A Framework for Progressive Trusting Services

Oana Dini
University of Besançon, France
oana.dini@univ-fcomte.fr

Pascal Lorenz
University of Haute Alsace, France
lorenz@ieee.org

Hervé Guyennet
University of Besançon, France
guyennet@lifc.univ-fcomte.fr

***Abstract* – The web-based transactions, web services, and service-oriented platforms require appropriate mechanisms to announce, select, and use different services. A user is always under the dilemma of 'use and trust' or 'trust and use' for different services based on the notion of service reputation. The interaction between every service provider and its users is regulated by the service level agreement and customer satisfaction feedback. The former is the basis for the technical audit, while the latter subjectively validates the user perception. Selection of a most appropriate service by correctly invoking is a challenge. This is due to the difficulties to correctly expose proper way to invoke a service, to the variety of services, from on-line services, software pieces, to shopping, and to different invoker behavior. When considering invoker feedback, service ranking based on user perception, or based on recommenders' statistics are relevant. A significant aspect is played by service similarity. The paper presents a framework and appropriate mechanisms to evaluate the services/providers in the light of their respective direct impact on user perception. To accurately evaluate the feedback after service/product consumption, we will refine the user profile by considering the dynamics of the feedback. The approach we propose deals with peaks in feedbacks. We consider quick negative and quick positive feedback as well as late vs. early feedback with respect to the time of the transaction. We propose formal concepts used in selecting an appropriate service. The paper presents adapted approaches to select services based on distance and similarity, and introduces a similarity taxonomy to better tune various kinds of service invocation under specific constraints, such feature relaxation, type of similarity, context, and service ranking. Selection is based on the feedback from the user. The proposed model is used for building a selection algorithm that allows variations on service invocation. The proposal is validated through use cases.**

Keywords – recommenders; reputation; dynamic feedback; services similarity; temporal similarity; use profiles, dynamic patterns.

I. INTRODUCTION

With the overwhelming amount of information, products, and services available over the Internet, it has become harder for the users to select the ones that fit best their needs or requirements. First of all, it is too difficult and time consuming to sort through hundreds of items and select the needed one. Also, there is the problem of trusting the provider for that item and not only that, but trusting that the provider is offering a product that meets the user's requirements. In order to assist the user in selecting the product or service that it needs, recommender systems have been proposed.

Recommender systems (RS) have been the subject of many studies and products over the last decade. The term was first brought up by Resnick and Varian [1], which, as mentioned in [2], it was mostly a replacement for "collaborative filtering" proposed in [3]. Recommender systems are defined as systems that collect ratings from users and then analyze the data to produce recommendations to other users [4]. There are several techniques used to generate recommendations, but the main categories are Content-based Filtering (CBF), Collaborative Filtering (CF), and Hybrid approaches [5].

RS are important in electronic commerce, especially for marketing [6] and they have been widely used in order to attract and retain customers. The relation between the loyalty of users and RS was studied in [7] using data from Amazon.com. Their findings showed that the presence of consumer reviews helps with retaining customers and also attracting new ones. In time, the business gains reputation, which usually translates to increase in business. There are a few challenges in optimally using the recommenders due to the variety of user's profile and its volatility and the reputation of different service providers. For dealing with these aspects, recommenders usually use product rating, confidence in service providers, and regularly update this information for an accurate suggestion for a given request.

In all the existing approaches, some improvable assumptions are considered for the purpose of easily

computing the reputation. Aspects like partial feedback, ignorance of customer confidence, and most importantly, lack of information on the service provider identity are major challenges for an accurate reputation per product, per service provider, per context, or per user profile. In this paper, we propose an approach taking into consideration the above challenges and deriving mechanisms for a more accurate reputation.

Classically, two notions concerning the quality of a delivered service are correlated for an accurate service evaluation, i.e., QoS (Quality of Service), and QoE (Quality of Experience). Specific to each service, there are particular service parameters that are agreed upon between a provider and a subscriber, commonly settled by the SLA (Service Level Agreement). On the provider side, the SLA parameters are used for technical audit and litigations (leading to penalties or bonuses towards a given user or class of users). Specific on-line and off-line measuring mechanisms for SLA metrics and specialized audit techniques have been proposed. On the consumer side, the subscribers' satisfaction is gathered and mapped to the audit results to validate a given service, to detect flaws in delivering a service, and to ultimately build a view on service reputation. In general, a record is handled per service or per products, with respect to a given subscriber or a class of subscribers. Customer feedback can be 'by request', or 'at will', and embraces various forms of on-line questionnaires. As a result, a customer might decide not to answer, or to answer exhibiting a particular behavior. Ultimately, a service provider might fake some feedbacks to increase its reputation. There are various factors that influence the computation of an accurate reputation, e.g., the volume of ordered services, the diversity of the subscriber classes, customer trust and loyalty, and the dynamics of the feedbacks. Practically, the main problem we try to find a solution for is to dynamically and accurately compute the reputation of a service/product, based on the system transactions. We propose a simple formula for reputation updates (1). The challenge is to identify the correct metrics in computing the updated reputation.

$$r_{real} = (1+ \lambda) \times r_{current} \quad (1)$$

where:
r_{real} represents the updated reputation, considering
$r_{current}$ represents the known and accepted reputation, and
λ represents the correction based on customer perception and feedback behavior, λ belongs to $\{(-\alpha, \alpha) \mid \alpha > 0\}$, usually having values in the vicinity of '0'.

The main achievement of our proposal is the following. The recommenders systems have a powerful mechanism to accurately indicate the real reputation, when selecting the best service provider from a service directory. Except some studies on QoE [17][19] that mainly consider technical metrics, we introduce and evaluate the customer behavior.

The paper focuses on dynamic aspects of customer feedbacks and formalizes mechanisms for a more accurate evaluation the reputation of a given service delivered by a given provider in establishing policies for λ.

The large spectrum of user behaviors (and, in general, the variety of needed services) leads to the need of similarity-based matching, when a given service is required. Traditionally, the notions QoS and QoE deals with these aspects. However, the perfect matching and the approximate-matching depend on a large number of factors. For example, if we consider Web Services dedicated to weather forecast, location, month/day/year, parameters (rain, wind, temperature, and pressure) can be appropriate parameters when inquiring. Definitively, there are several forecast services, and the experience of a particular user might differ from one forecast service to another. Some provide information that is more accurate than others (i.e., data is more frequently updated), history is better preserved by particular services, via backward search, e.g., Weather Underground, etc. A similar problem is observed when choosing and downloading a particular piece of software, when inquiring for a specialized on-line book shop, or for looking for a service providing the most updated world-wide information. Finally, some services offer a friendlier interface to search, order, and get delivered a particular need (i.e., personalized interface, myAccount, etc.).

There are meta-services, providing the service at choice. Such examples are those for buying flight tickets, where the cheapest, the quicker, or other selection criteria are used for service selection. Other meta-services are for selecting the most appropriate software to download, or to book a hotel. In most of the cases mentioned above, one criterion is usually considered to select from an existing service pool.

Two phases involve service features, (i) service discovery (locating) and (ii) service selection (in the case of a set of services, relatively satisfying the needs with similar degrees of satisfaction). Both phases require special mechanisms to assess service similarity. Meta-services have a restraint number of known services, well localized, the parameters of which are also in small number. Then, selection appears to be less complex. With a well known service and limited (usually one search/ selection criterion) similarity is relatively easy to be determined. The above considerations are not longer

valid for a large spectrum of properties a service might expose to satisfy a given service request. To satisfy a request, service similarity plays an important role for timely identifying and delivering, and for an optimal (maximal) customer (invoker) satisfaction. Customer satisfaction is expressed by QoE, on-line feedback, service ranking, and manifested by variations of QoS to keep service costs and satisfaction in synchrony.

The paper is structured as follows. Section II covers related works. An enhanced recommender model is presented in Section III. In Section IV presents a taxonomy of the dynamic feedback and a dual architecture. Section V introduces computation mechanisms for an accurate reputation of a service via dynamic reputation update policies and heuristics. A context-based similarity model, including distance and similarity metrics, a similarity taxonomy, and other facilities to consider service ranking and feature relaxations is introduced in Section VI. Section VII presents an algorithm to compute a minimum set of existing services satisfying a given query, following the newly introduced model. Section VIII concludes the paper and presents future investigations.

II. RELATED WORKS

As the proposed approach touches the recommendation and reputation on recommenders, service providers, and products, we first introduce some basic concepts.

2.1 Concepts

The core information of a recommender is a list of offers (products) and ratings of those products based on feedback received after a series of recommendations. The *rating* is subject to incomplete, fictitious feedback, volume of transactions for a given product or provider, and confidence in feedback. Based on the ratings, the recommender computes its own *ranking* per product.

s[r], P[r] represents a service or a provider with the rank r, where r is an integer.

Associated with the ranking is the notion of *reputation* that in fact determines the ranking. The reputation formula, while product oriented, it might not be accurate, as its computation cannot avoid some realities, such as some service providers have private relationships with recommenders (e.g., publicity, sponsorship) or indirect servicing (recommended product might not be produced by the front end provider, but simply delivered by it).

Reputation is an index associated with the service or a product based on user feedback that is taken into consideration when the ranking is calculated. The reputation index usually belongs to a set, *{outstanding, very good, good, acceptable, bad}***.** A recommender might increase the rank of a service when its reputation index, for example, passes from very good to outstanding [8]

Similarity is another concept used in generating recommendations. In order for a recommender to suggest products to a user, it needs to find a commonality among users (this applies in the collaborative approach) or among the products that were rated in the past by the user (this applies in contend-based approach). There are different techniques used to compute the similarity measure, but the most used are correlation-based and cosine-based techniques [5] [9]. Similarity is an index associated with two services or products. For example, $\mathbf{s_1}$ **[~/80%]** $\mathbf{s_2}$ means s_1 is similar with s_2 with an acceptance of 80% based on the service's features or in the same range of ranking.

2.2 Current approaches for recommenders

Recommenders are usually classified based on the approach for making the recommendations. There are three main categories of recommender types: content-based filtering, collaborative filtering, and hybrid filtering.

The *content-based filtering* recommends to users items that are similar to the ones searched by the users in the past [5][9]. This type of recommendation technique is mostly used to recommend text-based items such as documents and newspapers. In order to produce the recommendations, the system needs a profile of the user, which is represented by a set of terms. The profile can be obtained from the user through a questionnaire or it can be learned from their past transactions. This type of filtering has its shortcomings. Since it is content-based, it needs to have the representation of data in a matter that can be machine-parsable (e.g., article). It is harder to apply this technique in the case of movies, music, images, which are not machine-parsable.

The *collaborative filtering* (CBF) [3] tries to predict the relevance of an item based on the ratings done by other users. It accumulates ratings of products and whenever a request comes, the system identifies similar users and recommends the products rated by them. In this type of filtering, the user profile is defined by a vector of items and their ratings, which is updated over time. As opposed to the CBF, this type of filtering can

be applied to any kinds of items, not only to machine-parsable items. However, there are limitations with this approach, mostly caused by the lack of data points in initial stages: new user and new item.

The *hybrid algorithms* usually combine the content-based and the collaborative algorithms to overcome some of the limitations of the other two approaches. This approach has been adopted by some RS [10], [11]. There are different ways to combine the two algorithms and [5] present the different approaches in detail.

As we mentioned above, the reputation of a business is gained in time, mainly based on reviews from users. This brings up another point and that is obtaining accurate reviews from users. Many users are not willing to leave feedback after a transaction is completed. One reason for not leaving feedback is the lack of incentives. If there isn't some kind of payoff for the feedback, the user won't put the effort into posting one. An incentive mechanism is addressed in [12] where incentives are given to users who provide honest feedback through a side payment mechanism. Examples of incentives mechanisms are Amazon's "Top Reviewers" practice and Epinions.com referral fees practice [13]. Another reason for not leaving feedback is to purposely withhold information about a product that gives its user an advantage [14].

Another concern related to the validity of the reviews is the manipulation of the reviews by parties with direct vested interests. Businesses can review their own products in order to boost the sales. Also, the competition can leave or fabricate negative feedbacks to undermine the competitor's reputation. There are ways to filter out biased feedbacks and to prevent manipulation [15], but preventing coordinated collusion attacks is still an issue. eBay, for example, does not have a problem with feedback manipulation. The feedbacks can only be left by users who are registered with them and who made a purchase on eBay. However, if a group of users agree with a seller to leave positive feedback for fictitious auctions (e.g., the seller can post multiple 1 cent auctions on which the users can bid), the seller's ratings can be positively affected. These users are usually called shills. This approach would require quite an effort (the larger the number of shills, the bigger the impact), but it can be achieved.

Reputation is very useful in RS and eBay is one example of a reputation system that proves that their approach works well. However, having a centralized reputation system, such as eBay, can bring other issues, such as vulnerability and inflexibility of the system [14]. In [14], the authors propose a distributed trust and reputation management framework. The users choose a trust broker and after each transaction with a service, the user sends its rating to its trust broker. This way, the trust broker builds a reputation about a service based on the user's feedback. The brokers exchange reputation information among themselves in order to collect more information about the available services. This framework relies on the user's feedback only, ignoring the business model of the provider.

Recommender mechanisms [18] rank the products or services based on feedback received after a series of recommendations and successful transactions. The *rating* is subject to incomplete, fictitious feedback, volume of transactions for a given product or provider, and confidence in feedback. Based on the ratings, the recommender computes its own *ranking* per product, defining the reputation (r) of a service/product. A computational mechanism including user' confidence (c) and feedback expectation (e) was proposed in [16]. An attempt, rather static, of considering a static subjective evaluation of the quality of the voice service is described by the MOS (Mean Opinion Score). The MOS is an arithmetic value ranging between 1 and 5, expressing individual perception [17]. However, MOS apply strictly to voice-related services, on an individual basis. The metrics are purely technical and related to codec use, packet loss, packet reorder, packet errors, and jitter. Another standard for evaluating the speech QoE, also considering technical metrics, is captured by PESQ algorithms [19].

A dynamic approach for customer input is presented by byClick system [20][21], where there is an on-line click-counting on the number of service accesses. No customer behavior, subscription status, or feedback patterns are considered. However, in the current approaches, no correlation with the frequency of users' report and transaction peaks, as well as with the users' report patterns were considered.

Finding similar services (approximate, but satisfactory matching) is somehow similar to (i) text matching, (ii) schema matching, or (iii) software-component marching. For some text matching solutions (information retrieval) mechanisms based on term frequency are used [28][29]. In schema matching, special techniques are using semantics of the schemas to suggest schema matching [30]. Mainly, linguistic and structural analyses, as well as domain knowledge, are methods to handle schema matching. When expanding to software component matching [31] (considerably used in software reuse) component signature and program behavior (usually formally defined) are considered; in this case, data types and post-conditions should be considered for matching. However, these techniques are not suitable for Web

Services [27], as data types and post-conditions are not available. Usually, such a service has a name and text description in UDDI **(**Universal Description, Discovery, and Integration) registry, operation decryptions, and input/output descriptions; the last two are usually specified in WSDL (Web Service Description Language).

Dong *et al.* [27] proposed criteria for associating similar terms. They introduced the cohesion/correlation score, as a measure of how thing two terms are. However, they do not consider particular characteristics of a term. They applied the score only to Web Services. We start from the point that services similarity has a meaning only between services than can be context-oriented and belong to a cluster (e.g., invoking a service gives a list of similar operations with similar results). Other approaches consider both diversity and similar at the same time having the distance as a metric [32]. We adopt these metric (see Section III) and adapt them to the service similarity computation.

In fact, specific to each service, there are particular service parameters that are agreed upon between a provider and a subscriber, commonly settled by the SLA (Service Level Agreement). On the provider side, the SLA parameters are used for technical audit and litigations (leading to penalties or bonuses towards a given user or class of users). Specific on-line and off-line measuring mechanisms for SLA metrics and specialized audit techniques have been proposed. On the consumer side, the subscribers' satisfaction is gathered and mapped to the audit results to validate a given service, to detect flaws in delivering a service, and to ultimately build a view on service reputation. In general, a record is handled per service or per products, with respect to a given subscriber or a class of subscribers. Feedback can be used to enforce service similarity.

In this paper, we also expand the cluster-based similarity to service similarity and introduce similarity taxonomy, where the service consumer has a weight in deciding service similarity. The idea is to establish service ranking (and reputation) inside a given cluster, and define similarity considering service-provider and service-user feedback.

III. AN ENHANCED RECCOMENDER MODEL

In this section, we present a recommender model that can handle the sub-contract mechanism, yet keeping an accurate information on a given provider reputation (leading to an accurate ranking).

3.1 Setting the case

A simple scenario is presented in Figure 1, where the user is interested in service s_1 from P_1. The user asks the Recommender for the best provider for service s1 within specific parameters. The Recommender replies with either a provider that has the best reputation for service s_1 or with a list of providers $\{P_i\}$ for s_1. Let us assume P_1 is registered of being capable to deliver s_1 (others might be registered for s_1 as well). The Recommender cannot know if P_1 has the service or if it contracts it from a different provider. If P_1 is contracting s_1 from P_2, the transaction between P_1 and P_2 is transparent to both the Recommender and the user. At the end of the transaction, the user sends the rating of P_1 to the Recommender and P_1 receives all the credit for the transaction. This leads to an inaccurate reputation and altered ranking.

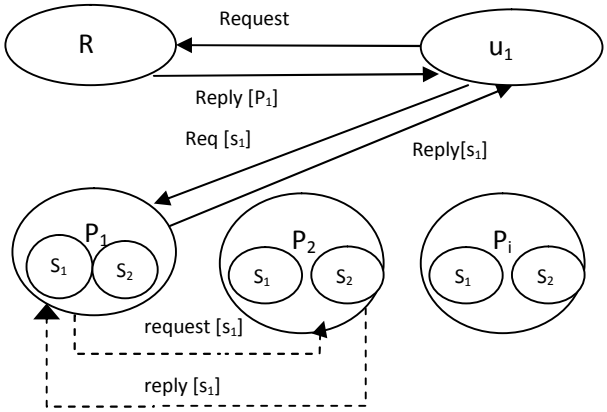

Figure 1. Indirect reputation

If the reputation of the provider is based only on the user's feedback, there is no way to assess the ultimate role of each provider. In order to have a more accurate picture of the providers' involvement, we propose that feedback from the providers be taken into account when establishing reputation. This includes both the front end provider (in our case P_1), as well as any subcontracted providers (in our case P_2). All feedback goes directly to the Recommender.

The ideal scenario would be when all the users and providers report 100% of the transactions. In reality, users don't always leave feedback and providers do not always report rendered services. In such a case, the Recommender is left to deal with an incomplete set of data. Moreover, some of the reported data may be fabricated by both users and providers.

3.2 Recommender representation model

Apart from the mechanism of collecting the feedback and interfacing with the users, the core information present in a recommender is stored in a service database. This allows a request to be replied to with a service or a list of services, eventually with a degree of similarity associated with each service. Usually, the recommender keeps information on relative ranking among these entities.

We propose an enhanced model, which takes into account the user's profile and behavior, and a list of potential providers for a given service. This allows a more refined ranking scheme where providers can be rated per service.

While ranking is based on user feedback, there is no appropriate mechanism to consider the user's expectation (e) and credibility (c). By user expectation we mean the probability of having the user leave feedback after a service was delivered. The credibility refers to the user's ability to give a trusted rating. Usually both, expectation and credibility are expressed as percentage.

In Figure 2, we present the enhanced recommender model. The recommender stores information about the available services, the providers and their services, plus the user profile, which includes its expectancy and credibility. Both services and providers are associated with a rating. The providers' rating is done within the context of a service. This way, the rating can be done per product and per provider for a specific product.

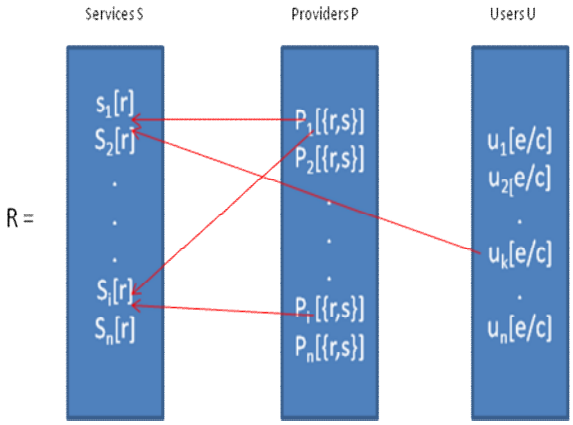

Figure 2. Enhanced Recommender Model

By keeping the relationships between the providers, their services, and also the users who requested the available services, the recommender can provide better suggestions and answer to more complex queries.

We classify queries in two categories, i.e., U-R and P-R. Some salient queries U-R might be:

Query 1:

input: $[s_1]$
output: $[s_1/P_1, s_1/P_2]$

The user asks for service s_1 and the recommender replies with a list of providers that offer s_1.

Query 2:

input: $[s_1]$ & $[s_1 (\sim/\mathcal{E})]$
output: $[s_1/P_1, s_1/P_2]$ & $[s_i/P_i]$

The user asks for service s_1 and/or a service similar to s_1. The recommender replies with a list of providers that offer s_1 and/or a list of providers who offer services similar with s_1.

"$\sim/\mathcal{E}$" represents the similarity of services with $\mathcal{E}$ as proximity

Query 3:

input: $[s]$ $[P_1, P_2]$
output: $[s_1/P_1, s_2/P_1]$ $[s_i/P_2, s_j/P_2]$

The user asks for a list of services offered by certain providers. The recommender replies with a list of services offered by those providers.

Query 4:

input: $[s \mid r > x]$
output: $[s_1/r_1, s_2/r_2]$

The user asks the recommender for a list of services, which has a ranking "r" higher than a certain value. The recommender replies with the list of services.

Some relevant queries P-R might be the following:

Query 5:

input: [u_i]
output: [u_i [e/c]]

The provider asks the recommender about user u_i. This may be relevant to the provider in order to assess the user's credibility. The recommender replies with the u_i expectation "e" and credibility "c".

Query 6:

input: [all U_i, e > α, c >β]
output: [u_i [e/c]]

The provider asks the recommender for a list of users whose expectation and credibility are higher than a certain value. This may be relevant to the provider in order to assess the user's credibility. The recommender replies with the list of user(s).

Based on the formula presented in the following section, complex information can be gathered and more accurate answers to different queries can be provided.

3.3 Computation mechanism

The enhanced model allows a more comprehensive schema for computing the reputation.

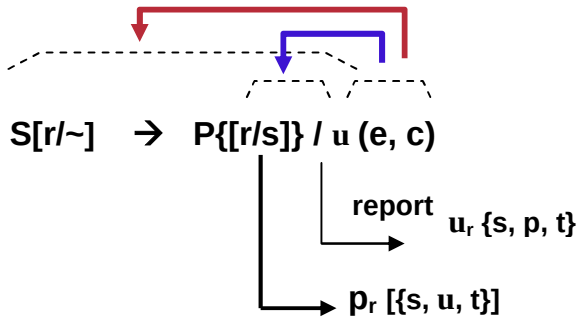

Figure 3. A computation schema for recommenders

In our framework, a recommender has mechanisms for representing services (S) with their reputation (r) and similarities (~), provider (P), with their reputation (r) linked to the reputation of their service providers (s), associated with user's (u) expectation (e) and credibility (c). A particular relation is valid at a moment (t). For example, a user x is expected to provide feedback with e = 80% and the confidence on its feedback is 70%. The feedback is on a provider (p) providing a service (s) at the time (t). The schema allows having a reputation view of a user at a given time, on a given provider delivering a given service. The schema also allows having a reputation of a provider, as perceived by a user at a given time, if delivered by a given service.

We are now going to concentrate on different scenarios dictated by the amount of data reported by users and service providers.

For example, a user sends a request to the recommender for the best cell phone provider that would meet certain parameters. The recommender replies with provider P_1. The user makes a request for a number of cell phones from P_1. After the transaction is completed, all the involved parties have the option to send feedback to the recommender. The recommender collects the data and based on the feedback, it updates the reputation of the involved parties. The nature of the collected data can be divided in three main cases:

3.3.1 Matching reports

The number of feedback reports from the user matches the number of reports from the service provider within a particular time window relevant to the service type. To continue with the example from above, the user sends the feedback to the Recommender, including the number of cell phones that it purchased. P_1 reports to the Recommender that the user purchased a number of cell phones from it. The numbers reported by both the user and P_1 match.

A subclass of this scenario would be when P_1 sub-contracts from a different provider, P_2. If P_1 receives a request for cell phones, it can send the products from its own stock, send part from its own stock and part from P_2, or get the entire order from P_2. In this case, the Recommender would receive reports from both providers, P_1 and P_2. The exact number reported would not match since P_1 will report that it sent the entire order to the user, and P_2 would report that it sent a certain number of phones to P_1, but the data can be correlated. The correlation is done by using the transaction completion time, the user identifier, and the provider identifier.

3.3.2 Over-reporting providers

The number of feedback reports from user and provider does not match. This can be caused by either providers exaggerating the amount of transactions completed, or by users who underreport. In this case,

some of the data can be correlated by the Recommender.

3.3.3 Underreporting provider

The number of feedback reports from user and provider does not match. This can be caused by either providers that do not report every transaction, or by users who exaggerate the amount of transactions completed. In this case, the Recommender can correlate some of the date.

3.3.4 Case study for reputation correction

Let us consider the following situation:

u -> [t] [p1] [s1], where u is the user, p1 and p2 are providers, t is the time of the request, and s1 is the service;
p1-> [t] [u] [s1], with p1 [r1/s1]
p2 → [t][s1], with p2 [r2/s1]

and the following transaction reports:

|u|: reports α transactions
|p1| reports β transactions (with $\beta < \alpha$)
|p2| reports γ transactions

then

$k = (\beta - \gamma) / \alpha$

In this case, for a given user u, and for the considered service s1, the real reputation is $r_1' = k \times r_1$, as there is an indirect service delivery form p2 via p1 to the user u. The schema allows having a more accurate view on who is delivering a service. Note that the number of transactions can be either reported or obtained by audit. In this use case, we consider that the providers are subscribed to an automated transaction report when delivering a service.

3.3.5 Discussion

In this section, we are comparing existing recommender systems with our proposal, on the basis of three main features: expectation, credibility, and user profile, as defined in Section 3.2.

We consider a few well known recommender systems and only selected those three main features as a basis of comparison. The existing recommenders do not incorporate in the user profile the expectation and credibility of a given user.

Table 1. Feature based comparison of several recommender systems as well as the proposed one

	eBay	Amazon.com	Barnes & Nobles	proposal
expectation	Not in profile	Not in profile	Not in profile	Included in profile
credibility	Not in profile	Not in profile	Not in profile	Included in profile
User profile	yes	yes	yes	yes

While the considered systems (eBay, Amazon.com, Barnes & Nobles) make use of the notion of profile when recommending a product, the main target is to identify potential similar services and products to either satisfy a request or recommend a particular service unknown to the user (using the similarity concept).

By including these features, the recommender can have a more complete view on user's satisfaction based on more accurate information maintained by the system on the user's behavior (the degree of responsiveness of the user ability to give trusted rating).

The performance and accuracy of a recommender system can be enhanced by including in the user's profile the user's expectancy and credibility. By having the expectancy of a user to leave a review and also its credibility, a recommender can better tune its suggestions to a user's requests with increased certainty. Ongoing experiments will identify the thresholds from where these features increase the accuracy of recommendations. Particular consideration will be given to the dynamics of user's feedback in terms of relationships between the frequency (volume) of the used services or products and the accuracy of the timely feedback.

IV. DYNAMIC FEEDBACK

The mapping QoS~QoE principally involves SLA' metrics. In our approach, we also introduce temporal and ethical metrics to quantify more accurately the customer feedback. Additionally, long term and short terms feedback patterns are identified, including spikes feedback.

We consider the basic introduced in [16], where by user expectation we mean the probability of having the user leave feedback after a service was delivered. The credibility refers to the user's ability to give a trusted rating. Usually both, expectation and credibility are expressed as percentage. The new mechanism proposed includes the status of the user and the feedback history.

Finally, we drive feedback-based polices for service reputation updates, by considering these metrics, based on extreme behaviors of customers in terms of feedback, e. g., feedback too late, too quick, too frequent, too rare, etc.

A. Dual reputation update architecture

Two views on reputation must be correlated for a given service/product, i.e., the provider view and the customer view. On the provider view, the perception of the service reputation, $r_{expected}$, represents the variation of several metrics, as the volume of sales, the number of new customers or lost customers in a given period. From the customer side, $r_{feedback}$ gathers customer perception on the reputation of a service.

Therefore, we propose a dual architecture to correlate and synchronize the two views. From the uniformity reasons, the update heuristics will follow a similar computation approach for both views, implemented by specialized engines.

Figure 4 depicts the main architecture and decisional and computation engines.

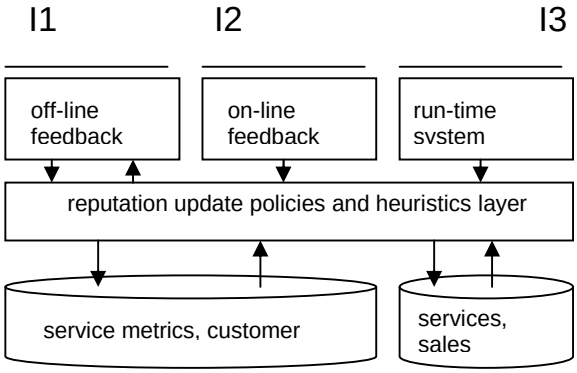

Figure 4. Dual reputation update architecture

The architecture considers two customer-facing interfaces (I1 and I2) handling the off-line (by_request, or at_will), and on-line (at_will), respectively, reputation updates. I3 is considering the provider-facing reputation updates. Three appropriate *off-line*, *on-line*, and *run-time* engines deal with the updates, by receiving and computing them via I1, I2, and I3, respectively. In the current paper, we only focus on aspects related to data collected via I1 and I3. For data collected via I3, an interesting implementation, nor related to our model, is presented by byClick [20][21]. While dynamic is different, the dual architecture can also support this approach. Hereafter, we will only refer to the off-line and run-time engines.

The *reputation update polices and heuristics layer* implements mechanisms to synchronize the two views, and to correlate the newly computed values. For example, a specific function is to trigger an update of $r_{feedback}$, when the, $r_{expected}$, has an unexpected variation, or when its variation trespass some thresholds. A concrete case could be when the volume of sales increases dramatically, with no appropriate variation of $r_{feedback}$. Appropriate heuristics will be presented in the next sections.

It is assumed that the customer and service records follow the model presented in [16] and enhanced in this proposal.

Without losing the generality, but for simplicity of the computation, we adopt the same formula, i.e., (1), as a core mechanism for reputation update engines; only the metrics will be specific to each view. Let us assume that a given service has a starting reputation r_0, on both views. We use the formula (2) for computing an updating value of the reputation:

$$r_{current} = r_0 \prod (1 + \lambda_i) \quad (2)$$

where: $\{ \lambda_i = k_i \times w_i \;\; i = 1.....M\}$

M: the number of considered metrics
i: a given i-th metric [I belongs to N]
k_i: basic normalized update due to the variation of the i-th metric [k_i belongs to R]
w_i: weight factor associated with the i i-th metric [w_i belongs to R]

The *off-line* and *run-time* engines compute the $r_{feedback}$ and $r_{expected}$, respectively, using (2) and appropriate heuristics for adopting λ_i. In the next section, we introduce a customer dynamic model and show how the λ_i are computed.

4.1 History metrics

We recall the main concepts of the enhanced recommender model [16], which we consider as the basis for dynamic feedback metrics:

s <r>: each service <s> has an associated reputation <r>
P_i<s,r_i>: each provider offers a service with its associated reputation
P_j<s,r_j>: another provider can offer the same service with a different associated reputation
u <e, c>: a user has a credibility and confidence metrics associated with

Note: for simplicity, we consider that <e, c> are the same for any service.

In evaluating the customer feedback, we consider individual metrics and metrics for a class of subscribers. In both cases, the feedback mode, i.e., 'by_request' or 'at_will' helps to differentiate between different extreme feedbacks.

feedbackMode::= {by_request, at_will}

4.2 Individual metrics

For individual metrics, 'subscription seniority', 'feedback timing', and the 'satisfaction' are relevant.

Seniority in profile::= {long term, regular, new}
FeedbackTiming::= {quick, regular, late }
SatisfactionDegree::= { x% | x = 0 - 100}

For policy-specification, we define satisfaction by metrics

satisfaction = satisfied, when $x > \beta_1$
= regular, when $\beta_2 \leq x \leq \beta_1$
= dissatisfied, when $x < \beta_2$

seniority = long term, when term $> \tau_1$
= regular, when $\tau_2 \leq$ term $\leq \tau_1$
= new, when term $< \tau_2$

feedback = quick, when $t < t_2$
= regular, when $t_2 \leq t \leq t_1$
= late, when $t > t_1$

With the above definitions, we assume that the architecture handles the seniority of the subscribers and the timestamps of their feedbacks after a service was consumed.

The following patterns of interest can be identified for each seniority profile:

a. &&<quick><satisfied>]
b. &&<quick><dissatisfied>]
c. &&<late><satisfied>]
d. &&<late><dissatisfied>]

The profile metrics quantified as 'regular' do not alter the computation of the reputation.

With the new metrics, a user is characterized by
- expectation
- credibility
- seniority

and a 'per service' feedback pattern. The feedback patterns comprise:
- feedback timing
- satisfaction degree
- feedback dynamics (#satisfied, #dissatisfied, repetitive replies, observation period)

There is a calibration phase for each system, where the appropriate values are tried and settled for the thresholds. For example, the following steps are considered by a calibration procedure for the feedback timing:

(1) An 'average' reply time is observed and recorded for both feedback modes for a given service.
(2) After the calibration period, the customer reaction is observed for that service, called 'average'.
(3) A policy can be defined by heuristics, as follows:

```
START
IF feedback mode = at_will
   IF 'feedback timing' is 'three times' than the
     'average'
        THEN feedback = late
IFNOT (feedback mode = at_request)
    IF 'feedback timing' is 'twice' than 'average'
        THEN feedback = late
END
```

Heuristic #1. Settling the feedback values
In a similar way, and based on calibration, policies for settling each threshold can be defined.

4.3 Metrics for classes of subscribers

For a class of subscribers to the same service, we propose feedback metrics capturing the community behavioral. In the case of a community, the individual profiles are aggregated. In order to capture the dynamicity of the feedback, we introduce a few feedback metrics describing a pattern structure. In a given observation period (Δ), we define the number of repetitive replies (m) and the number of satisfactions (ni+) an dissatisfactions (ni-), as well as n- = max {ni- | i = 1…} and n+ = min {ni+ \ i = 1….}. For example, in Figure 2, on the top, m = 2, n1+ = 4, n2+ = 5, and n+ = 4. In the third basic pattern, when both satisfactions and dissatisfactions are present, a pair (n+, n-) is attached to it.

Based on a series of observations periods, a profiling system is able to classify customers and have a coarse granularity on the feedbacks.

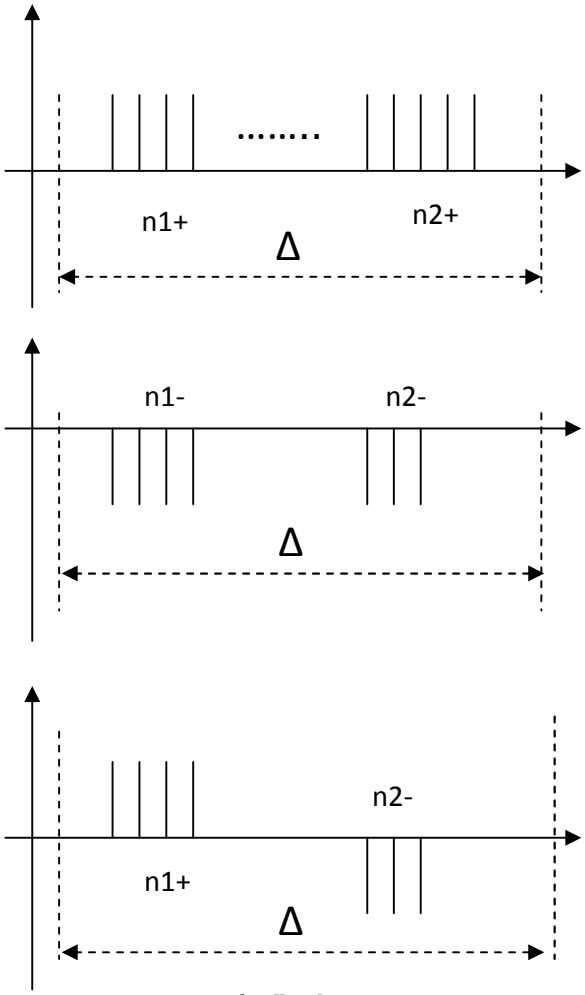

Figure 5. Basic feedback patterns

With these metrics, a reputation engine can trigger appropriate reputation update mechanisms. Definitively, they can be combined with the user history metrics to build more complex (and more accurate) updates.

4.4 Dynamics on service subscriptions

Service reputation is a main metric to justify the existence of that service, new investment in developing new features of the service, and new marketing activities to promote a given service.

In our model, we consider a few metrics that portray the dynamics of service sale, e.g., the volume of transactions, the number of new customers in a given period, and the number of lost customers in that period. For the last metric, we also consider the seniority, therefore distinguishing between long term and new customers.

Therefore, apart its features from the quality point of view, from the reputation perspective, a service is described by:

- r: reputation
- vol: variation in transaction volume [± %]
- new: new customers [%]
- lost_new: lost new customers [%]
- lost_long: lost long term customers [%]

(% is considered versus the numbers at the beginning of the observation period; usually every update represents the start of a new period)

A revision of the reputation of a service is always based on the above metrics.

An example of using heuristics for updating the reputation based on the number of transactions can be:

START
IF vol = - 10%
    THEN $r_{real} = (1- 0.1) \times r_{current}$
ELSE
    $r_{real} = r_{current}$
END

Heuristic #2. Updating the reputation versus variations of transactions

4.5 Conclusion on the reputation update model

We presented a model for updating the reputation of a service considering the user profile, its dynamic feedback, on the one side, and the dynamics of service subscriptions. In general, the reputation updates is triggered by a significant variation of one of the service subscription dynamics. This moment defines the origin of an updating period. It is assumed that a recommender system records other customer feedback information that is considered when updating the reputation.

In the following section, we present some basic heuristics to update service reputation, considering the metrics described above.

V. DYNAMIC REPUTATION UPDATING

There are several classes of updates, based on what metrics are used.

5.1 Satisfaction and feedback based policies

The simplest way of updating the reputation is considering the first four patterns

a. [<quick><satisfied>]
b. [<quick><dissatisfied>]
c. [<late><satisfied>]
d. [<late><dissatisfied>]

Following Policy #1, we associate with (a) and (b) a correction θ_1 and with (c) and (d) a correction θ_2, with $\theta_2 < \theta_1$, when the feedback_mode is 'at_will' and with $\theta_4 < \theta_3$, respectively, when the feedback_mode is 'by_request', with $\theta_4 < \theta_3 < \theta_2 < \theta_1$. The subjective justification of these values is given by the customer attitude in terms of promptness of reactions and their qualification.

The correction, in this case, is expressed by the following policy [Policy#1]

START
IF feedback_mode = *at_will*
  IF feedback = *quick*
    IF satisfaction = *satisfied*
      THEN $r_{real} = (1+ \theta_1) \times r_{current}$
    IFNOT (satisfaction = *dissatisfied*)
      THEN $r_{real} = (1- \theta_1) \times r_{current}$
  IFNOT (feedback = *late*)
    IF satisfaction = *satisfied*
      THEN $r_{real} = (1+ \theta_2) \times r_{current}$
    IFNOT (satisfaction = *dissatisfied*)
      THEN $r_{real} = (1- \theta_2) \times r_{current}$
IFNOT (feedback_mode = *at_request*)
  IF feedback = *quick*
    IF satisfaction = *satisfied*
      THEN $r_{real} = (1+ \theta_3) \times r_{current}$
    IFNOT (satisfaction = *dissatisfied*)
      THEN $r_{real} = (1- \theta_3) \times r_{current}$
  IFNOT (feedback = *late*)
    IF satisfaction = *satisfied*
      THEN $r_{real} = (1+ \theta_4) \times r_{current}$
    IFNOT (satisfaction = *dissatisfied*)
      THEN $r_{real} = (1- \theta_4) \times r_{current}$
END

Policy #1: Reputation updates #1

The values of all weights are done by validated calibration. For example, if the orders of a given service do not increase (or decrease), it means that the reputation is too high. Therefore, reputation is always 'in question', when the transactions for a service vary, or the service orders show a quick increase or decrease (in terms of volume, and in terms of new customers).

5.5 Satisfaction, feedback, and seniority based policies

Let us assume that a mechanism is in place for complying with Policy #1; additionally, the seniority must be considered. There are two strategies used in our model to update the reputation: (i) an optimistic one, and a (ii) pessimistic one.

Let us assume we have the following situation:

vol = +15%
new = 10%
lost_new = 2%
lost_long = 1%

In an optimistic strategy, would credit the 'vol' and 'new', while downplaying the 'lost_new' and 'lost_long'. Following the same approach of correcting by fraction representing the percentage, i.e., 10% is a correction of 0,1, we have the following heuristic:

START
IF vol = +15%
  new = 10%
  lost_new = 2%
  lost_long = 1%
  THEN $r_{real} = (1+ 0{,}15)\ (1 + 0{,}1)\ (1 - 0{,}02)\ (1- 0{,}01) \times r_{Policy\#1}$
END

Heuristic #3. Considering variations in transactions and subscribers (optimistic)

In a pessimistic approach, losing new subscribers or long term subscribers is an indication of service degradation from the quality point of view, of a violation of the SLA with a significant number of subscribers, or simply that the service was not upgraded at the expected standard.

In this case, a multiplicity factor can be used to consider the loss, e.g., k1 for loosing new subscribers and k2 for loosing long term subscribers.

START
IF vol = +15%
  new = 10%
  lost_new = 2%
  lost_long = 1%
  THEN $r_{real} = (1+ 0{,}15)\ (1 + 0{,}1)\ (1 - k_1 \times 0{,}02)\ (1- k_2 \times 0{,}01) \times r_{Policy\#1}$
END

Heuristic #4: Considering variations in transactions and subscribers (pessimistic)

Calibrating the values for k1 and k2 in this case follows also a given heuristic, as expressed below:

	1T	2T	3T
k1	3	2	1
k2	6	6	6

Heuristic #5: Multiplicity correction factors

In Heuristic #5, an example of selecting the multiplicity correction factors is presented. Assume that the unsubscribe event occurs in 1T, 2T or 3T time units for the enrolment, the example gives more weight to the loss of long term customers ($3T < \tau_2$ to correctly evaluate the 'new').

Note1: In the model presented above we considered no difference between the types of service, assuming that the QoS, from the provider perspective was delivered according to the SLA.

Note2: In the heuristics and the metrics presented above, we didn't consider any emotional feedback that might influence the feedback (such as accompanying gifts, bonuses, or penalties for QoS violations), nor particular interests of a customer in a service provider, such as stocks.

5.2 Reputation update considering feedback patterns

A fine grain reputation update considers the feedback patterns presented in Figure 5. While only one pattern can be considered to update the reputation, Heuristic #6 considers all three patterns (see Figure 5).

START
IF (Δ, m, n^+) $[\Delta > \tau_1]$
THEN $r_{real} = (1+ m^+xn^+/100) \times r_{current}$
IF (Δ, m, n^-) $[\Delta > \tau_1]$
THEN $r_{real} = (1+ m^-xn^-/100) \times r_{current}$
IF $((\Delta, m, n^+, n^-)$ $\&\&\Delta > \tau_1]$
THEN $r_{real} = (1+ m^-xn^-/100) \times (1+ m^+xn^+/100) \times r_{current}$
END

Heuristic #6. Reputation updated considering the feedback patterns

5.3 Reputation updating policies and heuristics

By their own nature, from both views (customer, provider), the reputation values on each view is a list, similar with time series, with

$r_{feedback} = \{r1, r2, r3, \ldots\ldots\}$, and (3)

$r_{expected} = \{r'1, r'2, r'3, \ldots\ldots\}$

at {t1, t2, t3, …..}

The reputation values can have the following position, as shown in Figure 6.

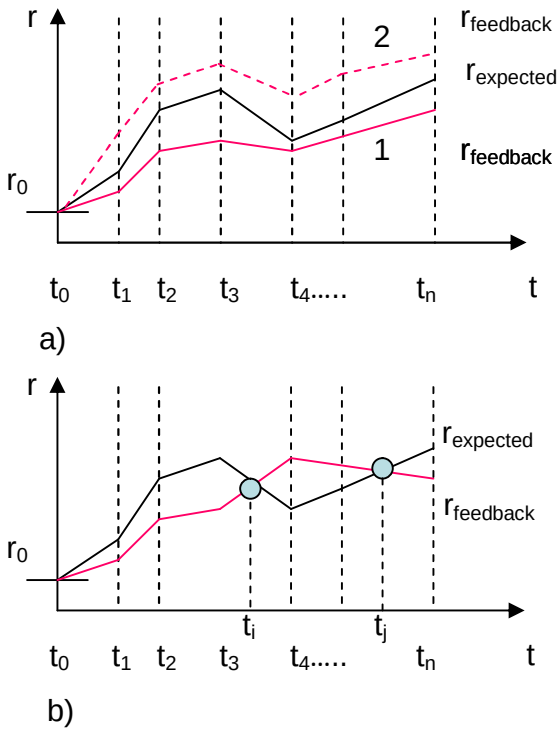

Figure 6. Relative position of reputation values

While computing the real reputation on both views, and considering the dynamics of customer feedback, we can observe a *channel trend* (Figure 6a) or local anomalies (Figure 6b). Formally, there are several cases for defining updating heuristics.

A chann*el trend* is considered when $| r_{expected} - r_{feedback}| < \varepsilon$ and an *anomaly* occurs when $| r_{expected} - r_{feedback}| > \delta$, where $\varepsilon << \delta$, for all t_i (ε and δ are thresholds that are established per services).

The model allows implementing five heuristics for synchronizing the feedback and expected reputation (see (3)). This allow to adjust the market from some actions; with no lost generality, we only consider three potential actions, i.e., 'increase/decrease the storage order', 'increase/decrease the price', and 'increase/decrease the offered QoS'. These actions are main contributors for the service costs.

1) Synchronization

In this case, both expected and feedback reputations are in synchronization with small variations.

IF
$| r_{expected} - r_{feedback}| < \varepsilon$
AND
$r_i > r'_i$ for all i
OR
$r_i < r'_i$ for all i
THEN "keep same storage order"
AND
"keep same prices"
AND
"keep same QoS/SLA agreements)
END

Heuristic #7. Regular computation

No particular actions are triggered, but regular reputation computation by off-line and run-time engines.

2) Pessimistic anomaly (from the provider)

In this case, the feedback reputation is much higher than the expected reputation; it is a situation to increase the objective function (benefits).

IF
$| r_{expected} - r_{feedback}| > \delta$
AND
$r_i > r'_i$ for all i
THEN
IF 'no QoS violation"
THEN "increase storage order"
ELSE
"offer lower QoS/SLA provider metrics"
(less costs)
END

Heuristic #8. Pessimistic policy

3) Optimistic anomaly (from the provider)

In this case, the expected reputation is much higher than the feedback reputation; it is a situation to decrease the objective function (benefits).

IF
$| r_{expected} - r_{feedback}| > \delta$
AND
$r_i < r'_i$ for all i
THEN
IF 'no QoS violation"
THEN "reduce storage order"
OR "reduce price"
ELSE
"offer better QoS/SLA provider metrics"
(more costs, to attract customers)
END

Heuristics #9. Optimistic policy

4) Under estimation (by the provider)

This case refers to the situation where the expected reputation decreases, but the feedback reputation increases.

IF
$| r_{expected} - r_{feedback}| < \varepsilon$
AND $r_i = r'_i$ for a given I &&see t_i in Fig. 3b]
AND $r_{i-1} < r'_{i-1}$
AND $r_{i+1} > r'_{i+1}$
THEN
(expectation decreases, customer satisfaction increases)
IF 'no QoS violation"
THEN "increase storage order"
OR "increase price"
ELSE
"offer lower QoS/SLA provider metrics"
(less costs))
END

Heuristics #10. Under estimation policy

5) Over estimation (by the provider)

This case refers to the situation where the expected reputation increases, but the feedback reputation decreases.

IF
$| r_{expected} - r_{feedback}| < \varepsilon$
AND $r_j = r'_j$ for a given I [see t_j in Fig. 6b]
AND $r_{j-1} > r'_{j-1}$

AND $r_{j+1} < r'_{j+1}$
THEN
(expectation increases, customer satisfaction decreases)
IF 'no QoS violation"
THEN "decrease storage order"
OR "decrease price"
ELSE
"offer better QoS/SLA provider metrics"
(more costs)
END

Heuristics #11. Over estimation policy

VI. A CONTEXT-BASED SIMILARITY MODEL

Then main idea of our approach is (i) having well defined service clusters, (ii) compute the distance between service feature, (iii) evaluate service similarity, based on service features, (iv) consider user-, service,- and producer-based similarity reflected by the appropriate reputations, and (v) evaluate how interchangeable two services are. When a service query is issued the algorithm we propose select the most appropriate service, considering both distance and similarity between services.

5.4 Identifying clusters of similar services

Expanding what was mentioned in [27], service cohesion of a service cluster must be strong (best potential to be similar), while correlation between two service clusters should be weak (service independence). We say that service s_1 is similar with s_2, and note $s_1 \sim s_2$, if the similarity confidence is greater than a given threshold δ. In a cluster S with $\|S\|$, where $\|x\|$ is the cardinality of x, we redefine cohesion and correlation as follows:

$$Cohe_S = \{(s_i, s_j) \mid s_i \sim s_j (\sim_{thres} > \delta)\} / (\|S\| \times (\|S\| - 1)) \quad (4)$$

and

$$Correl_{S,S'} = (A (S, S') + A (S', S)) / 2 \times \|S\| \times \|S'\|, \quad (5)$$

where

$$A (S, S') = \| \{s_i, s_j \mid s_i \in S, s_i \in S' \; si \sim sj \mid \sim_{thres} > \delta\}\| \quad (6)$$

with $$\sim_{score} = Cohe_S / Correl_{SS'} \quad (7)$$

defining the similarity score.

We notice that $\sim_{score}$ defines similarity classes based on the preexisting service clusters. To enhance the similarity score, clusters aggregation and clusters split operations are possible. Conditions and assessments for doing these are presented in [27].

6.2 Distance metrics for service similarity

Let us assume that a service *s* has *n* features (usually called data-points, as they are expressed by concrete values in an *n* dimensional space). The following distance methods are adapted for comparing services:

(a) Service Euclidian distance between two services in the *n* dimensional space

$$d(s_1,s_2) = 1/n \Sigma (a_{1i} - b_{2i})^{**2}, \text{ for all } i = 1 \ldots n \quad (8)$$

where a_i, b_i are service features.

(b) Service city-block distance

$$d(s_1,s_2) = 1/n \Sigma |a_{1i} - b_{2i}|, \text{ for all } i = 1 \ldots n \quad (9)$$

(c) Service Pearson correlation coefficient

$$r(s_1,s_2) = 1/n \Sigma ((a_{1i} - \underline{a})/\sigma_a) \times ((b_{2i} - \underline{b})/\sigma_b), \quad (10)$$

where $\underline{a}$ and $\underline{b}$ are the sample mean of a_i and b_i respectively, and σ_a and σ_b are the sample standard deviation of a_i and b_i.

The service Pearson distance is defined as

$$d(s_1,s_2) = 1 - r(s_1,s_2) \quad (11)$$

(d) Service Cosine similarity

$$d(s_1,s_2) = \cos(\theta) = (s_1 \bullet s_2) / (|s_1| |s_2|) \quad (12)$$

where $\bullet$ is the vector product of s_1 ans s_2.

By selecting a service distance metric, a clustering algorithm computes the distance matrix between two services. Mostly, (a) and (b) of the above are satisfying the triangle inequality, as true metrics.

6.3 Classes of similarities

In order to select the most appropriate service, we introduce producer-based similarity ($\sim_{prod}$), recommender-based similarity ($\sim_{recc}$), and user-based similarity ($\sim_{user}$). Producer similarity is based on the expectation, recommender's similarity is statistics-based, and user similarity is based on user feedback. In this taxonomy, $s_1 \sim_{prod} \{s_2, s_3, \ldots\}$ define a cluster of similar services, as defined by the producer.

To refine service similarity, we introduce the notions of *primary service features* and *secondary service features*, as shown in Figure 7,

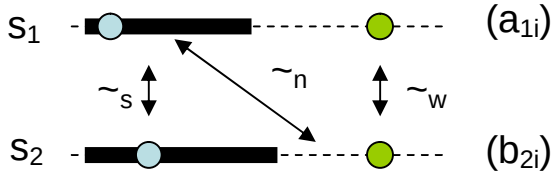

Figure 7. Similarity classes.

where the bold items represent primary service features (A_1 set), and the dashed items represent secondary service features (A_2 set) (similar for s_2)

We introduce strong, weak, and normal similarity, represented by $\sim_s$, $\sim_w$, and $\sim_n$, respectively.

Therefore, ($s_1 \sim s_2$) =

$= \sim_s$, iff all $a_{1i} \in A_1$ and $b_{2i} \in B_1$

$= \sim_w$, iff all $a_{1i} \in A_2$ and $b_{2i} \in B_2$

$= \sim_n$, iff there are $a_{1i} \in A_1$ and $b_{2i} \in B_2$ or
there are $a_{1i} \in A_2$ and $b_{2i} \in B_1$ (13)

Similarity composition allows to capture all possible combinations, e.g,, $\sim_{prod/s}$ represents a strong similarity defined by the producer, based on the primary service features.

A refinement of feature-based similarity is can be expressed when service features do not show a direct semantic matching, but feature composition might lead to such a match. Considering that a subset a service feature for a given service is equivalent with a feature for a service the similarity is computed for, we introduce feature composition-based similarity, as shown in Figure 2.

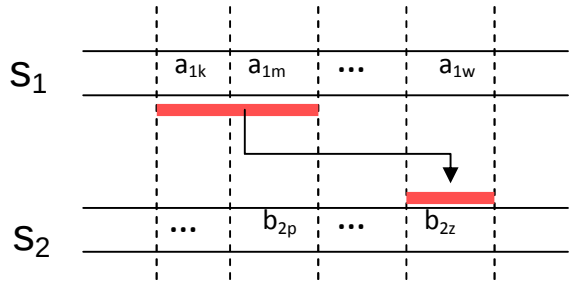

Figure 8. Feature composition-based similarity.

$s_1 \sim_{a1k,a1m\,/\,b2z} s_2$ (14)

with the semantic that the values of a_{1k} and a_{1m} composed are similar to the values of b_{2z}. Composition might be any arithmetic or Boolean operator, according to the nature of the features, e.g., if sets, then 'U' (union), if values, then '+' (addition), etc. If type, and a_{1k}:T1 and a_{1m}:T2, and b_{2z}:T3, then, then T3 is a subtype of either T1 or T2.

Combination between $\sim_s$, $\sim_w$, and $\sim_n$, and feature composition-based similarity can be applied following (13).

6.4 Updating similarity

When evaluating service similarities, perfect match of service features is desired, but rarely found, due to some continuous values of the features. For example, looking for a service offering the weather temperature with an accuracy of 0.1°F is not feasible. A query on what month the temperature is 67.3F might have no match; but, for a given location, a query on what moth shows [75-80] °F might be answered by April or May, if a Mediterranean area. We identify two possible relaxations when performing the matching.

6.4.1 Context-based feature migration

In time, and based on business models or customer feedback, some primary features become secondary, and vice-versa. Even more, in the same time, in different contexts, a feature can belong to either primary or secondary feature sets.

Let C = {c_i} a set of contexts and

$s1 ::= (A_1 \cup A_2)_{context = c1}$, with $A_1 \cap A_2 = \phi$

$s1 ::= (A'_1 \cup A'_2)_{context = c2}$, with $A'_1 \cap A'_2 = \phi$ (15)

then, the following is possible:

$s_1 \sim_{context = c1} s_2$

$s_1 \sim_{context = c2} s_3$ (16)

6.4.2 Feature relaxation-based similarity

Service features are not always perfectly matching (so goes for query matching, as well). Most of the time, the exact matching is not mandatory, e.g., if a service feature has a numeric value a variation of α_{1i} (usually symmetric, but not necessarily) of +/- α_{1i} is allowed. As a result, the similarity metrics presented in II.B can be relaxed. The same relaxation can be applied for similarity on data type/subtype, for similarity concerning the set of interface operations, or similarity concerning variations of an algorithm implementation. For example, when a query (with explicit relaxation of +/- 2ms) targets a service with a response delay of 10ms, any service offering a delay within [8ms, 12ms] is a desired matching. With no explicit relaxation delay, 10ms is mandatory. In this case,

$s1 \sim_{a1i +/- \alpha 1i}$ $s2 <=> b_{2i} \in [a_{1i} - \alpha_{1i}, a_{1i} + \alpha_{1i}]$ (17)

where a_{1i} and b_{2i} are the corresponding features of s_1 and s_2, respectively.

6.5 Recommender-based similarity

Recommender mechanisms rank [22] the products or services based on feedback received after a series of recommendations and successful transactions. The *ranking* is subject to incomplete, fictitious feedback, volume of transactions for a given product or provider, and confidence in feedback. Based on statistics, the recommender computes its own *ranking* per product, defining the reputation (r) of a service/product.

Considering a set of service clusters a recommender build based on type of services/products,, we define:

Cluster = {$cluster_i$}
with $s_1 \in cluster_i$ and $s_2 \in cluster_i$, for a given service feature

$$s_1 \sim_{feature = ai} s_2 ::= |rank_{s1} - rank_{s2}| < \varepsilon_{ai} \quad (18)$$

In general,

$$s_1 \sim_{Uai} s_2 ::= \max \{|rank_{s1} - rank_{s2}|\} < \min \{\varepsilon_{ai}\} \quad (19)$$

6.6 Customer feedback reputation-based similarity

Based on customer individual metrics, context, and potential query with relaxation, a reputation is associated with a service/product. Heuristics for updating the reputation have been presented in [22][23]. In general, the following information is available:

s <r>: each service <s> has an associated reputation <r>
P_i<s,r_i>: each provider offers a service with its associated reputation
P_j<s,r_j>: another provider can offer the same service with a different associated reputation
u <e, c>: a user *u* has a credibility and confidence metrics associated with
For simplicity, we consider that <e, c> are the same for any service.

For a given user, we define similarity in terms of r_s

$$s_1 \sim_{feedback} s_2 ::= |r_{s1} - r_{s2}| < \varepsilon_0 \text{ with } e > e_0 \text{ and } c > c_0 \quad (20)$$

In the following, the newly introduced model is used by an algorithm to identify the most suitable service to satisfy a query for a service.

VII. ALGORITHM FOR SERVICE RETRIEVAL USING SIMILARITY

We introduced a similarity model and classes of similarity that allow a user (invoker) to use for a service in a given context, allowing or not precise relaxation for some service features, and under different types of similarity (strong, weak, normal). Distance metrics were also adopted for services, in order to cluster the most suitable services for a particular query, before computing the similarity.

Based on the model previously introduced and on the user model [23] and reputation [22], a query for a service *s* can be expressed as

Q (s, similarity type, context, with/without relaxation on {a_{1i}})

The algorithm presented below illustrates the main steps to reach a service proposal that can be a set, a given service, or no service at all.

Algorithm for finding a requested service query **Q**, *based on similarity between potential satisfying services*

1: **begin**
2: **identify** the service cluster &&&see (4)]
3: **select** a distance metric &&&see (5)-(9)]
4: **calculate** distance between all s_i in the cluster
5: **select** a subset {s_k with min {$d(s_i, s_j) < \varepsilon$}
6: **if** Q with relaxation
7: **apply** (10) and (11) for all mentioned features
8: **if not**
9: **if** Q with context
10: **apply** (12) and (13)
11: **if not**
12: **compute** a subset {si} of the set found before step12
13: **select** {sl} from the subset of step 12, with rank (s_l) > δ_1 and $r_{feedbakc} > \delta_2$ &&&see (16) and (17)]
14: **select** a subset for the subset of step 13
15: **return** the subset of step 14
16: **end**

Note that the output of step 15 might be an empty set, or a set having many recommended services complying to the query conditions.

The complexity of the algorithm is given mostly by the number of services features that can be considered with relaxations.

A variation of the algorithm was experimented with relaxation conditions for a set of contexts. The number of features with relaxation, the number of contexts, and the number of services into a cluster determine the performance of the algorithm.

Different experiments on the on-line Barnes&Nobles system (on-line book shopping) show a reasonable improvement on the precision the algorithm returns after running various numbers of queries and varying different conditions.

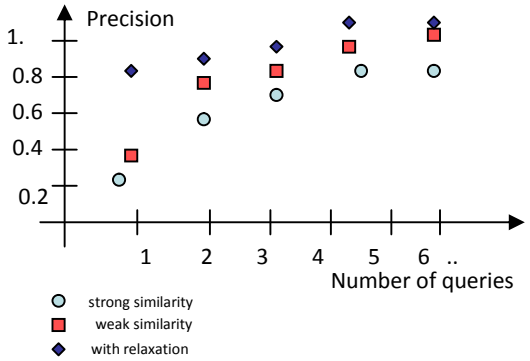

Figure 9. Precision of service returned to queries with different types of similarities

With no surprise, a service satisfying a query with relaxation riches quicker and with a higher precision the query expectation.

6.8 Similarity issues

We presented an approach for service invocation using similarity taxonomy where weak, strong, and normal similarity. Practically, services are clustered and service distance/similarity metrics were adopted from text-based domains. A reputation-based mechanism (introduced in [22][23]) is used in combination to context-based similarity and feature relaxation methods to identify a set of services that better serves a given query.

We also introduced the techniques of feature aggregation when similarity is evaluated and the continuous update of feature classification, i.e., primary/secondary, according to the context. More work should be done on these two items, as semantic-based aggregation should be considered.

6.7 Feature relaxation-based similarity

Service features are not always perfectly matching (so goes for query matching, as well). Most of the time, the exact matching is not mandatory, or, at least, the query can explicitly mention an acceptable variation. Usually, this is expressed as a constraint associated with a given service feature. For example, requiring a book delivery service, might have as a condition, *delivery costs* < *threshold*. In other cases, if a service feature has a numeric value a variation of α_{1i} (usually symmetric, but not necessarily) of +/- α_{1i} is allowed. As a result, the similarity metrics presented in II.B can be relaxed. The same relaxation can be applied for similarity on data type/subtype, for similarity concerning the set of interface operations, or similarity concerning variations of an algorithm implementation.

For s_1 [a_1, a_2, a_3,….a_n]
s_2 [b_1, b_2, b_3,…b_k]

Let us assume that a few service features a_i are associated with constraints. These constraints may be expressed as follows:

a_i > expression/threshold
a_i < expression/threshold
a_i ϵ &&&x, y] (*belongs to*, as an interval)
ai ϵ {x, y} (*belongs to*, as a set)

For a service selection, all expressions must be returned TRUE.

A query for a service can be represented by:

Q (s, similarity type, context, {(a_i, constraint$_i$)})

We express this as

$s_1 \sim_{constraint} s_2$ <=> bi satisfies a_i, and all constraint$_i$ are evaluated TRUE, for ALL *i* mentioned in the Q

For example, when a query (with explicit relaxation of +/- 2ms) targets a service with a response delay of 10ms, any service offering a delay within [8ms, 12ms] is a desired matching. With no explicit relaxation delay, 10ms is mandatory. In this case,

$$s1 \sim_{a1i +/- \alpha 1i} s2 <=> b_{2i} \epsilon [a_{1i} - \alpha_{1i}, a_{1i} + \alpha_{1i}] \quad (21)$$

where a_{1i} and b_{2i} are the corresponding features of s_1 and s_2, respectively.

As a note, similarity with constraints increases the chance to have a matching to a given query, on the expense of additional computation. A variation of this kind of similarity is when constraints are:

- Mandatory, for primary features (M)
- Optional (while desired), for secondary features (O)

For expressing these variations, a Q must be explicit on the categories of features

Q (s, similarity type, context, M:{(a_i, $constraint_i$)}, O:{(a_i, $constraint_i$)})

A response for the system should also contain the reference to the kind of feature/similarity, e.g.,

A response can be

$\{s_{1/M/nonO}, s_{3/M/O}, s_{8/M/nonO}\}$, or simply

$\{s_{1/M/nonO}, s_3, s8/M_{/nonO}\}$,

where *nonO* index represents the feedback of not all optional constraints were evaluated TRUE.

With no surprise, a service satisfying a query with relaxation riches quicker and with a higher precision the query expectation.

Two performance improving procedures are possible: (i) Building a query cluster each query belongs to (as a set of a priori known services that might satisfy the mandatory features of a given query, and (ii) Based on the responses, it is interesting that the same similarity criteria used to identify similar services can be used to evaluate the 'satisfaction' of the returned services.

The first procedure needs a look up in the previous query inventory, and group them by context and user. Then, looking by context and user ID, a subset of services are derived. A Q is now answered by considering a particular service cluster, sensitively smaller than the entire set of services.

By running a few situations, with 1,000 services, 5,000 contexts, 8,000 users, and queries with 4 mandatory features and 5 optional features, no constraints, the following results were obtained (Figure 10).

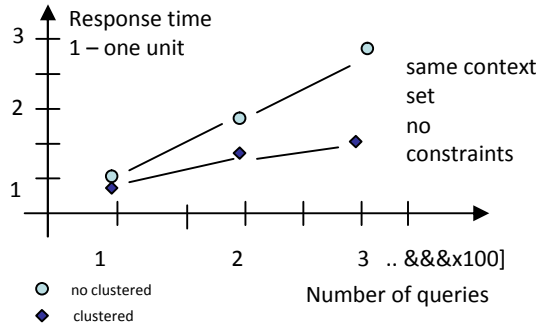

Figure 10. Response time versus number of queries in clustered and non clustered approaches

In the case where services are clustered (based on previous answers) the response time is dropping by almost half. However, a similarity (Q, A) must be also executed to evaluate hoe the recommended services satisfy a given query.

In terms of returned services, under the same setting, the results are presented in Figure 11.

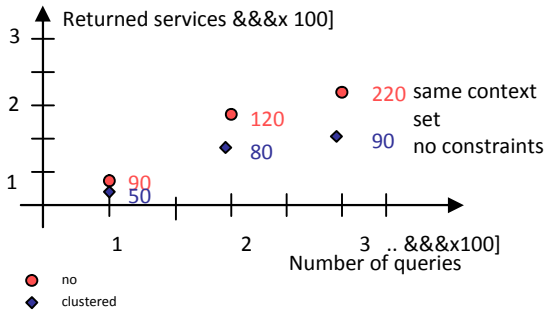

Figure 11. Returned services versus number of queries in clustered and non clustered approaches

A sensitive drop in returned services is observed in the clustered mode; however, it seems to be saturation with the number of queries increasing. This can be caused by the lack on context differentiation, or by the similarities of the queries. On the latter part, more experiments are needed.

The second procedure is used to apply similarity (Q, A) for each service in the retuned set, and select that service that has the max similarity. The procedure is simple, but requires to be applied, in turn, to all returned services. To simplify, one may select to run the similarity check only for the primary features. In this case, with the same settings for the experiment, the computation time is reduced by two thirds. This is due to the fact that similarity with constraints requires additional computation for each feature to validate that the constraint is TRUE.

To substantially reduce the computation time, and the cardinality of the returned services, a condition on service reputation reduced the time and the number of returned services.

VIII. CONCLUSION AND FUTURE WORKS

The paper presented a framework and appropriate mechanisms to evaluate the services/providers in the light of their respective direct impact on user perception. Essentially, the proposal considers several innovative ways of considering user impact on an accurate evaluation of a service/provider reputation.

The proposed schema can capture indirect service delivery and allow reputation correction based on the real transactions.

Future investigations should focus on a more formal definition of service/provider/feature similarity and the stability of the reputation accuracy over a longer period. This might lead to the reputation predictions; specialized metrics for assessing the accuracy of predictions in the light of indirect delivery are challenging but seen as very helpful in web-service driven environment.

On the user side, consistency feedback and reliability should be correlated with the frequency of users' report and transaction peaks, as well as with the user's report patterns. This will allow detection of potential 'off-market' agreements between providers and set an appropriate service level agreement policy.

One aspect that is left out, but worth to be mentioned, is that the reputation adaption function presented in the thesis should be refined. We adopted a linear $r = r_0 (1 + \lambda)$ reputation adaptation function. However, some services might listen to other forms of reputation adaptation function, as $r = r_0 (1 + e^{\lambda/r0})$, or $r = r_0 (1 + \log\lambda/r_0)$, etc. We think that this can be approached by defining on the customer side and on the provider side, a most suitable function for reputation update, considering the type of service and the context.

Another aspect that should be validated with more date sets is related to the duration of the trying period for endorsing the expected reputation. For example, 1-2 moth for a book service, 1-2 weeks for a coffee service, or 6 months for a piece of software. In this case, trying various validation periods might provide a more accurate reputation update.

Service reputation was calculated 'per context'. A global view, form the producer perspective will require studies on weighted cross-context reputation, in the case of a free-context query. For example, a formula might be

$$S_{cross\text{-}context} = (\Sigma\, w_i \times r_{contexti}) / \Sigma w_i \qquad (22)$$

In this case, large statistics are needed for an optimal tuning of w_i.

Another aspect that was surprising concerned the fact that new customers where quiet close to the estimated reputation; this raise the issue of a finer tuning considering customer profiles. The heuristics presented might need updates considering exceptions.

Finally, the fact that some service features shown strong impact on service reputation were originally classified as secondary (or, even miscellaneous) requires a more customer-oriented service design. Actually, the conclusion is quite the opposite with what is going on with the service launching, when a myriad of features are attached to a service, without a clear evaluation of a need. Even more, new features are added, without accurate validation of the use and customer evaluation of the existing ones.

IX. REFERENCES

[1] P. Resnick and H. R. Varian. Recommender systems. Communications of the ACM, 40(3):56–58, 1997.

[2] Esma Aimeur and Flavien Serge Mani Onana. Better control on recommender systems. E-Commerce Technology, 2006. The 8th IEEE International Conference on and Enterprise Computing, E-Commerce, and E-Services, The 3rd IEEE International Conference on Volume, Issue , 26-29 June 2006 pp. 38 – 38

[3] T. D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. Communications of the ACM, 35(12):61–70, 1992.

[4] Dieberger, A., Dourish, P., Hook, K., Resnick, P., and Wexelblat, A. Social navigation: techniques for building more usable system. Interactions, 7, 6, 2000, 36-45.

[5] Adomavicius, G. and Tuzhilin, A., Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. IEEE Transactions on Knowledge and Data Engineering, Volume 17, No. 6, June 2005. Pp. 734 – 749

[6] Chen, Y. B., and Xie, J. H. Online consumer reviews: a new element of marketing communications mix. Working paper, Eller College of Management, University of Arizona, Tucson, AZ, 2004.

[7] Kumar, N., and Benbasat, I. The influence of recommendations and consumer reviews on evaluations of Web sites. Information Systems Research, 17, 4, 2006, 425-439.

[8] Dini,O., Moh, M., Clemm, A.: Web Services: Self-adaptable Trust Mechanisms. AICT/SAPIR/ELETE 2005: 83-89

[9] Massa, P., & Bhattacharjee, B. (2004). Using Trust in Recommender Systems: An Experimental Analysis. Trust Management, (Proceedings of the 2nd International Conference, iTrust 2004).Oxford, UK, LNCS 2995, Springer, pp. 221-235.

[10] M. Pazzani, "A Framework for Collaborative, Content-Based, and Demographic Filtering, Artificial Intelligence Rev., pp. 393-408, Dec. 1999.

[11] A.I. Schein, A. Popescu, L.H. Ungar, and D.M. Pennock, Methods and Metrics for Cold-Start Recommendations," Proc. 25th Ann. Int'l ACM SIGIR Conf., 2002.

[12] R. Jurca and B. Faltings. "An Incentive Compatible Reputation Mechanism", Proc. IEEE Conf. on E-Commerce, pp. 285-292, Newport Beach, CA, June 2003.

[13] Pei-Yu Chen, Yen-Chun Chou, Kauffman, R.J. Community-Based Recommender Systems: Analyzing Business Models from a Systems Operator's Perspective. Proceedings of the 42nd Hawaii International Conference on System Sciences (HICSS-42). pp. 1-10, January 2009.

[14] Kwei-Jay Lin, Haiyin Lu, Tao Yu, and Chia-en Tai. A reputation and trust management broker framework for Web applications. Proceedings of The 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service. April 2005. pp. 262- 269

[15] Dellarocas, C. Strategic manipulation of Internet opinion forums: implications for consumers and firms. Mgmt. Sci., 52, 10, 2006, 1577-1593.

[16] O. Dini, P. Lorenz, and H. Guyennet; An Enhanced Architecture for Web Recommenders, SERVICE COMPUTATION 2009, IEEE Press, pp.

[17] Mean opinion score and metrics, http://technet.microsoft.com/en-us/library/bb894481.aspx [accessed: Jan 10, 2010]

[18] Adomavicius, G. Tuzhilin, A. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. IEEE Transactions on Knowledge and Data Engineering, Volume 17, No. 6, June 2005. Pp. 734 – 749

[19] PESQ, PESQ Algorithm firmeware, http://www.qoesystems.com/products/pesq_basic.htm &&acessed: January 13, 2010]

[20] Bruckner, R. M., List, B. and Schiefer, J., Striving Towards Near Real-Time Data Integration for Data Warehouses, In Proc. of the 4th Intl. Conf. on Data Warehousing and Knowledge Discovery (DaWaK 2002), Springer LNCS 2454, pp. 317–326, Aix-en-Provence, France, Sept. 2002.

[21] Schiefer, J., Seufert, A. 2005. Management and Controlling of Time-Sensitive Business Processes with Sense & Respond. In Proceedings of International Conference on Computational Intelligence for Modelling Control and Automation. Vienna, Austria.

[22] O. Dini, P. Lorenz, and H. Guyennet; An Enhanced Architecture for Web Recommenders, SERVICE COMPUTATION 2009, IEEE Press, pp. 372 – 378, ISBN: 978-1-4244-5166-1, Athens, Greece

[23] O. Dini, P. Lorenz, A. Abouaissa, and H. Guyennet, Dynamic Feedback for Service Reputation Updates, ICAS 2010, pp. 168-175 ISBN: 978-1-4244-5915-5, Cancun, Mexico

[24] C. Wu and E. Chang, Searching Services 'in the web': A Public Web Services Discovery Approach, SITIS 2007, The Third IEEE Conference on SignalImage Technologies and Internet-based Systems, pp. 321-328.

[25] M. Paolucci, B. Shishedjiev, Xh. Zenuni, and B. Raufi, GHSOM-based Web Service Discovery, 2010 European Computing Conference, ISSN: 1790-5117, 2010

[26] M. Szomszor, C. Cattuto, H. Alani, K. O'Hara, A. Baldassarri, V. Loreto, and V. D. Servedio, "Folksonomies, the semantic web, and movie recommendation," In 4th European Semantic Web Conference, Bridging the Gap between Semantic Web and Web 2.0, 2007.

[27] X. Dong, A. Halevy, J. Madhavan, E. Nemes, and J. Zhang, Similarity Search for Web Services, The 30th VLDB Conference, Toronto, 2004

[28] S. Cost and S. Salzberg, A Weighted Nearest Neighbor Algorithm for Learning Symbolic Features. Machine Learning, No. 10, 1993, pp. 57-78

[29] L.S. Larkey and W. Croft, Combining Classifiers in text Classifications Techniques, ACM SIGIR 1998.

[30] H.-H. Do and E. Rahm, COMA – A System for flexible Combination of Schema Matching Approaches, VLDB 2002

[31] A.M. Zaremski and J.M. Wing, Specification matching of software components. TOSEM, No. 6, pp. 333-369, 1997

[32] C. Bouras and V. Tsogkas, Improving text summarization using noun retrieval techniques, LNCS, Knowledge-based Intelligent Information and Engineering Systems, vol. 5178/2008, pp. 593-600