Towards Explanation-Aware Selection in Internet-Scale Infrastructures: Generating Rationale for Web Services Ratings and Reputation

Wanita Sherchan
Faculty of Information Technology
Monash University
Caulfield East, VIC 3145, Australia
Wanita.Sherchan@csse.monash.edu.au

Seng W. Loke
Dept. of Comp. Sc. and Engg.
Latrobe University
Bundoora, VIC 3086,Australia
S.Loke@latrobe.edu.au

Shonali Krishnaswamy
Faculty of Information Technology
Monash University
Caulfield East, VIC 3145, Australia
Shonali.Krishnaswamy@csse.monash.edu.au

Abstract

A collection of 1 billion publicly available web services can form an internet-scale infrastructure for building diverse applications. For a given application, selection of services and service providers from this collection becomes important and reputation is recognized as a key factor for this purpose. However, current reputation systems are limited in their ability to exchange reputation information between heterogeneous systems. To facilitate meaningful exchange and reuse of reputation information and for the overall determination of reputation, we identify the need to infer and explicate rationale for ratings. We present our knowledge based approach to inferring and explicating rationale for ratings. We show that this approach facilitates detection of deception and collusion, user preferences elicitation, explication of rationale behind user ratings and generation of personalized service recommendations.

1. Introduction

"Web services are a natural consequence of the evolution of the Web into an open medium which facilitates complex business and scientific application interactions" [1]. This transformation of the Internet from a source of information to a medium for the provision of Web/e-services has significantly changed the traditional paradigms of business interaction, resulting in an open distributed environment with possible interactions and dealings with hitherto unknown entities and short-term dynamic business relationships. In these emerging circumstances, traditional paradigms of business interactions fail to cope with the changing dynamics of electronic business. With the widespread proliferation of web service applications and the increasing diversity of service providers, selection of the most appropriate service (for a given set of requirements) poses a significant challenge. There has been a focus on extra-functional properties as the key distinguishing selection criteria. These extra-functional properties include Quality of Service (QoS) [7], trust and reputation [6].

Current reputation systems [8, 9, 13, 16] are typically based on ratings provided by the users. When there are no mechanisms in place to detect collusion and deception, combining user testimonies as such to form a providers reputation may not give an accurate assessment [3, 15], especially if the context of the ratings is not known. Moreover, such systems are vulnerable to manipulations by malicious users. Hence it becomes essential to establish the validity of the ratings prior to using them in formulating reputation based on such ratings.

Another significant issue that hasn't been addressed so far is the transferability of reputation between separate but potentially related environments [8]. For example, eBay and Amazon both provide e-marketplaces for online auctions / transactions. Currently, the reputation that a user has in eBay is not transferable when s/he goes to Amazon. This makes it easier for malicious users to deceive repeatedly. Such users can indulge in deceitful behavior in one context and this is not reflected as they move to another (User identity management is an important consideration, but outside the scope of this research). Because of lack of mechanisms for supporting transfer of reputation information, such dishonesty cannot be detected, nor stopped. For creating safer / more reliable environment for online interactions it is essential to be able to exchange / transfer reputation information. For a reputation management system which facilitates transfer of reputation information between domains, it is important to identify and include the rationale along with the ratings so that ratings can be reused (in the new domain) in a meaningful way relevant to the context. Moreover, reputation evaluation depends on a number of factors such as the length of history, the number of interactions and user biases, and can change considerably if the factors considered are different. Hence, “rationale”, which explicates these factors, plays an important part in understanding and using the reputation data. It is important to identify the rationale behind the ratings so that similar ratings (or ratings pertaining to a context) can be aggregated to obtain a reputation value meaningful in that context. Rationale provides justification, explanation, meaning and context to ratings and reputation values (which on their own are merely numbers). Without rationale, reputations as such have no meaning, especially if they were obtained from another reputation system (another context). For example, two reputation servers have reputation information regarding the same service but in different context, then in this case rationale provides a basis for combining these two information to obtain a more relevant and meaningful evaluation. Also, if the servers have reputation information regarding two different services, rationale provides a basis for making comparisons and selection of the most suitable one. Rationale is especially significant if the two reputation servers are heterogeneous (employ different mechanisms / algorithms for reputation evaluation). Therefore, to facilitate meaningful exchange and reuse of reputation information and for the overall determination of reputation, we identify the need to infer and explicate rationale for ratings and reputation.

Currently there has been little effort focused towards rationale inference and reasoning in the context of reputation. We aim to bridge this gap to facilitate a better understanding and holistic treatment of reputation in the specific context of Web Services. In this paper, we present one way of generating rationale for user ratings by comparing subjective and objective performance of the services/service providers. Using this approach we are able to support detecting deception and collusion (hence validating the ratings), identifying user preferences / detecting biases, providing recommendations to users and explicating the rationale for transfer between different reputation systems. Our proposed approach detects false or invalid ratings by comparing the ratings with the performance of the services, thereby reducing the possibility of maliciousness. Thus elucidated invalid ratings can be excluded from reputation evaluations. Moreover, based on such detection, we are able to identify deceptive and collusive users. The offending users could then be penalized in various ways, such as alienating the user in the society and barring the user from using the reputation system. Furthermore, the proposed approach infers user preferences from the users’ rating behavior. Thus inferred preferences can then be used for providing suitable recommendations to the user. In doing so, i.e. detecting deception, collusion and inferring user biases, we are able to reason about user ratings and infer the rationale behind user ratings. In addition, our model uses knowledge engineering for reasoning, hence we are able to generate explanations for the conclusions drawn and the inferences made. Provision of such explanations gives credibility to the deductions, makes the results more acceptable and accessible to the users [4, 12] and facilitates in exchange and reuse of rationale with reputation information.

The remainder of the article is structured in the following way. We first discuss our previous work and introduce our knowledge based modeling for reasoning about ratings in section 2. In section 3, we present the basic knowledge base of the reasoning system and discuss the rules in the knowledge base. We then discuss the operation of the basic reasoning model in section 4. In section 5, we discuss the prototype implementation to demonstrate how our knowledge based modeling provides a means to extract rationale for different aspects of reasoning with rationale such as detecting suspicious ratings, poor raters and identifying biases of users. We then discuss extensions to the basic reasoning model in section 6. Finally, we conclude and highlight the contributions of our model in reputation management in web services domain and point to future work in Section 7.

2. Generating rationale for user ratings

In our previous work [11], we proposed a fuzzy approach to extracting the rationale of user ratings and identifying user preferences. The proposed model uses a set of fuzzy inference rules which model unbiased rating and different possible biases towards particular attributes. To establish a particular bias, we compare the rating provided by a user with the rating generated by the inference rules based on the services’ actual performance.

We proposed to calculate the compliance values [5] for each parameter specified in the SLA and combine them using fuzzy inference rules to get the overall estimated rating for the service in that invocation. We proposed to estimate a set of ratings for a service based on its compliance in the different SLA parameters. Each rating is estimated on a certain basis, for example, unbiased towards any parameter, biased towards Response Time, biased towards Performance, biased towards Response Time and Performance, and so on. For this purpose, we proposed to define different sets of inference rules. The first set consists of unbiased rules, i.e. rules which treat all parameters as equal. The other sets of rules consider one parameter each as more important than other parameters. Using different sets of rules the estimated rating are different, for e.g. the estimated rating obtained by using unbiased set of rules will give an unbiased rating which will be different from that obtained by using a set of rules biased towards a particular attribute (e.g. response time). The rating given by the user is then compared against all of the estimated ratings obtained through the fuzzy inference and the estimated rating which is the closest match is considered equivalent to the users rating. Then the basis used for getting that estimation will be established as the rationale for that user rating. If the users rating does not match any of the estimated ratings, then the users rating is termed invalid because the estimated ratings are calculated for each possible case.

The work reported in this article is an extension of this previous work. In this paper, we use a knowledge based approach for reasoning about user ratings and explicating the rationale behind ratings. This has significant effects. First of all, we do not generate estimated ratings (as per the services' performance) and compare it with the user's ratings. Rather we use logic rules to directly compare the ratings with the service performance. This eliminates the problem arising due to estimation of ratings, because using the rules we are able to identify a user rating bias towards any number of quality attributes. Second, we are also able to detect collusion and deception in user ratings on a per transaction basis. Based on the results of this inference, we propose an approach for empirical analysis of user rating history to determine collusive and deceptive users. Third, since our model uses knowledge representation, we are able to explicate inferred rationale and generate explanations for the inferences. Fourth, the model infers all biases a user rating may have (i.e. bias towards any number of quality attributes), hence analysis/compilation of user bias history will give all the biases that a user has, which may be different in different invocations. Compiling all this information, we use a set of rules to determine biases of a user. This is significant in recommending services that perform well in those parameters that the user is consistently biased towards. This ability to capture/record varying biases is significant because in Web services context, changing bias is a valid occurrence. For example, at one time, price may not be a problem for a user, but may need a fast response time. At other time, accuracy may be most important due to critical nature of the service. Hence, for the same user, preferences are different in different invocations. Therefore, in these circumstances, traditional ways of determining preferences, such as statement of preferences by the users during registration are inadequate. It is required to infer users' preferences on a per invocation / transaction basis and compile them to determine the preferences of the user. In our model, we provide preference elicitation in per transaction basis and provide inference rules that perform empirical analysis of user biases to determine the preferences of the user. This is useful in generating recommendations that are more suitable for the user's preferences.

In addition, the proposed reasoning system is also capable of providing an explanation for its inferences, i.e. explaining how the system established these inferences. Such explanations give credibility to the deductions and the recipient of such information can use the information accordingly. For example, if the recipient of the information is a human, then textual explanations can be provided. Similarly, if the recipient is a computer program, then the sequence of facts and rules that led to the conclusion can be provided to the recipient system, which will use the information accordingly. Use of knowledge based approach facilitates such explanation.

Another advantage of using logic representation to store domain knowledge is that it provides a basis for defining vague concepts such as "deception" and "collusion". As such, we follow these definitions of "deception" and "collusion":

Deception: Deception is a situation where deliberately misleading ratings are provided which tarnishes the reputation of a service. For example, the service performs well, but user gives low ratings. In this situation, the user is trying to deceive other users by deliberately providing low ratings even when the service performed well. Such situations are termed as deception.

Collusion: Collusion is a situation where deliberately misleading ratings are provided that boost the reputation of a service. For example, the service performs poorly, but user gives high ratings. In this situation, the user is colluding with the service/provider and trying to boost its reputation, so even when it performed poorly, the user gave high ratings. Such situations are termed as collusion.

Our reasoning model identifies these situations in per transaction/ invocation level (i.e. deception/collusion in an invocation) as well as in users (i.e. a deceptive/collusive user).

In the next section, we describe the knowledge base of the reasoning system and discuss the rules in the knowledge base.

3. The Basic Knowledge Base

We use service performance measure compliance and user ratings to infer the rationale behind user ratings. For performance measurement and compliance calculation, readers are referred to Kalepu *et al.* [5] and Sherchan *et al.* [11]. We use fuzzy concepts to define performance levels and user ratings because the terms used to refer to both compliance (such as "compliant", "very compliant") and ratings ("poor", "moderate", "good", "very good") have a fuzzy element; hence fuzzy representation gives a more realistic mapping between the two. Also, the mapping between compliance and service performance is fuzzy. For example, we say, if a service has high compliance, then its performance is excellent, hence it should have high ratings. Whereas to map the same fact in numerical terms is next to impossible. Therefore, for the same reasons, we use fuzzy concepts to map compliance values to performance levels. In this paper, we use compliance as a performance measure. However, we would like to note that this model can support other forms of performance measures besides compliance.

We would also like to note that, if we take user ratings in the textual/descriptive form (which may be more appealing to the users), we do not need the rules that map numerical ratings to fuzzy ratings levels.

3.1. Fuzzy Terms

First, we start with the definitions of fuzzy terms/sets for ratings and performance (compliance). Table 1 lists these definitions. We have defined 4 terms (fuzzy sets) for ratings, namely, *poor*, *moderate*, *good* and *excellent*. Similarly, we have defined 3 fuzzy sets for compliance, namely, *lowCompliance*, *compliant* and *highCompliance*. However, we would like to note that the fuzzy categorization levels may be increased as per the granularity desired. The definitions as shown in Table 1 are for rating values in the range of 0 to 10, and compliance values in the range of -1 to 1. These fuzzy predicates define the degree of membership in different fuzzy sets for a given rating/compliance value. These definitions are consistent with declarations of fuzzy terms as used in Ciao Prolog [14].

For example, definition of the fuzzy predicate `poor` (refer to predicate (1) in Table 1):

- `poor(X,1)` $\Leftrightarrow$ X $\geq$ 0 $\wedge$ X $<$ 1.
  A rating value *X* has 1 degree of membership in the fuzzy set `poor` if X $\geq$ 0 and X $<$ 1.]

- `poor(X,M)` $\Leftrightarrow$ X $\geq$1 $\wedge$ X<3 $\wedge$ $2*M = 3 - X$.
  A rating value *X* has *M* degree of membership in the fuzzy set `poor` if X $\geq$ 1 and X $<$ 3 and $2*M = 3 - X$.

- `poor(X,0)` $\Leftrightarrow$ X $\geq$ 3 $\wedge$ X $\leq$ 10.
  A rating value *X* has 0 degree of membership in the fuzzy set `poor` if X $\geq$ 3 and X $<$ 10.

Similarly, other fuzzy predicates for rating and compliance fuzzy sets for have been defined. Table 1 lists the complete definitions.

Table 2 shows the definitions for ratings and compliance levels for given values of rating/compliance and membership degree. These predicates define the criteria for mapping ratings/compliance values to the different fuzzy sets. For example, the predicate `isPoorEnough/1` is defined as (refer to predicate (8) in Table 2):
`isPoorEnough(R)` $\Leftarrow$ `poor(R,M)`$\wedge$M$\geq 0.5\wedge$M≤ 1.
A rating `R` is “poor” enough (or it is somewhat poor)if `R` has a membership degree of 0.5 or more in the fuzzy set `poor`.

This mapping using two sets of rules (1 to 14) provides flexibility in the modeling and separates fuzzy membership definitions from the rest of the predicates. This lets us model the fuzzy membership functions as desired and the rest of the predicates need not change even if the fuzzy definitions change. Here, for the purpose of demonstration, we have used triangular membership functions for ratings and compliance fuzzy sets. However, we would like to note

Table 1. Fuzzy Terms

Rating Fuzzy Sets

Poor: **(1)**
`poor(X,1)` $\Leftrightarrow$ X $\geq$ 0 $\wedge$ X $<$ 1.
`poor(X,M)` $\Leftrightarrow$ X $\geq$ 1 $\wedge$ X $<$ 3 $\wedge$ $2*M = 3 - X$.
`poor(X,0)` $\Leftrightarrow$ X $\geq$ 3 $\wedge$ X $\leq$ 10.

Moderate: **(2)**
`moderate(X,0)` $\Leftrightarrow$ X $\geq$ 0 $\wedge$ X $<$ 1.
`moderate(X,M1)` $\Leftrightarrow$ X$\geq$1 $\wedge$ X<3 $\wedge$ $2*M1 = X - 1$.
`moderate(X,M2)` $\Leftrightarrow$ X$\geq$3 $\wedge$ X<5 $\wedge$ $2*M2 = 5 - X$.
`moderate(X,0)` $\Leftrightarrow$ X $\geq$ 5 $\wedge$ X $\leq$ 10.

Good: **(3)**
`good(X,0)` $\Leftrightarrow$ X $\geq$ 0 $\wedge$ X $<$ 3.
`good(X,M1)` $\Leftrightarrow$ X$\geq$3 $\wedge$ X<6 $\wedge$ $3*M1 = X - 3$.
`good(X,M2)` $\Leftrightarrow$ X$\geq$6 $\wedge$ X<9 $\wedge$ $3*M2 = 9 - X$.
`good(X,0)` $\Leftrightarrow$ X $\geq$ 9 $\wedge$ X $\leq$ 10.

Excellent: **(4)**
`excellent(X,0)` $\Leftrightarrow$ X $\geq$ 0 $\wedge$ X $<$ 6.
`excellent(X,M)` $\Leftrightarrow$ X $\geq$ 6 $\wedge$ X $<$ 9 $\wedge$ $3*M = X - 6$.
`excellent(X,1)` $\Leftrightarrow$ X $\geq$ 9 $\wedge$ X $\leq$ 10.

Compliance Fuzzy Sets

Low Compliance: **(5)**
`lowCompliance(X,1)` $\Leftrightarrow$ X $\geq$ -1 $\wedge$ X $<$ -0.3 .
`lowCompliance(X,M)` $\Leftrightarrow$
    X $\geq$ -0.3 $\wedge$ X $<$ 0 $\wedge$ $3*M =$ -10$*X$.
`lowCompliance(X,0)` $\Leftrightarrow$ X $\geq$ 0 $\wedge$ X $\leq$ 1.

Compliant: **(6)**
`compliant(X,0)` $\Leftrightarrow$ X $\geq$ -1 $\wedge$ X $<$ -0.3 .
`compliant(X,M1)` $\Leftrightarrow$
    X $\geq$ -0.3 $\wedge$ X $<$ 0 $\wedge$ $3*M1 = 10*X + 3$
`compliant(X,M2)` $\Leftrightarrow$
    X $\geq$ 0 $\wedge$ X $<$ 0.3 $\wedge$ $3*M2 = 3 - 10*X$
`compliant(X,0)` $\Leftrightarrow$ X $\geq$ 0.3 $\wedge$ X $\leq$ 1.

High Compliance: **(7)**
`highCompliance(X,0)` $\Leftrightarrow$ X $\geq$ -1 $\wedge$ X $<$ 0.
`highCompliance(X,M)` $\Leftrightarrow$
    X $\geq$ 0 $\wedge$ X $<$ 0.3 $\wedge$ $3*M = 10*X$.
`highCompliance(X,1)` $\Leftrightarrow$ X $\geq$ 0.3 $\wedge$ X $\leq$ 1.

Table 2. Definitions for Ratings and Compliance Levels

Rating	
Rating is "poor" enough	**(8)**
`isPoorEnough(R)` $\Leftarrow$ `poor(R,M)` $\wedge$ `M` ≥ 0.5 $\wedge$ `M` ≤ 1.	
Rating is "moderate" enough	**(9)**
`isModerateEnough(R)`$\Leftarrow$`moderate(R,M)`$\wedge$`M`$\geq 0.5 \wedge$`M`≤ 1.	
Rating is "good" enough	**(10)**
`isGoodEnough(R)` $\Leftarrow$ `good(R,M)` $\wedge$ `M` ≥ 0.5 $\wedge$ `M` ≤ 1.	
Rating is "excellent" enough	**(11)**
`isExcellentEnough(R)` $\Leftarrow$ `excellent(R,M)` $\wedge$ `M` ≥ 0.5 $\wedge$ `M` ≤ 1	
Compliance	
Compliance is "low" enough	**(12)**
`isLowEnough(C)` $\Leftarrow$ `lowCompliance(C,M)` $\wedge$ `M` ≥ 0.5 $\wedge$ `M` ≤ 1	
Compliance is "compliant" enough	**(13)**
`isCompliantEnough(C)` $\Leftarrow$ `compliant(C,M)` $\wedge$ `M` ≥ 0.5 $\wedge$ `M` ≤ 1	
Compliance is "high" enough	**(14)**
`isHighEnough(C)` $\Leftarrow$ `highCompliance(C,M)` $\wedge$ `M` ≥ 0.5 $\wedge$ `M` ≤ 1	

that our model can incorporate other types of fuzzy membership functions, for which only the definitions in Table 1 need change.

3.2. Axioms(Rules)

The axioms of the reasoning system are defined using the basic predicates defined in Tables 1 and 2. We discuss each of these rules in detail in this section.

Collusive Rating: (15)
This rule defines the condition when a particular rating (in a particular invocation) is collusive. Collusion is determined with respect to a service/provider. The rule is defined as follows:

$$\texttt{isInvalidCollusive(R, S, Inv)} \iff \texttt{ratingIn(R, Inv)} \wedge \texttt{serviceIn(S, Inv)} \wedge (\texttt{isExcellentEnough(R)} \vee \texttt{isGoodEnough(R)}) \wedge \forall A \in q.\texttt{complianceIn(C,A,Inv)} \wedge \texttt{isLowEnough(C)}$$

where q is the set of quality attributes in the system.

A rating (in a particular invocation) is identified as collusive with respect to a service/provider, if the rating is either "excellent" or "good", and the service had low compliance in all the quality attributes.

This means that the user rated the service highly even when the service performed poorly, hence a clear case of collusion between them.

Deceptive Rating: (16)
This rule defines the condition when a particular rating (in a particular invocation) is deceptive. The rule is defined as follows :

$$\texttt{isInvalidDeceptive(R, Inv)} \iff \texttt{ratingIn(R, Inv)} \wedge (\texttt{isPoorEnough(R)} \vee \texttt{isModerateEnough(R)}) \wedge \forall A \in q.\texttt{complianceIn(C,A,Inv)} \wedge \texttt{isHighEnough(C)}$$

where q is the set of quality attributes in the system.

A rating (in a particular invocation) is identified as deceptive, if the rating is either "poor" or "moderate", and in that invocation the service had high compliance in all the quality attributes.

This means the user rated the service poorly even when the service performed exceptionally well, hence a clear case of deception.

Bias of a Rating: (17)
This rule identifies the bias of a particular rating (in a particular invocation), i.e., it identifies which attributes the rating is biased towards. Here "bias" means that the user's rating is consistent with the performance delivered for that particular attribute, therefore the indication that the user places higher importance in that attribute because the performance in that attribute had higher significance in the user's rating. We declare that "a user rating has bias towards an attribute" if the rating properly reflects the performance in that attribute and therefore indicates preference to that attribute. The rule is defined as follows:

$$\texttt{biasOf(R, Inv, Attrib)} \iff \texttt{ratingIn(R, Inv)} \wedge \texttt{complianceIn(C, Attrib, Inv)} \wedge (\texttt{isPoorEnough(R)} \wedge \texttt{isLowEnough(C)}) \vee (\texttt{isExcellentEnough(R)} \wedge \texttt{isHighEnough(C)})$$

There are two conditions for determining bias. A rating (in a particular invocation) is biased towards a quality attribute, if the service performed poorly in that attribute and the user gave a low rating; or the service performed extremely well in that attribute and the user gave a high rating. The rule is based on the notion that an attribute that a user is biased towards has the most impact on the user's rating. Through multiple invocations of the rule, it is possible to identify all the biases of a particular rating.

Biased Rating: (18)
This rule defines the condition for declaring a rating as biased. The rule is defined as follows:

$$\texttt{isValidBiased(R, Inv)} \iff \texttt{ratingIn(R, Inv)} \wedge \exists A.\ \texttt{biasOf(R, Inv, A)} \wedge \neg\ \texttt{isInvalidCollusive(R, S, Inv)} \wedge \neg\ \texttt{isInvalidDeceptive(R, Inv)}$$

A rating is biased but valid if it is neither collusive nor deceptive, and has a bias towards an attribute.

This rule enforces that a biased rating is neither collusive nor deceptive, i.e. biasedness is mutually exclusive to both collusion and deception. Hence preserves the consistency of the knowledge base. This rule labels a rating as biased, whereas rule 17 identifies the biases of a rating.

Unbiased Rating: **(19)**
This rule defines unbiased rating.
`isUnbiased(R, Inv)` $\Longleftrightarrow$
    $\neg$ `isInvalidDeceptive(R, Inv)`
  $\wedge$ $\neg$ `isInvalidCollusive(R, S, Inv)`
  $\wedge$ $\neg$ `isValidBiased(R, Inv)`
Unbiased ratings are all those ratings that are valid, and there could be hundreds of rules to determine unbiased valid ratings. Hence, instead of declaring/defining numerous rules for valid unbiased ratings, we define a rating as unbiased, if it is neither biased, nor collusive, nor deceptive. The assumption is that there are only four types of ratings - collusive, deceptive, biased towards certain parameters, and the unbiased ones.

Valid Rating: **(20)**
This rule defines the validity of a rating.
`valid(R, Inv)` $\Longleftrightarrow$
    `isValidBiased(R, Inv)`
  $\vee$ `isUnbiased(R, Inv)`
A rating is valid if it is biased towards certain parameters, or if it is unbiased, i.e. all ratings identified as biased are valid, and all the ratings identified as unbiased are valid.

Invalid Rating: **(21)**
Ratings that are deceptive or collusive are invalid.
$\neg$ `valid(R, Inv)` $\Longleftrightarrow$
    `isInvalidCollusive(R, Inv)`
  $\vee$ `isInvalidDeceptive(R, Inv)`
All ratings that are collusive and deceptive are invalid. An invalid rating is the negation of a valid rating.

The basic knowledge base consists of the rules 1 to 21. These rules are complete enough for the reasoning system to be able to detect collusive and deceptive ratings and infer the biases of a rating. For further reasoning processes new rules can be added to the knowledge base. We discuss this further in a later section. At present, in the subsequent sections, we explain the operation of the reasoning system and discuss the prototype implementation of the basic reasoning engine.

4. Operation of the Reasoning System

Initially, the system contains only the axioms in its knowledge base, i.e. rules 1 to 21. Then once facts become available, the reasoning process starts. After each service invocation, the facts (Compliance details and the Ratings provided by the users) are entered in the system. Then for each rating provided, the system will determine whether it is collusive or deceptive or biased towards any attribute. If it is established to be none of these, then it is determined to be unbiased. The conclusion, i.e. the inference drawn (whatever it is among the four) is then added to the knowledge base, and can be used for further processing, such as determining user biases and determining a user as deceptive / collusive.

A deceptive / collusive rating is different from a deceptive / collusive user. Deceptive / collusive ratings are determined in a per transaction basis, whereas a user who gives consistently deceptive / collusive ratings is a deceptive / collusive user. Hence the distinction is appropriate.

First, we define the representation of the facts. The details of a particular rating are represented as:
`userIn(U, Inv)` $\wedge$ `serviceIn(S, Inv)`
$\wedge$ `ratingIn(R, Inv)` $\wedge$ `time(T, Inv).`
Where Inv is a unique invocation ID, U is the user ID of the user who invoked the service, S is the invoked service, R is the rating value provided by U in Inv, and T is the instance of time at invocation of S.

Similarly, details of performance statistics, i.e. Compliance statements are represented as:
`complianceIn(C1, Attrib1, Inv)`
$\wedge$ `complianceIn(C2, Attrib2, Inv)` $\wedge$ `...`
This representation format makes it possible to accommodate any number of attributes in the system without changing the representation. Moreover, new details pertaining to an invocation can be added into the knowledge base without significant changes in the representation format.

The reasoning system has fixed categorization levels for compliance levels and rating levels. To increase the categorization levels, the fuzzy rules (1-4 for Rating and 5-7 for Compliance) need to be changed accordingly, i.e., new rules need to be added into the knowledge base. This is reasonable because the categorization levels are fixed during the construction of the system and new categorization levels are not introduced later. Once the categorization levels are fixed, the system can function smoothly.

The system is flexible enough to allow any number of attributes to be used in the evaluation. Even different number of attributes or different attributes in different invocations will not hinder the operation of the system in any way. For example, in two invocations, two different sets of attributes are used, i.e. the system recorded performance statistics in different sets of attributes, then there may be no basis for comparing the results of the evaluations, but the system will, in each case, provide a valid conclusion irrespective of the number or types of attributes.

Furthermore, the system can provide justifications for its conclusions such as a rating is collusive or deceptive or biased, by tracing the rules used to arrive at the conclusion, this is the explicated rationale.

In the next section, we present our prototype implementation.

5. The Prototype Implementation of the Reasoning System for Generating Ratings Rationale

We developed a prototype knowledge-based reasoning engine in SWI-Prolog. The reasoning system operates as discussed in section 4. The reasoning system is then populated with random invocation results (compliance values) and ratings. Then the system is posed a set of queries -"Is the rating collusive / deceptive / biased / unbiased ?". The answers to these queries determine the validity of the ratings, which are helpful in making the subsequent decisions whether or not to include these ratings in the evaluation of the services. The system will terminate with success after finding a positive answer to any one of the queries. This is desirable because the properties deceptiveness, collusiveness, biasedness and unbiasedness are mutually exclusive, i.e. a rating is either deceptive or collusive or biased towards a set of parameters or unbiased, it can never be two or more of them at the same time. In addition, the system will find all the biases that the rating has which are then stored for future reference.

Then for generating explanations for the conclusions, we used the PML (Proof Markup Language) [2] developed by Knowledge Systems AI Laboratory at Stanford University. PML was developed for explaining the answers in the Semantic Web and provides portable and shareable explanations for question answering. Therefore, it is ideally suited for our purpose, that is to be able to exchange rationale along with ratings/reputation information between heterogeneous reputation systems. The explanations are in owl format and are portable, sharable, combinable and reusable.

The explanations can be represented as proof trees for visualization by human users and can be translated into plain English for human users. At the same time, the owl format is also machine readable. Therefore these explanations can be used to provide ratings/reputation rationale to human users as well as exchanged between reputation systems. For using these explanations in making decisions, we need to develop a decision support system, which is under investigation.

Having discussed the completed work so far, in the next section, we discuss enhancements to this basic reasoning system.

6. The Extended Knowledge Base

The basic knowledge base described in section 3 can be extended to include new reasoning features in the system. We discuss some of these extensions in this section.

The rules formulated in the basic knowledge base make it possible to identify collusive and deceptive rating and infer the biases of a rating, taking into account a single invocation. However, to determine collusion between a user and a service, to identify a deceptive user and infer the biases of a user, new rules should be added to the knowledge base. We discuss each of these extensions in the subsequent sub sections.

In the subsequent formulation of rules, we use the following notation:
We denote "there exists `e` such that `p(e)`" by

```
    ∃ e.  p(e)
```

and "for all `e` such that `p(e)`, `q(e)` is true" by

```
    ∀ e.  p(e):  q(e)
```

Identifying a Collusive/Deceptive User

A collusive user is one who provides consistently collusive ratings. Collusion between a user and a service can be determined by examining the ratings provided by the user to the service. If it is found that the user has been consistently providing collusive ratings to a service, then collusion is determined. The definite rule for determining collusion is when a user always gives collusive ratings to the service. This rule is defined as follows:

```
isCollludingWith(User, Service) ⟸
∀ Inv.  userIn(User, Inv) ∧
  serviceIn(Service,Inv) ∧ ratingIn(R,Inv):
  isInvalidCollusive(R, Service, Inv)
```

A user is colluding with a service if in all of the invocations of the service by the user, the ratings provided by the user to the service were found to be collusive.

However, this may not be the case in all situations. A user may provide collusive ratings only sometimes, which will not be detected by the above axiom. Hence, there is a need for a more distinctive measure for determining collusiveness in a user. However, from the definition of collusion itself, it is evident that a user that gives a single collusive rating is also collusive because a fair user would never give a high rating to a service, when the service performed poorly in all the quality attributes. In this situation, a collusive user will be identified using the following rule.

```
isCollludingWith(User, Service) ⟸
∃ Inv.  userIn(User, Inv) ∧
  serviceIn(Service,Inv) ∧ ratingIn(R,Inv) ∧
  isInvalidCollusive(R, Service, Inv)
```

Therefore, to differentiate between degrees of collusion, we can have a notion of degree of collusion, which can be defined as follows:

```
degreeOfCollusion(User,Service,Value) ⟸
   percentageCollusion(User, Service, Value)
```

Where the predicate `percentageCollusion(User, Service, Value)`, given User and Service, returns a Value which is defined as follows:
Value = Number of collusive ratings provided by User to Service/Total number of ratings provided by User to Service.

Here, the predicate `percentageCollusion/3` can be an external routine that performs this computation. Using this axiom, it is possible to identify a user as certain percent collusive instead of simply stating that the user is collusive.

Similarly, for determining a deceptive user, similar rules follow. The only difference is, deception is unrelated to a service and is a property of a user only, whereas collusion is with respect to a service.

Determining Biases of a User

The bias of a user will be towards those attributes towards which most of his/her ratings are found to be biased. Bias determination is useful for recommending suitable services to the user. The simplest case is when in every invocation the user (rating) shows bias towards the same (set of) attributes. In this case, the biases of a user are determined using the following rule:

```
userBias(User, Attrib) ⟸
∀ Inv.  userIn(User,Inv) ∧ ratingIn(R,Inv):
   biasOf(R ,Inv, Attrib)
```

A user is biased towards an attribute, if in all of the past invocations, the user has provided ratings biased towards that attribute.

Now, for aggregating all the biases, we can use an inbuilt predicate in Prolog `findall/3` to find all the biases of a user and store it in a list using the following axiom:

```
biasList(User, AttribList) ⟸
    findall(A, userBias(User, A), AttribList)
```

However, it may not always be the case that a user is found biased towards the same set of quality attributes in every invocation. The ratings provided by a user may have different biases in different invocations. Therefore, in such cases, we require a different mechanism for determining the biases of the user. For such situations, we propose the following possible solutions:

- Dominant biases: Consider the dominant biases, i.e. biases that occur in most of the invocations.
- Union of all biases: Combine all the biases in individual ratings (invocations).
- Weighted majority: Assign weights to different biases according to the frequency of occurrence of the biases.
- Most frequent: Consider only those biases that occur most frequently.

Biases thus determined can be used for providing suitable recommendations to the users. We discuss this in the following section.

Providing Recommendations

Using the various techniques discussed above, user biases (towards a set of attributes) are established. This information is then stored in the knowledge base. Based on such previously inferred biases, the system can make personalized recommendations to the user. There could be several techniques for providing recommendations. We list the following three:

- Recommend a service that has been rated highly by other users.
- Recommend a service that has been rated highly by other users who have similar preferences (similar to collaborative filtering recommendations).
- Recommend a service that has performed well in the attributes which are most important to the user:

  ```
  recommend(S, User) ⟸
     biasList(User, AttribList)
  ∧ ∀Inv.serviceIn(S,Inv)∧ a∈AttribList:
     complianceIn(C, a, Inv) ∧
    (isHighEnough(C)∨isCompliantEnough(C))
  ```

  Recommend the service to the user, which, in all of it past invocations, performed well (i.e. had high compliance or was compliant) in all the quality attributes the user has bias towards.

The first approach uses only ratings provided by the users, irrespective of the validity of the ratings and ignores the preferences of the user for whom recommendation is sought. Therefore, it is not very desirable.

The second approach takes user preferences into account, but it has some limitations. Firstly, it employs user similarity to find services that have been rated highly by similar users. It is difficult to determine similarity between users. One case would be: those users that have exactly the same preferences as the target user (for whom recommendation is being generated) are similar. In all other cases, it is difficult to determine similar users (i.e. users with similar preferences). If we consider only strict similarity, then there may be less number of users that are similar, and even less number of invocations for each service, hence the recommendation may not be accurate/suitable. Moreover, some ratings may be invalid (a possible occurrence because similar user does not mean that the ratings given by the users are all valid), further decreasing the number of invocations to be considered.

The third approach eliminates all of these issues because it uses only the actual performance of the services, so, all the invocations can be taken into account. Also, it utilizes the inferred biases of the target user in providing the recommendations. Hence, it is by far the most desirable recommendation technique.

The proposed axiom however, considers the most simplest case only and is inadequate for different situations. For example, the user for whom recommendation is being prepared (sought) may have preference towards more than one attribute and there may be no service that performed excellently in all those attributes in all the invocations. Alternatively, there may be more than one service that performed well in those attributes. In these situations, we need a more sophisticated mechanism.

Lets consider the first case, i.e. when there are no services that performed well in all the parameters that are important to the user. One solution to this problem could be generalizing the rule, i.e. removing a few parameters from the list of attributes that are important to the user. Then the question arises, which parameter to remove? In case the user has been found to be biased towards different attributes in different invocations, the attribute that the user is less frequently biased towards can be removed to find more matches for the recommendable services.

In the second case, i.e. when there are more than one service that have performed well in the parameters that are important to the user; if one of the services had high compliance in all the attributes important to the user more number of times than the others, the above rule does not detect this, and will recommend whichever service it evaluates first. To solve this problem (i.e. termination at finding of a single solution), we can ask a different query. Such as "Find all the services that performed well in all the parameters that are important to the user". Then on the result, we can apply different rules to choose between the services based on the number of times the service performed better than the other service.

7. Conclusions and future directions

We have established that analyzing user rating behavior against service performance gives insight into rationale behind user ratings. In summary, we have made the following contributions:
(i) Identification of invalid (collusive and deceptive) ratings
(ii) Identification of deceptive and collusive users
(iii) Identification of user preferences (bias towards certain quality attributes)
(iv) Inference and explication of rationale behind user ratings
(v) Mechanisms for personalized service recommendations based on the services' past performance and inferred user biases

Through identification of invalid ratings, the proposed model enhances the functionality of the Web Services Reputation System, making it robust from attacks by malicious entities. Also inclusion of rationale along with ratings makes it possible to personalize and tailor reputation evaluations to particular user requests (user requirements), subsequently enhancing the functionality of the service broker. Existing reputation systems which employ aggregation of user ratings as the basis for reputation assessment can make use of our reasoning engine to filter out invalid ratings. Also, the reputation models that take user preferences into account while assessing reputation [8, 10] can make use of our reasoning engine to infer user preferences instead of asking users explicitly. In addition, the system can provide explanation for its conclusions, i.e. how the system established that a particular rating / user is collusive / deceptive / biased towards ceratin quality attributes. Such explanations give credibility to the deductions, and the recipient of such information can use the information accordingly. Moreover, because the explanations are in owl format, they are portable and shareable and can be exchanged along with the ratings and reputation values between heterogeneous reputation systems operating in different domains.

However, to fully exploit the benefits of our reasoning engine, the reputation evaluator needs to have mechanisms to incorporate rationale into its reputation calculations, which is currently unavailable. Therefore, our next step is to design and develop a reputation evaluator which incorporates rationale inferred by the reasoning engine into its reputation calculations so that the system can deliver personalized and customized reputation assessment as per the user request.

In this paper, we presented one way of generating rationale for ratings (i.e. using objective service performance and subjective user ratings). However, various types of reputation systems implementing different reputation evaluation mechanisms / algorithms are in existence. For a fully functional network of heterogeneous reputation systems that can exchange reputation information, we need to be able to explicate rationale for every, or at the very least, a few unique types of reputation systems. Therefore, our next step is generating rationale for different reputation evaluation mechanisms. Moreover, to fully exploit the benefits of rationale explication, we need a decision support system that evaluates the rationale and provides a basis for comparison of the different kinds of rationale. This is also currently under investigation.

References

[1] F. Curbera, W. Nagy, and S. Weerawarana. Web services: Why and how. In *Proc.s of the Workshop on Object-Oriented Web Services, held in conjunction with OOPSLA 2001*, Tampa, Florida, USA, October 2001.

[2] P. P. da Silva, D. L. McGuinness, and R. Fikes. A proof markup language for semantic web services. *Information Systems*, 31:381–395, June-July 2006.

[3] C. Dellarocas. Immunizing online reputation reporting systems against unfair ratings and discriminatory behaviour. In *Proc.s of the 2nd ACM Conference on Electronic Commerce*, pages 150–157, October 2000.

[4] J. L. Herlocker, J. A. Konstan, and J. Riedl. Explaining collaborative filtering recommendations. In *Proc.s of 2000 ACM Conference on Computer supported cooperative work*, pages 241–250, Philadelphia, Pennsylvania, United States, 2000.

[5] S. Kalepu, S. Krishnaswamy, and S. W. Loke. Reputation = f(user ranking, compliance, verity). In *Proc.s of the IEEE International Conference on Web Services (ICWS 2004)*, pages 200–207, San Diego, California,USA, July 2004. IEEE Press.

[6] K.-J. Lin, H. Lu, T. Yu, and C. en Tai. A Reputation and Trust Management Broker Framework for Web Applications. In *Proc.s of the 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE'05)*, pages 262–269, Hong Kong, March 2005.

[7] A. Mani and A. Nagarajan. Understanding Quality of Service for Web Services . January 2002. http://www-106.ibm.com/developerworks/library/wsquality.

[8] E. M. Maximilien and M. P. Singh. Reputation and endorsement for web services. *SIGEcom Exchanges (ACM Special Interest Group on E-Commerce)*, 3(1):24–31, 2002.

[9] L. Mui, M. Mohtashemi, and A. Halberstadt. A computational model of trust and reputation. In *Proc.s of the 35th Hawaii International Conference on System Sciences*, volume 7, page 188. IEEE Press, 2002.

[10] W. Sherchan, S. Krishnaswamy, and S. W. Loke. Relevant past performance for selecting web services. In *Proc.s of the 1st International Workshop on Services Engineering (SEIW 2005), held in conjunction with the 5th International Conference on Quality Software (QSIC 2005)*, Melbourne, September 2005. IEEE Press.

[11] W. Sherchan, S. W. Loke, and S. Krishnaswamy. A fuzzy model for reasoning about reputation in web services. In *Proc.s of the 21st ACM Symposium on Applied Computing (ACM SAC 2006) - Trust, Recommendations, Evidence, and other Collaborative Know-how (TRECK) track*, Dijon, France, April 2006. ACM Press.

[12] R. Sinha and K. Swearingen. The role of transparency in recommender systems. In *CHI '02 extended abstracts on Human Factors in Computing Systems*, pages 830–831, Minneapolis, Minnesota, USA, April 2002. ACM Press.

[13] R. M. Sreenath and M. P. Singh. Agent-based service selection. *Journal on Web Semantics*, 1(3):261–279, April 2004.

[14] C. Vaucheret, S. Guadarrama, and F. Bueno. Ciao Prolog Reference Manual: Fuzzy Prolog . July 2004. `http://clip.dia.fi.upm.es/Software/Ciao/ciao_html/ciao_113.html#SEC467`.

[15] A. Whitby, A. Josang, and J. Indulska. Filtering out unfair ratings in bayesian reputation systems. In *Proc.s of the 7th International Workshop on Trust in Agent Societies, at AAMAS 2004*, New York, July 2004.

[16] B. Yu and M. P. Singh. An evidential model of distributed reputation management. In *Proc.s of the 1st International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 294–301, Bologna, Italy, July 2002. ACM Press.