An Agent-Based Service Marketplace for Dynamic and Unreliable Settings

Lina Barakat, Samhar Mahmoud, Simon Miles, Adel Taweel, and Michael Luck

Department of Informatics, King's College London, London, UK
{firstname.surename}@kcl.ac.uk

Abstract. In order to address the unreliable nature of service providers, and the dynamic nature of services (their quality values could change frequently over time due to various factors), this paper proposes a probabilistic, multi-valued quality model for services, capable of capturing uncertainty in their quality values by assigning each quality attribute with multiple potential values (or ranges of values), along with a corresponding probability distribution over these values. The probability distribution indicates the most likely quality value for an attribute at the current time step, but also notifies discovery applications of the possibility of other, possibly worse outcomes, thus ultimately facilitating more reliable service selection and composition via avoiding services with high uncertainty. Such uncertainty-aware, multi-valued quality models of services are maintained via an agent-based service marketplace, where each service is associated with a software agent, capable of learning the time-varying probability distributions of its quality values through applying online learning techniques, based on the service's past performance information. The experiments conducted demonstrate the effectiveness of the proposed approach.

Keywords: quality of service, probabilistic quality model, adaptive learning, dynamic environment, agent based marketplace.

1 Introduction

Service-oriented computing (SOC) is a promising paradigm for the sharing of resources and functionalities in open, distributed environments (e.g., the web and computational Grids). Via exposing such resources and functionalities as *services* [1], and utilising these services as elementary building blocks, this paradigm supports the rapid and economic development of complex, interoperable distributed applications.

Open distributed service-based systems, however, usually exhibit high degrees of dynamism and uncertainty for several reasons, either intentional or unintentional. For example, service providers, being autonomous and self-interested, may choose to act maliciously and announce false quality of service (QoS) capabilities in order to increase their own profit by attracting more customers. Even in cases where the providers are fully cooperative, it might be difficult (or simply

X. Franch et al. (Eds.): ICSOC 2014, LNCS 8831, pp. 169–183, 2014.

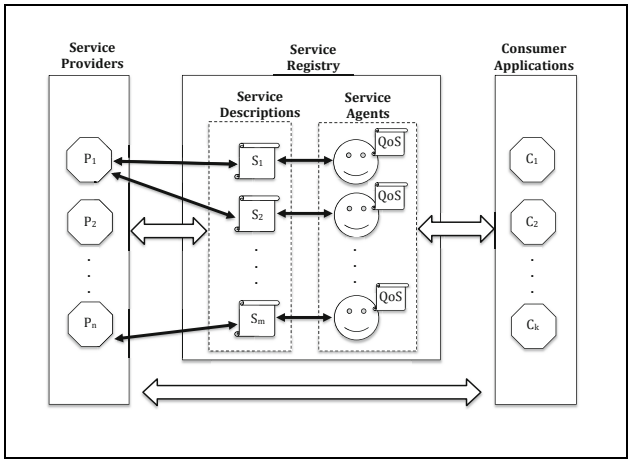

Fig. 1. Agent-augmented service marketplace architecture

not possible) to guarantee specific quality values for a service, because of their dependency on various run-time factors. For instance, the service response time at any particular moment could be significantly affected by the provider load and network traffic at that moment. Such dynamism and uncertainty can lead to highly undesirable situations during service execution (e.g. unfulfilled quality promises), and may demand costly corrective actions.

Consequently, as an attempt to minimise quality deviations of services at execution time, a number of efforts focus on providing more accurate estimation of service quality values, based on the available information regarding their past performance [9–13]. Specifically, assessing a quality attribute for a service is typically performed by applying some aggregation measure (e.g. a time-weighted average) to the previously observed values, which are obtained as feedback from service users, or from service-side monitors. Such a *single-valued* quality estimation model, however, does not capture the uncertainty in the service's quality values, and might produce inaccurate or invalid quality predictions, especially for attributes with high variance in values. For example, assume the values encountered in the past regarding the learning time attribute of a knowledge service are 10, 10, 10, 60, 60, 60 (minutes). Estimating the mean of these values would produce an expected value of 35 minutes, an imprecise indication of the attribute's actual outcome. Moreover, such a model is only limited to quantitative attributes, without the ability to accommodate qualitative cases.

In response, this paper proposes a probabilistic *multi-valued* quality estimation model, applicable to both numeric and categorical attributes. It captures uncertainty in quality values by augmenting these values with reliability scores, allowing more informative reasoning about the various potential quality outcomes of a service, thus enabling more reliable and proactive service selection. The responsibility of instantiating such quality models for services is distributed among a number of learning-enabled software agents, applying online learning

techniques to update the models on the availability of new service performance samples, without requiring storage of or iteration over all previous data.

The paper is organised as follows. The proposed agent-based service marketplace is introduced in Section 2. Section 3 and Section 4 present the basic (single-valued) service QoS model [7] and the proposed reliability-aware (multi-valued) extension, respectively. Section 5 provides the online quality learning algorithm, while Section 6 evaluates its effectiveness through experimental results. Section 7 discusses related work, and Section 8 concludes the paper.

2 Agent-Augmented Service Marketplace

A basic service marketplace, adopting the classic service-oriented architecture, provides support for the publication, description, discovery, and invocation of services, and involves interaction among three roles, the service provider, service consumer, and service registry. Specifically, a service provider describes its service using a standard format, and publishes this description in a public service registry so that the service can be discovered by potential clients. A service consumer searches the service registry to find a required service, and retrieve its binding details, which are then utilised to locate and invoke the service. Note that such a consumer could be an end-user application, a matchmaker agent (returning services that meet specific criteria), or a service composition engine (aggregating the functionalities of existing services into more sophisticated composite applications in order to fulfil particular high-level goals).

In addition to functional specification, service descriptions could also reference the quality of service (QoS) characteristics of services, indicating their non-functional capabilities. These attributes can be generic, such as price and response time, or domain-dependent, representing specific features and metrics of a particular domain. The QoS characteristics play an important role in differentiating between functionally equivalent services (those overlapping in their functional capabilities, but possibly varying in their QoS levels), and accommodating the different expectations of users (individuals or organisations). Yet, as stated earlier, the features advertised by service providers are not necessarily reliable, due to the untrustworthiness of these providers, and the dynamic nature of service environments, causing the quality values of services to deviate over time as a result of various environmental factors (which might be difficult to anticipate by providers). This could result in unfulfilled quality promises by services, and consequently a number of negative effects on the applications utilising these services, including unsatisfied users, money loss, or interruption in application execution while performing recovery re-planning.

To address this, we propose an extended service registry (see Figure 1), facilitating more reliable and self-adaptive service descriptions via the utilisation of software agents, capable of learning the actual QoS characteristics of services, and adapting their descriptions according to changes. Specifically, a *learning-enabled service agent* resides between a service description published by a provider and any discovery application, and exposes reliability and dynamism aware QoS information of the service to the latter by learning such

information based on collected service ratings after each interaction with the service. The ratings can be collected either directly from consumers via feedback interfaces, or automatically via appropriate monitors residing at the service side or over the network. Note that we assume in this paper that the ratings are honest and objective (false ratings can be handled through appropriate filtering and reputation mechanisms [15], but this is out of the scope of this paper).

In what follows, we first outline the traditional QoS model of services (the model corresponding to provider advertisements), and then focus on modelling the service agent, including an improved QoS model, augmenting the traditional model with reliability information, and a learning algorithm.

3 Basic QoS Model

The QoS model of a service registered within a marketplace can be defined as a tuple, $(AN, dom, type, value)$, as detailed below.

AN is the set of quality attributes that characterise the service. For example, $AN = \{\text{price}, \text{response time}, ...\}$.

$dom : AN \rightarrow 2^{AV}$ is an attribute domain function, which maps each quality attribute to its corresponding domain (the possible values of this attribute), where AV is the set of all possible quality attribute values (the union of the domains of all quality attributes). For example, $dom(\text{price}) = \mathbb{R}^+$.

$type : AN \rightarrow \{\text{CTG}, \text{DSC}, \text{CNT}\}$ is an attribute type function, which indicates whether the domain of a quality attribute is categorical (CTG), numeric and discrete (DSC), or numeric and continuous (CNT). For example, $type(\text{price}) = \text{CNT}$.

Finally, $value : AN \rightarrow AV \cup \{\text{undefined}\}$ is an attribute value function, which provides the value offered by the service for each quality attribute, such that $\forall a \in AN,\ value(a) \in dom(a) \cup \{\text{undefined}\}$. For example, $value(\text{price}) = 10$.

Generally, the QoS information of a service can be directly published into public service registries by the service provider, assessed from monitoring previous service performance by specialised proxies, or negotiated with the provider in terms of service level agreements (SLAs).

4 Uncertainty-Aware QoS Model

In order to capture uncertainty in the values that a service might deliver for the quality attributes at a particular time step $t \in T$, the service agent utilises a time-dependent, probabilistic model for describing the QoS features of the service. Specifically, each quality attribute is considered to be a random variable, and is associated with a probability distribution indicating the likelihood of each of its possible values at the current moment. Hence, to reflect this, the static single-valued attribute value function of the basic service model is modified by the service agent, as follows:

$$value_{ag} : AN \times T \rightarrow PROB$$

such that $\forall a \in AN,\ \forall t \in T,\ \ value_{ag}(a,t) = P(a,t)$ is the probability distribution over the possible outcomes of attribute a at time step t (with $PROB$ denoting the set of all possible probability distributions P). That is,

$$value_{ag}(a,t) = P(a,t) = \{p(a,v,t) \mid v \in dom^d(a)\}$$

with

$$\forall v \in dom^d(a),\ p(a,v,t) \in [0,1] \ \ \wedge \sum_{v \in dom^d(a)} p(a,v,t) = 1 \qquad (1)$$

where $p(a,v,t)$ is the likelihood of attribute a to take on value v at time step t, and $dom^d(a)$ is the discretised domain of attribute a. For categorical and discrete attributes, $a \in AN$ s.t. $type(a) \in \{\text{CTG}, \text{DSC}\}$, domain $dom^d(a)$ corresponds to the original value space, i.e. $dom^d(a) = dom(a)$. For continuous attributes, $a \in AN$ s.t. $type(a) = \text{CNT}$, domain $dom^d(a)$ is obtained via applying an appropriate discretisation algorithm on the original value space $dom(a)$ (a simple example is dividing $dom(a)$ into a number of equal ranges, with values $v \in dom^d(a)$ corresponding to the respective range representatives).

Such a probabilistic, multi-valued modelling of service quality features exposes more accurate and comprehensive details regarding the expected behaviour of the service, facilitating more informative and reliable service selection and accommodating the different needs of discovery applications, as opposed to the single-valued approach, where the discovery application is limited to a single, possibly inaccurate, summary attribute value. In particular, while general indications of the quality features may be sufficient for some discovery applications, others, performing more critical tasks, might favour accounting for the worst case scenario (i.e. selecting services by analysing the least desirable quality values that are probable in their cases). Note that, in the proposed approach, the expected (average) value for a numeric attribute a at time step t, $exp(a,t)$, can be easily derived from probability distribution $P(a,t)$, as follows:

$$exp(a,t) = \sum_{v \in dom^d(a)} v \times p(a,v,t)$$

Example. Consider a content provider, in the e-learning domain, offering a learning object (service) characterised by three quality properties, learning time (LT), difficulty level (DL), and interactivity type (IT), with:

$$type(\text{LT}) = \text{CNT} \ \wedge \ dom(\text{LT}) = [10,60]$$
$$type(\text{DL}) = \text{DSC} \ \wedge \ dom(\text{DL}) = \{1,2,3,4,5\}$$
$$type(\text{IT}) = \text{CTG} \ \wedge \ dom(\text{IT}) = \{\text{active}, \text{expositive}, \text{mixed}\}$$

Figure 2 shows example probabilistic value models for the learning object regarding the three quality properties, illustrating various uncertainty levels at time step t, with the highest uncertainty in value corresponding to attribute DL, where $p(\text{DL},v,t) = 0.2$ for all values $v \in \{1,2,3,4,5\}$.

Note that learning objects meet the conditions of our generic service definition, since they are self-contained, reusable units of instruction, which are made

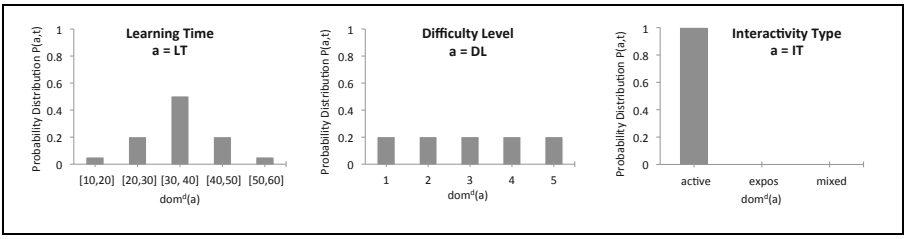

Fig. 2. Probabilistic attribute value function $value_{ag}(a,t)$ for a learning object

available for discovery through dedicated repositories, where their properties are described using a standard language (e.g. IEEE LOM[1]).

5 Learning Model

The learning problem of the service agent concerns devising a reliable QoS description for the service in the presence of uncertainty in the environment, where service providers might be untrustworthy, and may change their QoS policies without notification, either intentionally or unintentionally. Such learning is conducted by observing the behaviour of the service over time, and adapting its description accordingly. Specifically, the cycle of the service agent involves the following three steps.

(1) *Observe.* The agent receives new ratings for the service at time step t (e.g. user feedback after interaction with the service). Let $obs(t) = \{(a, rating(a,t)) \mid a \in AN\}$ denotes such ratings, where function $rating(a,t) \in dom^d(a)$ maps quality attribute a of the service to the value observed for a at time step t.

(2) *Learn.* The agent utilises the new observation history to update the probability distributions of the quality values of the service, so that the service behaviour is more accurately described for future selection. In other words,

$$value_{ag}(a,t) = P(a,t) = qoslearn(a, OBS_t) \tag{2}$$

where $OBS_t = \{obs(i)\}_{i=1}^{t}$ are the past observations of the service up to time t, and function *qoslearn* corresponds to the agent's learning algorithm.

(3) *Expose.* The agent makes the probability distributions, $value_{ag}(a,t)$, of the service's quality attributes $a \in AN$, available to discovery applications as the best generalisation of the behaviour of the service at time step t.

Next, we define the properties that need to be satisfied by the learning function *qoslearn*, followed by a learning algorithm achieving these properties.

5.1 Learner Requirements

Two desirable properties can be identified for a QoS learning algorithm: adaptivity and efficiency.

[1] `http://ltsc.ieee.org/wg12/`

Adaptivity refers to the ability of the learning algorithm to incorporate evolving data over time. It is required in the context of both stationary and non-stationary environments.

In a stationary environment, the underlying probability distributions of the service's quality values remain constant over time, but might not be known in advance due to missing, inaccurate, or untrustworthy QoS descriptions from providers. Hence, since observations of the service's actual behaviour only arrive incrementally (are not available at once), the learning algorithm should be able to increase the accuracy of the predicted quality model with more incoming data.

In a non-stationary environment, the probability distributions of quality values may experience changes over time, and therefore the learning algorithm should be able to accommodate these changes on their occurrence. Generally, probability distribution drifts may follow various patterns. A drift might occur *abruptly*, by suddenly switching from one probability distribution to another at some time step. Examples of such a drift include a significant degradation in a service's availability due to an unexpected network problem, or modification of service characteristics caused by an implementation change (e.g. additional content is added to the learning object, correspondingly affecting its learning time, difficulty level, etc). Alternatively, a drift may happen *gradually*, with the probability distribution exhibiting smaller differences over a longer time period. Examples of such a drift include a slow deterioration or improvement in a service's response time with increasing or decreasing load, respectively, during the day, or a gradual performance degradation of a hardware service due to wearing out with time.

Efficiency refers to the ability of the learning algorithm to operate in a timely and memory-effective manner. Specifically, since the learning is conducted at run time, i.e. while the service is in operation, sensitivity towards time limits becomes a critical feature to ensure that the learning cycle terminates, and consequently the QoS descriptions of the service are updated, prior to the next discovery attempt by a service consumer. Moreover, with the potentially continuous and long-lasting data input (for the duration of service operation), memory consumption is a major concern, and maintaining access to the whole set of past service data is very costly. In the most efficient form (both in terms of memory and processing time), data is discarded once the service's QoS model is updated upon data arrival, with the update being performed using the latest version of the model. That is, function *qoslearn* in Equation 2 should be modified as follows:

$$value_{ag}(a,t) = P(a,t) = qoslearn(a, obs(t), P(a,t-1)) \tag{3}$$

5.2 Learning Algorithm

For the purpose of instantiating the QoS learning function *qoslearn*, the service agent utilises an algorithm inspired by Policy Hill-Climbing [2], a rational learning algorithm for finding a policy that maximises an accumulative reward perceived from the environment. The idea of the proposed algorithm is as follows. At each learning cycle $t \in T$, and for each quality attribute of the service

Algorithm 1. Learning model of a service agent

1. Initialise the learning rate δ

2. $\forall a \in AN$, initialise $value_{ag}(a, t_0)$ according to the basic QoS model of the service:

2.1. if $value(a) \neq$ undefined **then**

$$\forall v \in dom^d(a), \quad p(a, v, t_0) = 1 \text{ if } v \text{ corresponds to } value(a)$$
$$p(a, v, t_0) = 0 \text{ otherwise}$$

2.2. else

$$\forall v \in dom^d(a), \quad p(a, v, t_0) = \frac{1}{|dom^d(a)|}$$

3. Repeat

3.1. Observe the behaviour of the service, $obs(t) = \{(a, rating(a, t)) \mid a \in AN\}$

3.2. Learn more accurate QoS policy, $value_{ag}(a, t)$, for each attribute $a \in AN$:

$$\forall v \in dom^d(a), \quad p(a, v, t) = (1 - \delta)p(a, v, t - 1) + \delta \quad \text{if } v = rating(a, t)$$
$$p(a, v, t) = (1 - \delta)p(a, v, t - 1) \quad \text{otherwise}$$

3.3. Expose $value_{ag}(a, t)$ to discovery applications

$a \in AN$, the currently maintained QoS policy $value_{ag}(a, t - 1)$ is improved according to a learning rate $\delta \in [0, 1]$, increasing the probability of the value with the highest utility (i.e. the value $v \in dom^d(a)$ observed for attribute a) at iteration t. The overall learning model of the service agent is illustrated in Algorithm 1, with the QoS policy re-evaluation rule (i.e. function *qoslearn*) being detailed at Line 3.2. It is easy to see that this rule keeps $value_{ag}(a, t)$ constrained to a legal probability distribution, i.e it satisfies Equation 1.

The agent initialises the QoS policy of each attribute a, $value_{ag}(a, t_0)$, according to provider advertisements, assigning a probability of 1 to the value indicated by the provider (Line 2.1 of Algorithm 1). In the case where an attribute is not instantiated by the provider, equal probabilities are initially assigned to all its possible values (Line 2.2 of Algorithm 1).

The QoS policy adjustment rule clearly satisfies Equation 3, thus fulfilling the *efficiency* requirement, while the choice of the learning factor δ governs the *adaptivity* property. Specifically, factor δ determines the rate at which the past data is forgotten, allowing a gradual discount of the impact of previous information and enabling responsiveness to more recent observations. As δ tends to unity, function *qoslearn* becomes a greedy function, removing the impact of all previous data up to step $t-1$, and accounting only for the latest observation. In contrast, lowering the value of δ decreases responsiveness to new data. Also, when factor δ is set to $\frac{1}{t}$ for each learning step t, all old and present observations are considered equally important. That is, in such a case, function *qoslearn* is

equivalent to simply deriving value probabilities from the frequencies of value occurrences over the entire set of observations up to time t, as illustrated below:

$$p(a,v,t) = \frac{\sum_{i=1}^{t} occur(a,v,i)}{t} = \frac{\sum_{i=1}^{t-1} occur(a,v,i) + occur(a,v,t)}{t}$$
$$= (\frac{t-1}{t})p(a,v,t-1) + (\frac{1}{t})occur(a,v,t)$$

where $occur(a,v,i) = 1$ if value v was observed for attribute a at time step i (i.e. $rating(a,i) = v$), and $occur(a,v,i) = 0$, otherwise. Further analysis of the effect of different values for factor δ is provided in Section 6.

6 Experiments and Results

In this section, we present an empirical evaluation of the proposed QoS learning framework, focusing on its performance in terms of producing reliable QoS estimates for services, in marketplaces of varying dynamism and uncertainty. The experiments are conducted on simulated datasets, allowing us to control the QoS policies of providers and their changes, thus facilitating evaluation under different settings.

A simulation run consists of a number of learning episodes (or cycles). At each episode t, the service provider delivers particular values for quality attributes $a \in AN$, which are observed by the service agent as ratings $rating(a,t)$. The generation of these quality values (i.e. the evaluation dataset) is governed by probability distributions $value_{prov}(a,t)$, representing the provider's actual QoS policy for each attribute a at time step t. Table 1 shows two distributions utilised in our experiments for specifying such policies: distribution Q_1 producing a fixed value v_i for attribute a, and normal distribution Q_2 over the possible values. Note that all the results reported are averaged over 100 simulation runs, and among different attribute types (i.e. categorical, discrete, and continuous). For simplicity, we only show the results from the perspective of one service and one quality attribute (other attributes and services exhibit similar trends).

Next, we first outline the strategies to be evaluated (Section 6.1) and the evaluation measure to be utilised (Section 6.2), followed by experimental results (Sections 6.3-6.4) and an overall result summary (Section 6.5).

6.1 Learning Strategies

Throughout the presentation of our experimental results, we refer to the following learning strategies.

SlideWindow_w: this is the sliding window learning strategy, a well known way of adapting to potential changes in incoming data and accommodating memory constraints [17]. It is adopted as a memory-based alternative for the purpose of estimating our model, as follows. At each time step, $value_{ag}(a,t)$ is rebuilt on a data window of size w storing the most recent w observations, according to the proportional frequencies of values in this window. Note that, by *SlideWindow_all*,

Table 1. Generative models of the provider's actual QoS values

$value_{prov}(a,t)$	Definition	Attribute Type
$Q_1(a,t)$	$q(a,v,t) = \begin{cases} 1 & \text{if } v = v_i \\ 0 & \text{if } v \in dom^d(a) \setminus \{v_i\} \end{cases}$ s.t. $v_i \in dom^d(a)$	$type(a) \in \{\text{CTG}, \text{DSC}\}$
$Q_2(a,t)$	Normal distribution over $dom(a)$, with mean μ and variance σ^2, s.t. $q(a,v,t) = \frac{1}{\sigma\sqrt{2\pi}} \int_a^b e^{-\frac{(x-\mu)^2}{2\sigma^2}} dx$ where $v \in dom^d(a)$ corresponds to range $[a,b]$ in $dom(a)$	$type(a) = \text{CNT}$

we refer to re-building the model using *all* the data observed so far, which is utilised as a baseline in our evaluation.

QoSLearn_δ: this is the learning strategy proposed in this paper, utilising a learning rate δ, with no memory requirement.

6.2 Evaluation Measure

In order to assess the reliability of the QoS model estimated by the service agent, we need to compare such a model against the actual QoS model of the provider. In other words, we are interested in quantifying the difference between the agent's estimated probability distribution, $value_{ag}(a,t) = P(a,t)$, and the provider's actual probability distribution, $value_{prov}(a,t) = Q(a,t)$, over the values of attribute a, at any time step t. For this purpose, we adopt the Hellinger Distance measure [3], which computes the distance, denoted $h(P,Q)$, between probability distribution $P(a,t) = \{p(a,v,t) \mid v \in dom^d(a)\}$, and probability distribution $Q(a,t) = \{q(a,v,t) \mid v \in dom^d(a)\}$, as follows:

$$h(P,Q) = \sqrt{\frac{1}{2} \sum_{v \in dom^d(a)} (\sqrt{p(a,v,t)} - \sqrt{q(a,v,t)})^2} \in [0, \sqrt{2}]$$

When $h(P,Q) = 0$, probability distributions P and Q are identical, whereas $h(P,Q) = \sqrt{2}$ corresponds to the maximum divergence between P and Q. Note that the Hellinger Distance measure is symmetric, i.e. $h(P,Q) = h(Q,P)$. Other probability distribution distances, e.g. the earth mover's distance (EMD), could also be utilised here.

6.3 Stationary Marketplace

In a stationary environment, the QoS policy of the service provider for attribute a remains static over time. Such an environment is simulated by generating

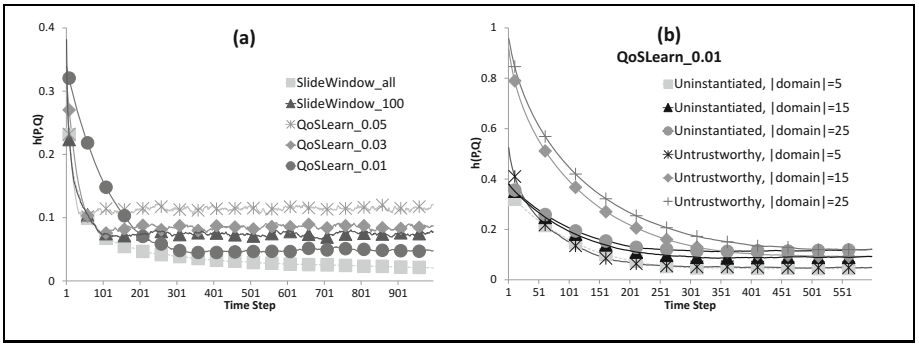

Fig. 3. Evaluation results in a stationary environment

the observations $rating(a, t)$, for all the time steps t, according to the same probability distribution $value_{prov}(a, t_0) = Q(a, t_0)$, s.t. $\forall$t, $Q(a, t) = Q(a, t_0)$ (i.e. value v_i and mean μ for distributions Q_1 and Q_2, respectively, remain fixed for all time steps). Here, we are interested in testing the ability of the proposed approach to learn probability distribution $Q(a, t_0)$, in the following two settings: *Untrustworthy Provider*, where the provider acts maliciously and advertises false capability $value(a)$ for attribute a, i.e. $value(a)$ does not correspond to v_i in the case of actual distribution Q_1 (see Table 1), and $value(a)$ differs significantly from μ in the case of actual distribution Q_2 (see Table 1); and *Uninstantiated Attribute*, where no performance indication regarding attribute a is available by the provider, i.e. $value(a) =$ undefined.

Figure 3(a) reports the results of the considered learning strategies. As expected, *SlideWindow_all* is the best performing strategy, with smaller window sizes achieving lower accuracy due to excluding relevant observations (all observations remain relevant in a static environment). By setting the learning rate δ to a small value of 0.01, the proposed learning strategy, *QoSLearn_0.01*, keeps the effect of older observations without necessitating their storage, and manages to approximate the performance of *SlideWindow_all*. However, such a small learning rate causes slower learning at the beginning, achieving an accuracy of about 0.2 only after 60 observations, compared to *SlideWindow_all* that achieves similar accuracy after just 15 observations. This initial learning period is further highlighted in Figure 3(b), distinguishing the two cases of *Untrustworthy Provider* and *Uninstantiated Attribute*, and varying the size of the attribute's domain dom^d. As can be seen, the effect of misleading providers generally takes longer to overcome, especially for a larger domain size, requiring a larger number of samples to accurately learn the actual underlying distribution.

6.4 Non-stationary Marketplace

In a non-stationary environment, the QoS policy of the service provider for attribute a changes over time. Two cases are distinguished depending on the change type, as detailed below.

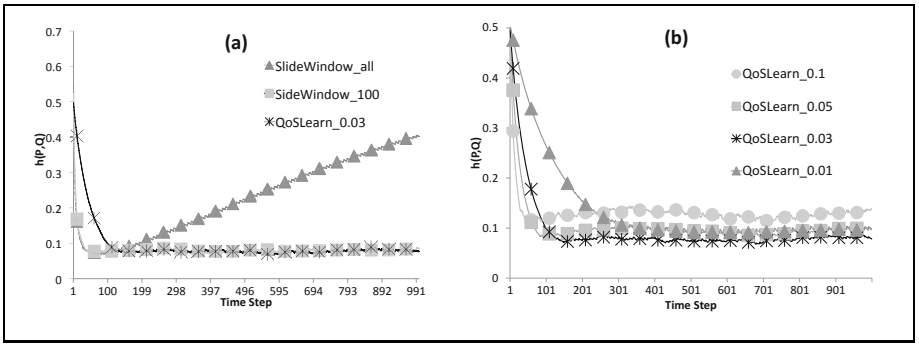

Fig. 4. Evaluation results in a dynamic environment (gradual change)

Gradual Change. Here, the generative model of the quality observations, $value_{prov}(a,t) = Q(a,t)$, changes slowly at each time step. For distribution Q_1, this is simulated by slightly decreasing the probability of value v_i, and correspondingly increasing the probability of another value $v_j \in dom^d(a) \setminus \{v_i\}$. For distribution Q_2, this is achieved by a slight repositioning of the mean μ.

Figure 4(a) shows the corresponding results of the considered learning strategies. As can be seen, the performance of *SlideWindow_all* slowly deteriorates with time as older observations become less relevant. Better prediction accuracy is obtained when the outdated data is gradually forgotten, favouring more recent observations, with well-performing strategies corresponding to settings *SlideWindow*_100 and *QoSLearn*_0.03 (note that Figure 4(a) only reports these settings for reasons of clarity). The effect of different learning rates for the proposed learning strategy is further studied in Figure 4(b), which demonstrates that setting δ to lower and higher values (in comparison with 0.03) would increase the prediction error due to intensifying the effect of irrelevant data and the lack of sufficient samples, respectively.

Abrupt Change. Here, the generative model of observations, $value_{prov}(a,t) = Q(a,t)$, experiences a considerable change every 200 time steps. Such a change is simulated by assigning probability 0 to value v_i and probability 1 to another value $v_j \in dom^d(a) \setminus \{v_i\}$ in the case of distribution Q_1, and switching to a significantly different mean μ in the case of distribution Q_2.

As depicted in Figure 5(a), strategy *QoSLearn*_0.05 (as well as *SlideWindow*_50) achieves a good tradeoff between reactivity and stability, allowing fast adaptation to a change ($h(P,Q)$ falls below 0.2 in less than 15 time steps) while assuring high accuracy ($h(P,Q) < 0.1$) in times of stability. *SlideWindow_all*, on the other hand, suffers from poor performance, especially after a change occurrence, where the learned model mostly reflects irrelevant observations. Figure 5(b) provides further analysis of the performance of strategy *QoSLearn*_δ between consecutive change points, for various learning rates δ. Clearly, the larger the learning rate, the quicker the observations are forgotten, resulting in

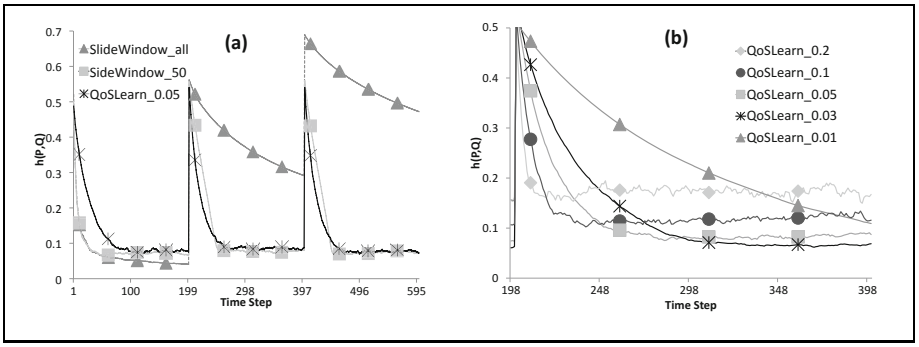

Fig. 5. Evaluation results in a dynamic environment (abrupt change)

faster reactivity after a change, but lower performance in stable periods. In contrast, smaller learning rates improve the accuracy of the learner due to capturing enough samples to reflect the current distribution, yet causes slower adaptation to a change since irrelevant data takes longer to be forgotten.

6.5 Result Summary

The results above demonstrated that, for both *SlideWindow_w* and *QoSLearn_δ*, appropriate parameter setting plays an important role for achieving accurate learning of the probability distributions of quality values, and depends on the environment dynamism. Moreover, the learning-rate-based strategy performs almost as good as the memory-based one, while achieving considerable saving in terms of storage and computation, especially with the increasing dimensionality and number of services in the marketplace. For instance, given a marketplace with 1000 services, each with 10 quality attributes, applying *QoSLearn_δ*, as compared to *SlideWindow_100*, in a gradually changing environment eliminates the need for storing and iterating over 100×10^4 quality ratings at each time step, with the gain increasing in static environments (which require larger window sizes).

7 Related Work

The dynamism and uncertainty of open distributed service-based systems, where the QoS features of the comprising services are unreliable and may exhibit high volatility, have been recognised by many researchers. Existing approaches in this regard can be categorised into reactive approaches and preventive approaches.

Reactive approaches aim at fault-tolerance during QoS-based service selection [8] or application execution [4–6], via performing appropriate corrective actions (e.g. service re-planning) that are triggered in reaction to a change or erroneous behaviour of a service. Such approaches, however, may suffer from

undesired effects such as reduced performance due to a high re-planning overhead, and in some cases, inability to find a satisfactory solution given the already executed services.

In response, preventive approaches (under which this paper falls) have been proposed. These aim at fault-avoidance through providing more accurate estimation of service quality values (typically from prior observations of service behaviour), thus allowing the discovery of more suitable services and minimising quality deviations at run time. A number of such efforts are concerned with modelling volatility in service response time, very often caused by the network. In this regard, Dai et al. [11] and Yang et al [12] predict changes in data transmission time (and consequently service response time) through a Semi-Markov Process. Aschoff et al. [10] model the response time of a service as a random variable, changing as a result of various factors related to the network and system resources (e.g. request queuing time). The exponentially weighted moving average is utilised for estimating the expected value of this variable at a particular time step, according to historical data. Similarly, time series modelling based on ARIMA (AutoRegressive Integrated Moving Average) has been proposed by Amin et al. [9] for the purpose of QoS forecasting. These approaches, however, mostly produce a single-valued quantification per quality attribute, and hence may suffer from inaccurate predictions, do not support reasoning about attribute value uncertainty, and are not suitable for categorical attributes.

Trust and reputation mechanisms have also been considered for the purpose of accurate quality predictions [13, 14, 16]. In particular, prior to an interaction with a service, an assessment of its overall trustworthiness [14, 16] or the trustworthiness of each of its QoS dimensions [13] is undertaken, to avoid selecting services that may not honour their promises. Typically, such an assessment is performed by producing reputation scores for the service based on feedback collected from its users (e.g. calculating a time-weighted average of past service ratings). Again, the reputation measures in these approaches are summarised by single aggregative values, thus suffering form similar limitations as above.

In contrast, we propose a probabilistic, multi-valued estimation model, which predicts multiple potential outcomes per quality attribute, and augments such outcomes with uncertainty degrees, thus facilitating more informative reasoning and reliable service selection. Moreover, it is applicable to both numeric and categorical attribute types.

8 Conclusion

The paper presented a probabilistic QoS learning model, tailored towards dynamic and untrustworthy service environments, where each service is associated with a software agent, able to learn, based on past performance information, the uncertainty degrees regarding the service's quality outcomes in the form of probability distributions over such outcomes. The learning is both efficient and adaptable to various degrees of environment dynamism via an appropriate choice of the learning rate, which is demonstrated through experimental results.

Future work involves investigating more complex stochastic models for the dynamic adjustment of the learning rate during the learning process when environment dynamics change over time, as well as accommodating the addition of new quality characteristics. Moreover, we intend to explore the social ability of software agents (e.g. collaboration among those monitoring services for the same provider) to improve QoS predictions in the proposed marketplace architecture, where the role of agents has been limited so far to individual learning.

References

1. Papazoglou, M.P., Traverso, P., Dustdar, S., Leymann, F.: Service-oriented Computing: State of the Art and Research Challenges. Computer 40, 38–45 (2007)
2. Bowling, M., Veloso, M.: Rational and Convergent Learning in Stochastic Games. In: 17th Int. Joint Conf. on Artificial Intelligence, pp. 1021–1026 (2001)
3. Simpson, D.G.: Hellinger Deviance Tests: Efficiency, Breakdown Points, and Examples. Journal of the American Statistical Association 84, 107–113 (1989)
4. Zeng, L., Benatallah, B., Ngu, A.H.H., Dumas, M., Kalagnanam, J., Chang, H.: QoS-aware Middleware for Web Services Composition. IEEE Trans. Softw. Eng. 30, 311–327 (2004)
5. Ardagna, D., Pernici, B.: Adaptive Service Composition in Flexible Processes. IEEE Trans. Softw. Eng. 33, 369–384 (2007)
6. Canfora, G., Penta, M.D., Esposito, R., Villani, M.L.: QoS-Aware Replanning of Composite Web Services. In: IEEE Int. Conf. on Web Services, pp. 121–129 (2005)
7. Barakat, L., Miles, S., Poernomo, I., Luck, M.: Efficient Multi-granularity Service Composition. In: IEEE Int. Conf. on Web Services, pp. 227–234 (2011)
8. Barakat, L., Miles, S., Luck, M.: Efficient Adaptive QoS-based Service Selection. Service Oriented Computing and Applications (2013), doi:10.1007/s11761-013-0149-z
9. Amin, A., Colman, A., Grunske, L.: An Approach to Forecasting QoS Attributes of Web Services Based on ARIMA and GARCH Models. In: IEEE Int. Conf. on Web Services, pp. 74–81 (2012)
10. Aschoff, R., Zisman, A.: QoS-driven proactive adaptation of service composition. In: Kappel, G., Maamar, Z., Motahari-Nezhad, H.R. (eds.) Service Oriented Computing. LNCS, vol. 7084, pp. 421–435. Springer, Heidelberg (2011)
11. Dai, Y., Yang, L., Zhang, B.: QoS-driven Self-healing Web Service Composition Based on Performance Prediction. Journal of Computer Science and Technology 24, 250–261 (2009)
12. Yang, L., Dai, Y., Zhang, B.: Performance Prediction Based EX-QoS Driven Approach for Adaptive Service Composition. Information Science and Engineering 25, 345–362 (2009)
13. Maximilien, E.M., Singh, M.P.: Agent-based Trust Model Involving Multiple Qualities. In: 4th Int. Joint Conf. on Autonomous Agents and Multiagent Systems, pp. 519–526 (2005)
14. Xu, Z., Martin, P., Powley, W., Zulkernine, F.: Reputation-Enhanced QoS-based Web Services Discovery. In: IEEE Int. Conf. on Web Services, pp. 249–256 (2007)
15. Vu, L., Hauswirth, M., Aberer, K.: QoS-based Service Selection and Ranking with Trust and Reputation Management. In: The Cooperative Information System Conference, pp. 446–483 (2005)
16. Malik, Z., Bouguettaya, A.: RATEWeb: Reputation Assessment for Trust Establishment among Web Services. The VLDB Journal 18, 885–911 (2009)
17. Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., Bouchachia, A.: A Survey on Concept Drift Adaptation. ACM Computing Surveys 46, 1–37 (2014)