# A Reputation Assessment Model for Trustful Service Recommendation

Okba Tibermacine[a], Chouki Tibermacine[b], Foudil Cherif[a]

[a]*Biskra University, P.B. 145 R.P, Biskra 07000, Algeria*
[b]*LIRMM, Univ Montpellier, CNRS, Montpellier 34000, France*

**Abstract**

Nowadays, software systems are mainly Web front-based, Cloud-deployed and accessible by a wide audience over the Internet. These online systems commonly rely on Service-oriented Architecture principles, where they are built as orchestrations of RESTful (and in some rare cases as SOAP-based) services. Integrating new services in an existing orchestration is a challenging and risky task because trustworthiness of these services is not guaranteed throughout their lifetime. Reputation of services is a good indicator about the overall quality of services, because it reflects consumer satisfaction regarding the service-offered functionality and quality. Thus, reputation of services could be considered in the selection and recommendation of trustworthy services. In this paper, we present a framework for the management of web service reputation to conduct a better service recommendation. We present a reputation assessment model that aggregates fair user feedback ratings. The model includes a mechanism that prevents the introduction of malicious feedback ratings, by penalizing detected specious users. In addition, this framework includes a bootstrapping technique for estimating reputation of newcomer services based on neighbor similarity and initial advertised QoS. A set of experiments has been conducted to evaluate the effectiveness of the proposed framework. The results of these experiments highlighted the potential of our framework. These are presented at the end of the paper.

*Keywords:* Software as a Service, Service Oriented Architecture, Web services, Service selection and recommendation, Trust and Reputation

## 1. Introduction

Software as a Service (SaaS) model tends to deliver software on-demand and price on-use. This concept can be successfully applicable over the internet by implementing Service-Oriented Architectures (SOA). Many cost-centric objectives could be guaranteed when using SaaS. However, integrating outsourced external software components (Services) in a new software development setting is challenging and even risky [1],

*Email addresses:* o.tibermacine@univ-biskra.dz (Okba Tibermacine), tibermacin@lirmm.fr (Chouki Tibermacine), foud_cherif@yahoo.fr (Foudil Cherif)

notably because delivered *Quality-of-Service* (QoS) and trustworthiness of these components are not always guaranteed during execution time.

### *1.1. Context and motivation*

With the proliferation of services on the Web, the selection of services to build trustful applications becomes a more challenging task. Therefore, it is crucial to provide effective recommendation and selection techniques that recommend optimal and trustworthy services from users' point of view. Practically, users are often interested not only in what functionalities a service may offer, but also on what qualities (e.g., *availability*, *price*, *response time*,*failure rate*, etc.) the services may ensure. QoS is usually used for describing and quantifying service nonfunctional characteristics. Thus, QoS-based selection of services has been extensively studied in the literature (e.g. [2]). In fact, three sources of QoS information are identified: i) Provider advertisements, ii) Active service monitoring, and iii) User feedback.

The advertised values of service providers may be subjective; many providers claim that they offer the best QoS values to attract more clients, without referral to a real comparison with other providers. In addition, QoS provided by service providers at publishing time is outdated after a short period of time. Thus, a continuous refreshment of this QoS information is indispensable due to the rapid change in their execution context (changing network traffic, for instance).

The second source of QoS information is the active monitoring of services. This solution is adequate for small systems with a few number of services. Indeed, it is a costly solution to permanently monitor thousands of services where scalability and deployment issues have to be seriously tackled.

The third source for gathering QoS information is by collecting and aggregating consumer feedback. We distinguish two kinds of consumer feedback; i) feedback with QoS values issued from clients during service execution monitoring, and ii) rating values on a given scale that reflect consumer satisfaction about both service functionalities and QoS. The collection of these two kinds of information not only allows an accurate evaluation of QoS from client perspective, but also accessing reputation and trustworthiness of services.

In the literature, the aggregation of consumer feedback is a mechanism for the assessment of services *trust* and *reputation* in many open systems such as E-marketing, peer-to-peer networking and multi-agent systems.

In this work, we consider the term *reputation*, which is defined as a collective measurement, seen as a community of users' opinion regarding their experience with the used service.Reputation as a Quality of Experience (QoE) metric reflects the trustworthiness and credibility of services and their providers [3]. Thus, QoE is considered as an important criterion for service selection and recommendation.

Although existing works have proposed some efficient and robust reputation measurement models, most of them suffer from the following shortcomings:

1. *The consideration of fair ratings*: The presence of unfair ratings in online reputation systems is nearly inescapable and its negative impact on the performance of such systems [4]. A feasible way to sustain the robustness of these systems is to detect and filter out unfair ratings despite the challenging nature of this task

[5]. Many unfair rating detection models have been proposed, e.g. temporal analysis with user correlation analysis (TAUCA), RM model, DARC model and iCLUB model [6]. These models usually detect unfair ratings based on mathematical models. They show good accuracy, but they provide a large space of configuration possibilities [6]. This does not guarantee good results without a fine-tuning of their parameters, which requires a lot of effort. In addition, some of these models do not provide a complete framework for reputation assessment that takes all the aspects, such as credibility of users, timing of ratings, and bootstrapping of the reputation of newcomer services.

2. T*he consideration of user credibility*: Given the relevance that ratings and user feedback have an Impact on the performance of online reputation systems, it is of primary importance to detect and automatically correct ratings manipulations through dishonest or fake users [7]. Therefore, finding robust and reliable ways to distinguish between types of users and rate their importance is a crucial component in such systems, and thus credibility of raters should be evaluated carefully. The concept of user credibility was introduced in [8]; it takes into account the distance between the rating of the user for a service and the ratings of the other users for the same service (the higher is the distance, the lower is the credibility). However, the formulation of credibility in this paper is such that the final ranking does not accurately reflect the actual preferences of the users. This inaccuracy happens because they weigh ratings by users reputations but do not normalize with the sum of weights (users credibility); indeed, they divide the weighted ratings sum by the number of raters. Hence, when all users rate an item with the same value, the ranking is below that value and can further be smaller than the minimum allowed rating (details are provided in [7]). To overcome this issue, we propose to evaluate the credibility of users based on majority voting principle, and calculate the credibility (i.e. user honesty) as the mean probability that users give positive or negative ratings according to the majority of users after classifying their rates as negative or positive.
3. *Reputation bootstrapping*: Reputation bootstrapping refers to the problem of assigning initial reputation scores to newly deployed services for which no record of rating history is available [9]. In fact, most of online reputation management systems such as [3, 10, 11] focus on the mathematical models used for assessing service reputation from received ratings and neglect the estimation of newcomers reputation evaluation. Though some solutions have addressed this issue (e.g. [12]), most of these solutions assign the same initial reputation value to every newcomer service. Adaptive approaches such as [13] propose to estimate the reputation of a newcomer by adapting the value of user/service peers that interact or behave like the newcomer. For instance, the authors in [14] proposed a bootstrapping solution that employs users observable features (a.k.a tags). Specifically, the distance between the values of the different tags are computed to quantify how behaviorally similar the new agents are to the existing ones. Once enough interactions and experiences are available, the reputation scores of agents is estimated during their lifetime based on their actual behavior. In [13], Malik et al. proposed two approaches to compute initial reputation values for the newly deployed services. The first approach capitalizes on the cooperation among services in a

community-based context to derive initial reputation values. Specifically, consumers bootstrap newcomer services proportionally to the rate of maliciousness of the community to which services belong. In the second approach, community providers are asked to assess newcomer services for a certain period of time and assign them initial reputation values accordingly. In contrast to some approaches (e.g. [9, 15]), we propose to include a model that estimates the reputation of newcomer services based on the correlation between the QoS of a service and its reputation using a regression based technique.

4. *The consideration of web service orchestration reputation*: Using atomic services collaboratively creates on-demand value-added services following composition (i.e. orchestration) principles. This later is a natural phenomenon in popular online service marketplaces (web services, Cloud services or microservices) where functionally-similar services exist with dynamic QoS values. Generally, it is challenging to access the reputation of on-demand composite services, as they usually have little or no direct consumer feedback [16]. The reputation of a composite service depends on the aggregated reputation of its component services [17]. Thus, we propose to evaluate the reputation of orchestrated services using the reputation of its composite services (either calculated from ratings history or through bootstrapping). In the proposed approach, we do not consider reputation transfer from providers or community to services, as it is suggested by [18].

### *1.2. Contribution*

In this paper, we propose a reputation management framework for service recommendation, providing solutions to the limitations previously discussed. The contribution of this paper is four-fold:

- First, we propose an architecture for the management of web service reputation to facilitate web service selection. We describe the main components that handle feedback rating acquisition, storage, aggregation and service recommendation.
- Second, we propose a reputation assessment model that is sensitive to rating time and users credibility factors. The first factor permits to assign more important weights to new feedback ratings. We believe that the quality of service providers is in continuous change due to their presence in a dynamic environment. Therefore, an effective reputation assessment model should assign more importance to recent feedback ratings. The second factor ensures the evaluation of user's honesty, which has its impact on considering fair feedback ratings. We apply a user "punishment" mechanism to exclude unfair ratings received from users who do not guarantee a certain credibility (i.e., a threshold-based solution).
- Third, we introduce a bootstrapping technique for evaluating the initial reputation of newcomer web services based on: i) their similarity with previously evaluated services, and ii) their initial advertised QoS. Besides, we propose a model to evaluate the overall reputation of service orchestrations.
- Finally, we conducted experiments to evaluate the proposed reputation assessment model and compared it with existing models. The results showed that our model outperforms existing ones in most cases.

It is worth mentioning that although our proposition focuses on evaluating accurately service reputation in the context of web service recommendation, it can be effortlessly applied to a SaaS Model on the Cloud.

### *1.3. Paper outline*

This paper is organized as follows. Section 2 presents the architecture of the proposed framework. Section 3 details the reputation assessment model. Section 4 shows the conducted experiment for evaluating the model. Section 5 describes the related work, and Section 6 concludes the paper.

## 2. Reputation management framework for service recommendation

In this section, we present the reputation management framework for a trustful service recommendation. We describe the main components of this framework and their roles for acquiring, storing, and aggregating user feedback ratings.

### *2.1. System Architecture*

Figure 1 illustrates the architecture of the reputation management framework. This framework enables users to search for services by providing search queries or by direct browsing of registries via the *Search and Selection Interface* (Component 1). After using a selected service, a user may send a feedback rating that represents her/his satisfaction to the system via the *Feedback Collector* (Component 2). Collected feedback ratings are stored in the feedback database.

The *Reputation Manager* (component 3) reassess periodically the reputation of services based on new updates that occur in the feedback database. This component employs the assessment model presented in Sect. 3 to assess service reputation. New assessed reputation scores are stored in the reputation database for future use during service recommendation.

The Search and Selection Interface proposes for its users sorted sets of services that correspond to their search queries. These sets are prepared by the *Service Selector* (Component 4) which: 1) retrieves services from registries, 2) extracts reputation scores from the reputation database, 3) sorts services based on their reputation scores, and 4) delivers results to Component 1 which recommends them to users.

### *2.2. Feedback Collector*

The role of the *Feedback Collector* component is to provide a human-interface for service users, allowing them to submit their feedback ratings. A user feedback is a quantification of her/his opinion about the consumed service. In the proposed architecture, feedbacks are ratings that range in a scale of 10, where 0 represents a complete dissatisfaction and 10 a total satisfaction. Every service has a unique identifier. Therefore, the user during feedback submission has to provide the service ID and the assigned feedback rating. During each feedback transaction, this component stores in the feedback database the following information:

- Feedback_ID : represents the identifier of the current feedback record;

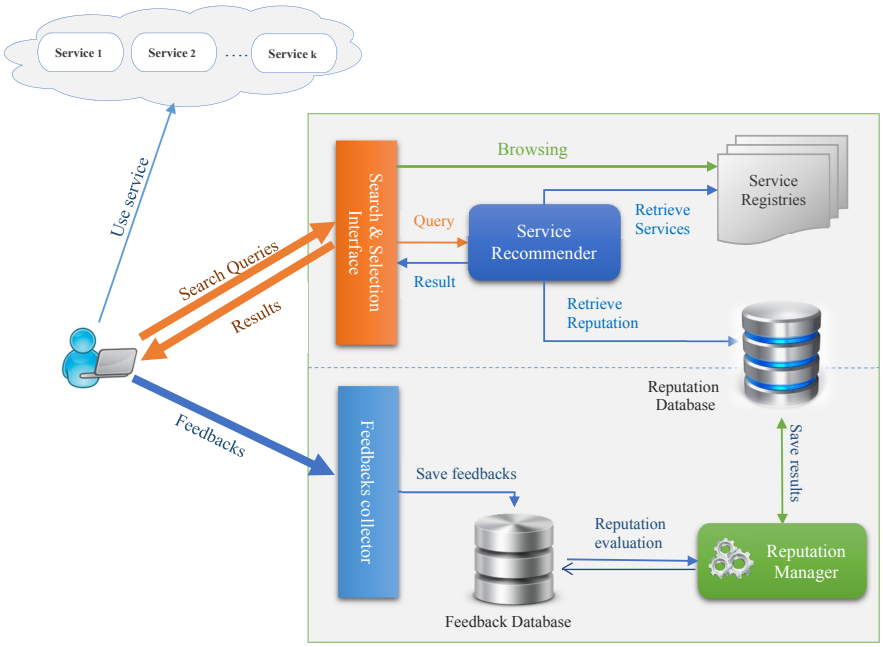


Figure 1: General Architecture of the Reputation Management Framework

- User_ID: represents the user identifier. In order to avoid user registration. The system considers the IP address of the user as its identifier;
- Service_ID: represents the identifier of the consumed service;
- Rating: represents the rate attributed by the user to the consumed service. Rates are unsigned integers that range between 0 and 10;
- Timestamp: represents the time/date of feedback reception;
- Modification_Nbr: represents the number of updates of the record. Initially, it takes the value 1.

Due to performance issues, for the same service and the same user, the feedback collector stores in the database only one record during an amount of time $T'$ (e.g. $T'$=24 hours). This record is updated when a new rating is introduced during that time interval. The feedback collector keeps only the last $N$ updates for the same record. $T'$ can be set based on administrator preferences.

### *2.3. Reputation Manager*

The *Reputation Manager* assesses the reputation of web services by aggregating feedbacks stored in the feedback database. We can summarize this component functionalities as follows:

- Retrieves new feedback ratings since the last assessment round from the feedback database;
- Selects, from the retrieved records, services whose reputation scores have to be assessed;
- Reevaluates the credibility (honesty factor) of the raters in respect to the model proposed in Subsection 3.4;
- Extracts all feedback ratings for each selected service, and assesses its reputation by applying the assessment model presented in Section 3;
- Stores assessment results in the reputation database.

  Stored records have the following structure:

  1. Service_ID: represents the identifier of the service in the system;
  2. Reputation: represents the assessed reputation value for the service;
  3. Timestamp: represents the time of the last assessment round;

- The reputation manager starts a new assessment round every time slot $T$. $T$ could be initialized by administrator based on the system performance.

### *2.4. Search and Selection Interface*

The Search and Selection interface allows users to interact with the system for selecting web services. The user via this component can browse directly service registries. Moreover, it can handle search queries. These queries are generally a set of keywords describing the sought services. The interface transfers queries to the service selector component, which analyses them and searches for appropriate services that match these queries. The selector returns sorted lists of services to the interface, which, by its turn, recommends them to users.

### *2.5. Service Recommender*

The service recommender component processes user queries through the following steps:

- **Step 1 – Query preparation**: the component analyses and prepares the query by removing stop words, stemming the remaining keywords, and adding synonyms to the query. The result of this step is a bag of words that represents the enriched initial search query.
- **Step 2 – Service retrieval**: the selector component searches for services that hold at least one element in the bag of words. We suppose that services in registries are tagged using one of the techniques proposed by Azmeh *et al.* [19] and Falleri *et al.* [20] or any other similar work. We assume that similarity values between services are assessed using the approach proposed by Tibermacine *et al.* [21] or by any other related work. This information is then stored in the database.

- **Step 3 – Reputation-based sorting**: After selecting web services, the component retrieves the reputation value of each service in the result set from the reputation database. Then, the component groups and sorts services based on their reputation and similarity values. Recommended results are sent back to the interface component.

## 3. Reputation Assessment Model

In this context, we consider reputation as an aggregation of user feedback ratings. Below, we present the mathematical model implemented by the reputation manager in the previous architecture.

### *3.1. Evaluation Metrics*

The reputation assessment model is built upon the following metrics. Some of these metrics have already been used individually in related works (i.e., [22–24]). However, these models do not incorporate all of these metrics together. As we will see later, this combination results in a more efficient model.

1. **User honesty degree (i.e., credibility)**: The credibility of a user has its impact on feedback ratings during reputation assessment; a dishonest user (i.e., a user with malicious behavior) can continuously decrease the reputation of a good service or increase the reputation of a poor service by providing a false feedback. Therefore, it is essential for an accurate reputation assessment to consider the credibility of users during reputation assessment.
2. **User rating history**: Users may behave maliciously; they can start as honest users, then they change their behavior over time. In consequence, the assessment model has to estimate and update the credibility of users with regard to their rating history.
3. **Punishment of suspicious users**: We consider a user as a suspicious user when her/his estimated credibility is less than a certain degree (i.e., a given threshold). For system safety, the model neutralizes feedback ratings of all suspicious users. This mechanism ensures the purity of feedback ratings used during the assessment of service reputation.
4. **Feedback history**: The reputation assessment model uses all feedback ratings stored in the database during each reassessment.
5. **Temporal sensitivity**: the module includes an exponential decay function with a decay factor (denoted as $\lambda$). The product of rating by the decay function gives more importance to newer ratings, which will have more influence on the reputation assessment.

### *3.2. Formulation*

Let $\delta_(i,k)$ be the feedback rating given by user $i$ for service $k$. The rating values range between 0 and 10, where 0 represents a total dissatisfaction about the functionalities and QoS of the used service, and 10 represents a total satisfaction. More precisely, let $\delta_(i,k) \in \{\delta^-_{(i,k)}, \delta^+_{(i,k)}\}$ , where $\delta^-_{(i,k)} \in \{0,1,2,3,4,5\}$ and $\delta^+_{(i,k)} \in \{6,7,8,9,10\}$. We

assume that $\delta^{-}_{(i,k)}$ is a negative feedback, and $\delta^{+}_{(i,k)}$ is a positive one. We group feedback ratings in two classes (positive and negative) in order to represent the fact that the rater's opinion is either positive or negative.

Let $\Phi(S_k)$ be the sum of all rates weighted by the time sensitivity factor $\lambda$ and the rater's honesty degree $H$. Mathematically, we define $\Phi(S_k)$ as follows :

$$\Phi(S_k) = (\sum_{i=1}^{n} \delta^{+}_{(i,k)} \times \lambda^{d_i} \times h_i) + (\sum_{j=1}^{m} \delta^{-}_{(j,k)} \times \lambda^{d_j} \times h_j) \tag{1}$$

Where :

- $\lambda^{d_i}$ is an exponential decay function with the base lambda ($\lambda$) and the exponent ($d_i$) which represents the age of the rating in days. $\lambda$ is a fraction between 0 and 1 and is set experimentally. Using this factor, every time the rating gets older, its effect on the reputation assessment becomes smaller.
- $h_i$ is the Honesty degree (i.e., the credibility factor) assessed for the user $i$ using Eq.4.
- $n+m$ is the number of all raters for the service $S_k$ where n is the number of raters who provided a positive feedback and m is the number of raters who provided negative feedback.

In addition, we define the intermediate function $\Omega(S_k)$ of a service $S_k$ as the ratio ($\frac{i}{ii}$) between (i) the difference between the evaluation of positive ratings (the sum of positive ratings weighted by their corresponding inclusion and honesty factors) and the evaluation of negative ratings (the sum of negative ratings weighted by their corresponding inclusion and honesty factors) and (ii) the global evaluation of ratings ($\Phi(S_k)$). We write:

$$\Omega(S_k) = \begin{cases} \frac{(\sum_{i=1}^{n} \delta^{+}_{(i,k)} \times \lambda^{d_i} \times h_i) - (\sum_{j=1}^{m} \delta^{-}_{(j,k)} \times \lambda^{d_j} \times h_j)}{\Phi(S_k)} & if \Phi(S_k) \neq 0 \\ -1 & Otherwise \end{cases} \tag{2}$$

Where:

- $\delta^{+}_{(i,k)}$ is the i-th positive rating, and $n$ is the number of positive rates for service ($S_k$). $\delta^{+}_{(i,k)} \in \{6,7,8,9,10\}$.
- $\delta^{-}_{(j,k)}$ is the j-th negative rating, and $m$ is the number of negative rates for service ($S_k$). $\delta^{-}_{(j,k)} \in \{0,1,2,3,4,5\}$.
- The range (co-domain) of the $\Omega$ function equates to $[-1,1]$.

### 3.3. Reputation

In case the function $\Omega(S_k)$ is evaluated to 1 (or -1), this means that all rates of service $S_k$ are positive (or negative) and then its reputation ($R(S_k)$) is calculated as the fraction of the average of feedback ratings. Otherwise, the reputation ($R(S_k)$) is the normalized value of the $\Omega$ function in the interval [0,1], which represents the relationships between positive and negative feedback ratings. We write:

$$R(S_k) = \begin{cases} \frac{(\sum_{i=1}^{n} \delta^{+}_{(i,k)} \times \lambda^{d_i} \times h_i)}{\sum_{i=1}^{n} \lambda^{d_i} \times h_i} & if \Omega(S_k) = 1 \\ \frac{(\sum_{j=1}^{m} \delta^{+}_{(j,k)} \times \lambda^{d_j} \times h_j)}{\sum_{j=1}^{m} \lambda^{d_j} \times h_j} & if \Omega(S_k) = -1 \\ \frac{\Omega(S_k)+1}{2} & Otherwise \end{cases} \tag{3}$$

### 3.4. Honesty Factor

The honesty factor (the credibility value) of a given user is the probability that this user gives an honest feedback according to the majority of raters. By default, a new user takes the value $\frac{1}{2}$. This value means that the user is neither honest nor dishonest. So, feedback ratings provided by new users do not affect heavily the reputation of the rated services. But by giving more ratings in the future, the honesty factor of this user will change, and it is calculated by the system as follows:

$$h_i = \frac{\sum_{s=1}^{t} (\frac{\eta(s)}{\eta^{+}(s)+\eta^{-}(s)})}{t}$$

$$\text{Such that :} \quad \eta(s) = \begin{cases} \eta^{+}(s) & if(\delta_{i,s} = \delta^{+}_{i,s}) \\ \eta^{-}(s) & if(\delta_{i,s} = \delta^{-}_{i,s}) \end{cases} \tag{4}$$

where,

- $t$: is the number of services rated by user $(i)$
- $\eta^{+}(s)$: denotes the number of positive ratings for service (s)
- $\eta^{-}(s)$: denotes the number of negative ratings for service (s)

For illustration, Figure 2 shows how to evaluate the honesty factor of a user $i$ who rated the service set composed of $\{WS_7, WS_{12}, WS_{19}, WS_{58}, WS_{62}\}$.

### 3.5. Suspicious User Punishment

As mentioned previously, the system considers a user with credibility value less than a fixed threshold as a suspicious user. Consequently, the effect of feedback ratings provided by this user have to be neutralized. Therefore, the system punishes suspicious users by setting their credibility to zero. In this way, the system ensures that service reputation is assessed only from fair feedback ratings (i.e., feedback ratings that are provided by users considered as honest). The credibility of a user is often subject to

| Rated Services | Other's feedbacks | | User (i) feedback | Probability |
|---|---|---|---|---|
| | Positive (+) | Negative (-) | | |
| $WS_7$ | 12 | 3 | - | $\frac{4}{16}=\frac{1}{4}$ |
| $WS_{12}$ | 1 | 10 | - | $\frac{11}{12}$ |
| $WS_{19}$ | 5 | 1 | + | $\frac{6}{7}$ |
| $WS_{58}$ | 16 | 0 | + | 1 |
| $WS_{62}$ | 6 | 3 | - | $\frac{2}{5}$ |
| | | | $H_i$ | 0,66 |

Figure 2: Sample of Honesty factor assessment for a user *i*

change due to her/his future behavior. Thus, the model reevaluates user credibility every time the system gathers new feedback ratings from that user. However, time sensitivity factor reduces the impact of old malicious feedback ratings in case that they are provided by a suspicious user in case she/he improved her/his behavior and has provided some new fair ratings.

### *3.6. Reputation bootstrapping*

System bootstrapping for evaluating reputation of **newcomer** services is a crucial and still challenging topic. Because neglecting the initial reputation scores of newcomer services might lead to subvert the performance of the whole system, making it vulnerable to many threats such as the Sybil attack [25].

In this model, rather than assigning default reputation values to newcomer services, we evaluate the reputation of a newcomer service by comparing its QoS to those of its similar services. We suppose that the reputation of a given service is the degree of user's satisfaction about service qualities. Generally, if we have a newcomer service that offers functionalities and QoS similar to some older services in the system, which already have reputation values (i.e., the system has already assessed their reputation), then we can evaluate the initial reputation of the newcomer service in terms of the reputation values of these similar services.

We assume that during the publication of a new service in the system, the provider publishes the QoS values of its new service. For simplicity, the system takes into account three qualities[1]: Response time, Availability, and Price, which are denoted respectively by $(T)$,$(A)$ and $(P)$. The process of evaluating the reputation of the newcomer service $S_{nc}$, is defined as follows:

1. **Similar services retrieval**: The first step is to find similar services from the service pool managed by the system. To that end, the system assesses the similarity

[1]The approach can however be easily generalized to more qualities.

between the newcomer service and existing services, using the similarity assessment approach proposed in [21]. The result of the similarity assessment is a set of similarity scores that represent the degree of resemblance between couples of services. These scores range between 0 and 1, where 0 represents a total dissimilarity and 1 a total similarity between the matched services. Services that hold a similarity-score greater than or equal to a fixed threshold are considered similar to the newcomer service. The result of this step is a set of similar services denoted by *SimResultSet*. s.t.
$SimResultSet = \{S_1, S_2, ..., S_n\}$ and $Sim_i, i = 1, ..., n$ denotes the similarity between the newcomer service $S_{nc}$ and $S_i$.

2. **QoS data preparation:** In the second step, the system retrieves QoS values and reputation scores for each similar service in the result set. Each service $S_i \in SimResultSet$ is represented by $S_i < T_i, A_i, P_i, R_i >$ where $T, A, P$ are the QoS values and $R$ is the service reputation. Likewise, the newcomer service is defined by $S_{nc} < T_{nc}, A_{nc}, P_{nc}, \hat{R}_{nc} >$, where $T_{nc}$, $A_{nc}$ and $P_{nc}$ represent the initial QoS values and $\hat{R}_{nc}$ (or $R(S_{nc})$) is the unknown reputation value.
3. **QoS normalization:** In this step, the system normalizes QoS data as follows:
   - Extracting the maximum value ($MaxVal$) and the minimum value ($MinVal$) for each quality.
   - For each value, replace it by ($NewQosVal$) which is calculated as follows:

$$NewQosVal = 10 \times \frac{QosVal - MinVal}{MaxVal - MinVal} \tag{5}$$

   For the sake of simplicity, we suppose that the reputation value is in a linear relationship with services QoS. This means that users are more satisfied by good QoS and vice-versa, and this follows a linear correlation. Thus, for each Service $S_i$ we have the following equation :

$$\alpha_i T_i + \beta_i A_i + \gamma_i P_i = R_i \tag{6}$$

   Where coefficient $\alpha_i, \beta_i$ and $\gamma_i$ could be computed as follows :

$$\alpha_i = \frac{R_i}{3 \times T_i}; \qquad \beta_i = \frac{R_i}{3 \times A_i}; \qquad \gamma_i = \frac{R_i}{3 \times P_i} \tag{7}$$

4. **Reputation evaluation:** To assess the reputation ($\hat{R}_{nc}$) of the newcomer service $S_{nc}$, the system evaluates the following equation:

$$\hat{R}_{nc} = \alpha_{nc} T_{nc} + \beta_{nc} A_{nc} + \gamma_{nc} P_{nc} \tag{8}$$

   Where, $\alpha_{nc}, \beta_{nc}$, and $\gamma_{nc}$ are computed using equations 9,10, and 11 respectively.

$$\alpha_{nc} = \frac{\sum_{i=1}^{n} (Sim_i \times \alpha_i)}{\sum_{i=1}^{n} (Sim_i)} \tag{9}$$

$$\beta_{nc} = \frac{\sum_{i=1}^{n}(Sim_i \times \beta_i)}{\sum_{i=1}^{n}(Sim_i)} \tag{10}$$

$$\gamma_{nc} = \frac{\sum_{i=1}^{n}(Sim_i \times \gamma_i)}{\sum_{i=1}^{n}(Sim_i)} \tag{11}$$

### *3.7. Provider Reputation*

The reputation of the provider mainly depends on the quality of its offered services and thus on their reputation. Consequently, we assess the reputation of a provider as the average of the reputations scores of its services. Given a provider $Pr_x$, let $Services(Pr_x) = S_1, S_2, ..., S_n$ be the set of the services provided by $Pr_x$. The reputation of this provider is assessed as follows:

$$RP(Pr_x) = \begin{cases} \frac{(\Sigma_{i=1}^{n} R(S_i))}{n} & If \quad Services(Pr_x) \neq \phi \\ \frac{1}{2} & Otherwise \end{cases} \tag{12}$$

where,

- $R(S_i)$ is the reputation of service $S_i$ that belongs to the provider's service set ($S_i \in services(Pr_x)$).
- $\phi$ denotes the empty set.

In case the provider is new, the initial reputation of the newcomer service is estimated using a regression model which considers QoS values and the reputation scores of all the existing services handled by the reputation management system. This regression model is detailed in [26].

### *3.8. WS Orchestration Reputation Assessment*

Many proposed reputation models focus on the selection of single services and do not consider service orchestrations. Ideally, advanced WS selection systems have to enable their users to select WS orchestrations among different possibilities based on reputation and service similarities. Therefore, a user can start by selecting a set of services to construct her/his orchestrations. The system assesses the reputation of this orchestration based on the reputation of single services. Then, it suggests similar orchestrations with better reputation values based on possible service substitutes (similar services that can replace initial ones in the orchestration).

In this section, we complete the previous model for enabling reputation-based selection of WS orchestrations.

In general, an orchestration is described in terms of a process which consists of a set of basic and structured activities. A basic activity relies on the use of an interface provided by a basic web service. It could be an invoke (invocation of a service), a receive (reception of message from a service) or a reply activity (delineating responses to messages that were previously received during the execution of receive activities). On the other hand, a structured activity prescribes the order of execution of a set of (basic or structured) activities; a structured activity may have one of the following types:

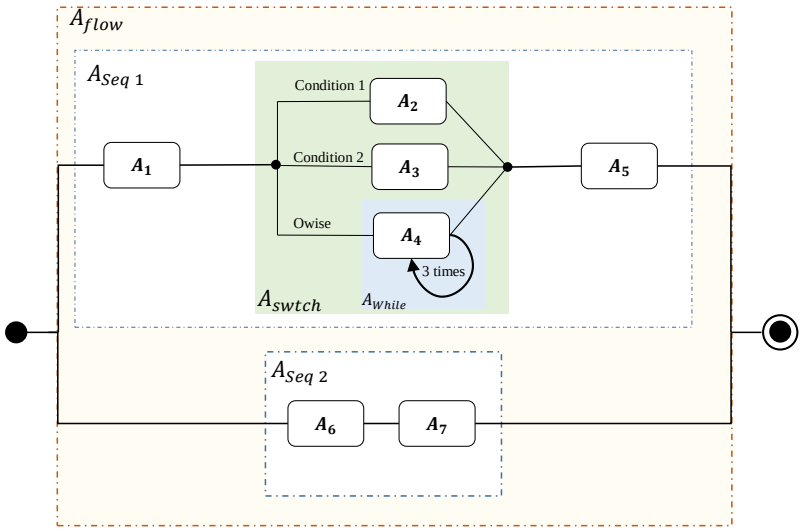


Figure 3: A hypothetical orchestration process

- **Sequence activities**: they consist of a set of activities that are executed sequentially.
- **Switch activities**: they consist of a set of ordered activities associated with conditions. Only the first activity whose condition is evaluated to true is executed in a switch activity.
- **While activities**: it holds a single activity that is executed iteratively for a given number of iterations.
- **Flow activities**: they consist of one or more activities, which, by default, are executed concurrently.

In this work, we propose to estimate the reputation of service orchestration based on the reputation values of its atomic services (i.e. orchestrated services). Table 1 shows rules to evaluate each type of activities. We note that in this model we ignore receive and reply activities because they always happen after invocation activities. The estimation of orchestration reputation is conducted by assessing the different activities.

For illustration, let us suppose the following hypothetical example. Let $O$ be the orchestration of 7 atomic services such that service $s_i$ is used in activity $A_i$ where $i = 1,..,7$. Figure 3 depicts the process that describes this orchestration. $O$ textually can be written as follows:

Table 1: Assessment rules

| Activity type | Description | Reputation assessment |
|---|---|---|
| **Invoke** | $A_{inv}(S_i)$ | $R(A_{inv}) = R(S_i)$ |
| **Sequence** | $A_{seq}(a_1, a_2, ..., a_m)$ | $R(A_{seq}) = average(R(a_1), R(a_2), ...., R(a_m))$ |
| **Switch** | $A_{swtch}((a_1\|cond_1), (a_2\|cond_2), ..., (a_m\|cond_m))$ | $R(A_{swtch}) = Min(R(a_1), R(a_2), ...., R(a_m))$ |
| **While** | $A_{while}(a_i) * n \quad times$ | $R(A_{while}) = R(a_i)^n$ |
| **Flow** | $A_{flow}(a_1, a_2, ..., A_m) = a1 \|\| a_2 \|\| ... \|\| a_m$ | $R(A_{flow}) = average(R(a_1), R(a_2), ...., R(a_m))$ |

$$O = A_{flow}\Big((A_{seq1}(A_1, A_{switch}(((A_2|cond_1), (A_3|cond_2), (A_{while}(A_4)*3)), A_5)), (A_{seq2}(A_6, A_7)))\Big)$$

For instance, given the reputation values for the atomic services as follows : $s_1 = 0.8, s_2 = 0.7, s_3 = 0.9, s_4 = 0.95, s_5 = 0.75, s_6 = 0.6$ and $s_7 = 0.85$, the reputation of the orchestration $O$ is evaluated by applying rules in Table 1 as follows :

- $A_1 = R(s_1) = 0.8$, $A_2 = R(s_2) = 0.7$,
  $A_3 = R(s_3) = 0.9$, $A_4 = R(s_4) = 0.95$,
  $A_5 = R(s_5) = 0.75$, $A_6 = R(s_6) = 0.6$, and
  $A_7 = R(s_7) = 0.85$.
- $A_{while} = (A_4)^3 = 0.857$
- $A_{switch} = Min(A_2, A3, A_{while}) = 0.7$
- $A_{seq1} = Average(A_1, A_{switch}, A_5) = 0.75$
- $A_{seq2} = Average(A_6, A_7) = 0.725$
- $A_{flow} = Average(A_{seq1}, A_{seq2}) = 0.737$

The reputation of the orchestration $O$ for the given example is equal to 0.737. Finally, it is important to mention that the goal from the orchestration reputation assessment model is to provide a good estimation of the reputation of the composite service before its deployment, so that the recommendation system can classify it.

## 4. Experimental Evaluation

In this section, we present the setting and the results of a set of experiments which were conducted to evaluate the reputation assessment model. Firstly, we study the impact of users' subjective and malicious feedback ratings on the accuracy of the proposed model. Secondly, we investigate the impact of the other parameters on the performance of the model. Finally, we conduct a comparison between the proposed model and some state-of-art approaches.

Table 2: Honest User Rates

| WS class | Rating | Description |
|---|---|---|
| **C1** | [0.7-1] | |
| **C2** | [0,3] | Users rate randomly |
| **C3** | [0.7-1] [0 - 3] | in the interval of $\pm$ 1 |
| **C4** | [0-3] [0.7-1] | of the RefVal |
| **C5** | [0-1] ($\pm$1) | |

*4.1. Experiment Description*

Due to the unavailability of feedback rating data, we have used a simulation in our experiments. We have built a concurrent Java program that simulates the interactions between Users, Services and the Reputation Manager (System). The program is designed in a way that the behavior of services is monitored, yet accurately captured. Hence, the program can simulate the behavior and feedback ratings of honest and malicious users accordingly. The system assesses reputation of services depending on collected feedback ratings, aging factor and user's credibility. The variance between assessed reputation scores of services and their ideal (expected) reputation (represented numerically in the interval of $[0, 1]$) is a key factor for the validation of the proposed reputation measurement model.

The programmed environment held a number of web services (*#Services*), and a number of users (*#Users*). We have simulated the interaction between these elements and the reputation manager during time slots. Each time slot is simulated as one day (*#Day*). A number of feedback rating transactions (*#TransactionsPerDay*) are issued from interactions between users and web services in each day. At the end of the day, the reputation manager calculates and updates the credibility factor of each user. Then, it assesses the reputation score of services using the model presented in Section 3. In order to have the most accurate results, the program uses a multi-round run (#Rounds) of simulation. The program lists the average of the assessed reputation scores of services in each class, along with their ideal reputation values. The simulation classes of web services are described in Subsection 4.1.1. User classes are presented in Subsection 4.1.2. Parameter tuning and the basic simulation algorithm are described in Subsection 4.1.3. Performance evaluation is presented in Subsection 4.1.4.

*4.1.1. Service classes*

The quality of a web service varies over time due to its presence in a dynamic environment where changes occur constantly. We can distinguish five different classes of service behaviors [23]; a first class of services that maintain a high level of performance, a second class of services that maintain a low level of performance, a third class of cheating services that start with a good performance then after a period of time, their performance degrades. A fourth class of services that start with a low performance and then their performance upgrades, and a fifth class of services with an oscillate performance.

We have implemented these five classes using a randomization function. In each class, the attribute performance value (*PerfVal*) represents the ideal reputation that corresponds to the actual behavior of a web service. (*PerfVal* ranges in the interval [0-1],

Table 3: Classes of simulated web services

| Class | Reputation | Execution time (ms) | Description |
|---|---|---|---|
| **C1** | [0.8 - 1] | [20-60] | Consistent high performance |
| **C2** | [0-0.2] | [100-150] | Consistent low performance |
| **C3** | [0.8 - 1] $\searrow$ [0 - 0.2] | [30-60] $\searrow$ [100-150] | High performance, then a degradation |
| **C4** | [0 - 0.2] $\nearrow$ [0.8 - 1] | [100-150] $\nearrow$ [30-60] | Low performance, then an enhancement |
| **C5** | [0-1] | [20-150] | Oscillate performance |

Table 4: Malicious User Rates

| WS class | Rating | Description |
|---|---|---|
| **C1** | [0 - 6] | Users rate randomly by value that are >>or <<than the RefValue |
| **C2** | [4 - 1] | |
| **C3** | [0 - 6] [4 - 1] | |
| **C4** | [4 - 1] [0 - 6] | |
| **C5** | [0-1] ($\pm$2) | |

where 0 denotes the lowest quality, and 1 represents the highest quality. The attribute Response Time (*ResT*) indicates the web service's maximum response time. Table 3 lists the expected parameters (Ideal reputation (*PerfVal*), and Execution time (*ResT*)) of each class.

Web service classes group all possible service behaviors, which ensures that experiment samples are representative of a real world environment.

*4.1.2. Users Classes*

To study the effect of users' credibility on the reputation assessment model, we propose to incorporate two classes of users that simulate the behavior (rating strategy) of honest and malicious users. Honest users give an organic (i.e. a fair) rate based on the performance value (PerfVal) that represents the ideal reputation of the service. A malicious (i.e. dishonest) user gives feedback that does not reflect the service's deserved performance value (PerfVal). The process of how the two classes provide a feedback rating is described in the following subsections.

**Malicious user class:**

The malicious user class represents a malicious user attempting to game the recommendation system by rating services using the following steps:

- The user selects a random, uniformly distributed service (ID: Identifier) from the service pool,
- Then, she/he reads the PerfVal (the performance value of the service) of the selected service; the PerfVal is a value that represents the qualities of the service and depicts its deserved/Ideal ranking score.
- Then, the user randomly decides whether she/he gives a malicious rate or a fair (organic) rate following the Bernoulli distribution (Bernoulli (0.71). 71% of the time, she/he gives a malicious rate. That is to say, more than two-thirds of the

Table 5: Simulation parameters

| Parameter | Values |
|---|---|
| **#Services** | 500 (100 per class) |
| **#Users** | 1000 |
| **#Days** | 100 |
| **#TransactionsPerDay** | 10000 (2000 per class) |
| **#Rounds** | 10 |
| **#HUserDensity** | ?% (variable) |

time, the user acts maliciously, and she/he acts honestly in one-third, attempting to hide her/his identity.

- If the user decides to act maliciously, she/he gives a feedback rating (following the discrete uniform distribution in the interval [0, (PerfVal*10 -2)] + [(PerfVal*10 +2), 10]. She/he gives a rate that does not reflect the deserved value estimated by the PerfVal of the service. For instance, if the PerfVal of the service is 0.6, then the user selects randomly one of these rates 0,1,2,3,4,8,9,10 to use as feedback.

- Otherwise (If the user decides to give a fair rate), she/he gives a random rate from the following three values (i) PerfVal*10 -1 or (ii) PerfVal*10, or (iii) (PerfVal*10) + 1. These values depict a natural variation between honest users opinions. For example, if the Perfval of the service is 0.6, then the user selects randomly one of these values 5,6,7.

**Honest user class:**
The honest user class represents a user that interacts with the recommendation system by rating services using the following simple steps:

- The user selects a random, uniformly distributed service (ID: Identifier) from the service pool,

- Then, she/he reads the PerfVal (the performance value of the service) of the selected service;

- The user rates the service using, randomly, one of the following values: (i) PerfVal*10 -1 or (ii) PerfVal*10, or (iii) (PerfVal*10) + 1. These values represent a natural variation between users opinions.

Tables 2 and 4 respectively summarize feedback ratings patterns generated by honest and malicious users for each class of services.

*4.1.3. Tuning*

Algorithm 1 represents the basic flow of one simulation round. We simulated a system with 500 services; we created 100 instances from each class of services. Then we created 1000 honest and malicious users according to the honest user density (#HUserDensity). For each time slot, which varies from 1 to the total number of simulated time

slots (#Days), we generated the ideal reputation of each service instance according to its service class. Then, we simulated the transactions (interactions) between randomly selected users and randomly selected services. Transactions are conducted simultaneously. After each transaction, the program stores the feedback rating generated by the involved user. At the end of each time slot, the program updates user credibility scores, and it assesses the daily reputation score of each web service. The program, at the end of all transactions, measures model performance and prints results. Table 5 lists simulation parameters.

**Algorithm 1** Pseudocode of the conducted simulation

**Input:** *#HUserDensity*
**Begin**

```
#Days = 100 ;
#services = 500;
#TransactionsPerDay = 10000 ;
Create services(#Services);
Create users(#HUserDensity);
for simDay = 1 to #Days do
    Generate_Ideal_Reputation_for_Services();
    for (class = 1 to 5) do
        for (i=0 to (#TransactionsPerDay/5)) do
            user = Select_random_user();
            service = Select_random_service_in_class(class);
            simulate_user_service_interactions(user, service);
            add_new_feedback_rating(user, service, simDay);
        end for
        evaluate_users_credibility();
        evaluate_daily_ideal_service_reputation();
        assess_daily_reputation_from_feedback_ratings();
        save_data();
    end for
end for
evaluate_results();
```

**End**

### *4.1.4. Performance metrics*

Performance metrics are classified into statistical accuracy and decision-support accuracy metrics [27, 28]. We adopt one metric from each class to evaluate the performance of the proposed reputation assessment model. *Mean absolute error* (MAE) is a representative metric of a statistical accuracy measure. The MAE metric is defined as follows:

$$MAE = \frac{\sum_{k=1}^{n} |Rep_a(WS_k) - Rep_i(WS_k)|}{N} \tag{13}$$

where, $Rep_a(WS_k)$ and $Rep_i(WS_k)$ are the assessed reputation score and the ideal reputation score of the service $WS_k$ respectively, and N is the number of services. F-Measure is the second used metric. It is a representative metric of the decision-support accuracy measure. *FMeasure* (or F-score) combines *precision* and *recall* metrics into a single value that determines how effectively the reputation measurement model assesses precisely the reputation of services, and handles correctly subjectivity and maliciousness of users' activity. First, Precision is defined in this context via *Normalized Mean Absolute Error* (NMAE) as follows:

$$precision = 1 - NMAE \tag{14}$$

where

$$NMAE = \frac{MAE}{r_{max} - r_{min}} \tag{15}$$

where $r_{max}$ and $r_{min}$ are the maximum and minimum rates respectively. *Precision* values range between 0 and 1, and decrease with the increase of NMAE. *Recall* is defined as the ratio of correctly evaluated services denoted $Nbr_{cs}$ to total number of services ($N$). *Recall* represents the probability that a service reputation is correctly evaluated. It is defined as follows:

$$Recall = \frac{Nbr_{cs}}{N} \tag{16}$$

F-Measure is defined based on precision and recall as follows:

$$F\text{-}Measure = \frac{2 \times recall \times precision}{recall + precision} \tag{17}$$

### *4.2. Reputation with maliciousness density variation*

The ultimate goal of this approach is to accurately assess the reputation of web services, even with the presence of a considerable number of unfair feedback ratings collected from malicious users. Thus, we present the results of the first simulation runs where we variate the number of malicious users. We have fixed simulation parameters to the values listed in Table 5, with a variation of maliciousness density. This density represents the percentage of malicious users in the system.

Figure 4 shows the ideal versus assessed reputation values obtained with three instances of maliciousness density (labeled 25%, 70% and 95%). In this figure, plots (a) through (e) are associated to the five web service classes described previously. For each class, we obtain the plotted reputation values as the average of its service reputation scores (100 service per class). Yet, each service reputation score is the average of 10 simulation-round values.

From the figure, we can see that with 25% of maliciousness density, the assessed reputation values in the five classes are almost equal to the ideal reputation values (showed by the solid line). These results are explained by the fact that honest users

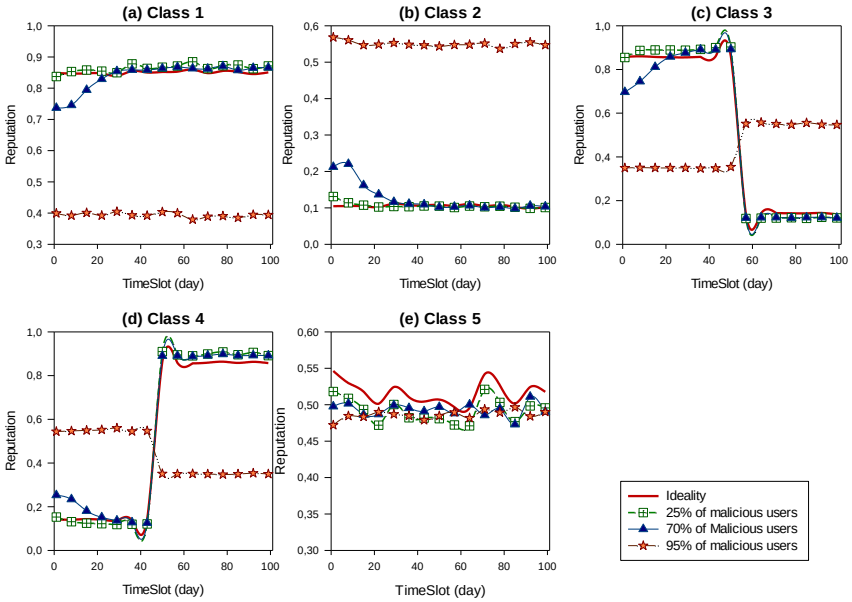


Figure 4: Ideal and assessed reputations with 25%, 70% and 95% of malicious user density

outnumber dishonest users, which permitted a successful user credibility evaluation and reputation assessment. The slight deviation occurred between the assessed and ideal reputation is natural, because it reflects the differences in opinions between honest users.

Second, we observe that, with malicious user density that equates to 70%, the assessed reputation is in decrease deviation from the ideal reputation in the first third of evaluation period (for time slots 1 to 30). Then, for the rest of the evaluation period, the assessed reputation becomes fairly and close to the ideal reputation (i.e., the assessed reputations converge to the ideal values). We note here that dishonest users outnumber honest users, which influences negatively the number of fair feedback ratings, the evaluation of user credibility, and hence the assessed reputation at the beginning of the evaluation. However, by the accumulation of fair and unfair feedback ratings, the model becomes able to distinguish between malicious and honest users based on their credibility values. Therefore, the system neutralizes the effect of unfair feedback ratings on the assessed reputation by punishing suspicious users with a credibility lower than 0.5.

Third, the experiment with a malicious density which equals to 95% shows that the assessed reputations are significantly deviating from the original reputation values. The model is unable to assess correctly reputation due to the very high number of malicious users. Fortunately, as stated in many works in the literature, like Whitby *et al.* [29] and Malik *et al.* [23], such high malicious user density in real world is unrealistic and much lower rates should be expected.

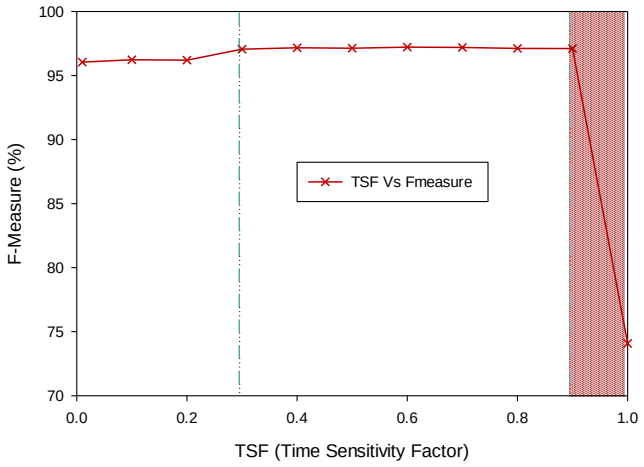


Figure 5: Effect of Time Sensitivity Factor on the F-Measure

Subsequently, We may conclude that the proposed assessment model is able to accurately assess the reputation of web services even with the presence of high malicious rates (up to 70%) in the reputation management system.

### *4.3. Impact of time sensitivity factor*

In the second instance of simulation runs, we have studied the impact of the Time Sensitivity Factor ($\lambda$) on the performance of the proposed assessment model. We have varied the value of $\lambda$ from 0.1 to 1 with a step value of 0.1. We fixed the number of time slots to 1000 ($\#days = 1000$).

Figure 5 depicts the effect of $\lambda$ on the global F-Measure. Values of global F-Measure are geometric means of the F-measure values of the five classes, which are assessed using Eq.17. Note that all used scores are the mean of 10 simulation round values.

The figure shows that: (i) F-measure is slightly increased when $\lambda$ varies from 0.1 to 0.3. (ii) F-measure is steady with the top value when $\lambda$ varies from 0.3 to 0.9. (iii) However, F-measure considerably decreases when $\lambda$ varies from 0.9 to 1. From these observations, we can conclude that the best performance of our reputation assessment model is when $\lambda$ ranges in the interval of [0.3, 0.9].

### *4.4. Effect of the punishment mechanism*

In the third instance of simulation runs, we investigated how the punishment mechanism affects the assessment of reputation scores. We fixed the simulation parameter with the values listed in Table 5, and varying maliciousness density from 0% to 100%. In each run, we assess service reputation scores in two manners: (i) by applying the punishment mechanism, and (ii) without applying this punishment mechanism. Figure 6 shows an F-Measure comparison between the obtained results. From this figure, we observe the following.

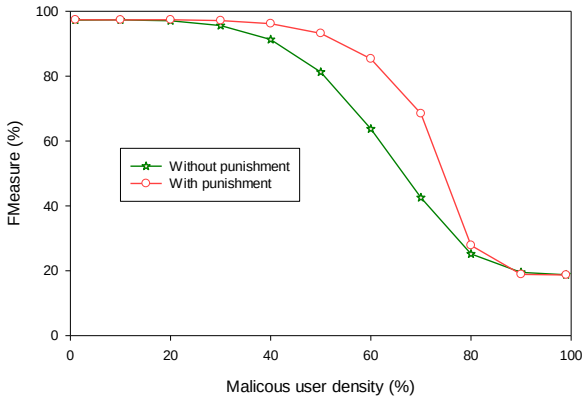


Figure 6: Effect of punishment mechanism on F-Measure performances

First, when malicious user density varies from 0% to 20%, the performance of the model with the application of the punishment mechanism is approximately equal to the performance of the model without the application of this mechanism. This is explained by the low impact of unfair feedback ratings (low number of malicious users) on the assessed reputation scores (i.e., the value of unfair feedback ratings multiplied by their user credibilities is neglected in comparison to the value of fair feedback ratings multiplied by their user credibilities).

Second, when maliciousness density varies from 20% to 90%, the performance of the model with the punishment application is enhanced (up to 20% when malicious density is in the interval [50%-70%]) relatively to the performance of the model without the punishment application. Thus, the model neutralizes effectively the impact of unfair feedback ratings from the assessed reputation scores.

Third, when maliciousness density is greater than 90%, the model with and without punishment application is unable to effectively assess the web service reputation scores due to the important number of malicious users, thus the number of unfair feedback ratings. Fortunately, as stated before, these malicious densities are unrealistic and much lower densities are expected in real-world.

Finally, we can safely draw the conclusion that the application of the punishment mechanism increases the performance of the proposed reputation assessment model.

### *4.5. Performance comparison*

In this section, we compare the performance of the proposed reputation assessment model with the three following reputation assessment approaches:

1- The normal approach (labeled **Normal**), where the reputation is assessed as the average of the collected feedback ratings, without considering any other metric such as the time sensitivity factor or the user credibility factor.

2- The approach proposed by Wang *et al.* [3] (labeled **Wang et al.**), where the

reputation score $q(s_j)$ of service $S_j$ is assessed as follows:

$$q(s_j) = \frac{1}{n}\sum_{i=1}^{n} r_i$$

where $r_i$ represents the i-th feedback rating, n (n=1,2, ...) is the number of feedback ratings. Note that the approach assesses reputation values using only pure feedback ratings (fair ratings or adjusted malicious ratings), because the approach applies a malicious feedback ratings prevention scheme based on the Cumulative Sum method (CUSUM). CUSUM monitors $n$ feedback ratings sample interval. For each sample interval, they assign a score $Z(y_i)$ which is assessed as follows:

$$Z(y_i) = \frac{\mu_1 - \mu_0}{\sigma^2}(y_i - \frac{\mu_1 - \mu_0}{2})$$

where, rating feedback sample intervals are represented by $\{y_1, y_2, ...\}$ and the variable $y_i (y_i = \sum_{i=1}^{m} r_i)$ $(i \leq j \leq n)$ $(m = 1, 2, ...)$, and $\mu_0$ and $\mu_1$ are the mean feedback rating traffic before and after the change. When a sample interval is available, the CUSUM $f_i$ is updated as follows:

$$f_i = \max(f_{i-1} + Z(y_i), 0)$$

if $f_i \geq h$ then a positive shift occurs in the n-th sample which means that there is an abnormal detection point (presence of malicious feedback rating). In our implementation of this scheme, we set $h$ to 0.7 based on the author's experiment settings.

3- The approach proposed by Mekouar *et al.* [30] (**labeled TrustWS**), where the reputation of a web service is assessed as the difference between positive and negative feedback ratings divided by the sum of both. Reputation is set to 0 when the sum of feedback ratings is equal to 0. This approach does not include neither time sensitivity factor nor the credibility of users for reputation assessment.

*4.5.1. Results of the Comparison*

We present in Figure 7 a comparison between the reputation scores assessed by our model and reputations scores obtained by the three approaches cited above, using 25 %as malicious user density parameter. The reference baseline of this comparison is the ideal reputation scores that are presented by a simple solid line in the different plots (a-e).

From the figure, we can see that our reputation scores are closer to the ideal reputation than scores obtained by other approaches, for the five different service classes.

We notice also that reputation scores assessed by Wang's approach are also steady and fairly close to the ideal reputation scores for the first, second and the fifth classes of services. However, the prevention scheme considers fair ratings received after an abrupt change of service QoS as malicious feedback rating, since they produce a positive shift detected by CUSUM. Therefore, the approach shows an insignificant deviation from the ideal reputation in the third and fourth service classes, during the second half of evaluation period as depicted in graph (c) and (d). This limitation is highlighted by the same authors in their paper [3] (Section 5.4).

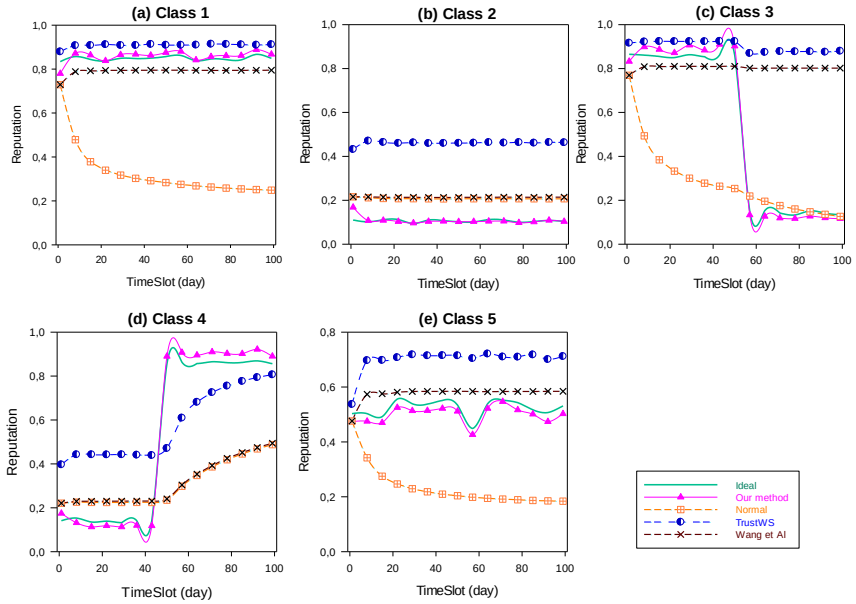


Figure 7: Ideal and compared assessed reputations with 25% of malicious user density

Moreover, TrustWS successfully assesses the reputation of services that maintain steady high QoS as depicted in Figure 7-(a), in the first half of graph (c), and the second half of graph (d).

In addition, reputation values assessed by the normal method converge to the ideal reputation for services with steady low QoS as we can see in graph (b). Unfortunately, the non-inclusion of credibility and time sensitivity factors affects the performance of TrustWS and the normal method to assess correctly reputation scores in the other cases.

*4.5.2. Performances comparison with alteration of malicious user density*

For further performance comparison between the four approaches, we conducted other simulation runs, using Table 5 parameters and varying the density of malicious users. Each experiment is run 10 times and the averages of MEA and F-Measure are measured. For better visibility, we plotted MEA and F-Measure results respectively in Figures 8 and 9. The results show the following:

- Under different settings, our reputation assessment model obtains the smallest MAE and the highest F-Measure values (up to 97%) consistently, within the interval [0%, 70%] of malicious user density. These results indicate a better accuracy of the reputation assessment.

- The approach of Wang *et al.* obtains smaller MAE and high F-Measure values when services maintain stable QoS (i.e., reasonable deviations are included). However, reputation scores diverge from the ideal values when service QoS is

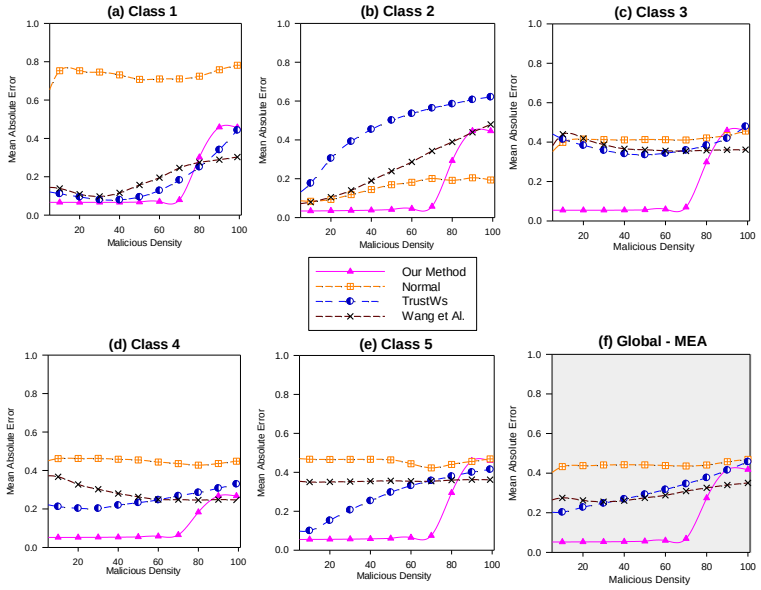


Figure 8: Mean Absolute Error comparison with the alteration of malicious users' density (a smaller MAE means a better performance)

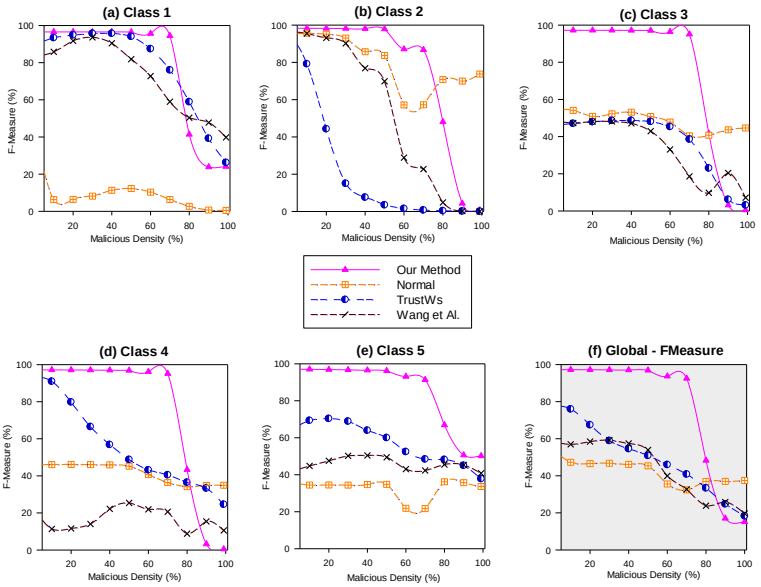


Figure 9: F-Measure performance comparison with the alteration of malicious users density

quickly and significantly upgraded or degraded (services suddenly change from good to bad or from bad to good). This divergence is presented by the increase of MAE and the decrease of F-Measure, such as the case with the services of classes 3 and 4. Hence, the global MEA and the global F-Measure are negatively influenced by this divergence, as it is depicted, respectively, in Figures 8 -(f) and 9 -(f).

- Even if it does not include credibility and time sensitivity factors, TrustWS measures accurately the reputation of services in the first class (services with consistent high QoS), as it is presented by high F-Measure scores in Figure 9-(a). Nevertheless, for the same reason, it fails to assess accurate reputation scores for the remaining classes, as it is shown in plots (b-f) in Figure 9.

- Likewise to TrustWS, the normal approach assesses correctly the reputation of the services of the second class (i.e., services with steady low QoS), as the associated MAE and F-Measure indicate in the figures. However, this naive method fails for the other classes.

- Indeed, the proposed method do not perform well when malicious densities reaches high densities (i.e. 75% or more of users are malicious). Mainly, this is due to the majority voting principle based on which the credibility of users are evaluated. So, when the majority are malicious in a system, obviously it becomes difficult to detect the honest users by trying to group them based on positive and negative rating that they provide. Fortunately, Whitby et al. [29] and Malik et al. [23] claim that high maliciousness densities are unrealistic in real world applications. Therefore, the method that we propose can safely be used in real world scenarios where maliciousness densities are below 70%.

### *4.6. Limitations & Threats to Validity*

The experiment shows that the proposed reputation assessment model provides an accurate measurement of service reputation. Nonetheless, few limitations can be highlighted:

- Our model is not adequate for assessing the reputation of old services with a very low number of feedback ratings, or services with a long discontinuous feedback rating. The more feedback ratings the system collects, the more accurate results the system provides;

- In case of open-systems where users do not login to the system, the use of IP addresses as a mechanism to identify users impose a problem when dynamic IP addresses are used, *i.e.* feedbacks sent by the same user with different IP addresses are not recognized to belong to the same user, and hence, threats to evaluate user credibilities and reputation scores raise;

- To guarantee the performance of the system, it is recommended that the system starts by a collection of fair feedback ratings.

## 5. Related work

Reputation and trust mechanisms have been a topic of interest in many fields, including networking [31, 32], edge computing [33], electronic negotiation and E-commerce systems [34–36]. In this section, we present an overview of the related work in the context of Web and cloud services. Comprehensive literature reviews about the topic are available in [37–40].

Dou et.al [10] presented a distributed trust evaluation protocol for Inter-Cloud ecosystems. In this work, feedback is protected by homomorphic encryption with verifiable secret key sharing. Second, to cater to the dynamic nature of Inter-Cloud, trust evaluation can be conducted in a distributed manner and is operational even when some parties are offline. Third, to facilitate customized trust evaluation, a mechanism is used to store feedbacks, such that it can be processed flexibly while protecting feedback privacy. Though this approach can be effective in evaluating trust, anonymizing users makes the evaluation of user credibility an impossible task. This approach assumes that all users are honest.

Malik and Bouguettaya proposed RateWeb [23]: a decentralized reputation system for web service orchestrations. The proposed architecture is based on a peer to peer (P2P) service model where each peer (service) is a consumer and a provider of services in the same time. RateWeb uses an ontology-based community model. The framework takes into account the presence of malicious raters that may exhibit oscillating honest and dishonest behavior. The system can thus consider and give more weights to the newest service rates. One of the strengths of this approach is the inclusion of many factors for the assessment of reputation. However, this solution do not consider a central reputation management entity that provides a unified assessment of reputation. That is, each peer in the system is responsible for collecting, updating and calculating the reputation of other peers. Such a system is thereby difficult to implement as a recommender system indexing thousands of stateless services.

Wang *et al.* propose a reputation measurement and malicious feedback rating prevention approach for web service recommendation systems [3]. The goal of this approach is to reduce the deviation of reputation measurement of web services, and to improve the success ratio of the service recommendation. The approach passes through two phases before computing services' reputation scores: the malicious feedback rating detection phase and the feedback rating adjustment phase. In the first phase, the authors apply the Commutative Sum Method (called CUSUM) to detect all the malicious collected feedback ratings. In the second phase, the authors deal with the computation of feedback similarity between different users using the Pearson Correlation Coefficient to adjust the feedback ratings. the authors propose a Bloom filter-based prevention scheme to identify the IP addresses with offending feedback ratings and filter them out. The approach considers that new and old feedback ratings have the same influence on the reputation assessment, but in real scenarios, service performances are in dynamic changes.

Xu et al [41] propose a selection framework of cloud services based on a user reputation perspective. Their goal is to identify reliable users in order to provide service selection with reliable QoS information. This model makes a strong assumption that each user has invoked the services and observed the QoS data. The user reputation is

then calculated based on the difference among the QoS data of other users. Unfortunately, such strong assumption that each user monitors all the services that she/he uses and reports data, is not applicable in real world scenarios, but more likely, users send one feedback rating that expresses their opinion.

Su et al [42] propose a trust-aware prediction approach for service recommendation. They start by calculating the reputation of users based on clustering and a beta distribution approach. Then, they identify similar services using the computed reputation values and similarity measures. Finally, the information of trustworthy similar users and similar services is then fully utilized to make accurate QoS prediction. The authors report that their method is accurate, however in case of matrix sparsity, the PCC (Pearson correlation Coefficient) used for similarity assessment may not give good results. In addition, the age of rating, which is an important factor, is not included in the evaluation process.

Li et al [11] propose a framework to enhance the security of cloud-based IoT systems. They assess the trustworthiness of cloud services based on a combination of reputation and security metrics. The framework employs an objective weight assignment approach to assign the respective relative importance weight factors to the security level and reputation level. Then it aggregates them in order to obtain the quantitative trustworthiness of cloud services. Reputation values in this approach are the aggregation of feedback ratings derived from cloud service providers. This approach assumes that security related information is available to be included in the evaluation process.

In [41], Xu et al. proposed MeURep, an approach that evaluates the reputation of users based on the historical QoS data provided by the users in personalized cloud services. The authors present two algorithms that consider that if the QoS data provided by a user is very different from the median, then this user is probably not reliable. The approach deals with reputation to measure QoS and not for ranking services.

The authors in [4] propose a solution for reputation escalation and unfair reputation detection by providing a solution for distinguishing between fake and trustworthy ratings. The solution is based on a randomized algorithm that approximates iteratively the mean ratings for each product (service) with high probability of confidence and thus with very low probability of randomly choosing unfair rating in the calculation of reputation. Despite the obtained accuracy in detecting unfair ratings, the approach does not show how to bootstrap the reputation of newcomers nor distinguish between old and recent ratings.

Ramos and Boratto in [43] divide users into classes based on the demographic attributes that define them and introduce the concept of disparate reputation (DR), capturing if users belonging to different classes are given systematically lower/higher reputation values. The authors propose an algorithm that ensures that reputation is independent of users sensitive attributes, and additionally they propose a step to introduce reputation independence that may be included in any ranking system which computes rankings as a weighted average of ratings. The approach heavily depends on demographic relationships between users, and thus it does not support bootstrapping new services.

Saude et al. [7] propose a reputation-based ranking system that uses multi-partite rating subnetworks. The proposed method clusters users by their similarities using three measures, two of them based on Kolmogorov complexity. The authors focus on

the resistance of user ranking to bribery and targeting to find optimal bribing strategies. The author propose to reflect the diversity of preferences by assigning different rankings to the same item in different groups of users. Based on the experiment, the solution is robust to different security attacks. This method calculates user scores through a weighted average instead of using majority voting and do not propose an efficient way to bootstrap the reputation of unranked items.

The authors in [5] investigate the existing unfair rating detection models by applying them on realistic application settings in an interactive manner. The process of the unfair ratings' detection is conducted by involving the interactions between the application system designer and these models. The proposed Customized Interactive System (CIS) allows an application designer to interact with these unfair rating detection models in one of the following five aspects: customizing scene, customizing attack, customizing model, customizing metrics and customizing result presentation. The CIS architecture consists of three layers: data layer, logical layer and interactive layer. After a series of interactions, the application designer is able to make a wise choice of the detection model with the appropriate parameters.

From a practical point of view, the most available information that exists in online service-based systems for calculating trust are ratings in the form of numerical values or textual raw data, users rating history records and time stamps of the recorded ratings. Thus, in this work, we proposed to leverage the available information to obtain an effective reputation evaluation. Our model includes a punishment mechanism of suspicious users to ensure an efficient reputation measurement from pure feedback ratings. This mechanism is derived from the evaluation of raters credibility using a majority voting principle and by distinguishing between positive and negative feedback ratings. The model also supports time sensitivity factor and use the credibility factor for weighting ratings during the assessment of reputation. A bootstrapping of newcomers' reputation is also considered in our work. It is based on the assumption that QoS scores are in correlation with reputation and thus a multiple regression model built from similar services can predict the reputation of the newcomer given its QoS values. Finally, the approach is a centralized solution which is appropriate for building recommendation systems for real-world services.

## 6. Conclusion

In this paper, we presented a reputation management framework for service recommendation. We focused in particular on presenting the mathematical model for assessing reputation from user feedback ratings. A mechanism of suspicious user punishment has been included in the model to ensure the assessment of reputation from fair feedback ratings. In addition, the model takes into consideration time sensitivity, where old user ratings are given less importance in the reputation assessment. Moreover, we introduced a new bootstrapping technique for estimating the initial reputation of newcomer services based on service similarity and initial advertised QoS. We reported simulation-based experiments, demonstrating the performance and effectiveness of the proposed reputation assessment model.

As a future work, we plan to complete the model with a machine/deep learning algorithm for reputation and QoS prediction, based on sentiment analysis in textual

user feedbacks. Besides, we will focus on the study of trust and reputation of cloud services where significant challenges have to be addressed, due to the highly dynamic, distributed and nontransparent nature of these services.

## References


[1] N. Limam, R. Boutaba, Assessing software service quality and trustworthiness at selection time, IEEE Transactions on Software Engineering, 36 (4) (2010) 559–574.

[2] M. Daaji, A. Ouni, M. M. Gammoudi, S. Bouktif, M. W. Mkaouer, Multi-criteria Web Services Selection: Balancing the Quality of Design and Quality of Service, ACM Transactions on Internet Technology (TOIT) 22 (1) (2021) 1–31.

[3] S. Wang, Z. Zheng, Z. Wu, M. Lyu, F. Yang, Reputation Measurement and Malicious Feedback Rating Prevention in Web Service Recommendation Systems, IEEE Transactions on Services Computing PP (99) (2014) 1–1, ISSN 1939-1374, doi:\bibinfo{doi}{10.1109/TSC.2014.2320262}.

[4] M. Rezvani, M. Rezvani, A randomized reputation system in the presence of unfair ratings, ACM Transactions on Management Information Systems (TMIS) 11 (1) (2020) 1–16.

[5] K. Lv, Y. Liu, J. Chen, D. Wang, Z. Tian, An Interactive System for Unfair Rating Detection Models in a Customized Perspective, in: 2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), IEEE, 871–878, 2021.

[6] A. Alqwadri, M. Azzeh, F. Almasalha, Application of machine learning for online reputation systems, International Journal of Automation and Computing 18 (3) (2021) 492–502.

[7] J. Saúde, G. Ramos, L. Boratto, C. Caleiro, A Robust Reputation-Based Group Ranking System and Its Resistance to Bribery, ACM Transactions on Knowledge Discovery from Data (TKDD) 16 (2) (2021) 1–35.

[8] R.-H. Li, J. Xu Yu, X. Huang, H. Cheng, Robust reputation-based ranking on bipartite rating networks, in: Proceedings of the 2012 SIAM international conference on data mining, SIAM, 612–623, 2012.

[9] O. A. Wahab, R. Cohen, J. Bentahar, H. Otrok, A. Mourad, G. Rjoub, An endorsement-based trust bootstrapping approach for newcomer cloud services, Information Sciences 527 (2020) 159–175.

[10] Y. Dou, H. C. Chan, M. H. Au, A distributed trust evaluation protocol with privacy protection for intercloud, IEEE Transactions on Parallel and Distributed Systems 30 (6) (2018) 1208–1221.

[11] X. Li, Q. Wang, X. Lan, X. Chen, N. Zhang, D. Chen, Enhancing cloud-based IoT security through trustworthy cloud service: An integration of security and reputation approach, IEEE Access 7 (2019) 9368–9383.

[12] Q. Wu, Q. Zhu, P. Li, A neural network based reputation bootstrapping approach for service selection, Enterprise Information Systems 9 (7) (2015) 768–784.

[13] Z. Malik, A. Bouguettaya, Reputation bootstrapping for trust establishment among web services, Internet Computing, IEEE 13 (1) (2009) 40–47.

[14] C. E. Player, N. Griffiths, Bootstrapping trust and stereotypes with tags, in: Proceedings of the 19th International Workshop on Trust in Agent Societies (Trust at AAMAS), 2017.

[15] V. L. Hallappanavar, C. M. Bulla, M. N. Birje, ANN based estimation of reputation of newcomer web services in fog computing, in: 2021 International Conference on Computer Communication and Informatics (ICCCI), IEEE, 1–7, 2021.

[16] S. Mistry, A. Bouguettaya, Reputation Bootstrapping for Composite Services using CP-nets, IEEE Transactions on Services Computing .

[17] L. Qu, A. Bouguettaya, A. G. Neiat, Confidence-aware reputation bootstrapping in composite service environments, in: International Conference on Service-Oriented Computing, Springer, 158–174, 2017.

[18] S. Mistry, L. Qu, A. Bouguettaya, Layer-based Composite Reputation Bootstrapping, ACM Transactions on Internet Technology (TOIT) 22 (1) (2021) 1–28.

[19] Z. Azmeh, J.-R. Falleri, M. Huchard, C. Tibermacine, Automatic web service tagging using machine learning and wordnet synsets, in: Web Information Systems and Technologies, Springer, 46–59, 2011.

[20] J.-R. Falleri, Z. Azmeh, M. Huchard, C. Tibermacine, et al., Automatic tag identification in web service descriptions, in: WEBIST'10: The International Conference on Web Information Systems and Technology, 2010.

[21] O. Tibermacine, C. Tibermacine, F. Cherif, A Practical Approach to the Measurement of Similarity between WSDL-based Web Services, Revue des Nouvelles Technologies de l'Information 6th French-speaking Conference on Software Architectures, RNTI-L-7 (2014) 03–18.

[22] Z. Xu, P. Martin, W. Powley, F. Zulkernine, Reputation-enhanced qos-based web services discovery, in: IEEE International Conference on Web Services (ICWS 2007), IEEE, 249–256, 2007.

[23] Z. Malik, A. Bouguettaya, Rateweb: Reputation assessment for trust establishment among web services, The VLDB JournalThe International Journal on Very Large Data Bases 18 (4) (2009) 885–911.

[24] L. Mekouar, Y. Iraqi, R. Boutaba, Incorporating trust in network virtualization, in: 2010 IEEE 10th International Conference on Computer and Information Technology (CIT), IEEE, 942–947, 2010.

[25] J. R. Douceur, The sybil attack, in: Peer-to-peer Systems, Springer, 251–260, 2002.

[26] O. Tibermacine, C. Tibermacine, F. Cherif, Estimating the reputation of newcomer web services using a regression-Based method, Journal of Systems and Software 145 (2018) 112 – 124, ISSN 0164-1212, doi:\bibinfo{doi}{https://doi.org/10.1016/j.jss.2018.08.026}.

[27] J. L. Herlocker, J. A. Konstan, L. G. Terveen, J. T. Riedl, Evaluating collaborative filtering recommender systems, ACM Transactions on Information Systems (TOIS) 22 (1) (2004) 5–53.

[28] J. Cao, Z. Wu, Y. Wang, Y. Zhuang, Hybrid Collaborative Filtering algorithm for bidirectional Web service recommendation, Knowledge and information systems 36 (3) (2013) 607–627.

[29] A. Whitby, A. Jøsang, J. Indulska, Filtering out unfair ratings in bayesian reputation systems, in: Proc. 7th Int. Workshop on Trust in Agent Societies, vol. 6, 2004.

[30] L. Mekouar, Y. Iraqi, TrustWS: A Trust Management System for Web Services, in: International Symposium on Web Services, At Dubai, UAE, 2010.

[31] J. M. J. Valero, P. M. S. Sánchez, M. G. Pérez, A. H. Celdrán, G. M. Pérez, Toward pre-standardization of reputation-based trust models beyond 5G, Computer Standards & Interfaces 81 (2022) 103596.

[32] Y. Ouyang, Z. Zeng, X. Li, T. Wang, X. Liu, A verifiable trust evaluation mechanism for ultra-reliable applications in 5G and beyond networks, Computer Standards & Interfaces 77 (2021) 103519.

[33] W. Zheng, B. Chen, D. He, An adaptive access control scheme based on trust degrees for edge computing, Computer Standards & Interfaces 82 (2022) 103640.

[34] O. Tafreschi, D. Mähler, J. Fengel, M. Rebstock, C. Eckert, A reputation system for electronic negotiations, Computer standards & interfaces 30 (6) (2008) 351–360.

[35] F. G. Mármol, G. M. Pérez, Towards pre-standardization of trust and reputation models for distributed and heterogeneous systems, Computer Standards & Interfaces 32 (4) (2010) 185–196.

[36] Y.-Y. Chang, S.-C. Lin, D. C. Yen, J.-W. Hung, The trust model of enterprise purchasing for B2B e-marketplaces, Computer Standards & Interfaces 70 (2020) 103422.

[37] D. D. S. Braga, M. Niemann, B. Hellingrath, F. B. D. L. Neto, Survey on computational trust and reputation models, ACM Computing Surveys (CSUR) 51 (5) (2018) 1–40.

[38] M. Chiregi, N. J. Navimipour, A comprehensive study of the trust evaluation mechanisms in the cloud computing, Journal of Service Science Research 9 (1) (2017) 1–30.

[39] F. G. Mármol, M. Q. Kuhnen, Reputation-based Web service orchestration in cloud computing: A survey, Concurrency and Computation: Practice and Experience .

[40] M. Nikravan, M. H. Kashani, A review on trust management in fog/edge computing: Techniques, trends, and challenges, Journal of Network and Computer Applications (2022) 103402.

[41] J. Xu, X. Du, W. Cai, C. Zhu, Y. Chen, MeURep: A novel user reputation calculation approach in personalized cloud services, PloS one 14 (6).

[42] K. Su, B. Xiao, B. Liu, H. Zhang, Z. Zhang, TAP: A personalized trust-aware QoS prediction approach for web service recommendation, Knowledge-Based Systems 115 (2017) 55–65.

[43] G. Ramos, L. Boratto, M. Marras, Robust Reputation Independence in Ranking Systems for Multiple Sensitive Attributes, arXiv preprint arXiv:2203.16663 .