An Approach for Context-Aware Service Selection Using QoS and User Preferences

Mohcine Madkour[1], Mohamed Bakhouya[2], Abdelilah Maach[1], and Driss El Ghanami[1]

[1] Ecole Mohammadia d'Ingenieurs, Mohamed V-Agdal University, Rabat, Morocco
mouhcine.madkour@gmail.com, {maach,elghanami}@emi.ac.ma
[2] Aalto University, Otakaari 4, 00076, Aalto, Finland
mohamed.bakhouya@aalto.fi

Abstract. Ubiquitous computing refers to building a global computing environment where seamless and invisible access to computing resources is provided to users. The successful application of pervasive services relies on its ability to provide efficient and cost-effective QoS (Quality of Service) support. This paper introduces a service selection approach based on QoS and user preferences. Simulations have been conducted and preliminary results are reported to show the usefulness and the importance of using QoS and user preferences in service selection process.

Keywords: Ubiquitous and pervasive computing, Context-aware service selection, service adaptation, QoS-aware, user preferences and service policies.

1 Introduction

Ubiquitous computing has revolutionized the way humans interact with the world around them. The widespread use of wearable devices and the seamless connectivity between them have transcended the traditional computing era to the pervasive computing era. The later enables new opportunities for a user to perform his/her operations everywhere and anytime [16,17,5]. Each user has different expectations for a particular service, which requires a context-aware and user-centric selection process. However, the process has to overcome challenges exposed by uncertainty and fuzziness of context information. These challenges make it difficult to select a service that provide the appropriate QoS and meet user preferences. Current approaches for service selection are typically based on exact matches between offered and requested service capabilities. These approaches are inefficient when the context is fuzzy or uncertain [3,4].

In ubiquitous and pervasive environments, a service usually possesses a set of QoS parameters, though many of them are of dynamic nature, i.e., related to the service execution environment, a service can still advertise its assumed QoS [6,7]. QoS policies are rules related to QoS parameters, such as bandwidth and response time. QoS policy of services includes both functional and non-functional properties. Functional properties can be measured in terms of throughput, latency, response time, whereas

M. Matera and G. Rossi (Eds.): MobiWIS 2013 Workshops, CCIS 183, pp. 110–119, 2013.

non-functional properties contain non quantifiable metrics, such as integrity, reliability, availability, and security [1]. Therefore, QoS parameters should be taken into account by the service selection process. Furthermore, user preferences are underlying criteria that are required to make the service adaptable and useful.

The contribution of this paper is twofold: First, we focus on non-functional and QoS aspects in service description to select a set of services according to their appropriateness to quality attributes of user context. Second, we define a policy-based adaptation scheme as the automatic selection of the best policy for delivering the service to user. The remainder of this paper is organized as follows. Section 2 presents briefly related work. Section 3 introduces the selection process and describes the policy-based and QoS-aware adaptation scheme. Section 4 presents the preliminary simulation results. Conclusions and future work are given in Section 5.

2 Related Work

Service descriptions are used to advertise the service capabilities, interface, behavior, and quality. Publication of such information about available services provides the necessary means for discovery and selection of services [17]. In particular, the QoS description [2] provides functional and non-functional service quality attributes, such as service metering and cost, performance metrics (e.g., response time), security attributes, integrity, reliability, scalability, and availability. Moreover, quality of service plays an important role in automatic service selection. It is mainly used to establish valid and reliable service and identify the best offers from a set of functionally similar ones.

Recent studies emphasized that selection process can be viewed as graph matching problem. More precisely, with the development of service-oriented computing environments, QoS-aware service selection has been a more and more important research issue. In service selection, QoS evaluation in matching process are always aggregated for computing the QoS of matches, which has been reported in many previous studies. The work [14] and [15] emphasized the definition of QoS aspects and metrics. In [12], a model for web service discovery with QoS constraints, called WSDM-Q, is proposed. A set of QoS categorization *tModel* and a kind of reputation categorization *tModel* are defined in the model to describe the QoS attributes of a service and the degree of guaranteed QoS delivered by a service respectively. The concept of quantification is introduced in the model to transform between the QoS attributes and QoS categorizations. In [13], a QoS-Guaranteed and distributed mechanism of Web service discovery is proposed. It supports Web service discovery with QoS constraints and enhances the QoS of service discovery system.

However, the above approaches have several limitations. For example, there are situations that some QoS attributes cannot be matched exactly with user query due to the fuzziness of context. In other words, it is difficult to get from user query the same expressions as in service description, e.g., a user wants a hotel that is five min far away from the city center, while in service description we acquire the hotel is 15 km from city center, or a user requests maximum reliability, whereas in the description

the reliability is set to be 75%. Furthermore, QoS values always vary based on different contexts for different types of service invocation. When multiple preferences exist over the QoS attributes of services, it is always difficult to maximize all the QoS attributes because there might always be anti-correlated relations between them.

3 Selection and Adaptation Process

3.1 Service Selection

A process model can be represented as a workflow graph, where the vertices represent "activities" and the edges represent the precedencies between "activities". Fig. 1-a and Fig.1-b show an example of workflows for user's query and service respectively. They are represented as graphs annotated with semantic QoS attributes, similar to the approach proposed in [8].

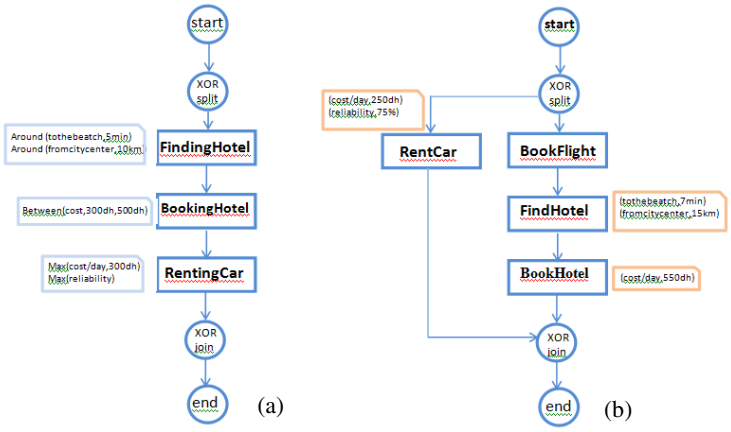


Fig. 1. a) Request workflow graph, b) Service workflow graph

In order to select the best suitable service for a given query two similarity functions are used, *structural similarity* and *semantic similarity*. For computing structural similarity, we consider that a process is represented by a graph $G=(N,E)$, where N is the nodes and E is the edges. Given two graphs, a similarity score can be obtained by computing their *graph-edit distance* as described in [9]. The *graph-edit distance* between two graphs is the minimal number of graph-edit operations that is required to get from one graph to another. This operation can be seen as a mapping process between two graphs. For example, let us consider $G_1=(N_1,E_1)$ and $G_2=(N_2,E_2)$ are two workflow graphs of process models. The mapping between G_1 and G_2 is defined as the partial injective mapping $M:(N_1 \rightarrow N_2) U (E_1 \rightarrow E_2)$ that maps nodes and edges. Indeed, to go from G_1 to G_2, three operations are possible: *i)* inserting nodes or edges, *ii)* deleting nodes or edges, and *iii)* substituting nodes or edges (the alteration of the label of node or edge).We assume that *ID* (Inserted or Deleted) is the set of all inserted and deleted nodes or edges and *SB* (SuBstituted) is the set of all substituted nodes or

edges. The mapping distance is defined as D=|ID|+|SB|+$\sum_{n,m\in M} 1 - SynSym(n,m)$, where *SynSym* is the syntactic similarity [22]. The *graph-edit distance* similarity, denoted by *Str-Sim*, is then computed as one minus the average of the following values: the fraction of cardinal of inserted or deleted nodes, the fraction of cardinal of inserted or deleted edges, and the average distance of substituted nodes [22]:

$$Str\text{-}Sim = 1 - \left\{ \frac{|ID|}{|N_1|+|N_2|}, \frac{|SB|}{|E_1|+|E_2|}, \frac{2.\sum_{n,m\in M} 1-SynSym(n,m)}{|N_1|+|N_2|-|ID|} \right\}$$

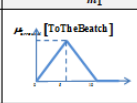
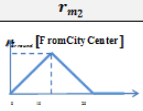
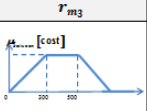
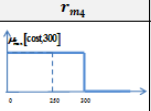
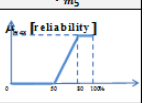
Instead of using a plain average of the three components, we can use a weighted average. It's worth noting that each service query process supports a set of QoS requirements according to the current context, such as throughput, reliability, response time, and cost. These QoS attributes are always attached to its semantic values. Usually the majority of QoS attributes of a given context are not exactly value-fixed in service request. Therefore, for evaluating the QoS attributes of a given context in a query process, we consider the membership functions that represent the predicates interpreting the quality attributes. For example, as illustrated in Fig. 1, where each pair *(r_i; a_j)* is linked with a membership function, r_i is a request attribute of node i and a_j is its corresponding annotation in applicant service *a*. An example is provided in Table 1. The work presented in this paper considers that a mapping algorithm is already given, such as the one presented in [10] in order to map two process models.

Table 1. QoS semantic similarity elements

Request attribute	Serive annotation	QoS Semantic evaluation
r_{m_1}: ***around*** (tothebeatch,**5**min)	a_{m_1}: (tothebeatch, **7min**)	$QSem - Eval(r_{m_1}, r_{m_1})$
r_{m_2}: ***around*** (fromcitycenter,**10**km)	a_{m_2}: (fromcitycenter, **15km**)	$QSem - Eval(r_{m_2}, r_{m_2})$
r_{m_3}: ***between***(cost,**300**dh,**500**dhs)	a_{m_3}: (cost/day,**550**dh)	$QSem - Eval(r_{m_3}, r_{m_3})$
r_{m_4}: ***max***(cost/day,**300**dh)	a_{m_4}: (cost/day,**250**dh)	$QSem - Eval(r_{m_4}, r_{m_4})$
r_{m_5}: ***max***(reliability)	a_{m_5}: (reliability, **75%**)	$QSem - Eval(r_{m_5}, r_{m_5})$

After mapping the two graphs, a list of nodes that match each other is obtained. Then, the semantic similarity can be computed from atomic pairs *(r_i; a_j)*. To do so, we consider the membership function of each semantic QoS expression in order to compute the truth degree of each expression r_i as shown in Table 2.

Table 2. QoS semantic similarity computing between matched pairs

Request attribute	r_{m1}	r_{m2}	r_{m3}	r_{m4}	r_{m5}
Membership function of the request attribute					
Evaluation	**0,65**	**0,5**	**0,9**	**1**	**0,85**

We aggregate the semantic similarity values by using the linguistic quantifier "***almost all***"[11]. More precisely, the natural interpretation of the similarities between a request *r* and an applicant *a* process model is **"**Almost all semantic preferences of *r* are satisfied by *a***"**. For example, using values provided in Fig. 1, the overall semantic QoS similarity can

be computed as follows: $\mathrm{QSem}-\mathrm{Eval}(r_{m_4}, a_{m_4}) = \mathbf{1} \geq \mathrm{QSem}-\mathrm{Eval}(r_{m_5}, a_{m_5}) = \mathbf{0,9} \geq \mathrm{QSem}-\mathrm{Eval}(r_{m_3}, a_{m_3}) = \mathbf{0,85} \geq \mathrm{QSem}-\mathrm{Eval}(r_{m_1}, a_{m_1}) = \mathbf{0,65} \geq \mathrm{QSem}-\mathrm{Eval}(r_{m_2}, a_{m_2}) = \mathbf{0}$. Therefore, the overall similarity degree can be obtained by computing the following expression $\boldsymbol{max}(\min(\mathbf{1}, \mu_Q\left(\frac{1}{5}\right), \min(\mathbf{0,9}, \mu_Q\left(\frac{2}{5}\right), \min(\mathbf{0,85}, \mu_Q\left(\frac{3}{5}\right), \min(\mathbf{0,65}, \mu_Q\left(\frac{4}{5}\right), \min(\mathbf{0,5}, \mu_Q\left(\frac{5}{5}\right)$. The value obtained is 0.65, which means that 65% of nodes are semantically similar. This is expected since there are three nodes among five that have a similarity degree greater than 0.65.

It should be noted that using these functions, several services could match the user request. In order to select the best suitable one, two ranking methods could be used. In the first method, the average value of both structural and semantic degrees (either with weighing them or without) is used. Thus the resulting degree handles both criteria, and the ranking is based on this overall degree. In the second method, the success rate for the structural degree is considered. If it reaches some degree, which is an acceptable matching, then we rank-order by only the semantic degree satisfaction, otherwise, the matching is not reachable and thus the process model is ignored. This service selection mechanism addresses the quality attributes in service description to enable the fully-automatic selection of services. It supports semantic QoS service description and matchmaking, works system-wide, and could yield to a good retrieval rate.

3.2 Policy-Based Adaptation

The adaptation approach taking into consideration service policies is described as follows. The first *M* services are first taken from the output of the context-aware service selection process described in Section 3.1. The number *M* depends on user preferences and service features to let for more flexibility in service selection and adaptation schemes. Indeed, the policies of the *M* selected services are considered in the adaptation process. This process is a fuzzy-based adaptation using fitness functions. More precisely, let us consider$S = \{S_1, S_2, , \dots, S_M\}$ a set of selected services, where $S_i, (1 \leq i \leq M)$ represents the *i-th* service, and *M* is the number of services. In the rest of this section, some definitions and terminologies are first given and adapted from the work presented in [18].

- **Policy:** represents a mode used to deliver a service with a certain resource requirement and a quality-of-service condition. Let's consider $P_i = \{p_i^1, p_i^2, \dots, p_i^{m_i}\}, i \in [1, M]$ a set of policies, which can be adopted for delivering the $\mathrm{i}-\mathrm{th}$ service, where m_i is the number of the policies of service S_i.
- **Preference Context:** Let's consider $C = \{c_1, c_2, \dots, c_n)$ a set of context attributes used to build preferences in order to monitor all services S_i, where c_i represents the *i-th* context information and *n* is the number of monitored contexts.
- **Preference Situation:** A preference situation (PS) is a combination of preferences. Let's consider $LV = \{Lv_1, Lv_2, \dots, Lv_k)$ a set of linguistic values. The preference situation at time *t*, denoted by $PS(t)$, can be represented by a set of 3-element tuples: $\boldsymbol{PS(t)} = \{(\boldsymbol{c_a}, \boldsymbol{Lv_b}, \boldsymbol{\mu_{c_a Lv_b}}(prefered - value(\boldsymbol{c_a}, t))\}$, where $(prefered - value(\boldsymbol{c_a}, t))$ represents the suitable value of context $\mathrm{c_a}$ at time *t*; and $\boldsymbol{\mu_{c_a Lv_b}}(x)$ is the predefined membership function of $c_a\ is\ Lv_b$".

It is worth noting that each policy p_i^j of $S_i \in S$is associated with a most suitable preference situation (PS).Therefore, the Standard Reference of $\mathrm{p_i^j}$ can be computed as follows: $SR(\mathrm{p_i^j}) = \{(\boldsymbol{c_a}, \boldsymbol{Lv_b}, \boldsymbol{\mu_{c_a Lv_b}}(Most - Suited - value(\boldsymbol{c_a}, t))$. $SRD(P_i)$ is the Standard Reference Depository, which represents the aggregation $\{SR(\mathrm{p_i^1}), SR(\mathrm{p_i^2}), \ldots SR(\mathrm{p_i^{m_i}})\}$. Therefore, the fuzzy adaptation process can be defined as a mapping process from the set of preferences situation PS to a set of all candidates' policies $P_{candidates}$. It aims to find the most suitable policies from $P_{candidates}$ by making the tradeoff between QoS and user preferences. So, the adaptation process computes the appropriate fitness function of all elements in $P_{candidates}$ in order to select the most suitable policy. We propose to use reciprocals of Manhattan [20], Minkowsky [19] and Chebychev [21] distances as fitness functions:

$$FF - Manhattan(\boldsymbol{p_i^j}) = \frac{1}{\sum_{k=1}^{sizeof(SR(p_i^j)} |\mu(Most - suited - value(\boldsymbol{c_k}, t)) - \mu(prefered - value(\boldsymbol{c_k}, t))|}$$

$$\mathrm{FF - Minkowsky}\ (\boldsymbol{p_i^j}) = \frac{1}{\sum_{k=1}^{sizeof(SR(p_i^j)} |(\mu(Most - suited - value(\boldsymbol{c_k}, t)) - \mu(prefered - value(\boldsymbol{c_k}, t))|^{p_k})^{1/r_k}}$$

$$\mathrm{FF - Chebychev}(\boldsymbol{p_i^j}) = \frac{1}{max_{k=1}^{sizeof(SR(p_i^j)} (|\mu(Most - suited - value(\boldsymbol{c_k}, t)) - \mu(prefered - value(\boldsymbol{c_k}, t))|)}$$

Where $sizeof(SR(p_i^j)$ represents the number of tuples in $SR(p_i^j)$, $\mu(x)$is the membership function predefined for i-th vector, and we consider two set of integers $p=\{p_k, k\ is\ integer\}$ and $r\{r_k, k\ is\ integer\}$. It is worth noting that in Minowky-based fitness function we independently configure the two set of powers p and r to find the balance between the large number of different elements and the importance of the difference between two terms.

The concept of fitness function is inspired by the membership function in classical fuzzy logic theory. But the two are different, i.e., the value of fitness degree is a positive number but not limited into [0,1], and the sum of fitness degree for all the policies of a service is not equal to 1. The value of fitness degree only indicates to which degree a policy is suitable for the current environment. In the above three functions, the divisors are for the calculation of the distance between PS and $SR(p_i^j)$. After obtaining the *crisp distance*, we compute its inverse to get the fitness degree. For instance, if the distance between PS and $SR(p_i^j)$ is 0, which means that current Preference Situation and $SR(p_i^j)$ have a perfect match, then the fitness degree is infinite. In general, the fitness degree decreases with the increase of the distance and vice versa.

4 Simulation Results

In order to evaluate the selection and adaptation process, we have used the following scenario. Let us consider that Otman is situated in a University campus and uses a PDA devise, which is equipped by a context-aware Campus Assistant application. Otman can then exchange emails and messages with his colleagues inside the campus. Let's assume that the application runs *chat* and *email* services. Otman uses the PDA to send and

receive emails through the university's email server, and to chat with colleagues as well. The services provided by the Campus Assistant application have alternative policies according to real-time context. For example, in order to distribute messages among chatting participants, Campus Assistant service helps in making a suitable choice among three policies: *textChat* to deliver text messages; *voiceChat* to exchange voice messages; and *videoChat* to exchange video messages. The email service operates in a similar fashion. Students can check their emails using one of the following five policies: *headMail*: which delivers only the mail header and a notice signifying that the transmitting operation is not over and will continue when an elevated network bandwidth and other circumstances are available; *fullMail*: which conveys mail in full-text; *encryptedMail*: which sends encrypted email; *bigMail*: which conveys email in full-text and attachments; and *encryptedBigMail*, which conveys encrypted big-emails.

We assume that the output of the selection process and their corresponding policies are: S = {chat, email}; P_1 ={textChat(p_1^1), voiceChat(p_1^2), videoChat(p_1^3)}; P_2 ={headMail (p_2^1) ,fullMail (p_2^2) ,encryptedMail (p_2^3) ,bigMail (p_2^4) , Mail (p_2^5)}. We consider the following context types over which the users have preferences:

C={Network_bandwidth(c_1),Processor_clockRate(c_2),Response_delay(c_3),Memory_freeSpace($c_{4)}$}; and we concider the linguistic terms: LV = {high, low}.

So, PS(current-time)={

($\mathbf{Network-maxRate}$, high, $\mu_{\text{Network-maxRate,high}}(prefered-value(network-maxRate,t))$),

($\boldsymbol{CPU-ClockRate}, high, \mu_{CPU-ClockRate,high}(prefered-value(CPU-ClockRate,t))$),

($\boldsymbol{Network-Delay}, low, \mu_{Network-Delay,Low}(prefered-value(Network-Delay,t))$,

($\boldsymbol{RAM-freeSpace}, high, \mu_{RAM-freeSpace,High}(prefered-value(\text{RAM}-\text{freeSpace},t))$)}

The membership function of the used linguistic terms is represented in Fig. 2. We have also other preferences such as User Privacy Intervention (UPI), if a user want to protect its privacy then $P_1' = \{p_1^1, p_1^2\}$ and $P_2' = \{p_2^1, p_2^3, p_2^5\}$, else $P_1' = P_1$ and $P_2' = P_2$, and energy saving, if the energy level of the battery is less than 5%, then$P_1' = P_1$ $and P_2' = \{p_2^1\}$. The suitable context values for all policies of P_1 and P_2 , considered in the evaluations, are shown in Table 3.

Table 3. Most suitable context values for policies

	c_1 (kbps)	c_2 (MHz)	c_3 (ms)	c_4 (MB)
p_1^1	4	20	500	0,2
p_1^2	200	300	10	4
p_1^3	10000	1000	0,2	200
p_2^1	2	4	---	0,2
p_2^2	10	10	---	0,4
p_2^3	10	100	----	10
p_2^4	500	50	---	2
p_2^5	500	1000	---	100

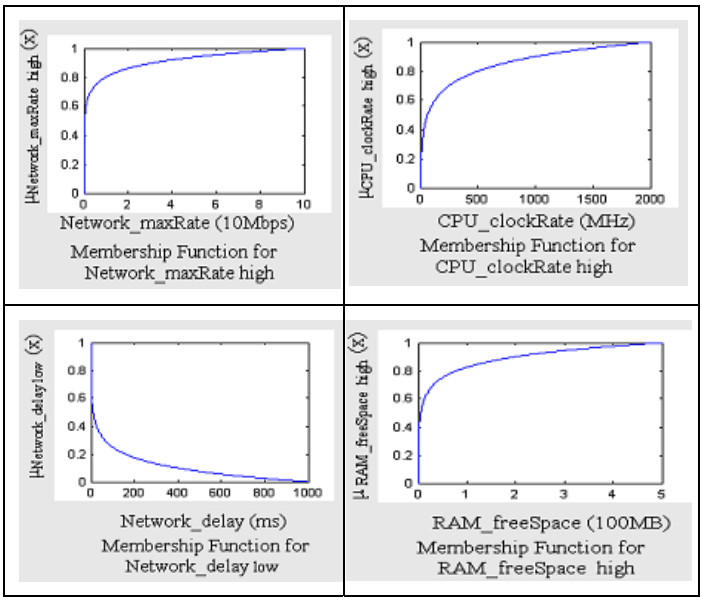


Fig. 2. Predefined membership functions

Network_bandwidth=10kps	*Processor_clockRate= 10 MHZ*	*Response_delay=100ms*
Memory_freeSpace= 10 MB	I_{UPI} *=0*	*Energy_Level= 40%*

Fig. 3. Current QoS information

After computing the preference situation, the service adaptation process can be performed. The following steps are then used to compute SRD using the values shown in Fig. 3:

- Use the fuzzy sets to interpret current preference situation: PS(current) = {(Network_maxRate, high, 0.20), (Processor_clockRate, high, 0.23), (Nework_delay, low, 0.25), (Memory_freeSpace, high, 0.58)}.
- Use privacy preferences and battery energy to filter P_1 and P_2, and get $P_1' = P_1$ and $P_2' = P_2$
- Use the membership functions depicted in Fig. 2 to calculate SRD(P_1') and SRD(P_2'). Results obtained are shown in Table 4.
- We simulate the three fitness functions. For the Minkowsky fitness function, we consider that all *r values are* equals to 1 and *p* takes 2 for chat service, for email service we consider *p=2* in case of Network_maxRate andProcessor_clockRate and *p=3* relative to Nework_delay and Memory_freeSpace. As so we consider that Network_maxRate and Processor_clockRate are more significant than Response_delay and Memory_freeSpace in making the decision for service adaptation. Results are depicted in Fig. 4. They show that the policy *VideoChat* is the most suitable adaptation strategy.

Several results have been also obtained using Minkowsky fitness function by considering different values of p and r. From those results, we realize that properly assigning values to these parameters will make Minkowsky distance-based fitness function much better to response to the current context. A systematic solution for automatically choosing the right values of elements of p and r is required; however it will lead to an optimization problem. Furthermore, selecting the best suitable fitness function could provide better adaptation policies to the actual context and user preferences.

Table 4. SRD(P_1') and SRD(P_2')

		$c_1, high$	$c_2, high$	$c_3, high$	c_4, low
SRD(P_1')	SRD(p_1^1)	0,12	0,33	0,08	0,15
	SRD(p_1^2)	0,46	0,72	0,50	0,48
	SRD(p_1^3)	0,80	0,90	0,92	0,90
SRD(P_2')	SRD(p_2^1)	0,06	0,10	---	0,15
	SRD(p_2^2)	0,20	0,23	---	0,23
	SRD(p_2^3)	0,20	0,57	---	0,58
	SRD(p_2^4)	0,54	0,47	---	0,40
	SRD(p_2^5)	0,54	0,90	---	0,83

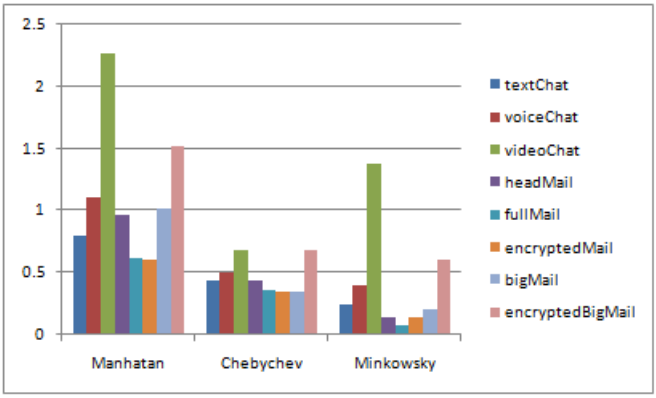


Fig. 4. Fitness degrees for service policies

5 Conclusions and Future Work

In this paper, we use QoS parameters and policies in both service selection and adaptation. The selection process allows getting the best-fitting services while the second is used for customizing the selected service to the actual context. The model represents QoS requirements and user preferences with fuzzy sets to allow mimicking the human manner in expressing desires and performing the aggregations. Ongoing work tackles the adaption issues in which parameters and the required fitness function could be automatically selected according to the context and user preferences.

References

1. Zhou, C., Chia, L.-T., Lee, B.S.: Semantics in Service Discovery and QoS Measurement. IT Professional 7(2), 29–34 (2005)
2. Chaari, S.: Youakim. B., Biennier, F.: Enhancing Web Service Selection by QoS-Based Ontology and WS-Policy. In: SAC 2008 Proceedings (2008)
3. Madkour, M., El Ghanami, D., Maach, A.: QoS-based approach for context-aware service selection with fuzzy preferences handling. JCAT 47(4), 379–391 (2013)
4. Madkour, M., El Ghanami, D., Maach, A.: Context-Aware Service Adaptation: An Approach Based on Fuzzy Sets and Service Composition. J. Inf. Sci. Eng. 29(1), 1–16 (2013)
5. Bakhouya, M., Campbell, R.H., Coronato, A., De Pietro, G., Ranganathan, A.: Introduction to special section on formal methods in pervasive computing. ACM Transactions on Autonomous and Adaptive Systems 7(1), 6 (2012)
6. Lee, K., Jeon, J., Lee, W., Jeong, S., Park, S.: QoS For Web Services: Requirements and Possible Approaches. Technical report, W3C, Note 25, Web Services Architecture Working Group (2003)
7. Yu, W.D., Radhakrishna, R.B., Pingali, S., Kolluri, V.: Modeling the measurements of QoS requirements in web service systems. Simulation Journal 83(1), 75–91 (2007)
8. Dumas, M., García-Bañuelos, L., Polyvyanyy, A., Yang, Y., Zhang, L.: Aggregate Quality of Service Computation for Composite Services. In: Maglio, P.P., Weske, M., Yang, J., Fantinato, M. (eds.) ICSOC 2010. LNCS, vol. 6470, pp. 213–227. Springer, Heidelberg (2010)
9. Bunke, H., Shearer, K.: A graph distance metric based on the maximal common subgraph. Pattern Recognition Letters 19, 255–259 (1998)
10. Grigori, D., Corrales, J.C., Bouzeghoub, M., Gater, A.: Ranking bpel processes for service discovery. IEEE Transactions on Services Computing 3, 178–192 (2010)
11. Yage, R.R.: Families of OWA operators. Fuzzy Sets and Systems 59, 125–148 (1993)
12. Yang, S.W., Shi, M.L.: A model for Web service discovery with QoS Constraints. Chinese Journal of Computers 28(4), 589–594 (2005)
13. Guo, D.K., Ren, Y., Chen, H.H.: A QoS-Guaranteed and Distributed Model for Web Service Discovery. Journal of Software 17(11), 2324–2334 (2006)
14. Ran, S.: A Model for Web Services Discovery with QoS. SIGecom Exchange 4, 1–10 (2003)
15. Lee, K.C., Jeon, J.H., Lee, W.S., Jeong, S.H., Park, S.W.: QoS for Web Services: Requirements and Possible Approaches. W3C Working Group Note 25 (2003)
16. Weiser, M.: Hot topics: Ubiquitous computing. IEEE Computer (1996)
17. Bakhouya, M., Gaber, J.: Approaches to Ubiquitous Computing. In: Labiod, H. (ed.) Wireless Ad hoc and Sensor Networks, pp. 111–142. ISTE Publishing Knowledge/John Wiley and Sons Inc., London (2008)
18. Cao, J., Xing, N.: Service adaptation using fuzzy theory in context-aware mobile computing middleware. In: Proceedings of the 11th IEEE Conference on Embedded and Real-Time Computing Systems and Applications, pp. 496–501 (2005)
19. Kruskal, J.B.: Multidimensional scaling by optimizing goodness of fit to a non metric hypothesis. Psychometrika 29(1), 1–27 (1964)
20. Stenström, P.: High performance embedded architectures and compilers. In: Third International Conference HiPEAC, Göteborg, Sweden (2008)
21. Abello, J.M., Pardalos, P.M., Mauricio, G.C.: Handbook of Massive Data Sets. Springer (2002) ISBN 1402004893
22. Dijkman, R., Dumas, M., Van Dongen, B., Käärik, R., Mendling, J.: Similarity of business process models: Metrics and evaluation. Inf. Syst., 498–516 (2011)