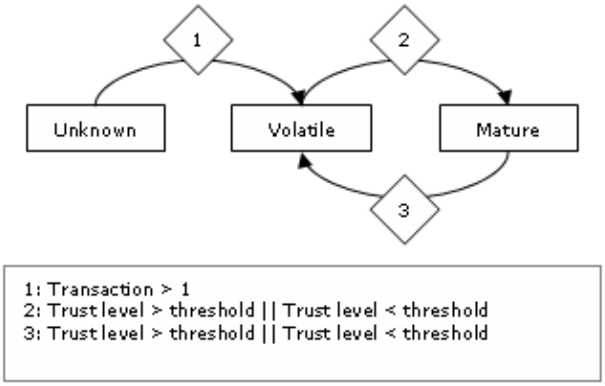

Figure 4. Phases of trust relationship - Arrows indicate possible direction of phase transition

2. Verify the trustworthiness of a service. This is usually performed by exploring the reputation of the service (global trust value) and combining it with the user's central view of the service (local trust value). The verification process results in a trust value assigned to the service.

3. Make a practical decision on whether to trust or distrust a service. The decision making is concluded by taking the user's trust policy into account.

4. If trust, evaluate/rank the service after a transaction with the service has taken place.

A trust relationship exists between a user and a service once the user makes a trust decision about the service's trustworthiness whether the trust decision is to trust or distrust the service. For online interactions, particularly consumers' interactions with e-commerce sites, research has shown that the user's trust relationship with the e-commerce site goes through various phases [3]. Their findings showed that the relationship starts with users being unaware of the site followed by a period where the user starts building some amount of trust towards the site. The user then looks for evidence or clues to confirm his trust and once confirmed further experience is required to maintain the trust relationship. In the context of our framework, we adopt the same phases that a trust relationship goes through. At any point in time, a trust relationship will exist in one of three phases: `unknown`, `volatile` or `mature`. See Figure 4

The first phase of a trust relationship is the `Unknown` phase. A relationship between a service consumer and a service is in the `unknown` phase if the consumer has not interacted with the service in the past. If the consumer decides to interact with the service, the relationship enters the `Volatile` phase. The trust relationship enters the `Mature` phase when the user develops a mature experience with the service by either being satisfied that the service's trustworthiness has reached a level whereby a more stable trust relationship can be maintained or being assured that the service's trustworthiness has dropped to a level whereby a relationship with the service is certainly vulnerable.

5.5.1 Capturing user's policy

The first requirement in the trust management's life cycle is interpreting the user's trust disposition. For this purpose, a policy model is introduced. The trust policy provides a model and corresponding syntax to describe the user's trust disposition. It provides a model for the expression of the trust requirements and characteristics. The policy is defined as:

$$\text{Trust Policy} = (\,[t_1, t_2], <\textit{phase-specific rules}>_1, <\textit{phase-specific rules}>_2, \ldots, <\textit{phase transition rules}>_1, <\textit{phase transition rules}>_2, \ldots)$$

where the range [*t1, t2*] is the time interval over which the trust policy is valid, <*phase-specific rules*> is a set of rules identified for a particular phase in the trust relationship, and <*phase transition rules*> is a set of rules that govern the transition of the trust relationship from one phase to another.

<*phase-specific rules*> formally consists of the <*target phase*>, <*trust sources*>, <*trust beliefs*> and <*trust utility*>. <*target phase*> indicates the trust relationship phase (unknown, volatile or mature) in which the rules are applied to. <*trust sources*> defines the trust sources in the

form (*name*, *influence*), where name indicates the name of the source and influence is the level that the source influences the trust value. <*trust beliefs*> defines a sequence of beliefs that the user is concerned of. Each element in the sequence is of the form (*name*, *influence*), where name indicates the name of the belief and influence is the level that the belief influences the trust value. <*trust utility*> defines a set of the user subjective utilities. Four types of utilities are identified: the utility when the service does not deceive the user, the utility when the service deceives the user if he trusts the service, the utility when the service does not deceive the user if he distrust the service, and the utility when the service deceives if the user distrusts the service.

<*phase transition rules*> is a sequence of <*phase transition rule*>. Each <*phase transition rule*> consists of <*target phase*>, <phase to> and <*condition*>. <*target phase*> specifies the trust relationship phase (unknown, volatile or mature) in which the rule is applied to. <*phase to*> specifies the phase in which the target phase should be changed to. The <*condition*> details the condition in which the phase of the relationship should be changed. Figure 5 illustrates a fragment of a trust policy.

```
<trust policy>
  <phase_rule target="unknown">
    <belief_source name="QoE" influence="0.7">
    <nelief_source name="QoC" influence="0.5">
    <belief name="reliability" influence="0.6">
    <belief name="promptness" influence="0.7">
    <belief name="availability" influence="0.66">
    <belief name="cost" iinfluence="-0.41">
    <utility action="trust" decieve="yes" value="0.7">
    <utility action="trust" decieve="no" value="0.8">
    <utility action="distrust" decieve="yes" value="0.5">
    <utility action="distrust" decieve="no" value="0.5">
   </phase_rule>
   <phase_transition target="unknown">
    <phase_to>volatile</phase_to>
    <condition>
     <trust_level operand="greaterThan" value="0.5">
     <number_of_transactions operand="greaterThan"
                  value="1">
    </condition>
   <phase_transition>
    ...........
    .......
</trust policy>
```

Figure 5. A fragment of a trust policy

5.5.2 Verifying Trustworthiness

Verifying the trustworthiness of a service is the process of computing the trust level of a service. The computed trust level is a degree of trust instead of a simple probability factor since it evaluates the trustworthiness in a rational way. As the trust level is articulated as a function of the subjective certainty of beliefs, an implementation of the function is needed.

An implementation of such function, which we also use in this model, is proposed by Falcone et al [8]. The authors' implementation is based on Fuzzy Logic and more specifically on a special type of fuzzy system called Fuzzy Cognitive Maps (FCM) [14]. An FCM is an additive fuzzy system with feedback; it is suited for representing a dynamic system with cause-effect relations. An FCM has several factors; representing belief sources, and edges, representing the casual power of a factor over another one. A positive cause means casual increase, whereas a negative cause means casual decrease. The degree of how much a factor causes another is computed from two values: the value of the causing factor and a value of the edge which represents how much the cause factor influences the other factor. The values of all the edges are assigned by a human and propagate in the FCM until a stable state is reached; so the values of the other factors are computed. Once the initial values for the first layer (i.e. belief sources) are set, the FCM starts running. The state of a factor N at each step s is computed using the following algorithm:

```
Loop i over all factors of system
    Current value of factor i is Vi
    Loop j over all factors of system
    Calculate new value of factor i as
    affected by factor j:
    Vij = Vi + Eij (Vj - Vi) Iij / 100
    Level Vij must be between 0 and 100.
        If Vj < 0 then snap to Vij = 0.
        If Vij > 100 then snap to Vij = 100.
    End of loop j
End of loop i
```

Formally speaking, the state of a factor N at each step s is computed taking the sum of all the inputs, i.e., the current values at step $s-1$ of factors with edges coming into N multiplied by the corresponding edge weights. The value is then squashed (into the -1,1 interval) using a threshold function. The FCM run ends when an equilibrium is reached, i.e., when the state of all factors at step s is the same as that at step $s-1$.

Suppose we want to use the fuzzy cognitive maps to model the behaviour of the user's trust level towards a service s which he has not interacted in the past. The user's pertinent beliefs for s are: the reliability, promptness, availability and cost. Those beliefs are obtained from two sources: the user own experience registry as QoE and the compliance registry as QoC. The user relies on the QoE more than the QoC and, therefore, he weights the QoE and QoC by 0.7 and 0.5 respectively. The user believes that the promptness and reliability of the service are two important elements for increasing his trust level. Hence, he assigns high positive weights for them (+0.6 and +0.7). The availability of the service is not very important but will also add to the trust level and therefore a small positive weight is allocated to it (+0.3). The cost of the service has a significant impact on the trust level. An increase in the cost leads to a

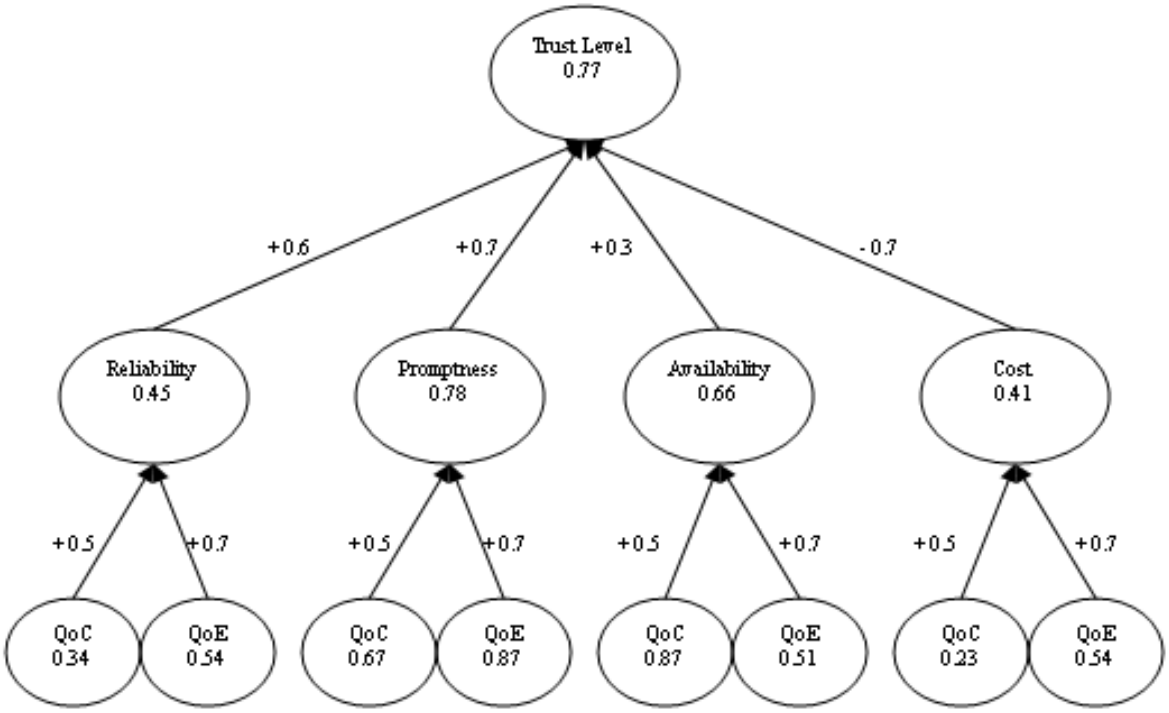

Figure 6. An example of FCM to compute trust level

decease in the trust level. Therefore, a high negative weight is allocated to it (-0.7). The user specifies these information in the trust policy as illustrated in Figure 5. The FCM model for this scenario is illustrated in Figure 6.

Obtaining the trustworthiness value is the second steps in the trust management life cycle. In order to make a decision to trust, other factors must be taken into consideration, such as risk and utility.

5.6. Deciding to Trust

Decision making is the cognitive process of selecting a course of action from among multiple alternatives mutually exclusive actions. In other words, from among the alternatives, one and only one choice can be made. Each of these choices might have one or more possible consequences that are beyond the control of the decision maker, which again are mutually exclusive. The rational procedure in such a situation is to identify all possible outcomes, determine their values (positive or negative) and the probabilities that they will result from each course of action, and multiply the two to give an expected value. The action to be chosen should be the one that gives rise to the highest total expected value. In reality people do not behave like this otherwise no-one would either gamble or take out insurance. Within behavioural decision theory, this has led to various dilutions of the expected value theory; for example, objective probabilities can be replaced by subjective estimates, and objective values by subjective utilities, giving rise to the subjective expected utility or SEU theory, developed by Savage [20]. Subjective expected utility is a method in decision theory in the presence of risk. It combines two distinct subjective concepts: a personal utility function and a personal probability analysis based on Bayesian probability theory. If an agent A believes an uncertain event has possible outcomes x_i each with a utility to A of $u(x_i)$ and where A believes that the probability of each outcome is $P(x_i)$, then A's subjective expected utility will be

$$\sum_i u(x_i)p(x_i)$$

A may be able to make a decision which changes the possible outcomes of y_j in which case A's subjective expected utility will become

$$\sum_j u(y_j)p(y_j)$$

Which decision the agent prefers depends on which sub-

jective expected utility is higher. Different agents may make different decisions because they may have different utility functions or different beliefs about the probabilities of different outcomes. The SEU theory applies perfectly in the trust decision making. In any situation, a user A endowed with the opportunity of utilising a service s has two course of actions: (1) to trust or (2) to distrust. We calculate the expected value associated with each possible course of action, and select the course of action that has the highest expected value. To calculate the expected value for a course of action, we multiply each possible payoff associated with that course of action with its probability, and sum up all the products for that course of action. In order to do that for the two actions, we should consider the following abstract scenario (illustrated in Table 1) where we call:

- U_{ta}: The utility when the service s does not deceive if user A trusts s.
- U_{tb}: The utility when the service s deceives if user A trusts s.
- U_{da}: The utility when the service s does not deceive if user A distrusts s.
- U_{da}: The utility when the service s deceives if user A distrusts s.

In the above scenario, the agent's preference to trust must be greater than the agent's preference to distrust. If we replace the probability of each outcome $P(x_i)$ by the trust value of the outcome, then applying the SEU theory, in order to trust we must have:

$$(TL)(U_{ta})+(1-TL)(U_{tb}) > (TL)(U_{da})+(1-TL)(U_{db})$$

where TL is the trust level that agent A has in s and which it is calculated in the previous section. Considering the example given in Figure 5 and the computed trust level, we get:

$$((0.77)(0.8)+(1-0.77)(0.7)) > (0.77)(0.5)+(1-0.77)(0.5)$$

$$\cong 0.777 > 0.5$$

Since the expected value for the trust action is higher than the expected value for the distrust action, the decision is concluded to trust the service.

5.7. Evaluating a Transaction with a Service

Evaluating a transaction with a service is the cognitive process of making a judgment to assign a set of qualities for the experience that the user had with the service in that transaction. When a user A requests a task t from a service s, the quality of completing the task reflects A's experience with s. Therefore, in experience, the qualities drawn from that experience are important. In the context of the our model, two types of qualities are identified: (1) Quality of Experience (QoE) and (2) Quality of Compliance (QoC). QoE is used as a feedback to the user's own experience registry while QoC is used as a feedback to a public compliance registry. The main difference between the two qualities is that QoE is used solely by the user who evaluates them while QoC is used by other users too. Therefore, a formal framework for evaluating and updating the QoC must be presented. This framework is discussed in details in [22].

QoE is expressed as a set of characteristics of the environment, perception and opinion of the user as he interacts with the service. Formally, we define QoE as:

$$QoE = exp, opn$$

The *exp* is the expectation that the user had about the service before being involved in a transaction with the service. The expectation contains the expected trust level before the transaction takes place. The *opn* is the opinion that the user has about the service after the transaction has been taken place. The value represents the user's subjective view of how the service was obeyed to contract's terms. How the subjective view is computed is beyond the scope of this paper.

The concept of QoC, on the other hand, stems from the fact that there has been no practice of recording the achieved service levels once a transaction has been completed. Doing so gives an insight into the providers past performance by providing necessary data to progressively assess the compliance levels over a range of past transactions. Refer to [23] for more information on how the QoC is computed.

6. Experiment

The primary goal of our experiment is to show the benefits of our framework and precisely the benefits of introducing a semantic service discovery and a trust approach for service selection. In the experiment we also show how the selection of services change when the components of trust change.

We construct a simulation that allows the creation of various data mining Web Services, e.g. classification and clustering Web Services. In addition, the simulation allows the creation of sources of beliefs, e.g. QoC and QoE, which feed a service consumer with belief values for the availability, reliability, promptness and cost for each Web Service. The user uses the information given to make a trust-based decision to select one of the Web Services.

Action	Utility	mmm	Expected Value
	s does not deceive (probability = p_a)	s deceive (probability = p_b)	
Trust	U_{ta}	U_{tb}	$(p_a)(U_{ta})$ + $(p_b)(U_{tb})$
Distrust	U_{da}	U_{db}	$(p_a)(U_{da})$ + $(p_b)(U_{db})$

Table 1. Analysis of trust decision making

6.1. Setup Summary

We conduct two experiments. In the first experiment, we show how the framework works by discovering and selecting a matching service. In the second experiment we show how the selected service changes if the trust parameters defined in the user's trust policy change.

6.2. Experiment 1

In this simulation we create six Web Services: three classification Web Services (J48 Classifier), and three clustering Web Services (Cobweb cluster). `J48` is an implementation of C4.5, a standard algorithm that is widely used for practical machine learning producing decision tree models. This algorithm works by forming pruned partial decision trees (built using C4.5's heuristics), and immediately converting them into a corresponding rule [27]. Semantically, the `J48` algorithm can be found in the taxonomy by traversing the tree structure as follows: `classifiers -> trees -> J48`. The function attribute of the algorithm is `forming pruned partial decision trees`.

We also create two sources of beliefs: QoC and QoE which feed the user with the belief values for the six Web Services as shown in Table 2.

We send the following service request to the user-agent to discover and select a service considering the matching parameters and trust policy.

As a result of the service request shown above in Figure 7, the service request is sent to the DMS and forwarded to the MMS. As a first step the MS reads the service request parameters responsible for the matchmaking identified by the `matchingParameters` tag. Having the two attributes allows to discover the algorithm semantically. The parameter `trees` breaks the search further down and results into one out of four algorithms. The function attribute `partial decision trees` then matches the `J48 algorithm`. As there are four J48 Web Services available in the service repository, the MMS returns those services to DMS. In this step, the same result is obtained each time.

Once DMS forwards the three matching services to SS, the later computes the trust value for each service. To do so, SS acquires the belief values for each requested trust parameter from our simulated sources of beliefs (i.e. QoC and QoE) and computes the trust value using a FCM as discussed in section 5.5.2. The computed trust value for the three services are 77, 48, 24 respectively. Since the first service has the highest trust value, the service is selected. The next step is to decide to trust or distrust the service. Using the formula in section 5.6 we get:

```
<ServiceRequest> <Service name=''>
 <matchingParameters>
   <parameter name="trees" value="">
   <parameter name="function"
       value="partial decision trees">
  </matchingParameters>
 <trust policy>
  <phase_rule target="unknown">
    <belief_source name="QoE" influence="0.7">
    <nelief_source name="QoC" influence="0.5">
    <belief name="reliability" influence="0.6">
    <belief name="promptness" influence="0.7">
    <belief name="availability" influence="0.66">
    <belief name="cost" iinfluence="-0.41">
    <utility action="trust" decieve="yes" value="0.7">
    <utility action="trust" decieve="no" value="0.8">
    <utility action="distrust" decieve="yes" value="0.5">
    <utility action="distrust" decieve="no" value="0.5">
   </phase_rule>
   <phase_transition target="unknown">
    <phase_to>volatile</phase_to>
    <condition>
     <trust_level operand="greaterThan" value="0.5">
     <number_of_transactions operand="greaterThan"
                 value="1">
    </condition>
   <phase_transition>
    ...........
    .......
 </trust policy>
</ServiceRequest>
```

Figure 7. An example of a service request

$$((0.77)(0.8)+(1-0.77)(0.7)) > (0.77)(0.5)+(1-0.77)(0.5)$$

$$\cong 0.777 > 0.5$$

Since the expected value for the trust action is higher than the expected value for the distrust action, the decision is concluded to trust the service.

6.3. Experiment 2

In the second experiment, we perform the same operation as in the first experiment but we modify some of the parameters in the trust policy. Mainly, we change the importance values of the reliability, availability, promptness

Table 2. "Experiment 1 values"

Service	Reliability		Availability		Promptness		Cost	
	QoE	QoC	QoE	QoC	QoE	QoC	QoE	QoC
1	54	34	51	87	87	67	54	23
2	46	58	66	78	45	65	22	13
3	37	22	16	19	14	23	18	17
4	65	34	32	23	54	33	43	35
5	62	65	32	56	75	67	76	88
6	12	23	42	44	23	41	15	17

and cost into 30, 20, 30 and 90 respectively. This modification reflects the user's strong disposition towards the cost of the service.

On running the experiment we get the following trust value for the three services: 51, 88, 15 for service 1, 2, and 3 respectively. The result shows that the second service has the highest trust value in this case. This reflects the fact that service 2 costs less than the first service, and the user's desire for cheaper services.

7. Conclusion

Online service provision commonly takes place between parties who have never had transactions with each other before, in an environment where the service consumer often has insufficient information about the service provider and about the goods and services offered. In such environment, several challenges regarding discovering and selecting the "right" service need to be addressed. In this paper, we introduced a framework for semantic discovery and selection of Web Services using a trust management model. The introduced trust management model identifies four phases in trust management: (1) capturing user's trust disposition, (2) verifying trustworthiness level, (3) making trust decision and (4) evaluation after a transaction has taken place.

The significance of our framework is proven by the experiment results. Without the MatchMaker service, the user would require to have a low level knowledge about the various data mining algorithms. Without the trust-based ServiceSelector, the user would select one of the three services randomly which might not match the user's trust policy. Therefore, we believe that our framework has significant advantages regarding the traditional service discovery and selection approach.

Applying the approach and framework using the case study shows how the trust factors that produce the final trust value for each service are dependent on:

- the initial strength of the different beliefs (on which trust is based on), but also
- how much a specific belief impacts the final trust (the casuality power of a belief).

References

[1] Faehim. available at: http://users.cs.cf.ac.uk/ali.shaikhali/faehim/index.htm. last viewed: July 2005.

[2] The weka toolkit, university of waikato. *Available at: http://www.cs.waikato.ac.nz/ml/weka/. last viewed: November 2004*.

[3] E-commerce trust study. *Cheskin Research and Studio Archetype*, 1999.

[4] *Roget's II: The New Thesauru*. Houghton, 2003.

[5] A. Abdul-Rahman and S. Hailes. Using recommendations for managing trust in distributed systems. *Proceedings IEEE Malaysia International Conference on Communication*, 1997.

[6] R. Falcone and C. Castelfranch. Cognitive anatomy, social importance and quantification. *Proceedings of the International Conference on Multi-Agent Systems*, 1998.

[7] R. Falcone and C. Castelfranch. Social trust: A cognitive approach. *Trust and Deception in Virtual Societies Journal, pp. 55-90*, 2001.

[8] R. Falcone, G. Pezzulo, and C. Castelfranch. A fuzzy approach to a belief-based trust computation. *Lecture Notes on Artificial Intelligence, pp. 73-86*, 2631, 2003.

[9] T. X. R. Group. Xsb. *http://xsb.sourceforge.net*.

[10] T. Gruber. Ontolingua: A mechanism to support portable ontologies. *Technical Report KSL 91-66, Knowledge Systems Laboratory, Department of Computer Science, Stanford University*, 1992.

[11] C. K. H. Tangmunarunkit, S. Decker. Ontology-based resource matching in the grid - the grid meets the semantic web. *In the proceedings of the First Workshop on Semantics in Peer-to-Peer and Grid Computing (SemPG03). In conjunction with the Twelfth International World Wide Web Conference 2003. Budapest, Hungary*, 2003.

[12] C. Jonker and J. Treur. Formal analysis of models for the dynamics of trust based on experiences. *Proceedings of the Autonomous Agents '99 Workshop on Deception, Fraud, and Trust in Agent Societies*, 1999.

[13] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. *Proceedings of the Twelfth International World Wide Web Conference*, 2003.

[14] B. Kosko. *Fuzzy Thinking: The New Science of Fuzzy Logic*. Hyperion, 1994.

[15] S. A. Ludwig. Flexible semantic matchmaking engine. *Proceedings of 2nd IASTED International Conference on Information and Knowledge Sharing (IKS), AZ, USA*, 2003.

[16] D. Mandell and S. McIlraith. A bottom-up approach to automating web service discovery, customization, and semantic translation. *Proceedings of the 12th International World Wide Web Conference, Workshop on E-Services and the Semantic Web(ESSW'03), Budapest*, 2003.

[17] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. *Stanford Digital Library Technologies Project*, 1998.

[18] P. H. R. Fikes and I. Horrocks. Daml query language. *Abstract Specification*, 2002.

[19] J. Sabater and C.Sierra. Regret: a reputation model for gregarious societies. *Proceedings of the 1st International Joint Conference on Autonomous Agents and Multi-Agents Systems*, 2002.

[20] L. J. Savage. *Foundations of Statistics*. New York: John Wiley & Sons, 1954.

[21] M. Schillo, P. Funk, , and M. Rovatsos. Who can you trust: Dealing with deception. *Proceedings of the Autonomous Agents '99 Workshop on Deception, Fraud, and Trust in Agent Societies*, 1999.

[22] A. Shaikh Ali, S. Majithia, O. Rana, and D. W. Walker. Reputation-based semantic service discovery. *Concurrency and Computation: Practice and Experience*, 2005.

[23] A. Shaikh Ali, O. Rana, and D. W. Walkeri. Ws-qoc: Measuring quality of service compliance. *Proceeding the 2nd International Conference on Service Oriented Computing (short paper)*, November 2004.

[24] A. Shaikh Ali, O. F. Rana, and R. J. Al-Ali. Evidence-aware trust model for dynamic services. *Book Chapter, in "High Performance Computing: Paradigms and Infrastructure"*, edited by Laurence Yang and Minyi Guo, John Wiley, 2005.

[25] A. Shaikh Ali, O. F. Rana, and I. J. Taylor. Web services composition for distributed data mining. *Workshop on Web and Grid Services for Scientific Data Analysis (WAGSSDA), Oslo, Norway*, 2005.

[26] M. Witkowski, A. Aritikis, and J. Pitt. Experiments in building experiential trust in a society of objective-trust based agents. *Trust in Cyper-societies, pages 111-132*, 2001.

[27] I. H. Witten and E. Frank. Data mining: Practical machine learning tools and techniques with java implementations. *Morgan Kaufmann, ISBN 1-55860-552-5*, 1999.