Assessing Software Service Quality and Trustworthiness at Selection Time

Noura Limam and Raouf Boutaba, *Senior Member*, *IEEE*

Abstract—The integration of external software in project development is challenging and risky, notably because the execution quality of the software and the trustworthiness of the software provider may be unknown at integration time. This is a timely problem and of increasing importance with the advent of the SaaS model of service delivery. Therefore, in choosing the SaaS service to utilize, project managers must identify and evaluate the level of risk associated with each candidate. Trust is commonly assessed through reputation systems; however, existing systems rely on ratings provided by consumers. This raises numerous issues involving the subjectivity and unfairness of the service ratings. This paper describes a framework for reputation-aware software service selection and rating. A selection algorithm is devised for service recommendation, providing SaaS consumers with the best possible choices based on quality, cost, and trust. An automated rating model, based on the expectancy-disconfirmation theory from market science, is also defined to overcome feedback subjectivity issues. The proposed rating and selection models are validated through simulations, demonstrating that the system can effectively capture service behavior and recommend the best possible choices.

Index Terms—Software as a service (SaaS), software selection, service utility, review and rating, trust and reputation, risk management, SLA monitoring.

1 Introduction

Growing competition within the IT industry has created a strong incentive for developing solutions to support more agile and more competitive businesses. The long-term success of commercial off-the-shelf (COTS) software as a time-effective alternative to custom "in-house" developed solutions is still being compromised by the involved cost of ownership, installation and maintenance time, and effort. Therefore, the IT industry has started to move toward a new model for software delivery—one that is easy to deploy, maintenance-free, and cost-effective.

The Software as a Service (SaaS) model, where software is delivered on-demand and priced on-use, has been made possible by the widespread adoption of fast Internet access, combined with the widespread acceptance of SOA-based solutions. By reducing the cost of ownership and alleviating the burden of software installation and maintenance, SaaS has gained popularity in recent years. As enterprises have started to outsource some of their software infrastructure and development projects to SaaS vendors, the number of SaaS offerings has expanded dramatically, even among vendors of traditional on-premises software.

However, integrating outsourced software into project development can be challenging or even risky. In particular, the performance or quality of the external software may not be satisfactory at the time of execution. SaaS somewhat lowers this risk due to its on-use pricing. While traditionally acquired COTS may be prohibitively expensive to replace despite unsatisfactory performance, the SaaS model provides consumers with a looser, more flexible relationship to software or service providers. To some extent, SaaS provides a low-risk alternative to large investments. Nevertheless, the success of SaaS integration depends on the behavior of the provider. Since the software is being delivered as a service, it is hosted at, and maintained by, the provider, leaving the consumer with a low degree of control on its performance. As long as the service provider fulfills its obligations to the consumer—provides the needed support, undertakes the required management and maintenance tasks, and generally behaves well—then the risks of failure remain low. However, the behavior of service providers is unknown until the service is rendered. The risk of bad behavior cannot be excluded and can have adverse effects on the project outcomes.

An empirical study of the risk factors related to the development using external software (COTS-like) components along with associated risk reduction activities has been reported in [1]. It showed that risk reduction at software selection time is negatively correlated with occurrences of most project development-related risks. In fact, selection must be driven by quality constraints, with selection time evaluation of component quality and choice of appropriate service providers all essential to successful integration. However, in practice, the evaluation of service quality cannot be performed until the service is acquired. Consequently, quality evaluation is typically limited to the evaluation of quality offers by comparing the quality level that providers promise to the quality requirements. Compliance cannot be guaranteed at selection time, so it is essential to choose a provider that is trusted to respect its commitments.

- *N. Limam is with the Division of IT Convergence Engineering, POSTECH—Pohang University of Science and Technology, San31, Hyoja Dong, Nam Gu, Pohang, Gyungbuk 790-894, Republic of South Korea. E-mail: noura.limam@gmail.com.*
- *R. Boutaba is with the D.R. Cheriton School of Computer Science, University of Waterloo, 200 University Avenue West, Waterloo, Ontario, N2L 3G1, Canada. E-mail: rboutaba@uwaterloo.ca.*

Manuscript received 28 Mar. 2008; revised 3 Apr. 2009; accepted 30 Nov. 2009; published online 6 Jan. 2010.
Recommended for acceptance by A. Wolf.
For information on obtaining reprints of this article, please send e-mail to: tse@computer.org, and reference IEEECS Log Number TSE-2008-03-0120.
Digital Object Identifier no. 10.1109/TSE.2010.2.

The trustworthiness of service providers is commonly measured by their reputations. Reputation systems not only record and track providers' behavior, but can create an incentive for good behavior by providing consumers with some control over market quality. However, existing systems tend to rely on customers' ratings of past service experiences. This creates major issues in terms of subjectivity and rating unfairness.

This paper introduces a framework for reputation-aware software service selection and rating. The key characteristic of the proposed framework is to automate both the selection and the rating of software services, in this way not only alleviating a potentially tedious and time-consuming task, but also increasing the objectivity of the service quality reports. The ultimate aim underlying the development of the framework is to reduce at selection time the risk associated with the utilization of external software services in development projects. The service selection algorithm acts as a user-centric and reputation-aware *service recommender* while determining a service's suitability to a particular user's preferences in terms of quality and cost. As opposed to previous works on software service selection where reputation is either neglected [2] or dealt with as any quality parameter [3], service reputation is considered in this work as a predictor for the conformance of service offers to the resulting delivered service quality. In order for a reputation mechanism to be fair and objective, it is essential to compute reputation on the basis of fair and objective feedbacks. While most of the works that addressed this latter issue are on evaluating the fairness of existing feedbacks [4], our work focuses instead on the process of generating objective and fair feedbacks. Grounded on works in the area of Service Level Agreement (SLA) monitoring such as the one in [5], a computational model is provided to objectively evaluate the delivered service based on the actual measurement of the conformance of the execution quality to the contracted SLA. A novel algorithm is also devised to automate the rating process based on the expectancy-disconfirmation theory from market science.

The remainder of the paper is organized as follows: Section 2 discusses the motivations and the contributions of this work. Our automated feedback and reputation computation models are described in Section 3, followed by our automated service selection algorithm in Section 4. Section 5 presents the results of the simulation studies conducted to evaluate the proposed system. Section 6 considers the implications of these results and describes a number of topics for future work. In Section 7, related works are studied and compared to this work. Finally, Section 8 concludes the paper.

2 Motivations and Contributions

Service selection is a multicriteria decision-making problem whose resolution commonly involves a trade-off between quality and cost. As explained before, there is no guarantee of service quality at selection time; however, reputation can help in predicting the likelihood of a quality offer to be met. As a matter of fact, selection can translate into a three-criteria decision-making problem involving reputation, quality, and cost. This problem can be reduced to a single-criterion decision-making problem provided that quality reputation and cost are aggregated into a single selection metric. Although recent literature works agree on the necessity of considering reputation for service selection, there is no general consensus on the role of reputation in decision support. Considering service reputation as the aggregation of "arbitrary" consumers' feedback makes it hard to clearly define what exactly feedback refer to (credibility, reliability, etc.) and what exactly reputation stands for.

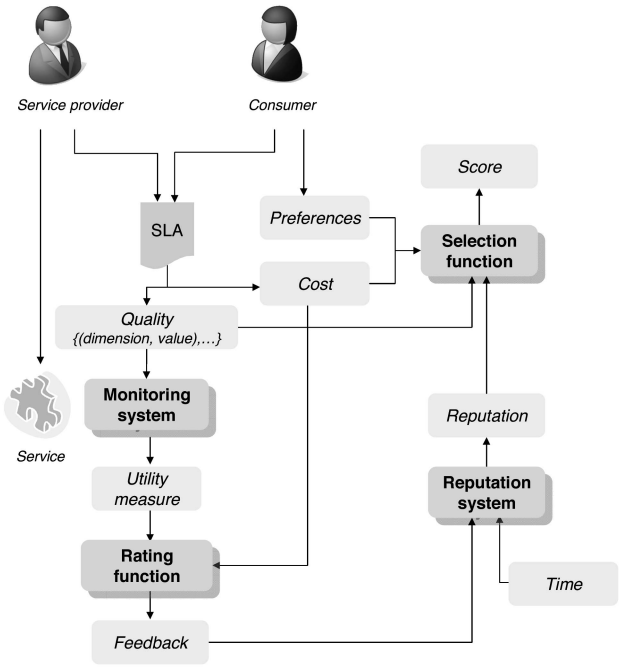

Fig. 1. Automated selection and rating framework.

Ensuring the veracity of reputation reports is also a critical issue. First, feedback can be subjective since it is based on consumers' "personal" expectations and opinions. Second, consumers may have an "obstructed" view of a service and its performance, especially when the latter is part of a composite service. Third, reputation systems are prone to attacks by malicious consumers who may give false ratings and subvert service reputation. Generally, it is harder to maintain a per-consumer reputation system than a per-service reputation system, mainly because services are less versatile, more traceable, and come in a smaller number. Moreover, it is harder to manage user identities—especially for malicious users who are likely to change theirs quite often (e.g., sybil attacks [6]). Finally, consumers may have little incentive to leave feedback; they are often more eager to leave negative feedback when they are dissatisfied with the experienced service than to leave positive feedback when they are satisfied. This introduces a bias against positive ratings and leads to unfair reputation reports. For all of these reasons, the first step toward establishing the foundations of an automated reputation-aware selection framework is to unambiguously define the feedback as a computable nonarbitrary metric and to devise an objective rating system.

As shown in Fig. 1, our contributions range from feedback computation to the derivation of the selection metric, and include reputation calculation. The focus in this work is on software services, contracted with monitorable Service Level

Service Level Agreement

Obligations
- 99.99 % uptime
- 600 ms response time

Compensations
- Percentage of total charges paid by customer

UPTIME (PER 15 MIN. CONTROL)	CREDIT
99.99% - 100.00%	0%
99.98% - 98.00%	5 %
97.99% - 97.00%	10 %
96.99% - 95.00	20%
<95%	50%

RESPONSE TIME (MEAN PER 15 MIN. CONTROL)	CREDIT
0ms - 600 ms	0%
600ms - 633ms	5 %
634ms - 666ms	10 %
666ms - 699ms	15%
>799ms	20%

Fig. 2. Example SLA: small subset of SaaS SLA.

Agreements (SLAs) [7]. Monitorable SLAs are a useful tool that govern consumer-provider relationship. Fig. 2 shows a short version of an SLA (say, for service *GenericSaaS*) as can be found on the Web [8], [9], [10], [11] and in a more formal representation [12]. From the customer's perspective, the SLA introduces a level of accountability and a means to monitor service quality and performance. In this example, two quality parameters are considered: *uptime* and *response time*. The SLA also suggests that service *uptime* and *response time* are estimated each 15-minute period in the billing cycle. From the provider's perspective, the SLA is used to set realistic expectations (commitments in terms of service quality—here *uptime* is set to 99.99 percent and *response time* to 600 ms). SLAs also include methods of compensation should the provider's commitment not be met—in the example, a credit defined as a percentage of total charges paid by the consumer. Indeed, SLAs create an incentive for good behavior among SaaS providers but still do not guarantee any good behavior at service delivery time. For this reason, we devise a proactive strategy for risk reduction at selection time.

We introduce a *rating function* that makes use of quality monitoring results and service cost to produce feedback. Only measurable quality parameters are considered for monitoring. Monitoring is achieved in compliance with the SLA, i.e., the metrics used and the way measurements are achieved must be the same as those described in the SLA. Measurements are assumed to be conducted in a trustworthy manner in order to be credible and accurate. Skene et al. [7] suggest, for instance, the use of monitoring software executing on trusted computing platforms or temper-proof hardware. In addition to being possibly implemented on the consumer's and provider's side, monitoring can also be supported by third parties trusted by both the consumer and the provider. Third parties will have to arbitrate any dispute between the consumer and the provider. Third party monitoring can be supported by the companies that manage service directories since they are likely to be concerned with the quality of the advertised services [13], or any specialized companies hired to achieve QoS monitoring tasks [14]. The trustworthiness of the monitoring system together with our proposed rating function ensures the credibility of feedbacks.

We also propose a *reputation derivation* model that aggregates all of the feedback into an overall rating (i.e., reputation) while taking into account the time factor for a more realistic result. Finally, we devise a *selection function* that derives a single selection metric out of the reputation of the service as provided by the reputation system and the offered quality and cost. In order for the selection metric to be more compatible with the request of the consumer, we allow the latter to define her/his preferences and priorities in terms of the different aspects of the service offer.

3 Automated Rating and Reputation Computation Models

The goal of the rating function is to provide objective feedback on a delivered service without human intervention. We define in the following a *feedback forecasting* model that translates service execution quality into feedback so that any quality monitoring system can be enhanced with such a rating function.

3.1 Feedback versus Satisfaction

In essence, feedback is a measure of a user's satisfaction with the service. Thus, it is necessary that the automated rating process provide feedback that corresponds to the level of satisfaction/dissatisfaction with service delivery. The *expectancy-disconfirmation* theory from market science [15] provides a conceptual framework for the study of consumer satisfaction versus service quality. According to this theory, consumer satisfaction is the outcome of the comparison between consumers' *preconsumption expectation* and *postconsumption disconfirmation*, where confirmed expectations lead to moderate satisfaction, positively disconfirmed (i.e., exceeded) expectations lead to high satisfaction, and negatively disconfirmed (i.e., underachieved) expectations affect satisfaction more strongly than positive disconfirmation and lead to dissatisfaction. This theory has been extensively used to investigate the antecedents and consequences of consumer satisfaction, as well as the inferred relationship with customer retention. As a matter of fact, several customer satisfaction models have been devised in the literature; Xiao and Boutaba [16], for instance, present a network service-oriented customer satisfaction model that links network service quality to customer satisfaction and then to provider revenue and profit.

Existing customer satisfaction forecasting models rely mainly on customer's *expectation* and *perception*, which are inherently subjective concepts. We found our feedback forecasting model on the former models, however, to ensure the objectivity of our model, objective, nonarbitrary, and

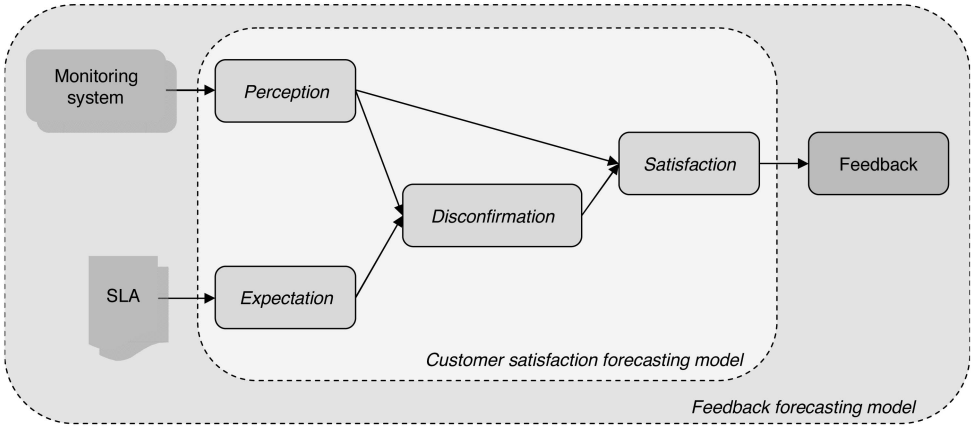

Fig. 3. Feedback forecasting model.

measurable parameters are substituted for the subjective concepts. Namely, SLAs are used to quantify quality expectations and trusted quality measurements (i.e., quality monitoring results) to quantify quality perceptions (see Fig. 3).

3.2 Mathematical Formulation

We use a single scalar metric to quantify quality perception. This metric is basically the *utility function* of the delivered service. Utility expresses the conformance of service execution quality to the agreement. The utility function can be considered as the distribution function of the probability that the observed quality meets the agreed quality level during service execution. Thus, the utility function can be estimated from quality monitoring results.

We denote by υ the utility function of the service. Service quality can be defined as a vector of N dimensions, where N represents the number of quality parameters $QoS_{dim} = Q_1, Q_2, \ldots, Q_N$.

The utility function υ is defined in [17] as a weighted product of the utilities associated with each parameter Q_i. Compared to a weighted mean, which moderates the impact of low utility levels, a weighted product better reflects the intense impact that a failure in a single quality aspect may have on the overall performance of a QoS-sensitive service. For instance, high server response time may cause the client side application to time out before the service is properly rendered. From the consumer's perspective, the impact of a high response time (i.e., low levels in the *response-time*-related utility) should not be moderated by the permanent availability of the service (i.e., high levels in the *uptime*-related utility) as it leads to the failure of the service from the consumer's perspective.

υ is expressed as follows:

$$\upsilon = \prod_{Q_i \in QoS_{dim}} F_{Q_i}^{c_{Q_i}}, \tag{1}$$

where, for each QoS parameter Q_i in QoS_{dim}, F_{Q_i} is a function that gives the utility associated with the parameter Q_i and the weight $c_{Q_i} \in [0,1]$ reflects how much the user cares about the quality parameter Q_i. c_{Q_i} is user specific; in our model, we consider $c_{Q_i} = 1$ for each parameter Q_i. We also define the function F_{Q_i} as the probability for a measured value of Q_i to meet the quality requirement. For instance, the utility of a *GenericSaaS* is computed as follows: $\upsilon = F_{uptime} * F_{response\text{-}time}$.

3.2.1 Utility Computation

For each quality parameter Q_i, $dom(Q_i)$ denotes the domain of Q_i, $(q_i)_e$ the agreed (expected) value, and q_i the measured (perceived) value of Q_i. We define the function $Accept_i$ as follows:

$$Accept_i : \begin{cases} dom(Q_i) \longrightarrow 0,1 \\ q_i \longmapsto Accept_i(q_i) = \begin{cases} 1 & \text{if } q_i \textit{ better than or} \\ & \textit{equal to } (q_i)_e, \\ 0, & \text{otherwise.} \end{cases} \end{cases} \tag{2}$$

The $Accept_i$ function follows a *Bernoulli distribution*. The probability p_i for a measured q_i to meet the quality requirement $(q_i)_e$ represents the utility of the quality parameter Q_i.

Let n_i be the number of trusted accurate measurements conducted on the quality parameter q_i (i.e., the measurements that conform to an accuracy constraint if any has been agreed upon [7]). For each trusted measurement k, $(q_i)_k$ denotes the measured quality value and $(X_i)_k$ the associated $Accept_i((q_i)_k)$. $(X_i)_1, (X_i)_2, \ldots, (X_i)_{n_i}$ are independent Bernoulli trials, identically distributed with success probability p_i, then

$$Y_i = \sum_{k=1}^{n_i} (X_i)_k \sim \text{Binomial}(n_i, p_i). \tag{3}$$

Under certain conditions, an approximation to $\text{Binomial}(n_i, p_i)$ is given by the normal distribution $\text{Normal}(n_i p_i, n_i p_i (1 - p_i))$. Because the normal approximation is not accurate for small values of n_i, the normal approximation is commonly used with the rule of thumb $n_i p_i > 10$ and $n_i(1 - p_i) > 10$. It is worth noting that the approximation does not apply well if the proportion of successful measurements is too close to 0 or 1 and fails when the proportion is equal to 0 or 1.

Let $\widetilde{p_i}$ be the proportion of successful measurements. $\widetilde{p_i}$ is computed as follows:

$$\widetilde{p_i} = \frac{1}{n_i}\sum_{k=1}^{n_i}(X_i)_k. \quad (4)$$

According to the central limit theorem, $\widetilde{p_i}$ approximates p_i with a standard deviation

$$\sigma = \sqrt{\frac{\widetilde{p_i}(1-\widetilde{p_i})}{n_i}}.$$

p_i is within approximately two standard deviations, i.e., falls in the interval $[\widetilde{p_i} - z_{95\%} * \widetilde{\sigma}, \widetilde{p_i} + z_{95\%} * \widetilde{\sigma}]$ with probability 95 percent.

Let F_{Q_i} be the lowest bound of p_i—considering that rating is part of a risk reduction strategy:

$$F_{Q_i} = \max\left(\widetilde{p_i} - z_{95\%} * \sqrt{\frac{\widetilde{p_i}(1-\widetilde{p_i})}{n_i}}, 0\right). \quad (5)$$

Assume, for example, that a *GenericSaaS* service is contracted with the SLA that appears in Fig. 2, and the *uptime* is monitored and compared against the terms of the SLA. Measurements are conducted repeatedly and the *uptime* of the service is observed during 15 min each time as requested by the SLA. The effective *uptime* is calculated by subtracting from 100 percent the error rate experienced during the 15 min observation period. If no error has been detected, then the *uptime* at the end of the observation period is 100 percent. Assuming, for example, that measurements have been conducted 50 times ($n_{uptime} = 50$), and that no error has been detected the first 49 measurements, i.e., $(X_{uptime})_1 = (X_{uptime})_2 = \cdots = (X_{uptime})_{49} = 1$. Assuming that, during the last 15 min, 10 out of the 50 attempts to connect to the service have returned internal service errors, the service *uptime* at the 50th measurement is 80 percent. This also means that the SLA has been violated within the 50th measurement, i.e., $(X_{uptime})_{50} = 0$. This leads to $\widetilde{p_{uptime}} = 0.98$ and

$$\widetilde{p_{uptime}} - z_{95\%} * \sqrt{\frac{\widetilde{p_{uptime}}(1-\widetilde{p_{uptime}})}{n_u ptime}} = 0.94.$$

$\widetilde{p_{uptime}}$ is too close to 1, and according to the rule of thumb, n_{uptime} is not large enough given that $50 * (1 - 0.94) = 3 < 10$. This means that 0.94 is not a good approximation, therefore, it is more accurate to set F_{uptime} to 0.98.

3.2.2 Feedback Computation

As shown in (6), customer satisfaction $CSAT$ is defined in [16] as a linear combination of a perception function and a disconfirmation function. The former maps the perceived utility to the customer's "baseline" satisfaction, and the perceived disconfirmation to the customer's "referred" satisfaction (i.e., considering customer's expectation as a reference point):

$$CSAT(s) = f_p(v) + f_d(v - v_e), \quad (6)$$

where v denotes the perceived utility, v_e the expected utility, f_p the perception function, and f_d the disconfirmation function. The perception function is described as a concave function considering that the customer is less sensitive to changes in high utility values than to lower ones. The disconfirmation function is expressed as a two-piece convex-linear function grounded on the fact that customer satisfaction increases mildly (linearly) when perceived utility exceeds expectation and decreases significantly (exponentially) when perceived utility falls below expectation.

We describe our feedback function $FEEDBACK(s)$ as an increasing function bounded between 0 and 1 and view it as a combination of a perception function and a disconfirmation function similarly to $CSAT$. According to [16], customer expectation, i.e., v_e, evolves over time depending on the experienced disconfirmation, in the way that positive disconfirmation increases future expectation while negative disconfirmation has the opposite effect. For the sake of objectivity, we consider that expectation is solely governed by the quality agreement and does not evolve over time. As a matter of fact, it is possible to consider v_e as constant; the customer expects the agreement to be respected, thus, $v_e = 1$. This also implies that the perceived utility will not exceed the expected utility; the disconfirmation function is strictly convex.

The feedback function can be formalized as follows:

$$FEEDBACK(s) = f_p(v) + f_d(v-1) = f(v), \quad (7)$$

where f is an increasing function defined in $[0,1]$ and bounded between $f(0) = 0$ and $f(1) = 1$. f should combine the characteristics of both the perception and disconfirmation functions, in particular, the concavity of the perception function and the convexity of the disconfirmation function. We consider that f varies slightly for utility values close to 0 as well as for utility values close to 1. The concavity of f changes at a particular utility value, say, U_0 in $[0,1]$. Those assumptions can be formalized as follows:

$$\begin{cases} f''(U_0) = 0, \\ f''(v) \geq 0 \quad for \quad v \in [0, U_0], \\ f''(v) \leq 0 \quad for \quad v \in [U_0, 1]. \end{cases} \quad (8)$$

Similarly to [16], we adopt a polynomial rate of change for f. The above assumptions lead to the following expression:

$$f''(v) = \mu(U_0 - v) \quad \text{with} \quad \mu \geq 0. \quad (9)$$

After integration, and considering that f is an increasing function with $f(0) = 0$ and $f(1) = 1$, we obtain the following expression:

$$f(v) = -\frac{\mu}{6}v^3 + \frac{\mu U_0}{2}v^2 + \left(1 + \mu\left(\frac{1}{6} - \frac{U_0}{2}\right)\right)v, \quad (10)$$

with μ respecting the following constraints:

$$\begin{cases} 1 + \mu\left(\dfrac{U_0}{2} - \dfrac{1}{3}\right) \geq 0, \\ 1 + \mu\left(\dfrac{1}{6} - \dfrac{U_0}{2}\right) \geq 0. \end{cases} \quad (11)$$

The equations in (11) define the relationship between the parameters μ and U_0. As shown in Fig. 4, U_0 and μ control the shape and curvature, respectively, of the feedback function f. For similar μ values, U_0 delays the convexity of f—that is the sensitivity of the feedback function to variations in utility—to higher utility values. Fig. 4 shows how f turns from strictly concave with $U_0 = 0$ to strictly convex with $U_0 = 1$, with μ set

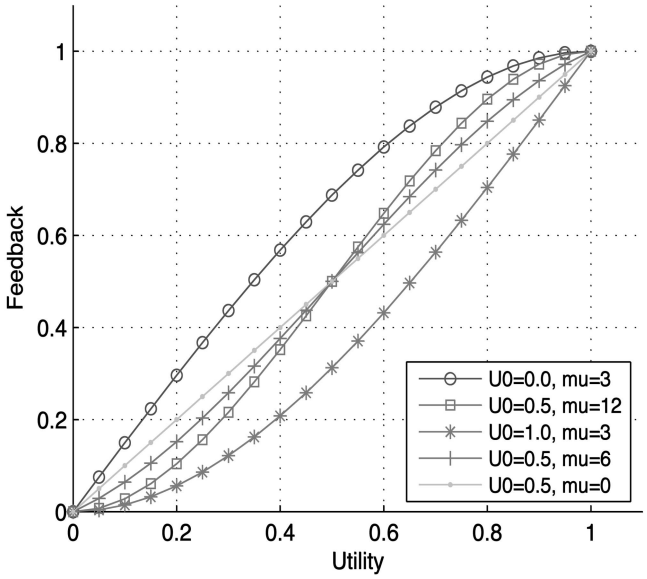

Fig. 4. Impact of U_0 and μ on the feedback function.

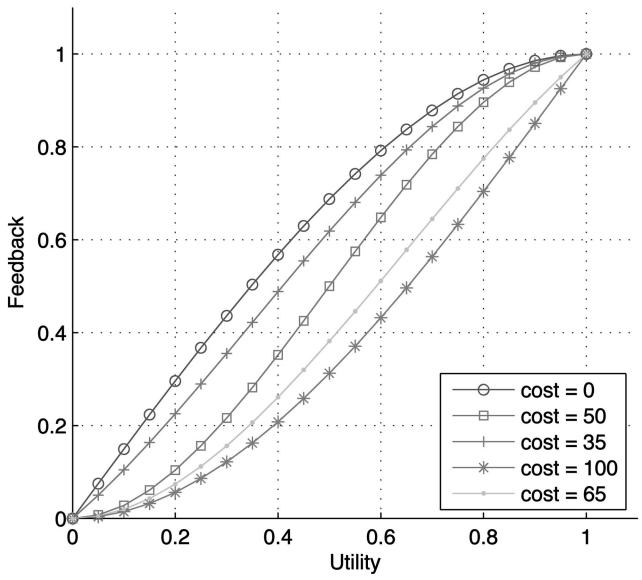

Fig. 5. Impact of cost on the feedback function.

to the same value. On the other hand, for similar U_0 values, μ softens or amplifies the curvature of f, hence, the impact of variations in utility, nearby extreme utility values. When U_0 is set to 0.5, for instance, the curve reaches maximum (respectively, minimum) curvature when μ is set to 12 (respectively, 0). More generally, maximum curvature is attained with μ set as follows:

$$\begin{cases} \mu = \dfrac{6}{2-3U_0} & for \quad U_0 \in \left[0, \dfrac{1}{2}\right], \\ \mu = \dfrac{6}{3U_0-1} & for \quad U_0 \in \left[\dfrac{1}{2}, 1\right]. \end{cases} \tag{12}$$

The impact of U_0 on the feedback function suggests that U_0 and the cost of the service could be inversely related. It is common sense that cost has an impact on customer quality expectation; the higher the cost of the service, the higher the expectation and the more sensitive the customer is to quality disconfirmation. In fact, the customer gets easily dissatisfied with pricey services as soon as the utility drops below the expected value. As opposed to U_0, the higher (respectively, lower) the cost of the service, the more convex (respectively, concave) the feedback function is at nearby high utility values. This suggests that the higher the cost is, the lower U_0 should be, and vice versa.

Let c denote the cost of the service and ϕ the function that relates service cost to U_0. ϕ is a positive increasing function with a value in $[0, 1]$. Let c_{min} be the lowest possible service cost and c_{max} the highest. We can simply define ϕ as follows:

$$\phi : \begin{cases} Dom(c) \longrightarrow [0,1] \\ c \longmapsto \phi(c) = \begin{cases} \dfrac{1}{2} & \text{if} \quad c_{max} - c_{min} = 0, \\ \dfrac{c - c_{min}}{c_{max} - c_{min}}, & \text{else.} \end{cases} \end{cases} \tag{13}$$

This way, $U_0 = 1$ for $c = c_{max}$, and $U_0 = 0$ for $c = c_{min}$. Fig. 5 shows how the shape of the feedback function evolves from convex to concave with decreasing cost values.

In the following, we experiment with our automated rating process and demonstrate that the design requirements have been properly implemented. To this end, we consider 10 service instances S_j, $j = 1..10$. Each comes with a randomly generated cost value between 0 and 100, and an *SLA violation* probability $fail_j$. For each service instance, we simulate 100 quality measurements. Each measurement violates the SLA with the probability $fail_j$. From the measurement results, we estimate service utility v. The probability the SLA is respected $(1 - fail_j)$ is referred to as the *effective utility*, while *estimated utility* refers to v. Finally, we compute the feedback for each service instance as explained at the beginning of this section (see Table 1).

Fig. 6 shows the impact of utility and cost variations on feedbacks. For equal perceived (i.e., estimated) utility values, e.g., S_8 and S_9, the higher the cost of the service, the lower the feedback, whereas for equal costs, e.g., S_2 and S_4, the higher the perceived utility, the higher the feedback. However, the impact of variations in cost remains lower than variations in utility. We also observe that the higher the cost of the service, the more important the impact of

TABLE 1
Experiment Settings and Results

Service ID	SLA violation rate	Cost	Estimated utility	Feedback
S_1	0.02	9	0.88	0.97
S_2	0.04	95	0.88	0.82
S_3	0.13	79	0.81	0.75
S_4	0.05	95	0.70	0.56
S_5	0.11	9	0.70	0.87
S_6	0.13	34	0.65	0.80
S_7	0.51	6	0.42	0.59
S_8	0.43	91	0.37	0.18
S_9	0.51	10	0.37	0.52
S_{10}	0.75	11	0.17	0.24

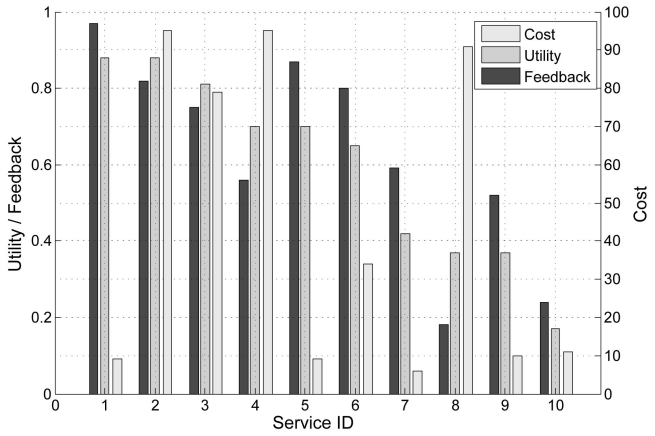

Fig. 6. Feedback versus utility and cost.

variations in utility values on the feedback. For instance, S_1 and S_5 have equal low costs and high utilities, while S_2 and S_4 have equal high costs and have the same high utilities as S_1 and S_5, respectively. However, we observe that the deviation in feedbacks ($\Delta feedback$) between S_2 and S_4 is almost three times as high as between S_1 and S_5. Variations in $\Delta feedback$ are in fact driven by variations in the curvature of the feedback function.

3.3 Reputation Computation Model

As discussed previously, reputation is computed on the basis of past feedbacks. It helps consumers predict the credibility of the service offer and the trustworthiness of the service provider prior to reaching an agreement.

Past feedback reflects the past behavior of a service and may give an indication of its future behavior; feedback may be randomly distributed when a service's behavior is not deterministic; they may follow a trend, e.g., increasing feedback may reflect an improvement in service quality, or they may be cyclic when there is a periodicity in a service's behavior, e.g., quality may decrease in rush hours leading to low feedback at those times.

When the feedback does not show any trend, it is difficult to predict a service's future behavior. However, when feedback exhibits a trend, this should be taken into account by the reputation function as it could help to predict future behavior. Smoothing and forecasting techniques, like *Moving Average*, *Weighted Moving Average*, or *Exponential Smoothing* [18], can be used to predict near future behavior from past behaviors. Other more specific techniques like Holt's Linear Exponential and Holt-Winters' Forecasting are more suitable for long-term forecasting over data showing a trend and periodicity, respectively.

In the following, we will evaluate some of the previously mentioned techniques under different circumstances, namely, *moving average* (MA), *weighted moving average* (WMA), *simple exponential smoothing* (SES), and *Holt's linear exponential* (HLE). We first consider a service with nondeterministic behavior. Then, we assume that the quality of the service is degrading. For the sake of simplicity, we make the following assumptions: 1) Feedback is left each time unit and 2) we want to predict the behavior of the service for one time unit ahead.

We generate N utility values and compute the corresponding feedbacks as explained in Section 3.2.2. We compare the user's feedback series $f(t)$ against the reputation series $R(t)$. The latter is computed as follows:

$$R(t): \begin{cases} \frac{1}{T}\sum_{\tau=t-T+1}^{t} f(\tau) \quad \text{with} \quad \text{MA}, \\ \frac{T}{1+\cdots+T} f(t) + \cdots + \frac{1}{1+\cdots+T} f(t-T+1) \quad \text{with} \quad \text{WMA}, \\ \alpha f(t) + (1-\alpha) R(t-1) \quad \text{with} \quad \text{SES}, \\ L(t) + T(t) \quad \text{with} \quad \text{HLE}, \end{cases} \tag{14}$$

where $L(t) = \alpha f(t) + (1-\alpha)R(t-1)$ is the exponentially smoothed estimate of the level of the series at time t and $T(t) = \beta(L(t) - L(t-1)) + (1-\beta)T(t-1)$ is the exponentially smoothed estimate of the trend of the series at time t.

To simulate nondeterministic behavior, we generate randomly N utility values. We simulate the trend by generating an exponentially decreasing distribution with a certain amount of distortion. We have set the smoothing period T for MA and WMA to 5, i.e., the forecast reputation value corresponds to, respectively, the mean and weighted mean of the feedbacks collected in the last 5 time units.

Fig. 7 shows that the reputation is more smoothed with the MA technique. Very recent variations in the feedback series have a low impact on reputation. With a higher smoothing period, the resulting series would be even smoother and the reputation would be less sensitive to variations in recent feedbacks.

Whereas in MA, past observations are weighted equally, the WMA technique assigns decreasing weights as the observation gets older. The reputation is more sensitive to recent variations in feedbacks when WMA is used, yet WMA—like MA—cannot be used until the first smoothing period has elapsed. Similarly to MA, the longer the smoothing period, the smoother the reputation series.

The SES technique is very simple to use and implement compared to the other techniques. The smoothing factor α, which must be between 0 and 1, acts as a weight for the most recent observation but also acts recursively as an aging factor for all past observations. In fact, each forecast $R(t)$ is computed as a weighted average of the latest observation, i.e., last feedback $f(t)$ and the previous forecast $R(t-1)$. The closer α is to 1, the more responsive the reputation series is to the latest changes in the feedback series, while the closer α is to 0, the higher the smoothing effect is. The choice of an *appropriate* smoothing factor is purely judgmental. α can be adjusted in order to optimize the gap between observations and forecast, e.g., α can be determined such that $\sum (R(t-1) - f(t))^2$ is minimized. Fig. 7 shows that α set to 0.8 brings a visible smoothing level while the recent trends in feedback variations are still captured.

The HLE smoothing method adds to the SES technique a *trend* component to create a linear trend in the forecast. HLE is therefore more appropriate for forecasting when the observations show an exponential growth or decline. By setting the HLE parameters α and β to 0.7 and 0.6, respectively, as shown in Fig. 7, we can see how quickly the reputation function reacts to changes in the feedback

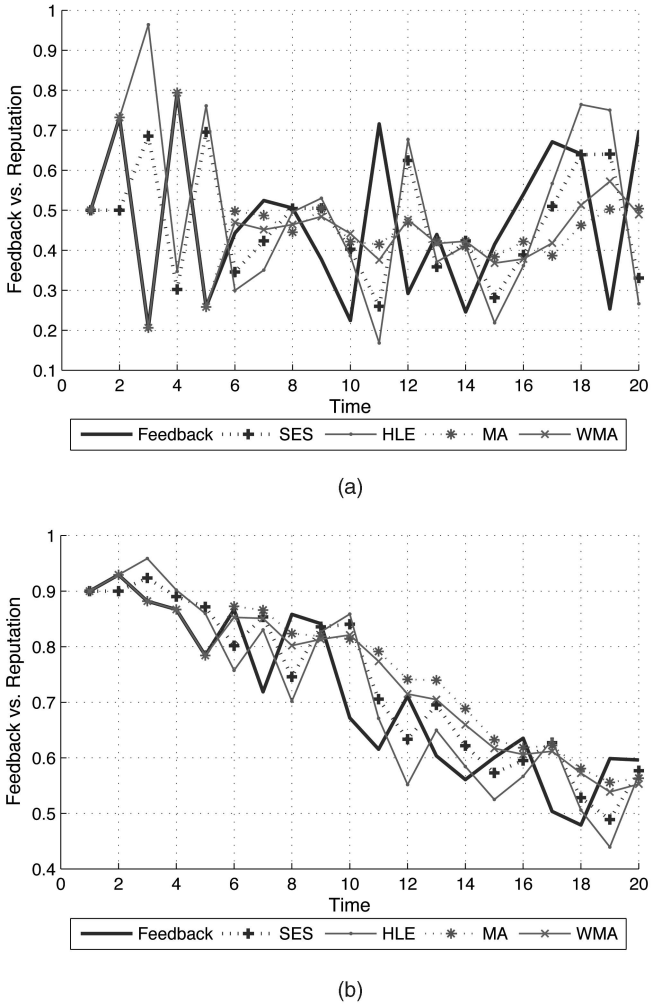

Fig. 7. Reputation forecasting by smoothing. (a) Feedbacks without trend and (b) feedbacks with a trend.

series and follows the recent trend. A smaller α would lead to a more visible smoothing.

The simplicity, convenience, and adaptability of the Simple Exponential Smoothing technique make it a good candidate for reputation computation. However, the accuracy of any smoothing or forecasting technique depends on the constancy of the observations; continuous and regular feedbacks are required to accurately capture the behavior of the service and help making more informed choices.

4 Automated Selection Process

As outlined before, services are compared against three criteria (quality, cost, and reputation) before any selection is made. In the previous section, we have discussed the need for fair reputation reports in order to make enlightened choices, and to this end, devised a rating system that ensures the objectivity of feedbacks. In this section, we focus on the selection process. We investigate the ways in which quality, cost, and reputation can be combined in support of decision making. We devise an algorithm that aggregates the three parameters into a single ranking metric. This algorithm involves three steps: *match making*, *evaluation*, and *ranking*.

4.1 Match Making

This step consists of comparing service offers against user requirements. All of the offers which do not meet user requirements, commonly expressed in terms of quality and cost constraints, are ignored. The QoS offer is commonly defined as a vector of (Q_i, q_i) pairs, where Q_i refers to a quality dimension and q_i refers to the corresponding value (e.g., (*uptime*, 99.99 percent),(*response time*, 600 ms)). Quality dimensions may have properties [3] such as different definition domains and evaluation rules. For instance, service availability (*uptime*) is defined on $[0, 1]$ and obeys the "more-is-better" evaluation rule, whereas service responsiveness (*response time*) is defined on $[0, +\infty[$ and, as opposed to the availability dimension, obeys the "less-is-better" evaluation rule. The evaluation of a QoS offer is challenging, considering that it requires the knowledge of such properties. In this perspective, it is essential to provide a formal specification of quality dimensions. The definition of QoS dimensions is a critical issue, which has received a lot of attention over the last decade. However, only a few works have provided quality dimension taxonomies. Among the Web Services-related works, WSOL [19], WSLA [20], DAML-QoS [21], OWL-S [22], and OWL-Q [23] provide some attempts to formalize the description of QoS dimensions.

For each candidate service s, let qos be the offered quality vector, R its reputation, and $cost$ its cost. Let $qos_r = (q_1)_r, (q_2)_r, \ldots, (q_N)_r$ be the vector of quality constraints.

We denote by QOS^+ the subset of *more-is-better*-like QoS parameters and QOS^- the subset *less-is-better*-like parameters.

Let S be the set of preselected services,

$$s \in S \quad \text{if} \quad \forall i = 1..N \begin{cases} \text{if} \quad Q_i \in QOS^-, & (q_i)_r \leq q_i(s), \\ \text{if} \quad Q_i \in QOS^+, & (q_i)_r \geq q_i(s). \end{cases} \tag{15}$$

Similarly, restrictions on cost may apply; unaffordable services are also ignored.

To illustrate the matchmaking step, consider four instances of *GenericSaaS*, S_1, S_2, S_3, and S_4, with the respective offers:

$$\begin{aligned} &S_1(qos = \{99.99\%, 600 \text{ ms}\}, cost = 100), \\ &S_2(qos = \{98.99\%, 699 \text{ ms}\}, cost = 90), \\ &S_3(qos = \{97.99\%, 600 \text{ ms}\}, cost = 90), \\ &S_4(qos = \{96.0\%, 699 \text{ ms}\}, cost = 70). \end{aligned}$$

After the matchmaking step, only S_1, S_2, and S_3 are kept for further evaluation when the QoS requirements are set to $qos_r = \{96.99\%, 699 \text{ ms}\}$ given that the offer of S_4 in terms of *uptime* is lower than requested ($96.0\% < 96.99\%$) .

4.2 Evaluation

At this stage, all of the eligible services offer a quality level that is equal to or higher than requested and come at affordable costs. We will thus evaluate service offers in terms of the *gain* in quality and cost that is proposed. Let Q and C be the evaluation metrics of gains in quality and cost, respectively. We hereby define R, Q, and C as scalar values between 0 and 1.

We first evaluate the gain in each quality dimension Q_i. For each Q_i, we define two parameters $(q_i)_{max}$ and $(q_i)_{min}$ as follows:

$$\begin{cases} (q_i)_{max} = \begin{cases} \max_{s\in S} q_i(s) & \text{if} \quad Q_i \in QOS^+, \\ (q_i)_r & \text{if} \quad Q_i \in QOS^-, \end{cases} \\ (q_i)_{min} = \begin{cases} \min_{s\in S} q_i(s) & \text{if} \quad Q_i \in QOS^-, \\ (q_i)_r & \text{if} \quad Q_i \in QOS^+. \end{cases} \end{cases} \tag{16}$$

In the previous *GenericSaaS* example, $uptime_{max} = 99.99\%$, $uptime_{min} = 96.99\%$, $response\text{-}time_{max} = 699$ ms, and $response\text{-}time_{min} = 600$ ms.

The scaling function $Scal_i$ is defined on $dom(Q_i)$ and takes values in $[0, 1]$. $Scal_i$ is increasing for $Q_i \in QOS^+$ and decreasing for $Q_i \in QOS^-$:

$$Scal_i(q_i) = \begin{cases} \dfrac{q_i - (q_i)_{min}}{(q_i)_{max} - (q_i)_{min}} & \text{if} \quad Q_i \in QOS^+, \\ \dfrac{(q_i)_{max} - q_i}{(q_i)_{max} - (q_i)_{min}} & \text{if} \quad Q_i \in QOS^-, \\ 1 & \text{if} \quad (q_i)_{max} - (q_i)_{min} = 0. \end{cases} \tag{17}$$

We can easily show that, for all $i = 1..N$, $Scal_i((q_i)_r) = 0$.

For instance, the scaled quality parameters of the above $GenericSaaS$ services are as follows: $qos(S1) = \{1, 1\}$, $qos(S2) = \{0.5, 0\}$ and $qos(S3) = \{0, 1\}$.

We now derive the scalar metric Q from the vector $(Scal_i(q_i))$. $W = (w_1), (w_2), \ldots, (w_N)$ denotes the consumer's quality preferences, where $0 \le w_i \le 1$ and $\sum_{k=1}^{N} w_i = 1$. $(Scal_i(q_i))_{i=1..N}$ represent the coordinates of the candidate service s in the N-dimensional euclidean space, where the origin is $s_0(Scal_i((q_i)_r)_{i=1..N}$. We compute Q as the weighted euclidean distance $Q = \|s - s_0\|$ as follows:

$$Q = \sqrt{\sum_{i=1}^{N} w_i Scal_i(q_i)^2(s)}, \tag{18}$$

where Q also represents the weighted root-mean-square of $(Scal_i(q_i))_{i=1..N}$.

Finally, C is computed as follows:

$$C = \frac{(C)_{max} - C}{C_{max} - C_{min}}.$$

In the previous *GenericSaaS* example, assuming that the quality parameters *uptime* and *response time* are equally weighted, the scaled QoS and cost (Q, C) offers of the preselected service instances S_1, S_2, and S_3 are, respectively $(1, 0)$, $(0.47, 1)$, and $(0.74, 1)$.

4.3 Ranking

This is the final step, where R, Q, and C are combined into $SCORE(s)$, the ultimate selection metric. The service with the highest $SCORE(s)$ is then selected. In [3], a weighted mean-like aggregation function on all quality parameters including cost and reputation is used, where weights are user-defined constants. We believe that reputation should not be considered as a quality parameter, but instead, as a moderator between service quality and quality guarantees. Moreover, a weighted mean-like score function implies that the score is evenly sensitive to variations in reputation. Although we can agree that the score function should be evenly sensitive to variations in Q or C, we believe that it should be less sensitive to variations in "low" reputation values than to variations in higher values. In fact, we observe that under a reputation threshold, consumer's "trust" in the service is lost, which means that the service is no longer believed to deliver the expected quality.

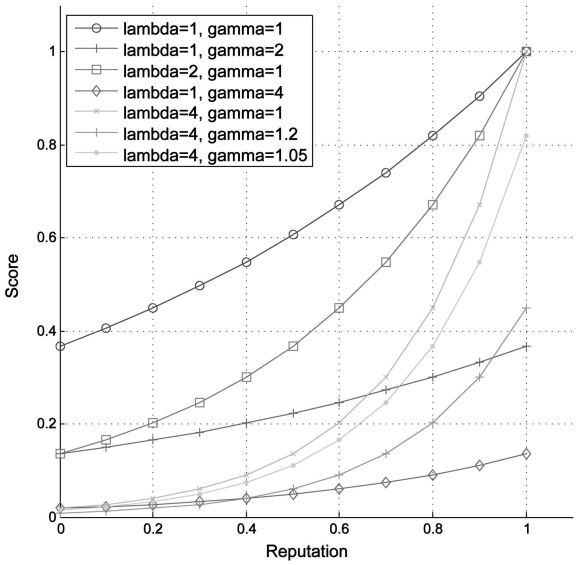

Fig. 8. Impact of λ and γ on the $SCORE$ function.

According to the above observations, $SCORE(s)$ should increase linearly with Q and C and exponentially with R; this would emphasize the insensitivity of $SCORE(s)$ to variations in low R values. We also consider that $SCORE(s)$ takes values in $[0, 1]$. We denote by $SCORE_R(s)$, $SCORE_Q(s)$, and $SCORE_C(s)$ the partial derivatives of $SCORE(s)$ with respect to R, Q, and C, respectively. We obtain the following:

$$\begin{cases} SCORE_R(s) = \lambda\, SCORE(s), \\ SCORE_Q(s) = \alpha, \\ SCORE_C(s) = \beta, \\ \lambda > 0 \qquad and \quad \alpha, \beta \in [0, 1]. \end{cases} \tag{19}$$

We also have

$$SCORE(s) = 1 \quad \text{if} \quad R, Q, C{=}1, \tag{20}$$

$$SCORE(s) = 0 \quad \text{if} \quad R, Q, C = 0. \tag{21}$$

We integrate the partial derivatives of $SCORE(s)$ from (19), obtaining

$$SCORE(s) = e^{\lambda(R-\gamma)} + \alpha Q + \beta C + \psi. \tag{22}$$

With $\gamma > 1$ and $\psi = -e^{-\lambda\gamma}$ (from (21)). α and β weight the impact of the quality and the cost attributes, respectively, on the score function. As shown in Fig. 8, λ and γ control the impact of reputation on the score function; λ controls the growth rate and γ the function's range. We observe that the higher λ is, the more convex the function is, while the closer γ is to 1, the larger the function's range is. A high λ value and a γ value close to 1 are more compliant with the desired characteristics of the score function.

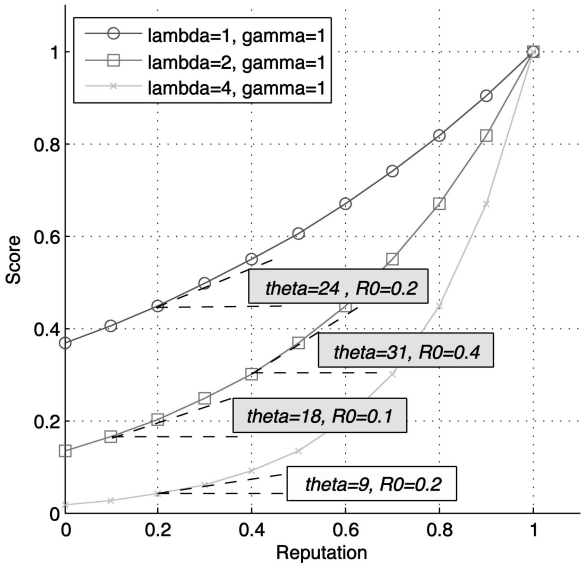

Fig. 9. The $SCORE$ function: λ versus θ and R_0.

Moreover, we obtain a very desirable simplification of $SCORE(s)$ by setting γ to 1:

$$SCORE(s) = e^{\lambda(R-1)} + e^{-\lambda}(\alpha e^{\lambda} Q + \beta e^{\lambda} C - 1). \quad (23)$$

From (20), we obtain

$$\alpha e^{\lambda} + \beta e^{\lambda} = 1. \quad (24)$$

In the remainder of the paper, the relative weights ω_c and ω_q will denote βe^{λ} and αe^{λ}, respectively:

$$SCORE(s) = e^{\lambda(R-1)} + e^{-\lambda}(\omega_q Q + \omega_c C - 1). \quad (25)$$

We denote by R_0 the threshold under which a reputation value is considered unsatisfactory, i.e., the value under which $SCORE(s)$ is much less sensitive to variations in R. R_0 should be the point at which the score function's growth rate turns from "low" to "high," i.e., graphically speaking, the point at which the tangent angle gets sharper. We denote by θ the tangent angle at R_0. θ is defined as follows:

$$\theta = \arctan(SCORE_R(R_0)). \quad (26)$$

Due to the convexity of the $SCORE$ function, the higher R_0 is, the lower θ will be. Fig. 9 also shows that θ decreases with an increasing λ. By choosing R_0, and the tangent angle $\theta \in [0, \frac{\pi}{4}]$ that we consider sharp enough, we can derive λ from the following equation:

$$\lambda \exp(\lambda(R_0 - 1)) - \tan(\theta) = 0. \quad (27)$$

It is worth noting that the choice of R_0 should depend on the properties of the service. Subscribers to pricey or mission critical services have higher expectations with respect to service quality; R_0 should be set to the higher side for those kinds of services.

Assuming that R_0 is set to 0.3, λ to 2.6, ω_c and ω_q to 0.6 and 0.4, respectively, and that the reputations of the *GenericSaaS* instances S_1, S_2, and S_3 are, respectively, 1, 1, and 0.9, then the score of S_1 is 0.95, the score of S_2 is 0.98, and the score of S_3 is 0.76. Given that $score(S_2) > score(S_1) > score(S_3)$, the outcome of the selection process is S_2. Now if reputations were set to 0.2, 0.2, and 0.1, respectively, service scores would be 0.08, 0.1, and 0.089, respectively. S_2 would still be selected, but we notice that, this time, S_3 is ranked second as opposed to third in the previous setting. This is due to the fact that the QoS and cost offer of S_3 are more attractive than S_1's.

5 Simulation and Evaluation

In a previous work [24], simulation experiments were conducted to evaluate the responsiveness of the system to permanent changes in service behavior. The experiments showed that the user ends up selecting the service instance that delivers the highest utility almost 83 percent of the time, and that she/he would not have been more satisfied with a service other than with her/his choice more than 87 percent of the time.

In the following, a new simulation setting is established to evaluate the system under a more realistic environment (Section 5.1). Additional experiments have also been conducted (Section 5.2) to assess the impact of service behavior on the effectiveness of the selection system.

5.1 Analysis of System Behavior

Consider a set of 20 software services S_i, $i = 1..20$, say, for instance, 20 storage services, with 20 different QoS offers (*uptime* ranging from 97.99 to 99.99 percent) and different costs (price ranging from 70 to 90). All services are assumed to have successfully passed the matchmaking test (Section 4.1).

Commonly, new services are constantly introduced into the market. In order to reproduce this feature, services are assigned a randomly generated parameter Arrival Time to Market (ATTM) that refers to the time at which the service starts being available. The time at which services get their first customers, and more importantly, their first feedback at Time to First Feedback (TTFF), where $TTFF \geq ATTM$.

We assume in this evaluation study that services have a similar "overall" good behavior. In other words, we assume that service providers respect their commitments to their customers most of the time. Failures may occur from time to time and services recover shortly after. Two more randomly generated simulation parameters are defined for each service instance: Time Between Failures (TBFs), which refers to the *shortest* time that separates two consecutive failures, and Time to Recovery (TTR), which refers to the *longest* time the service will take to recover from any failure. At each failure, the recovery time is randomly generated with TTR as an upper bound (the upper bound for all TTRs being set to 50), and the next failure will take place after a randomly generated time, not shorter than TBF (the lower bound for TBFs being set to 50 and the upper bound to 400). As long as the service is operational, its utility is equal to 1. During failures, the utility will vary between 0 and 1.

Feedback is derived according to our rating function. Discontinuity is introduced in the feedback series assuming that reports are not provided each single time unit. The reputation series is then derived using the SES technique. Scores are computed and selection is made every time unit according to our selection algorithm. The system's behavior is observed for 400 units of simulation time.

TABLE 2
Service Parameters

ID	Cost	Uptime (%)	ATTM	TTFF	TBF	TTR
0	74.0	97.99	0	0	165	14
1	98.0	97.99	86	130	184	25
2	81.0	99.99	44	59	133	43
3	71.0	98.99	83	103	151	47
4	98.0	97.99	71	97	105	11
5	75.0	99.99	43	43	177	26
6	76.0	97.99	28	64	182	42
7	100.0	99.99	92	125	156	44
8	77.0	99.99	46	70	116	16
9	81.0	99.99	82	118	188	32
10	74.0	99.99	89	89	123	26
11	75.0	98.99	71	82	111	48
12	98.0	97.99	63	92	186	17
13	81.0	97.99	66	109	142	34
14	81.0	99.99	65	76	180	42
15	73.0	98.99	73	119	120	44
16	80.0	99.99	10	10	149	29
17	81.0	99.99	6	6	176	20
18	88.0	97.99	6	39	102	30
19	92.0	97.99	41	89	185	14

Service parameters are listed in Table 2. The other simulation parameters (including the SES smoothing parameter α, R_0, λ, w_q, and w_c) are set as indicated in Table 3.

Fig. 10 shows the utilities and reputations of a representative set of service instances. Utility functions start at different times according to the ATTM of the corresponding service. At ATTM, the reputation of the service is set to R_0 and stays so until the first feedback is reported (see Fig. 10f).

TABLE 3
Simulation Parameters

Parameter	Definition	Value
α	SES smoothing parameter	0.8
λ	SCORE function convexity parameter	2.6
R_0	Satisfactory reputation threshold	0.3
w_q	Relative quality weight	0.4
w_c	Relative cost weight	0.6

Fig. 10 also shows how fast the reputation system reacts to changes in service behaviors. However, it is worth noting that the more frequently the service fails and the longer the service takes to recover, the longer it takes for its reputation to get back to higher values.

The simulation results show that the system adapts to the market dynamics. Five service instances (S_0, S_{17}, S_{16}, S_5, and S_{10}) are respectively considered for selection during the first 200 time units, whereas only two services (S_{10} and S_5), particularly S_{10} out of the former five, are considered later on (see Fig. 11). In fact, during the first 200 time units, new services with potentially better quality and/or cost offers are constantly coming into the market. Provided that new services have behavior at least as good as the older ones, they are considered as better choices. As soon as the market reaches a stable state, the selection converges toward the

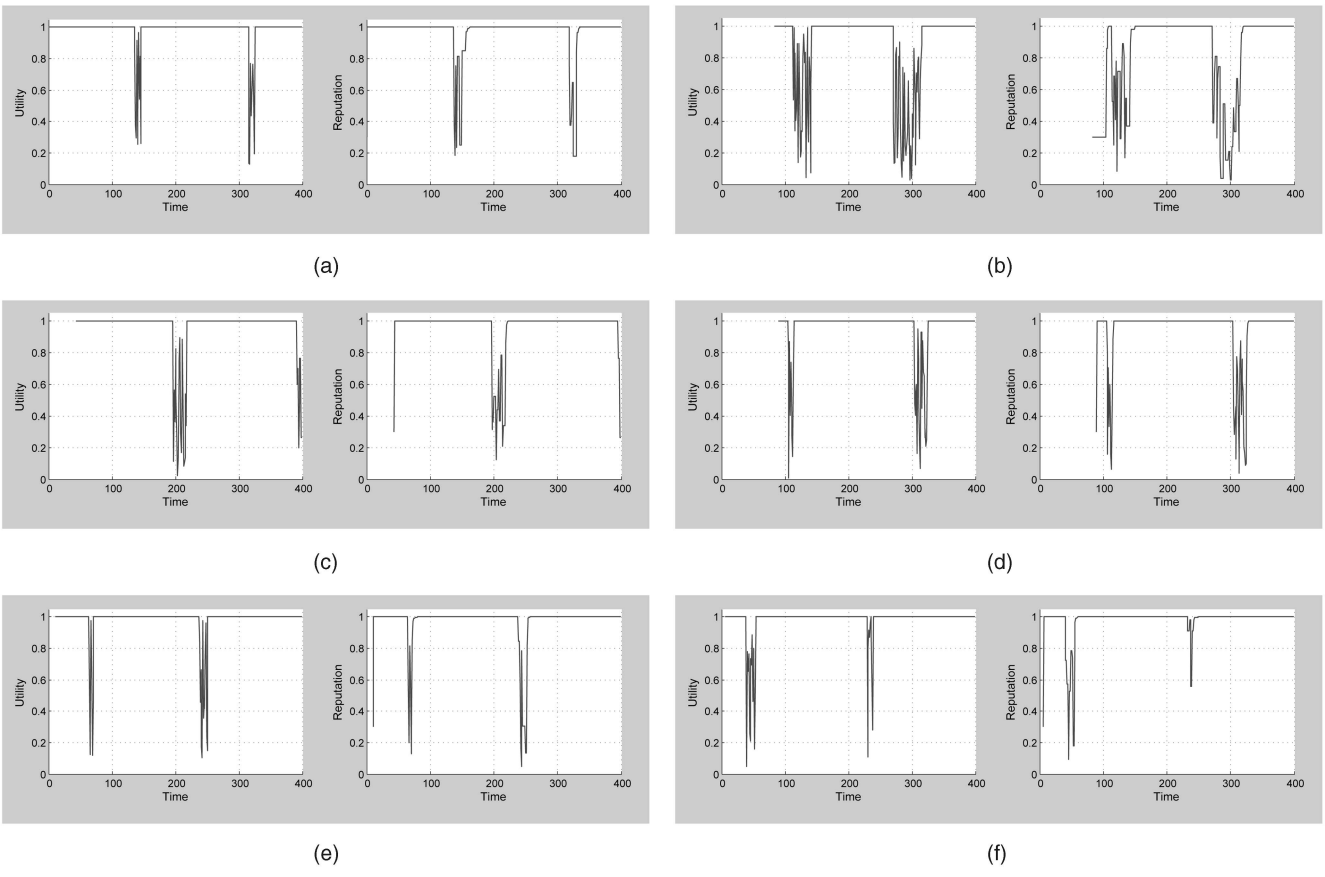

Fig. 10. Utility and reputation evolution of (a) S_0, (b) S_3, (c) S_5, (d) S_{10}, (e) S_{16}, and (f) S_{17}, respectively.

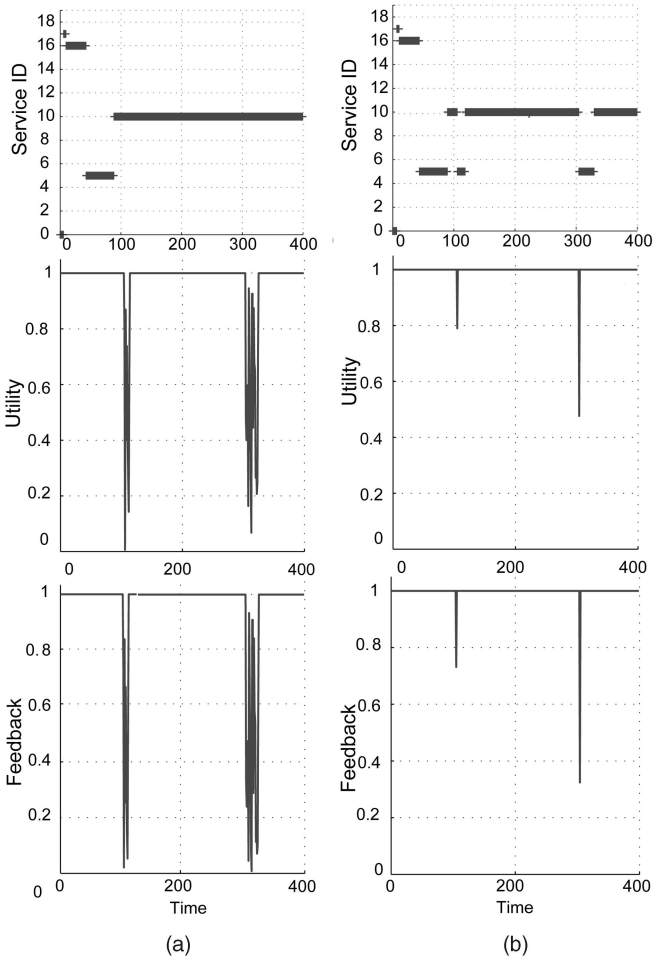

Fig. 11. Selected services, experienced utility, and customer satisfaction. (a) Without reputation awareness and (b) with reputation awareness.

services which offer the best quality/cost ratio and reputable behavior.

As shown in Fig. 11, users experience better utility and better satisfaction with the delivered service when reputation is considered in decision support. More generally, users experience the best possible utilities and leave the highest possible feedbacks most of the time when our devised selection mechanism and automated rating system are used. It is worth noting that the most selected service during market stability, S_{10}, comes at a low cost (third lowest cost), high quality (the highest quality offer), relatively sparse failures, and low recovery times (see Fig. 11).

This experiment has been repeated 100 times in order to evaluate more accurately the success rate of the system. Two parameters have been observed: the mean *absolute* success rate, i.e., how often the user experiences maximum satisfaction, and the mean *relative* success rate, i.e., how often user's satisfaction with the selected service is enhanced when reputation is considered in decision support. As shown in Table 4, the mean absolute success is on the very high side (around 99 percent) both in dynamic and stable environments. However, it is worth noting that the satisfaction of the user is only slightly enhanced (5.61 to 7.58 percent) by considering reputation in decision support. This is due to the fact that all candidate services behave relatively well,

TABLE 4
Simulation Results

	Dynamic market	Stable market
Absolute success	99.37%	99.41%
Relative success	5.61%	7.58%

here with the lower (respectively, upper) bound of all TBFs set to 100 time units (respectively, 200 time units) and the upper bound of all TTRs set to 50. In the following section, we will evaluate the impact of service behavior on the enhancement induced by reputation awareness.

5.2 Impact of Service Behavior on System Performance

In order to evaluate the impact of service behavior on the effectiveness of our system, we have run additional experiments with variable TBFs. We have varied the upper bound of TBFs from 20 to 390 (the lower bound of TBFs and the upper bound of TTRs are set at 20 this time). The outcome of the experiments is shown in Fig. 12.

The absolute success rate of the system is consistently on the higher side (more than 96 percent most of the time). It drops slightly under extremely variable utility conditions, i.e., when the TBF upper bound is close to 20. The drop is more meaningful when the market is stable (success rate below 90 percent). On the other hand, the relative success rate is perceptibly higher when the market is stable compared to when the market is dynamic. Under extremely variable utility conditions, we notice that the satisfaction of users can be significantly enhanced when reputation is considered in decision support. For instance, Fig. 12 shows

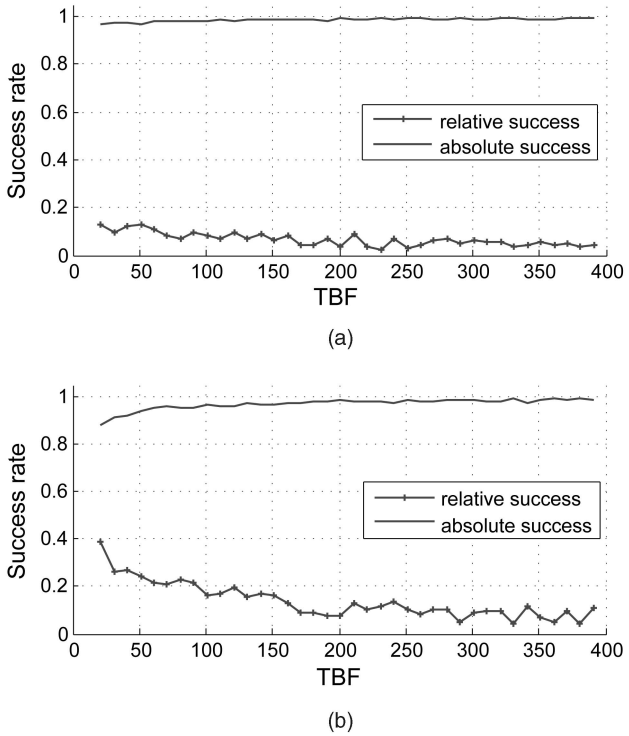

Fig. 12. Success rate versus TBF in (a) dynamic market and (b) stable market.

that enhancement in user satisfaction can reach 40 percent when the market is stable (13 percent otherwise).

The conducted experiments show that the overall system evolves well over time. Furthermore, the obtained results demonstrate that our automated selection system, in conjunction with our automated rating mechanism, succeeds in capturing service behavior and providing the best possible choice.

6 Discussion

With the quasi-globalization of fast Internet access, the success of SOA infrastructures, and the growing interest in cloud computing, enterprises as well as software vendors are increasingly adopting the SaaS on-demand delivery model. The unknown quality of the service and the unknown reliability of the provider at selection time are risks that need to be considered before an SaaS is acquired. Because the software is hosted by its provider, the consumer needs guarantees that the service will perform as expected and that the provider will maintain the service—to prevent failures—and provide support on request.

In this work, we have identified risk management at selection time as essential to the long-term success of the SaaS model. In this perspective, a reputation-aware selection mechanism and an automated rating system have been designed. The simulation results have demonstrated that the devised system has successfully met our primary objectives and can be an important component in a risk management strategy for software development with SaaS.

The designed system can be configured so that it can adapt to the needs of service integrators, to the particularities of each service, and to market dynamics. It does not involve any heavy implementation or complex infrastructure; the selection algorithm can be implemented easily, and it can make use of any reputation system using a patch to any monitoring system to provide the rating functions.

In developing this system, several design choices were made. Although the conducted simulations have shown successful results, there are a number of areas that could provide room for improvement. The following discusses the impact of several key design decisions and highlights points for future investigation.

6.1 Reputation versus Feedback versus Utility versus Performance Reports

The lack of governance in today's Internet and the widespread growth of the service market have led to the popularity of reputation systems. The use of reputation systems in risk reduction is motivated by the need for project managers to 1) make more informed service selections, 2) trace the behavior of contracted service providers in order to anticipate performance degradation, and 3) trace the behavior of competing providers, in case a substitute service is required. The choice of using feedback and reputation to measure the behavior of the service instead of plain monitoring reports is in part based on the widespread acceptance of reputation systems, and the corresponding maturity of related protocols and infrastructures. However, it is worth noting that feedback is more storage-effective and easier to maintain, while reputations are easier to retrieve. Using monitoring reports would have necessitated extra effort in investigating and designing description schemes and management protocols.

In this work, feedback is derived from service utility, exclusively measuring the performance of the service in terms of the delivered quality level. However, more aspects could be considered in the feedback computation. For example, the cost and sensitivity of the service to quality degradation could be considered in order to penalize expensive and misbehaving services. It is worth noting that the feedback computation (10) is configurable; the value of μ can be chosen, with setting μ to 0 resulting in $feedback = utility$.

6.2 Feedback Computation

In this work, the feedback is computed as a forecast of the consumer's satisfaction with the service's execution quality. Alternatively, classical as well as Bayesian forecasting techniques could be applied to the consumer's past feedback on the same service (feedback *time series*), or on other consumers' feedback on the same service (feedback *samples*) to predict or *infer* future feedback. These options have been considered and evaluated at an early stage of this work and withdrawn because of the subjectivity and bias issues. Generally, applying forecasting techniques on time series introduces a bias toward past feedback and past service behavior. Furthermore, forecasting on feedback samples does not solve the issue of feedback subjectivity since there is no guarantee that the sampled feedback is objective and fair. In order for the feedback forecast to be objective, fair, and unbiased, the forecasting technique must be solely applied to the current experience with the service. None of the above forecasting techniques meets this requirement.

The expectancy-disconfirmation theory form market science has been widely used for explaining or predicting the acceptance or rejection of marketed products. It is a well-accepted and tried theory from which a number of customer satisfaction models have been derived and empirically validated. According to these models, in particular the model presented in [16], the satisfaction of a customer with a delivered service can be inferred from her/his expectation and perception of the service utility. Customer satisfaction models fit well in our framework indeed.

6.3 Reputation Computation

Several techniques for reputation computation have been investigated in this work. Our primary goal was to find an alternative to using simple mean of feedback, which would not take short-term fluctuations in feedback into consideration. We have focused on forecasting and smoothing-based techniques due to their desirable features. Which technique is more appropriate is a matter of judgment. Other aspects could also be considered in this decision, for instance, it is desirable that the chosen technique be easy to implement, work effectively with a small number of feedbacks, and be configurable. The Simple Exponential Technique fulfills these requirements as outlined in Section 3.3. Choosing an appropriate smoothing parameter is also discretional. A high smoothing level prevents reputation from dropping significantly when a failure occurs. However, it also prevents the reputation from recovering rapidly. Sensitivity versus tolerance to failures is a trade-off that must be decided on.

6.4 Discontinuity in Feedback

A major discontinuity in feedback (period without any feedback) may affect the accuracy of the reputation as a predictor for future service behavior. In this situation, it may be risky to maintain a high reputation when no activity has been reported for a while. To prevent such risks, it is essential to age reputations. One option worthy of consideration is to reset reputations to a neutral value (R_0) when there is an interval without feedback. However, this provides an opportunity for misbehaving service providers. If their reputation falls below R_0, it may be advantageous for them to leave the market temporarily in order for their reputation to be reset.

In order to prevent such behaviors, the required period of discontinuity must be long enough to dissuade providers (by interrupting revenue). However, longer discontinuities can negatively affect the accuracy of reputations. An alternative would be to institute a multilevel aging function; while high reputation values could be reset to R_0, low reputations are reset to 0. Further investigations of these solutions are planned for the near future.

7 Related Works

Service selection and rating is a research topic that emerged recently with the advent of SOA and SaaS. A literature survey reveals that the few works in this area have addressed different facets of the topic, such as the assessment of service quality at selection time, the evaluation of the credibility of quality reports, the computation and management of service reputation, and the measurement of service quality at execution time.

For example, [3], [25], and [2] focused on service composition with end-to-end QoS constraints. Zeng et al. [3] consider two models for service selection. The first model promotes the selection of those services that present the optimal service-level quality offers. We have adapted this model to our service-level quality-constrained selection problem. Unlike this model, the second model promotes service selection with global planning. In this perspective, Integer Linear Programming (ILP) is used to determine the optimal solution. Considering the complexity of ILP optimization, Yu et al. [25] propose heuristics to find near-optimal solutions in polynomial time. Similarly, Anselmi et al. [2] provide a Mixed Integer Linear Programming (MILP)-based formulation of the selection problem and consider a greedy heuristic to find near-optimal solutions. The above works consider service selection as a multicriteria constraint satisfaction problem but ignore the fact that services may not deliver, at execution time, the promised quality at selection time. Although the reputation factor is considered in [3], it is defined as a quality parameter and used as any other quality parameter in the ranking function.

Similarly to our work, Wu et al. [26] address the issue of predicting the capability of a service to deliver the level of quality that would meet user's requirements. In this perspective, a Bayesian network-based QoS assessment model is used along with a fuzzy logic-based reasoning approach for inferring service capability under various combinations of users QoS requirements. The performance of the service is tracked and recorded while the service is being executed and the compliance between the user's QoS requirements and the service's delivered QoS is computed. Based on these compliance values, a fuzzy reasoning approach is introduced to infer the service's overall capability and the corresponding Bayesian network is updated with the assessment outcome. Similar to [26], we address the need for an automated service recommender to evaluate the suitability of a service to users' preferences.

Yang et al. [27], Ali et al. [28], and Wishart et al. [29] promote the use of trust and reputation to rate candidate services in the selection process, and focus on the representation and computation aspects of service reputation. Yang et al. [27] use, for instance, a multidimensional trust model to evaluate service and service provider's properties like credibility, quality, and reliability. A "trust" vector describes users' experience with those service aspects. The trustworthiness of the service, referred to as the confidence in the service, is estimated using the Hypothesis Evaluation theory, whereas in [28], service reputation is linked to an execution context: Services are evaluated with regard to a specific execution context (application domain or user type). Reputation is then computed as a weighted mean of time-decaying feedback within the desired context. A time-decaying factor, called the "forgetting" factor, is also used in [29], along with a Bayesian estimation model to compute the reputation of a service on the basis of past users' testimonials. As apposed to the above works, we consider a more general reputation computation model where the trustworthiness of a service provider is evaluated with conformance to the terms enclosed in the offer (i.e., SLA) and do not only consider the timeliness of service ratings but also take into account the trend they may show.

Vu et al. [4] consider the assessment of the credibility of quality reports. Few trusted parties are assumed to provide credible reports on the conformance of the delivered quality to the offered one. These reports are used to evaluate the credibility of other reports. False reports are then detected and ignored in the selection process. The future conformance of the delivered quality to the offer is predicted using a linear regression method on past QoS conformance reports, each weighted by its evaluated credibility. Indeed, false quality reports may jeopardize the selection process. However, the report evaluation system in [4] raises a critical issue. User reports cannot be objectively and fairly compared to reports made by trusted monitoring agents unless both monitoring measurements are achieved under the same circumstances. If trusted monitoring agents can be deployed within the same context as users, then user reports are no longer needed. Our solution has similarities with the above works in that we are also concerned with the credibility of service ratings. However, we focus on the process of generating credible feedbacks and promote a trustworthy automated rating process instead.

Our work has been in part grounded on literature works in the area of SLA monitoring like [7], [30], and [5]. In particular, Skene et al. [7] present a comprehensive framework for the monitorability of SLAs and the assessment of measurement credibility and accuracy. This work is complementary to ours in that we build our automated rating system on the assumption that SLAs are monitorable and that SLA monitoring is conducted in a trusted and accurate fashion. Findings in [7], [30], and [5] validate this fundamental assumption indeed.

8 Conclusion

In this paper, the issue of risk management has been addressed in the context of project development using external software service components. In this perspective, we have presented an automated quality and reputation-based framework for service rating and selection. Although a few previous works have considered quality and reputation for service selection, none have considered the automation of the service rating process.

The proposed service rating allows feedback to be assigned to a delivered service that objectively reflects the satisfaction or dissatisfaction with the rendered performance and quality. To this end, the compliance of the rendered quality with the quality agreed upon is monitored and translated into a utility metric. A feedback computation model, derived from the expectancy-disconfirmation theory from market science, is then proposed to generate a feedback from service utility and cost. A reputation derivation model has also been proposed to aggregate feedback into a reputation value that better reflects the behavior of the service at selection time.

The selection algorithm has been designed to assist customers in selecting the most appropriate service offering considering quality and cost constraints. Reputation is used to predict the credibility of the quality offer and the conformance of this offer to the delivered quality. We devised a service ranking function that aggregates the quality, cost, and reputation parameters into a single metric that is used to evaluate service offerings against each other.

Simulation studies have been conducted in order to evaluate the proposed rating and selection algorithms. We have considered several services with different offers and utility functions. Our rating algorithm was run on each service over a long period of time, while simultaneously conducting service selection on behalf of the virtual customers. The results show that the overall system evolved well over time, with misbehaving services being detected quickly. Customers can then avoid these services—in fact, we show that the system selects services that the customer could not have been more satisfied with. Our rating and selection systems succeeded in capturing the service behaviors and translating them into the best possible choices for customers.

This work contributes to a framework that aims to design an infrastructure that supports the integration of software services in application development and provisioning projects. Interesting avenues for future research include the generalization of our selection mechanism to composite services with global quality constraints and the investigation of advanced algorithms for multicriteria decision making and multiobjective optimization such as multiobjective evolutionary algorithms.

Acknowledgments

This work was supported in part by the Natural Science and Engineering Council of Canada (NSERC) Discovery program and in part by the WCU (World Class University) program through the Korea National Research Foundation funded by the Ministry of Education, Science and Technology (Project No. R31-2008-000-10100-0). The authors would like to express their gratitude to Brent K. Ishibashi for helping with paper editing and proofreading, and to the associate editor and anonymous reviewers for their encouragements and insightful comments.

References

[1] J. Li, R. Conradi, O.P. Slyngstad, M. Torchiano, M. Morisio, and C. Bunse, "A State-of-the-Practice Survey of Risk Management in Development with Off-the-Shelf Software Components," *IEEE Trans. Software Eng.*, vol. 34, no. 2, pp. 271-286, Mar./Apr. 2008.

[2] J. Anselmi, D. Ardagna, and P. Cremonesi, "A QoS-Based Selection Approach of Autonomic Grid Services," *Proc. Workshop Service-Oriented Computing Performance: Aspects, Issues, and Approaches,* 2007.

[3] L. Zeng, B. Benatallah, A.H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "QoS-Aware Middleware for Web Services Composition," *IEEE Trans. Software Eng.*, vol. 30, no. 5, pp. 311-327, May 2004.

[4] L.-H. Vu, M. Hauswirth, and K. Aberer, "QoS-Based Service Selection and Ranking with Trust and Reputation Management," *Proc. 13th Conf. Cooperative Information Systems,* 2005.

[5] J. Skene, F. Raimondi, and W. Emmerich, "Service-Level Agreements for Electronic Services," *IEEE Trans. Software Eng.*, vol. 36, no. 2, pp. 288-304, Mar./Apr. 2010, http://doi.ieeecomputersociety.org/10.1109/TSE.2009.55.

[6] J.R. Douceur, "The Sybil Attack," *Proc. First Int'l Workshop Peer-to-Peer Systems,* 2002.

[7] J. Skene, A. Skene, J. Crampton, and W. Emmerich, "The Monitorability of Service-Level Agreements for Application-Service Provision," *Proc. Sixth Int'l Workshop Software and Performanc,* pp. 3-14, 2007.

[8] BelGOnet, "Service Level Agreement," http://www.belgonet.com/website/UK/Service_Level_Agreement_UK.pdf, 2008.

[9] EZSM, "EZSM Service Level Agreement," http://www.easyservermanagement.com/sla.php, 2008.

[10] Amazon, "Amazon s3 Service Level Agreement," http://aws.amazon.com/s3-sla/, 2007.

[11] Intacct, "Intacct Buy with Confidence," http://us.intacct.com/downloads/08datasheets/DS_Buy_with_Confidence.pdf, 2008.

[12] H. Ludwig, R.P. King, and A. Keller, "Web Service Level Agreements," http://www.research.ibm.com/wsla/sample-outsourced.wsla, 2002.

[13] Y. Wang and J. Vassileva, "A Review on Trust and Reputation for Web Service Selection," *Proc. 27th Int'l Conf. Distributed Computing Systems Workshops,* 2007.

[14] T. SaaS, "Trust Saas: Putting the Trust in Software as a Service (SaaS)," http://trustsaas.com/, 2008.

[15] R.L. Oliver, "A Cognitive Model of the Antecedents and Consequences of Satisfaction Decisions," *J. Marketing Research,* vol. 17, pp. 460-469, Nov. 1980.

[16] J. Xiao and R. Boutaba, "Assessing Network Service Profitability: Modeling from Market Science Perspective," *IEEE/ACM Trans. Networking,* vol. 15, no. 6, pp. 1307-1320, Dec. 2007.

[17] V. Poladian, J.P. Sousa, D. Garlan, and M. Shaw, "Dynamic Configuration of Resource-Aware Services," *Proc. 26th Int'l Conf. Software Eng.*, 2004.

[18] P. Brockwell and R. Davis, *Introduction to Time Series and Forecasting.* Springer-Verlang, 2002.

[19] V. Tosic, K. Patel, and B. Pagurek, "Wsol—Web Service Offerings Language," *Proc. Workshop Web Services, e-Business, and the Semantic Web,* 2002.

[20] A. Keller and H. Ludwig, "The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services," *J. Network and System Management,* vol. 11, no. 1, pp. 57-81, 2003.

[21] C. Zhou, L.-T. Chia, and B.-S. Lee, "DAML-QoS Ontology for Web Services," *Proc. IEEE Int'l Conf. Web Services,* 2004.

[22] M. Burstein, J. Hobbs, O. Lassila, D. Mcdermott, S. Mcilraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne, E. Sirin, N. Srinivasan, and K. Sycara, "OWL-S: Semantic Markup for Web Services," http://www.w3.org/Submission/2004/SUBM-OWL-S-20041122/, 2004.

[23] K. Kyriakos and P. Dimitris, "A Semantic QoS-Based Web Service Discovery Algorithm for Over-Constrained Demands," *Proc. Third Int'l Conf. Next Generation Web Services Practices,* 2007.
[24] N. Limam and R. Boutaba, "Assessing Service Quality and Trustworthiness at Selection Time," *Proc. 11th IEEE/IFIP Network Operations and Management Symp.,* 2008.
[25] T. Yu, Y. Zhang, and K.-J. Lin, "Efficient Algorithms for Web Services Selection with End-to-End QoS Constraints," *ACM Trans. Web,* vol. 1, no. 1, 2007.
[26] G. Wu, J. Wei, X. Qiao, and L. Li, "A Bayesian Network Based QoS Assessment Model for Web Services," *Proc. IEEE Int'l Conf. Services Computin,* 2007.
[27] S.J.H. Yang, J.S.F. Hsieh, B.C.W. Lan, and J.-Y. Chung, "Composition and Evaluation of Trustworthy Web Services," *Proc. IEEE Int'l Workshop Business Services Networks,* 2005.
[28] A.S. Ali, S. Majithia, O.F. Rana, and D.W. Walker, "Reputation-Based Semantic Service Discovery," *Proc. 13th IEEE Int'l Workshops Enabling Technologies: Infrastructure for Collaborative Enterprises,* 2004.
[29] R. Wishart, R. Robinson, J. Indulska, and A. Josang, "Superstringrep: Reputation-Enhanced Service Discovery," *Proc. 28th Australasian Conf. Computer Science,* 2005.
[30] F. Raimondi, J. Skene, and W. Emmerich, "Efficient Online Monitoring of Web-Service SLAs," *Proc. 16th ACM SIGSOFT Int'l Symp. Foundations of Software Eng.,* pp. 170-180, 2008.

Noura Limam received the BS degree from the National School of Computer Science, Tunisia, in 2001, and the MSc and PhD degrees from the University of Paris VI, France, in 2002 and 2007, respectively. She is currently a postdoctoral fellow in the Division of IT Convergence Engineering at POSTECH, Republic of South Korea. Formerly, she was a researcher at Ucopia Communications, Inc., France, and a research assistant at the School of Computer Science at the University of Waterloo, Canada. She received the Fred W. Ellersick Prize IEEE Communications Society Award in 2008. Her research interests include service management, service engineering, and service-oriented architectures.

Raouf Boutaba is a professor of computer science and a Cheriton Faculty Fellow at the University of Waterloo (Canada). His main research interests are in network, resource and service management. He is the founding Editor-in-Chief of the *IEEE Transactions on Network and Service Management* and is on the editorial boards of other journals. He served as a distinguished lecturer of the IEEE Communications and the IEEE Computer Societies. He also served as the chairman of the IEEE Technical Committee on Information Infrastructure and the IFIP Working Group on Network and Distributed Systems Management. He has received several recognitions, such as the Premiers research excellence award, the IEEE Harold Sobol, Fred W. Ellersick, and Joe LoCicero awards, and the Don Stokesburry award. He is a senior member of the IEEE.