EADRM: A Framework for Explanation-Aware Distributed Reputation Management of Web Services

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

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

Abstract

We propose the EADRM (Explanation-Aware Distributed Reputation Management) framework for sharing web services' reputation in heterogeneous environments. This framework advocates rationale extraction for meaningful exchange of reputation and includes a decision support algorithm for combining rationale-fortified recommendations.

1. Introduction

Reputation is recognized as a key factor for services selection and reputation systems aid in this purpose. Current reputation systems typically operate within a centralized context/domain or on the basis of a peer-to-peer social network. In the centralized approach [5, 6, 7, 4], a single reputation server generates, stores and disseminates reputation data centrally. In the peer-to-peer approach [8, 15, 1, 14, 3, 16], each user generates and manages reputation data locally and distributes it to the users seeking reputation information through its "social" network. The centralized and peer-to-peer approaches complement each other well, with the centralized approach providing ease of use and the peer-to-peer approach providing personalization. However, both of these approaches lack support for excahnge of reputation information between heterogeneous reputation systems. This issue is particularly significant in the context of web services. Web services are inherently distributed and large-scale distribution of this domain is anticipated [8], hence it is reasonable to expect existence of distributed reputation systems, which may be heterogeneous (i.e., using different reputation evaluation algorithms). Existing reputation systems fail to deliver in such cases because they do not cater for sharing reputation information between heterogeneous systems. Hence, this paper proposes a hybrid approach for managing reputation in open distributed systems, which incorporates the benefits and eliminates the limitations of the centralized and peer-to-peer approaches, provides the users meaningful and relevant evaluations of services reputation and facilitates exchange and reuse of reputation information across heterogeneous reputation systems.

For sharing of meaningful reputation information between heterogeneous reputation systems, in addition to the reputation information, extra information that helps in understanding and using the reputation value should also be included. This extra information is termed as *rationale*. Rationale provides justification, explanation, meaning and context to ratings and reputation values and plays an important part in understanding and using the reputation data. Therefore, an important part of this framework is to determine how such reputation values and associated rationale (obtained from various sources) can be meaningfully integrated. Therefore, in this paper, we propose a decision support algorithm to combine recommendations and rationale obtained from various reputation servers and with the help of a case study demonstrate the operation and the feasibility of the proposed reputation management framework and the decision support algorithm.

The remainder of the article is structured in the following way. We first discuss our proposed framework for sharing reputation information in section 2. In section 3, we propose a decision support algorithm to combine recommendations with rationale. In section 4, we present a case study to demonstrate the operation of the EADRM framework and the decision support algorithm. Finally, we conclude and highlight the contributions of our framework in Web Services reputation management and point to future work in Section 5.

978-0-7695-3473-2/08 $25.00 © 2008 IEEE
DOI 10.1109/APSCC.2008.258

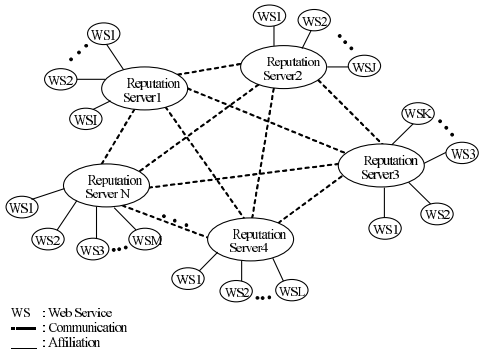

Figure 1. Overview of the Explanation-Aware Distributed Reputation Management (EADRM) Framework.

2. Overview of the Explanation-Aware Distributed Reputation Management (EADRM) Framework

In this section we describe our proposed hybrid approach for reputation management in open distributed systems, which we have named EADRM (Explanation-Aware Distributed Reputation Management) framework. Figure 1 shows the conceptual overview of the EADRM framework.

The EADRM framework consists of a collection of "Reputation Servers" interacting and exchanging meaningful reputation information (i.e., with rationale). Web Services register with one or more reputation servers for their reputation management. Services that are willing to be evaluated on their QoS provision by third parties (reputation servers) are regarded as reliable by the users. Therefore, services have the incentive to register with reputation servers for reputation management. However, some services may not want to be evaluated, and therefore, will have the option not to register with any of the reputation servers. Services that register with one or more reputation servers will have their reputation computed by the corresponding reputation servers. Services that register and allow their reputation to be computed will benefit from the value of openness and willingness to be evaluated, and will be considered favorably by the users as opposed to those services that do not want to be evaluated. This intuition provides the basis for our assumption that services will register with one or more reputation servers for their reputation management. We note that service providers may register with service registries such as UDDI for their service advertisements, which is different from their registration with a reputation server. Registration with the UDDI-like registries allows the services the opportunity to advertise their products or offerings whereas registration with reputation servers allows the services to be evaluated for their performance and thereby increases their likelihood of being selected by the users (based on their reputation). In the EADRM framework implementation, reputation servers could be associated with the service registries or operate independently of the registries. Next, we describe the major components of the EADRM framework.

Reputation Evaluation System (RES): The Reputation Evaluation System (RES) component of a Reputation Server is responsible for managing reputations of services registered with the Reputation Server. Therefore, any reputation system (based on centralized approach) existing in the literature can be implemented as the RES component in the EADRM framework.

Rationale Generation System (RGS): The Rationale Generation System (RGS) generates rationale for the reputation information computed by the reputation evaluation system (RES). Typically, three types of reputation information are available for each entity: (i) ratings, (ii) reputation and (iii) ranking. In each of these cases, the associated rationale generation is performed by the RGS. Therefore, three types of rationale can be generated: (i) rating rationale, (ii) reputation rationale, and (iii) ranking rationale. Readers are referred to [12, 13, 11] for discussions on rationale types and generation.

Rationale Representation System (RRS): The rationale generated by the RGS has to be represented in a standard format so that it can be shared with other reputation servers. The problem for the Rationale Representation System (RRS) is non trivial since various types of reputation systems are in existence in literature, for which various pieces of rationale can be generated. Therefore, representing a wide variety of reputation related rationale information in a single standard format is a challenge. The Rationale Representation System (RRS) component is responsible for performing this representation. We prescribe the **R**eputation R**a**tionale **M**arkup **L**anguage (RaML) for this representation. The output of the RSS is ready to be shared with other reputation servers and to be used in decision making by the Decision Support System (DSS).

Decision Support System (DSS): If a Reputation Server does not contain reputation information regarding a particular service or is not confident on the information that it contains, then the Reputation Server contacts other reputation servers and combines obtained information to provide a comprehensive evaluation to the user. The evaluation and aggregation of information obtained from various reputation servers is performed by the decision support system (DSS). The DSS could be embedded within each Reputation Server or within special web services that provide reputation service (i.e., reputation services). If a reputation server does not have a DSS component, then it can outsource the

decision making process to external services that provide decision support service or to other Reputation Servers that have the DSS component. The main objective of a DSS is to combine reputation information plus rationale obtained from various reputation servers. The problem is trivial if all the reputation servers recommend the same service or provide the same reputation value for a service. However, this is highly unlikely, specially if the reputation servers are heterogeneous, i.e., their RES component uses different algorithms for computing reputation. When there is a significant mismatch between the recommendations or reputation evaluations from various Reputation Servers, the DSS plays a decisive role and provides a mechanism for combining conflicting recommendations. Any decision support system can be implemented to work with the EADRM framework as long as it has the capability to combine rationale with the recommendations and reputation values and provide explanations for its final recommendations.

Communication Protocol: The Reputation Servers communicating with each other form a Peer-to-Peer type of network, therefore any standard communication protocol designed for Peer-to-Peer networks is suitable for the EADRM framework. In addition, for exchange of rationale-fortified reputation related information, we have developed a specific language, the **R**eputation R**a**tionale **M**arkup **L**anguage (RaML), which is available from http://www.csse.monash.edu.au/∼wanitas/phd.html.

Each Reputation Server is centralized in its domain of operation, therefore, the Reputation Server is responsible for calculation, evaluation, updating and management of reputation of Web Services registered with it and end users are unconcerned about the computations and need not spend their valuable resources on reputation management. This incorporates the benefits of the centralized approach into the EADRM framework. If a user wants personalized reputation evaluation based on its own previous interactions or its biases (i.e., incorporating the benefits of the peer-to-peer approach), the user can register with one of the reputation servers, which will then conduct all reputation management on behalf of the user. User specific information such as the user's preference for certain attributes, quality requirements and desired period of history to consider can be taken into account for personalized reputation evaluation of the services.

Four issues need to be addressed for the realization of the EADRM framework: rationale generation, representation, communication and integration. Readers are referred to [12, 13, 11] for discussions on rationale types and generation. In the next section, we address rationale integration and present a decision support algorithm for combining service recommendations with rationale. Representation and communication of rationale is currently under study.

3. A Decision Support Algorithm for Combining Reputation Information Obtained from Heterogeneous Sources

In this section, we discuss the Decision Support System (DSS) component of the EADRM framework and present an example decision support algorithm. Rationale refers to information that is useful for understanding the context of the provided information and helps in making an informed decision. Therefore, information that constitute rationale need to be analyzed and compared. The rationale accompanying a reputation value consists of various pieces of information such as the type of reputation value, the type of information used to build the reputation value, the source of the reputation value, the size and length of history included in the computation of the reputation value and the biases of the computation. The recipient of this information can use its own discretion to use a subset of this information to make decisions. For example, (say) a user distrusts reputation computations based on subjective user ratings. Then in this case, the DSS may choose to discard all reputation information whose rationale state that the reputation computation was based on subjective user ratings, and may only include those based on objective performance metrics. Similarly, a user may prefer that ratings older than a certain date be excluded from the reputation evaluation. Another user may prefer that only those ratings provided by users with the same preference be included.

Alternatively, instead of using one piece of information as the deciding factor, a Reputation Server may use several pieces of rationale to make a decision. in such cases, the pieces of rationale need to be combined in a meaningful way so as to enable comparison of the rationale obtained from several sources. This process is termed as *rationale strength evaluation*. Evaluation of *rationale strength* could be quantitative as well as qualitative. Quantitative rationale strength evaluation refers to use of mathematical functions for relating various pieces of rationale, whereas qualitative rationale evaluation could be use of logic rules to relate various pieces of rationale. Therefore, DSSs vary according to the way they present and process rationale information and exact implementation and application of two DSSs may vary greatly even though they use the same pieces of rationale for making decisions. The policies of the DSS and the preferences of the user play a great role in the use of rationale and therefore it is unrealistic to prescribe a generic decision support algorithm for use by everyone.

The decision support algorithm proposed in this section uses rationale for a particular rank (or *rank rationale*) as the basis for decision making. We selected *rank rationale* as the basis because *rank rationale* is the discriminating factor to consider when assessing rankings. Typically, two pieces of information will always be available as part of the *rank ra-*

tionale for each service (irrespective of their ranks): (i) the number of past invocations of the service (size of history) and (ii) the length of history of invocations of the service. Typically, this information will be different for each service in the ranking because it is unlikely that each service in the ranking will be invoked the exact number of times and for the exact duration. Hence these two pieces of information are always the differentiating factors to consider. For example, a service may be ranked highly by a reputation server, but the associated rationale may indicate that the service hasn't been invoked for the last two months (i.e., ranking was based on two months old information), in this case the user may choose to give priority to another service ranked lower but based on more recent information.

Next, we propose our decision support algorithm for the following two scenarios:

Scenario 1: All the Reputation Servers provide one recommendation each (with rationale).

Basic Algorithm: Select the recommendation with the strongest rationale. So, the choice function is $c = strongest(Rationale)$, where *Strongest* is a function which takes a set of *Rationale* and returns the strongest rationale, where strength of rationale is determined via some algorithm. For example, if using quantitative evaluation, the rationale strength evaluation algorithm could be a function relating the pieces of rationale to obtain a quantitative value. We present one quantitative rationale evaluation function in section 4.1.

If rationale strength is evaluated as a quantitative value (referred to as *rationale points*), the algorithm would take the following form:

Let there be *n* Reputation Servers $(RS_1, ..., RS_n)$. Each Reputation Server provides one recommendation (with rationale). Let each service be represented by S_i, and the list of recommended services be $S_1, S_2, ..., S_m$. Also, let $Rationale^i_{RS_j}$ be the rationale points associated with the recommendation of service *i* by the Reputation Server *j*. Then, for each web service, *i*, calculate the *total rationale points* as the sum of the *rationale points* for each recommendation of *i* by a Reputation Server. The Web Service with the highest *total points* will be the final recommendation to the user.

Total points for service i, $P_i = \sum_{j=1}^{k} Rationale^i_{RS_j}$

where k is the number of Reputation Servers that recommended service S,

Therefore, in this case the choice function is $c = max(P_i)$

Scenario 2: All the Reputation Servers provide a ranked list of services (plus rationale) as recommendation. In this case, we propose to combine our basic algorithm (described above) with a voting system.

Algorithm: Select the recommendation with the strongest combination of votes and rationale. So, the choice function is $c = Strongest(Vote, Rationale)$, where *Strongest* is a function which takes a set of rationale and vote combinations and returns the the strongest combination of rationale and votes, where (again) the strength of the combination is determined by an algorithm.

Vote captures the order of recommendation of the services. Rationale strength refers to the support for the recommendation. For example, a Reputation Server recommends two web services in order WS1,WS2; then the votes for WS1 will be higher than that for WS2. However, the corresponding rationale for the recommendations may not reflect the ranking order. Say, WS1 was recommended based on 10 invocations, and WS2 on 100 invocations. Then in this case the size of history gives the quantitative rationale strength and comprises the *Rationale* part of the choice function.

Again, similar to the basic algorithm, for quantitative rationale strength evaluation, the algorithm would take the following form:

Algorithm: Sum the combination of the votes and "rationale points" to compute "total points" for each Web Service and recommend the service with the highest "total points". One major benefit of combining "rationale" with voting is resolving ties in case of equal number of votes for two or more services (a typical case, cyclic preference for candidates in situations where the number of voters equals the number of candidates). In such situations, rationale serves to differentiate among the Web Services even though they receive equal number of combined votes from the Reputation Servers.

For the same situation as described in the scenario 1. Let $Vote^i_{RS_j}$ be the vote for Service *i* by Reputation Server *j*. Then, in this case, total points is calculated as:

Total points for service i, $P_i = \sum_{j=1}^{k} Vote^i_{RS_j} \times Rationale^i_{RS_j}$

where k is the number of Reputation Servers that voted for the service i

Similar to the basic algorithm, in this case also, the choice function is $c = max(P_i)$

In both the cases, the final result or recommendation can be provided as a recommendation for a single Web Service (the one that has the highest total points), or a ranked list of Web Services (say top 3 Web Services with the highest total points). This recommendation can then be supported by its own rationale. The rationale for the final recommendation could include (1) average percentage of the recommended service(s) in the ranked list out of the total set of services supported by all the Reputation Servers, (2) percentage of occurrence of the service in the Reputation Servers (say, the service was ranked/available in 60% of the Reputation Servers), (3) percentage of occurrence of the service in the top 3 rank (say). This extra information allows the users the flexibility to make their own decisions based on their preferences. For example, a user may not be particularly bothered about the rationale and may want to use a service

that has been ranked topmost by most Reputation Servers. Another user may prefer a Web Service that is ranked in most Reputation Servers as it shows consistency in the Web Service's performance (because unranked implies that either there was no information available or that the performance was not good). Provision of rationale along with the recommendation allows this flexibility to the users.

As can be seen from the above discussion, the algorithm requires two distinct types of information: the rationale points and the votes. The rationale points are computed using a *rationale strength evaluation* function and determined by the pieces of rationale information available. The votes are determined by a voting system and can be specified for all decision support systems to use. Therefore, in the next section, we discuss our choice of voting system to be used in this decision support algorithm.

Choice of voting system: Voting systems are typical methods of making social choices by a collection of individuals [9]. Many different kinds of voting procedures exist in the literature. Among the various preferential voting systems, we selected Borda count voting [9] as our choice of voting algorithm because it allows integration of rationale with votes. The major focus of this paper is not in the development of voting procedures. We merely wish to illustrate how a voting system may be used with rationale to facilitate decision making when using reputation and rationale.

In Borda count, each voter (i.e., Reputation Server in our case) ranks the candidates in their ballot (i.e. the Web Services). If there are n candidates, the first-place candidate receives n points, the second place candidate receives n-1 points and so on with the last place candidate receiving 1 point. Another common variant is assigning n-1 points to the first-placed candidate and so on with the last place candidate receiving 0 points. We prefer the former method as the latter approach unfairly penalizes last place candidates, and even though a candidate (here, a web service) is placed last, it might be supported by stronger rationale so assigning it at least 1 point ensures that rationale is considered. Yet another variant exists for assigning points in which points are assigned in increasingly smaller fractions: the first preference gets 1 point, the second preference gets 1/2 points, the 3rd preference gets 1/3 points and so on. This method favors the candidate with more first preferences as compared to the normal Borda Count. We refer to this method as Borda Count Variant.

In the next section, we demonstrate the operation of the EADRM framework through a case study of a reputation server employing relevant past performance based service recommendation algorithm proposed in [10]. We explain what constitutes rationale, discuss how rationale strength could be evaluated and demonstrate how the the decision support algorithm could be implemented in this case.

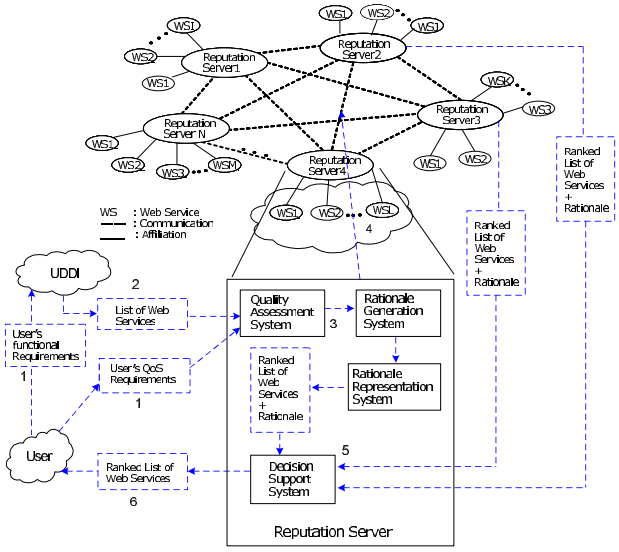

Figure 2. Operation of the EADRM Framework (with Expanded View of a Reputation Server).

4. Case Study: Web Service Recommendations with Rationale

The main aim of this case study is to demonstrate the feasibility of EADRM framework by presenting the operation of the EADRM framework. Hence, the sub aims are:

- to demonstrate rationale extraction and
- to demonstrate implementation of the decision support algorithm through illustrative scenarios

In this case study, we consider a system of Reputation Servers communicating and interacting with each other and providing service recommendations to users within the EADRM framework. Figure 2 depicts the components of a Reputation Server that employs the personalized service recommendation algorithm proposed in [10]. The figure shows that the Reputation Server consists of the four components - (i) the reputation evaluation system, (ii) the rationale generation system, (iii) the rationale representation system and (iv) the decision support system. The Quality Assessment System comprises the relevant past performance based quality evaluation algorithm [10]. This algorithm is based on evaluating services' quality / reputation according to the current (requestor) user's QoS requirements. Therefore, first, a user's request is divided into functional requirements and QoS requirements. The functional requirements are provided to the UDDI (Universal Description, Discovery, and Integration) like registry to obtain a

list of services that provide the desired functionality (step 1). The QoS requirements are provided to the quality assessment system (step 1). The list of web services obtained from UDDI registry is then input to the quality assessment system (step 2).

Taking the user's QoS requirements and past performance history of the services, the quality assessment system then ranks the list of services (step 3). The Rationale Generation System generates the rationale for the ranking of the services and the Rationale Representation System explicates the generated ranking rationale. If the Reputation Server does not have information regarding all the services, it will then contact its neighboring Reputation Servers for reputation information regarding some or all of the services (step 4). The neighbor Reputation Servers provide the information in the form of a ranked list (in the example shown in the figure) and associated rationale. As a point to note, the neighbor Reputation Servers may or may not implement the same service recommendation algorithm to generate the web service rankings.

The Decision Support System combines the ranked list of services obtained from the neighbors and then computes the final ranking for the user (step 5) using a decision support algorithm. The decision support algorithm should enable the Reputation Server to combine the ranked recommendations with the associated rationale. The final ranking (along with corresponding rationale) is then forwarded to the user (step 6).

In the next section (4.1), we discuss how quantitative rationale can be given to complement the service recommendations generated by a reputation server which implements the relevant past performance based service recommendation algorithm. Then in the subsequent section (4.2), we present the prototype implementation of the decision support algorithm followed by some illustrative scenarios extracted from the implementation in section 4.3.

4.1. Evaluation of Rationale Strength for Algorithms using Similarity in User Requests

Rationale extraction is specific to the reputation evaluation algorithm. Hence there is no general method of extracting rationale. Rationale constitutes any information that is helpful in understanding the reputation data, especially information that enables distinguishing between two web services' reputation. For examples of rationale generation, readers are referred to [12] and [13]. In this section, we discuss generation of rationale and evaluation of rationale strength for algorithms that use similarity in user requests (for example, algorithms proposed in [10] and [2]). Both of these algorithms use similarity in user requests to segregate invocations so that the services can be evaluated based on their performance in similar requirement conditions. This incorporates contextual information into the quality evaluation process, making the evaluation tailored to user's particular needs.

Inherent in the algorithms are two major factors that support recommendations/rankings: (1) Number of similar invocations: n (higher n is desirable so that quality evaluation is based on a large history of interactions), and (2) Similarity threshold: δ (lower δ is desirable so that invocations identified as similar are indeed very similar to the current user request). Therefore, intuitively, a recommendation that is supported by high n and low δ would be desirable, as high n would mean that the service had many similar interactions in which it performed consistently and low δ means that these past interactions that were used to evaluate the service's quality were indeed very similar to the user's requirements (greater personalization). The strength of the rationale is higher for higher n and lower δ, therefore, a very simple rationale strength evaluation function in this case is $f(Rationale) = n/\delta$.

Therefore, the decision support algorithm uses n/δ as the rationale strength evaluation function. To note, use of n/δ does not necessarily mean all reputation servers used the relevant past performance algorithm to compute the rankings. All algorithms that use similarity in invocations as a basis for evaluation of service rankings would provide n and δ as part of the rank rationale (for example, [2] and [10]). Therefore, service rankings obtained from reputation servers that implement these algorithms can be combined using our proposed decision support algorithm. We would like to note that rationale obtained with the rankings may contain other pieces of information besides n and δ. However, this particular decision support algorithm uses only n and δ as the major criteria for decision making and therefore uses the rationale strength evaluation function $f(Rationale) = n/\delta$ to compute rationale for each service recommendation. Using this function and Borda count voting method, we developed a prototype implementation of the decision support algorithm which we present in the next section 4.2 followed by some illustrative scenarios in the subsequent section (4.3).

4.2. The Prototype Implementation of the Decision Support Algorithm

In our prototype implementation, we compared the operation of the following 4 algorithms:

(i) **Rationale only** (sum the rationale points (n/δ) for each of the recommended Web Services, whichever has the highest points will be recommended to the user):

Total points for Service i, $P_i = \sum_{j=1}^{m} (n/\delta)_j$

where m is the number of Reputation Servers that voted for the service i

$(n/\delta)_j$ is the rationale points corresponding to the recommendation of i by the j^{th} Reputation Server

Referring back to section 3, this algorithm corresponds to scenario 1, i.e., all the reputation servers provide one recommendation each.

(ii) **Voting only** (sum the votes (using Borda count) for each Web Service, the Web Service with highest vote will be recommended to the user):

Total points for Service i, $P_i = \sum_{j=1}^{m} Vote_j$

where m is the number of Reputation Servers that voted for the Service i,and

$Vote_j$ is the vote provided by the j^{th} Reputation Server to the Service i

(iii) **Borda Count with rationale** (our algorithm combining normal Borda count voting (votes from n to 1) with rationale points), and

(iv) **Borda Count Variant with rationale** (our algorithm combining Borda count variant voting (votes from 1 to 1/n) with rationale points)

For both algorithms (iii) and (iv):

Total points for Service i, $P_i = \sum_{j=1}^{m} Vote_j \times (n/\delta)_j$

where m is the number of Reputation Servers that voted for the Service i

$Vote_j$ is the vote provided by the j^{th} Reputation Server to the Service i, and

$(n/\delta)_j$ is the rationale points corresponding to the recommendation of i by the j^{th} Reputation Server

4.3. Illustrative Scenarios

The following test cases are selected from our prototype implementation. In all of the following cases, the tables show information (i.e, rationale strength and the votes) obtained from 3 reputation servers RS1, RS2 and RS3 and the results (ranked recommendations) after applying each of the 4 algorithms.

Table 1. Case 1

Case

Reputation Server	δ	Web Service	Size of "similar" history (n)	Rationale Strength (n/δ)	Vote
RS1	0.12	WS1	70	583.33	1
		WS2	50	416.66	2
		WS3	160	1333.33	3
RS2	0.12	WS1	150	1250.0	3
		WS2	80	666.66	1
		WS3	70	583.33	2
RS3	0.12	WS1	160	1333.33	2
		WS2	80	666.66	3
		WS3	170	1416.66	1

Results

Algorithm	Rationale Only	Voting Only	Borda Count(with rationale)	Borda Count Variant(with rationale)
Ranking	WS3	WS1,WS2,WS3	WS1	WS1
	WS1		WS3	WS3
	WS2		WS2	WS2

In the case shown in Table 1, each web service receives equal combined votes, therefore, in this case, Voting Only algorithm gives a 3-way tie. The tie is diffused and results provided by all 3 algorithms that include rationale. However, note the different recommendation provided by the different algorithms. In this case, Borda Count and Borda Count Variant provide the same recommendation (WS1), whereas Rationale Only provides a different recommendation (WS3).

Table 2. Case 2

Case

Reputation Server	δ	Web Service	Size of "similar" history (n)	Rationale Strength (n/δ)	Vote
RS1	0.129	WS1	80	620.15	1
		WS2	70	542.63	2
		WS3	190	1472.86	3
RS2	0.259	WS1	50	193.05	2
		WS2	60	231.66	3
		WS3	70	270.27	1
RS3	0.388	WS1	50	128.86	2
		WS2	60	154.64	3
		WS3	70	180.41	1

Results

Algorithm	Rationale Only	Voting Only	Borda Count(with rationale)	Borda Count Variant(with rationale)
Ranking	WS3	WS2	WS3	WS3
	WS2		WS2	WS2
	WS1	WS1,WS3	WS1	WS1

In the case depicted in Table 2, two Reputation Servers RS2 and RS3 recommend WS2, however the recommendation of RS1 (i.e.,WS3) is supported by stronger rationale. Therefore, all the algorithms that incorporate rationale recommend WS3 (note Voting only algorithm recommends WS2). This case illustrates the significance of rationale.

Table 3. Case 3

Case

Reputation Server	δ	Web Service	Size of "similar" history (n)	Rationale Strength (n/δ)	Vote
RS1	0.129	WS1	70	542.63	1
		WS2	50	387.59	2
		WS3	60	465.11	3
RS2	0.129	WS1	70	542.63	2
		WS2	50	387.59	3
		WS3	80	620.15	1
RS3	0.129	WS1	70	542.63	2
		WS2	50	387.59	3
		WS3	60	465.11	1

Results

Algorithm	Rationale Only	Voting Only	Borda Count(with rationale)	Borda Count Variant(with rationale)
Ranking	WS2	WS2	WS2	WS2
	WS3		WS1	WS3
	WS1	WS1,WS3	WS3	WS1

In the case depicted in Table 3, the recommendations provided by all 3 Reputation Servers have equivalent "Rationale Strength", therefore, the final recommendation is guided by the votes each web service receives (in this case two reputation servers support Web Service WS2, therefore all the algorithms recommend WS2, however, note the difference in the lower order recommendations).

These illustrative test cases demonstrate that the combined algorithms (integrating votes and rationale, i.e., Borda Count with rationale and Borda Count Variant with rationale) provide a better means of providing a solution to a potentially tied situation are superior to both voting only and rationale only algorithms. These scenarios also demonstrate the difference between the 4 algorithms.

Observations from the prototype implementation

1. The final recommendation is directly proportional to the size of history 'n' and inversely proportional to the similarity threshold δ. If a recommendation is supported by larger 'n' and lower δ, then it is favored in the final recommendation.

2. The algorithms ordered with respect to increasing sensitivity to 'n' and δ are: (1) Voting only, (2) Normal Borda count voting with rationale, (3) Variant of Borda count voting with rationale, and (4) Rationale only

3. Although a service is recommended by higher number of Reputation Servers, it may still be overlooked over another service recommended by fewer Reputation Servers but supported by stronger rationale (n/δ).

4. Given two Web Services, if they get equal total votes, the combined algorithms behave like the rationale-only algorithm.

5. Given two Reputation Servers' recommendations supported by "similar" rationale, the combined algorithms behave similar to the voting-only algorithm.

6. In cases where voting gives a tie between two or more Web Services, the combined algorithms provide a means to break this tie (with the help of rationale).

5. Conclusions and Future Work

In this paper, we proposed a hybrid approach for Web Services reputation management. The framework advocates inclusion of rationale with reputation information for better transfer, exchange and reuse of reputation information and for informed aggregation of reputation obtained from various sources. The core of the framework consists of reputation evaluation system, rationale generation system, rationale representation system, decision support system and communication protocols. The reputation evaluation system could comprise any existing reputation system or algorithm based on the centralized approach, and the corresponding rationale generation system would include rationale extraction mechanism for the algorithm. The rationale representation system would use a standard language (RaML) for representing rational generated by the rationale generation system. Within a case study, we demonstrated how quantitative rationale can be included to complement service recommendations. We presented and demonstrated the operation of an example decision support system that combines rationale with recommendations.

Standard communication protocols designed for Peer-to-Peer networks are suitable for the EADRM framework. In addition, for exchange of reputation rationale information, we are currently working towards defining a **R**eputation R**a**tionale **M**arkup **L**anguage (RaML) that provides a standard format for representation and communication of reputation-rationale between heterogeneous reputation systems. Existence of a standard platform such as RaML for representing rationale would facilitate comparison of rationale obtained from various sources and in various formats. Provision of RaML would make rationale easily usable and applicable in not only reputation systems but also in various other problem domains such as service and product recommendations and multi-agent negotiations.

References

[1] E. Damiani, S. D. C. di Vimercati, S. Paraboschi, and P. Samarati. Managing and Sharing Servents' Reputations in P2P Systems. *IEEE Transactions on Knowledge and Data Engineering*, 15(4):840–854, 2003.

[2] V. Deora, J. Shao, W. A. Gray, and N. J. Fiddian. A Quality of Service Management Framework Based on User Expectations. In *Proceedings of the 1st International Conference on Service Oriented Computing (ICSOC 2003)*, pages 104–114, Trento, Italy, 2003.

[3] T. D. Huynh, N. R. Jennings, and N. R. Shadbolt. An Integrated Trust and Reputation Model for Open Multi-agent Systems. *Autonomous Agents and Multi-Agent Systems*, 13:119–54, 2006.

[4] A. Josang and J. Haller. Dirichlet Reputation Systems. In *Proc.s of ARES'07*, pages 112– 119, Washington, DC, USA, April 2007. IEEE Computer Society.

[5] R. Jurca and B. Faltings. An Incentive Compatible Reputation Mechanism. In *Proc.s of the IEEE Conference on E-Commerce*, 2003.

[6] S. Kalepu, S. Krishnaswamy, and S. W. Loke. Reputation = f(User Ranking, Compliance, Verity) . In *Proc.s of ICWS'04*, 2004.

[7] Y. Liu, A. H. Ngu, and L. Zeng. QoS Computation and Policing in Dynamic Web Service Selection. In *Proc.s of WWW'04*. ACM Press, 2004.

[8] E. M. Maximilien and M. P. Singh. Reputation and Endorsement for Web Services. *ACM SIGecom Exchanges*, 3(1):24–31, 2002.

[9] H. Nurmi. Voting Procedures: A Summary Analysis. *British Journal of Political Science*, 13(2):181–208, April 1983.

[10] W. Sherchan, S. Krishnaswamy, and S. W. Loke. Relevant Past Performance for Selecting Web Services. In *Proc.s of QSIC'05*, 2005.

[11] W. Sherchan, S. Krishnaswamy, and S. W. Loke. Explaining Reputation for Informed Web Services Selection. In *Proc.s of ICWS'08*, 2008.

[12] W. Sherchan, S. W. Loke, and S. Krishnaswamy. A Fuzzy Model for Reasoning about Reputation in Web Services. In *Proc.s of ACM SAC (TRECK'06)*, 2006.

[13] W. Sherchan, S. W. Loke, and S. Krishnaswamy. Towards Explanation-Aware Selection in Internet-Scale Infrastructures: Generating Rationale for Web Services Ratings and Reputation. In *Proc.s of EDOC (MWS'06)*, 2006.

[14] R. M. Sreenath and M. P. Singh. Agent-Based Service Selection. *Journal on Web Semantics*, 1(3):261–279, April 2004.

[15] B. Yu and M. P. Singh. An Evidential Model of Distributed Reputation Management. In *Proc.s of AAMAS'02*, pages 294–301, New York, NY, USA, July 2002. ACM Press.

[16] R. Zhou and K. Hwang. PowerTrust: A Robust and Scalable Reputation System for Trusted Peer-to-Peer Computing. *IEEE Transactions on Parallel Distributed Systems*, 18(4):460–473, 2007.