

## Service Quality Evaluation by Personal Ontology\*

CHIUNG-HON LEON LEE<sup>†</sup>, ALAN LIU AND JENG-SHIN HUNG

<sup>†</sup>*Department of Computer Science and Information Engineering  
Nanhua University*

*Dalin, Chiayi, 622 Taiwan*

*Department of Electrical Engineering and Center for Telecommunication Research  
National Chung Cheng University  
Chiayi, 621 Taiwan*

How to find and select a suitable service for services users are a challenging task. In this paper, we proposed a method of concept categorization with the use of personal ontology to evaluate and select more suitable service according to the preferences of users. We use the attributes of 'reputation', 'compliance', 'verity', and 'user ranking' for our evaluation of service quality. In addition, a new attribute called 'reformation' is also added to evaluate service quality. Based on the evaluation attributes, a fuzzy reasoning method is used for service selection. Our method is to aide the evaluation process with user preferences to reflect the fact that the user difference will affect the selection of different service providers. A system prototype has been built for determining the feasibility of our approach.

**Keywords:** web services, personal ontology, fuzzy reasoning, services selection, user preference

### 1. INTRODUCTION

Web services are a good choice for loosely coupled architectures and reusable software components which could be published, located, and invoked via the Internet. However, to fully satisfy the requirements of business applications, the current technologies still have many research issues such as security, selection, composition, and semantic problems to be explored [1].

Many approaches have been proposed to enable the Web services. The technologies of SOAP, WSDL, and UDDI, provide syntactic level standards for Web services. The languages of Web Services Flow Language [2] and BPEL4WS [3] give the basis for assembling value-added composite services from atomic services. The Semantic Web services [4] solve problems on the semantic level of Web services and address Web services descriptions as a whole. The semantic markup languages OWL-S and its previous release DAML-S [5] are proposed to describe the capabilities and contents of the Web services in a computer interpretable language and improve the quality of service discovery, selection, invocation, composition, monitoring, and recovery.

Although there are many techniques and standards which have been proposed to facilitate the Web services access, it is still too naive to wish that one can directly use the

---

Received October 31, 2007; accepted June 27, 2008.

Communicated by Jonathan Lee, Wei-Tek Tsai and Yau-Hwang Kuo.

\* This paper was partially supported by the National Science Council of Taiwan, R.O.C. through grant No. NSC-96-2218-E-343-002-MY2 and Department of Industrial Technology, Ministry of Economic Affairs, Taiwan, R.O.C. through grant No. 96-EC-17-A-02-S1-029.

results returned from the service providers to satisfy the user's request. The system might have to interact with the user to elicit enough information for Web services acquisition, discover, select, and compose services, process information returned from the services, and demonstrate the results to the user for user's request satisfaction.

How to find and select suitable services for services users is one of the key challenges in Web services research. In this paper, we use the concept of Quality of Service (QoS) [6] to evaluate the services to the users according to user preferences. For such requirements, personal ontology [7] is used in modeling user preferences and also user backgrounds. By establishing personal ontology for each user, we may automate the matching process for finding more suitable service providers for the users. This work uses goal models [8] to capture user intention to aide the personal ontology for providing more information for the system to filter the services.

Personal ontology is used for storing information about each individual user. How much information and what information to keep raise a new problem and will be discussed in the latter section. The work in [9, 10] uses personal ontology to record the habit and interests of a user in browsing Web pages. A system then uses such information for recommending the user to view certain Web pages. The authors in [11] use personal ontology in an agent-system for a meeting scheduling system. In [12], personal ontology and domain ontology were used in keeping personal information and food preferences for finding Web services.

There are many service providers offering similar services, and the services delivered by different providers may differ depending on situations and platforms. Quality attributes that are usually considered include execution price, latency, response time, throughput, availability, reliability, reputation, *etc.* In addition, the attributes like security and transaction may also be considered [6]. In using quality attributes in selecting services, the work reported in [11] uses QoS for evaluating service composition. The authors in [12, 13] define ontology for service quality to provide strategies for QoS between providers and requestors. Such strategies will be used for selecting matching services. Service Level Agreement (SLA) [14] is to protect the requestors for receiving quality services. For providers, they can use this index for making themselves more appealing to the requestors.

Table 1 shows the features used by the systems mentioned in the previous paragraph. All systems use community usage history which is the record of services used for selecting services. Most of them use some kinds of service quality parameters and make a good use of user feedback. However, Personal service selection policy is supported by only one system.

**Table 1. Comparison of features.**

	[11]	[12]	[16]	[17]
Service quality parameters	O	O	O	x
User feedback	O	x	O	O
User usage history	x	x	O	O
Community usage history	O	O	O	O
Personal selection policy	x	O	x	x
User preferences	x	x	x	x

The discussion above on related work is the motivation to our work in blending personal ontology with the QoS concept. This paper is organized as follows. How to construct personal ontology is presented in section 2. Section 3 discusses on the subject of service quality, while section 4 presents the reasoning method for service selection. Section 5 explains the simulation work and the last section states the conclusion.

## 2. PERSONAL ONTOLOGY CONSTRUCTION

Our approach in defining a personal ontology for recording user interests and preferences is introduced in this section. We first identify the types of concepts and then discuss the relationship between them.

### 2.1 Types of Concepts

In constructing a personal ontology, we use a concept to represent the preferences of a user for some domains. For example, the term ‘price’ for the shopping domain means how much to spend for goods, and this concept can be used in considering how a user behaves in shopping. A word may have different senses, and different terms may be used in different domains. If the term ‘price’ is associated with another term ‘Java’, then it may mean the price for a bag of coffee beans or the airfare to a resort island. Even if the domain is set as shopping a book for programming in Java, it may also differ according to users to determine whether a book price of USD 50 is expensive or not. Thus ‘expensive book’ or ‘inexpensive book’ may be associated with ‘price’, but the value associated to it may be different according to users.

Following the discussion above, we classify a concept from domain point of view and user point of view. Three attributes are used to further define concepts, “independent”, “dependent”, and “specific”. Our approach uses six types of concept to associate domains and user needs as listed below:

**Domain Independent Concept (DIC)** A concept may appear in different domains, but the meaning of the concept is the same despite of different domains.

**Domain Dependent Concept (DDC)** The meaning of a concept differs in different domains.

**Domain Specific Concept (DSC)** A concept may be used only in a specific domain.

**User Independent Concept (UIC)** The meaning of a concept is agreed by all users in the same domain.

**User Dependent Concept (UDC)** The meaning of a concept is different according to different users.

**User Specific Concept (USC)** A concept is only meaningful to a specific user.

System design may benefit from the types above. The following shows how we can incorporate such information:

1. If the concept is of DIC, then system does not need to worry about the ambiguity in the meaning.

2. If the concept is of DDC, then the system may not immediately determine the meaning, but need to collect more information to clarify its meaning.
3. If the concept is of DSC, then the system interprets its meaning in that particular domain.

Using the definition above, we can determine nine different kinds of concept which is shown in Table 2. The table shows how domain dependencies and user dependencies can be combined together.

**Table 2. Categories of concepts.**

User Aspect / Domain Aspect	User Independent	User Dependent	User Specific
Domain Independent	User Independent & Domain Independent	User Dependent & Domain Independent	User Specific & Domain Independent
Domain Dependent	User Independent & Domain Dependent	User Dependent & Domain Dependent	User Specific & Domain Dependent
Domain Specific	User Independent & Domain Specific	User Dependent & Domain Specific	User Specific & Domain Specific

**2.2 Relationships Between Concepts**

From the discussion above, we define three kinds of relationships between concepts, such as below:

**hasInformation** This associates the information between different users or between preferences.

**hasPreference** This shows the possession of preferences for a concept.

**needInterpretation** This connects ‘Interpretation’ and UDC or USC. As shown in Fig. 1, needInterpretation has domains ‘UDC’ and ‘USC’. Its range is of the concept of ‘Interpretation’.

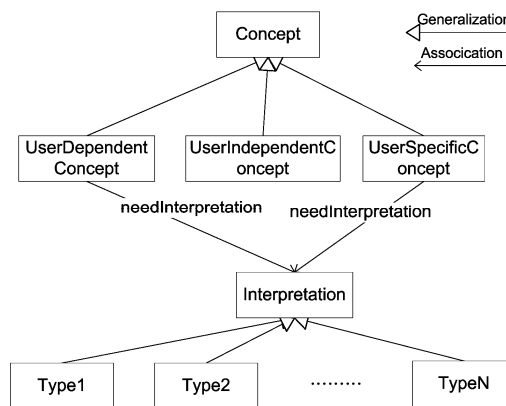


Fig. 1. needInterpretation relation.

### 2.3 Process of Constructing Personal Ontology

The procedure of constructing personal ontology is shown in Fig. 2. The steps are described below:

**Step 1:** Determine the domains that will be used for personal ontology.

**Step 2:** Find the items of interests (concepts) in the chosen domains for the users.

**Step 3:** Categorize the concepts according to the meaning in domains.

**Step 4:** Categorize the concepts according to the user's point of view.

**Step 5:** Determine whether the concept is of UDC or USC. If so, more information is needed. If not, the step 6 can be skipped.

**Step 6:** Use 'Interpretation' to determine the user's view on such concept.

**Step 7:** Collect the weights between the personal information and the user preferences.

**Step 8:** Produce personal ontology.

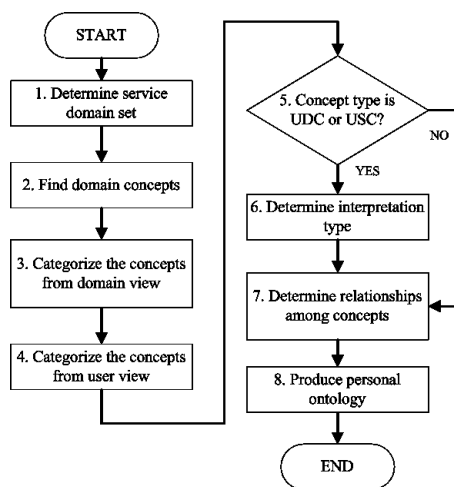


Fig. 2. Personal ontology construction process.

Fig. 3 shows an example in a transportation system. A partial view of the personal ontology shown here has four sub-domains. From the domain point of view, the attributes 'price', 'company', and 'time' all belong to DIC. The train type and ticket rank belong to DSC. Following the steps in Fig. 2, an ontology instance derived from Fig. 3 is shown in Fig. 4.

## 3. SERVICE QUALITY EVALUATION

According to the definition of 'reputation' given in [15], a system must consider that the usage experiences will be kept and the values kept track will determine the outcome of the decision of the participants. We use the idea of 'reputation' introduced in [16, 17] for our evaluation of service quality. Besides the original three criteria 'ompliance', 'verity', and 'user ranking', we introduce 'reformation' as an additional criterion.

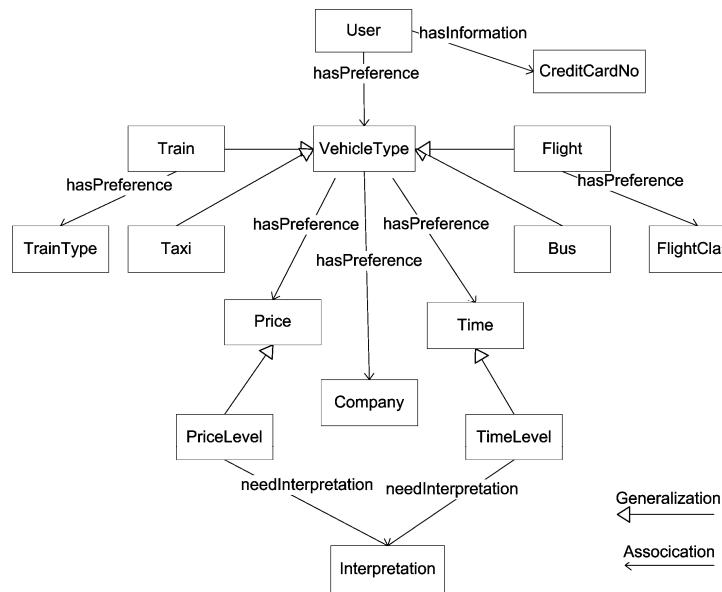


Fig. 3. Partial list of personal ontology in the transportation domain.

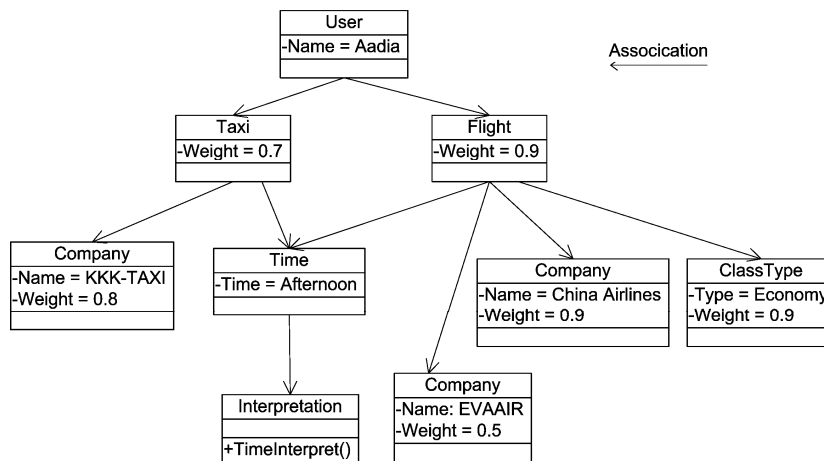


Fig. 4. A partial list of personal ontology instance.

### 3.1 Compliance

Compliance is used for recording the integrity of the projected value and delivered value of a service provider. As presented in SLA [10], how a provider can guarantee a service quality is evaluated. In this paper, we use  $a$  to determine service quality value,  $p$  to determine the guarantee value,  $d$  to determine the delivery value.  $a_i p_n$  represents the  $n$ th guarantee value of a service quality value,  $a_i$ .  $a_i d_n$  means what  $a_i$  received for the delivery value after using such services for  $n$  times. A history of a service provider is shown in a list like below:

$$\{(a_1p_1, a_1d_1), (a_1p_2, a_1d_2) \dots (a_1p_n, a_1d_n)\}.$$

Let  $N_{a_i}^j$  be the normalized guarantee value and  $L_{compl}^{a_i}$  be the compliance of  $a_i$  for the past  $n$  services. They are defined as

$$N_{a_i}^j = \frac{a_i p_j - a_i d_j}{a_i p_j}, L_{compl}^{a_i} = \frac{\sum_{j=1}^n N_{a_i}^j}{n}. \quad (1)$$

We can have the compliance of a service,  $S_{compl}$ , where  $r$  is the number of quality attributes.

$$S_{compl} = \frac{\sum_{i=1}^r L_{compl}^{a_i}}{r} \quad (2)$$

### 3.2 Verity

Verity determines the degree of change (differences in service quality and delivery) in a service provider. This can be calculated from the history data. We can use the value of compliance to determine the average value of compliance for the past  $n$  usages,  $\mu_{compl}$ . By using this value, we can define the verity value,  $S_{verity}$ . The formulas are given below:

$$\mu_{compl} = \frac{\sum_{i=1}^n S_{compl_i}}{n}, S_{verity} = \sqrt{\frac{\sum_{i=1}^n (S_{compl_i} - \mu_{compl})^2}{n}}. \quad (3)$$

The smaller the value of verity, the more reliable the service provider is. Thus, the ideal value for verity is 0. If the value is large, it indicates that the reliability of a service provider is low

### 3.3 User Ranking

User ranking tells how satisfied a user is about a certain service provider. This value is subjective according to each user. Thus, the value may differ depending on different users' grading preferences. The value,  $S_{rating}$ , shows the average of the ranking given by the same user. The values are between (0, 10) in this paper, where the default value is set as 0 and the formula is defined as below:

$$S_{rating} = \frac{\sum_{i=1}^n R_i}{n}. \quad (4)$$

### 3.4 Reformation

Reformation is the degree of improvement for service quality. The advantage of ap-

plying this criterion is that it gives a chance of updating the values of quality value. With this criterion, the system can give a chance to continued improved services. The value,  $\mu_{a,d}$ , is the average of what a user received for a particular service. The value,  $\mu_{a,D}$ , is the average of difference between the promised value and real value. The value,  $R_{a_i}$ , shows the degree of improvement. If the degree of improvement is only the proportion, then for a person with 30 points to receive 45 points is a 50% improvement while a person with 90 points to receive perfect 100 is just 11% improvement. In order to show the differences in the degree of progress in the lower level and the higher level, the value,  $W_{a,d}$ , gives a weighted evaluation with  $C_i$  as a base.  $C_i$  can be set by system designer or derived by data mining approaches. In this paper, we assume that  $C_i$  is set by system designer.

$$\mu_{a,d} = \frac{\sum_{j=1}^n a_i d_j}{n}, \mu_{a,D} = \frac{\sum_{j=1}^n a_i p_j - a_i d_j}{n} \quad (5)$$

$$R_{a_i} = \frac{a_i p_{n+1} - \mu_{a,D}}{\mu_{a,d}}, W_{a,d} = \frac{\mu_{a,d}}{C_i}$$

Finally, putting all together, we can have the value for reformation,  $S_{reform}$ , defined as

$$S_{reform} = \frac{\sum_{i=1}^r W_{a,d} \times R_{a_i}}{r}. \quad (6)$$

### 3.5 Reputation

With the availability of compliance, verity, user rating, and reformation, we can define reputation as a quadruple below:

$$S_{reput} = (S_{compl}, S_{verity}, S_{ratings}, S_{reform}). \quad (7)$$

There are different ways to derive reputation criterion. For example, the designer can simply add the values of compliance, verity, user rating, and reformation to get reputation, or considering the promises and reliability from the past experiences with a service provider, a user may add some weights to the above. Thus, we will have

$$S_{reput} = W_1 S_{compl} + W_2 S_{verity} + W_3 S_{ratings} + W_4 S_{reform}. \quad (8)$$

In next section, we introduced a fuzzy reasoning method to derive reputation for service selection.

## 4. FUZZY REASONING METHOD FOR SERVICE SELECTION

We use fuzzy reasoning method in our approach for service evaluation. A service quality model consisting of 'reputation', 'compliance', 'verity', 'user ranking', and 'reformation' is used and 'bad', 'normal', and 'good' are used for defining a input linguistic value.

Right and left linear membership functions are used to model the ‘bad’ and ‘good’ fuzzy set.

$$\text{right\_linear}(x : a, b) = \begin{cases} 1 & x < a \\ (b-x)/(b-a) & a \leq x \leq b \\ 0 & x > b \end{cases} \quad (9)$$

$$\text{left\_linear}(x : a, b) = \begin{cases} 0 & x < a \\ (x-a)/(b-a) & a \leq x \leq b \\ 1 & x > b \end{cases} \quad (10)$$

The ‘normal’ fuzzy set is described by the triangle function.

$$\text{triangle}(x : a, b, c) = \begin{cases} 0 & x < a \\ (x-a)/(b-a) & a \leq x \leq b \\ (c-x)/(c-b) & b < x \leq c \\ 0 & x > c \end{cases} \quad (11)$$

We also provide some hedges with the definition of ‘extremely’, ‘very’, ‘normal’, ‘somewhat’, and ‘less’ to modify fuzzy linguistic variables. Fig. 5 shows a fuzzy set with ‘normal’ hedge, where Quality Value shows the arguments entered. The y-axis, Degree is the membership degree with the values between 0 and 1.

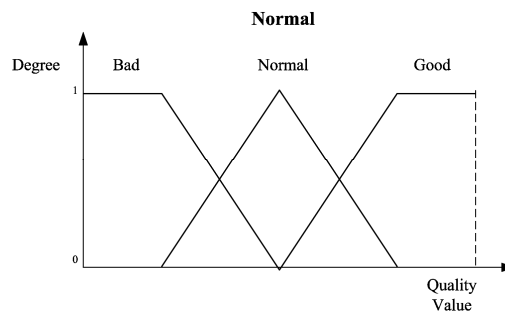


Fig. 5. Definition ‘Normal’.

A degree of concern,  $Q_A$  can also be determined. The degree of concern for ‘normal’ then is defined as  $S_A$ .

$$Q_A = (\mu_{Bad}, \mu_{Normal}, \mu_{Good}), S_A(U_{Normal}, Q_A) \quad (12)$$

The degree of concern, ‘Extremely’ can be defined as Fig. 6. Fig. 7 shows the degree of concern for ‘Very’. In the same way ‘Somewhat’ and ‘Less’ can be determined and are depicted in Figs. 8 and 9.

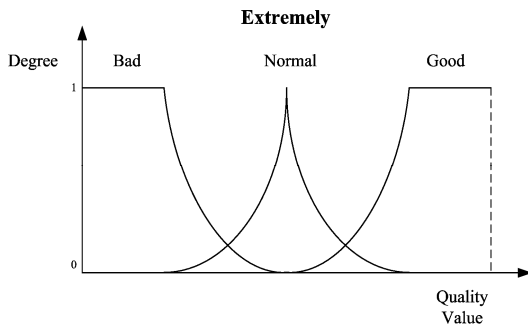


Fig. 6. Definition 'Extremely'.

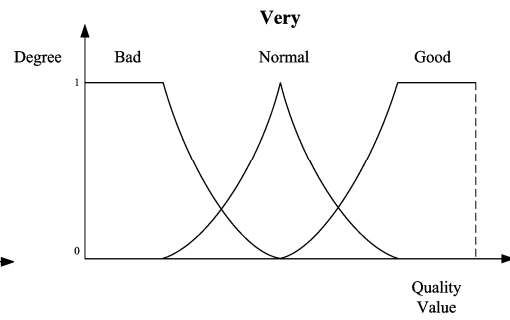


Fig. 7. Definition 'Very'.

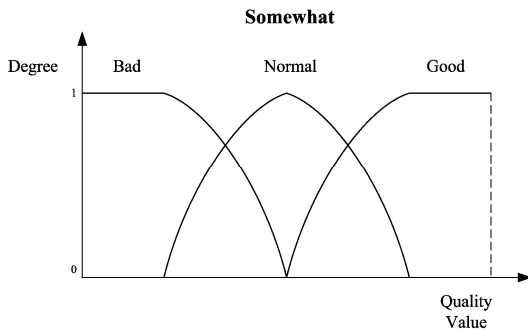


Fig. 8. Definition 'Somewhat'.

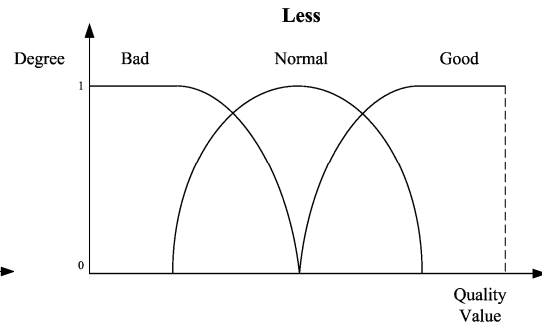


Fig. 9. Definition 'Less'.

$$S_A(U_{Extremely}, Q_A) = [S_A(U_{Normal}, Q_A)]^3 \tag{13}$$

$$S_A(U_{Very}, Q_A) = [S_A(U_{Normal}, Q_A)]^2 \tag{14}$$

$$S_A(U_{Somewhat}, Q_A) = [S_A(U_{Normal}, Q_A)]^{1/2} \tag{15}$$

$$S_A(U_{Less}, Q_A) = [S_A(U_{Normal}, Q_A)]^{1/3} \tag{16}$$

With the above functions defined, we let the user express his strategies and preferences for using the services. The strategies are kept in personal ontology to let the system determine user's preference.

Table 3 shows the first 10 rules from 81 fuzzy rules, using values, P (prime), G (good), N (normal), B (bad), and W (worst). Here we may see a rule, IF Compliance = Good and Verity = Good and UserRanking = Good and Reformation = Good, then MatchingValue = Prime.

The system delivers a fuzzy variable for matching value of a service. The range is 0 to 1. Fig. 10 shows the matching value with 5 types of linguistic values: 'Prime', 'Good', 'Normal', 'Bad', and 'Worst'.

The Max-Min Operation along with Center of Gravity is used in the fuzzy reasoning. We consider the accuracy of the output and then to consider the performance issues.

**Table 3. Fuzzy rule base.**

Rule	Input				Output
	Comp.	Verity	User Ranking	Reform.	Matching Value
1	G	G	G	G	P
2	G	G	G	N	P
3	G	G	G	B	G
4	G	G	N	G	P
5	G	G	N	N	G
6	G	G	N	B	G
7	G	G	B	G	G
8	G	G	B	N	G
9	G	G	B	B	N
10	G	N	G	G	P

P: Prime G: Good N: Normal B: Bad W: Worst

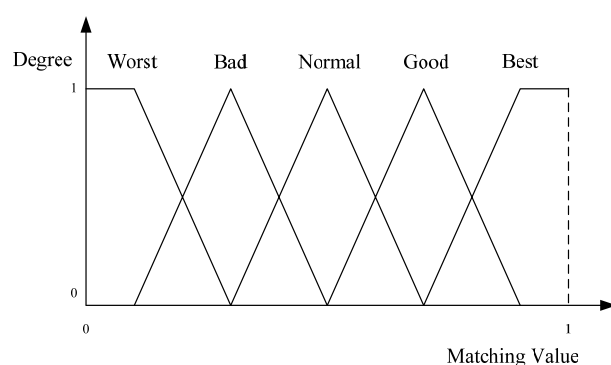


Fig. 10. Matching values.

## 5. SIMULATION

A system prototype has been built for determining the feasibility of our approach. The system was written in Java, and personal ontology was constructed through Protégé [18] and is recorded as an XML file. Jena [19] was used in reasoning, and FuzzyJ Tool-Kit [20] was used in handling fuzzy values.

Fig. 11 illustrates our prototype system. Two external components are the Service Planner from our previous results [21] and Service Matchmaker (SM) which is similar to [22]. The Service Planner processes the goal model along with personal preferences from the Personal Ontologies to derive plans for the Service Selector, which prepares a request document in OWL-S and send it to the SM, which in turn finds some suitable service providers. The Fuzzy Decision Module then is to reason according to the user preferences to select the most suitable service for the user. At the same time, the Policy Generator determines which policy is best for the occasion according to the ontology or directly from the user. The QoS Calculator uses the attributes presented in section 3 and the execution history is kept in a database.

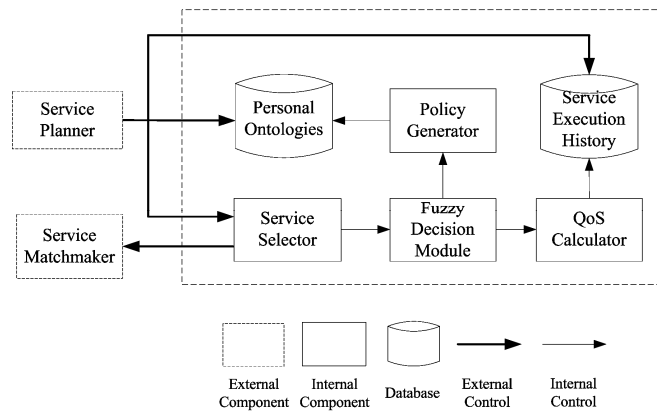


Fig. 11. Prototype system architecture.

The purpose of this simulation is to evaluate the differences between ‘Prime’, ‘Good’, ‘Normal’, ‘Bad’, and ‘Worst’. To start with a simple experiment, we first consider ‘response time’ as a factor in determining how five services which are viewed as similar services may differ. For this simulation, ‘User Rating’ is assumed to be all ‘Normal’. To setup the simulation, we assume that we have five services, namely Service 0, Service 1, *etc.* The services are defined as Table 4:

Table 4. Sample services.

	Service 0	Service 1	Service 2	Service 3	Service 4
time	slow	slow	medium	fast	fast
stability	bad	good	good	bad	good

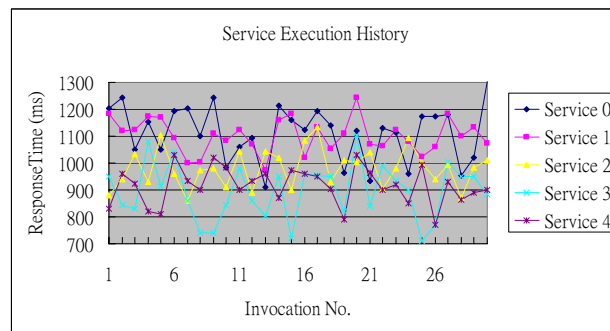


Fig. 12. Time for service execution.

Fig. 12 shows the result of performing 5 different services for 30 times. We can see individual characteristics of each service. Service 0 and Service 3 have a great variation in response time, and this causes unstable services. Table 5 shows the statistical data concerning response time.

**Table 5. Response time (Unit: ms).**

	Service 0	Service 1	Service 2	Service 3	Service 4
Average	1110.93	1100.93	981.4	895.966	915.633
Std. Div.	102.664	62.9827	71.6281	103.048	68.4218
Max.	1302	1242	1132	1102	1031
Min.	911	972	861	711	771

**Table 6. Polices based to quality attributes.**

	Compliance	Verity	User Rating	Reformation
Policy 1	Less	Less	Less	Less
Policy 2	Normal	Less	Less	Less
Policy 3	Extremely	Less	Less	Less

In our service quality evaluation mode, ‘Compliance’ keeps the record of whether the service is completed every time. ‘Verity’ shows the stableness of the quality. Table 6 shows three different policies with four attributes set in different values. This setting is to simulate a user whose main concern is on compliance.

By taking Service 3 as example, we can see that Service 3 is sensitive to the compliance value, so that the value varies greatly according to the degree of ‘less’, ‘normal’, and ‘extreme’ as shown in Fig. 13.

Table 7 shows the result of having completed all 30 service selections except for the very first one which does not a previous service. In this simulation ‘Compliance’ and

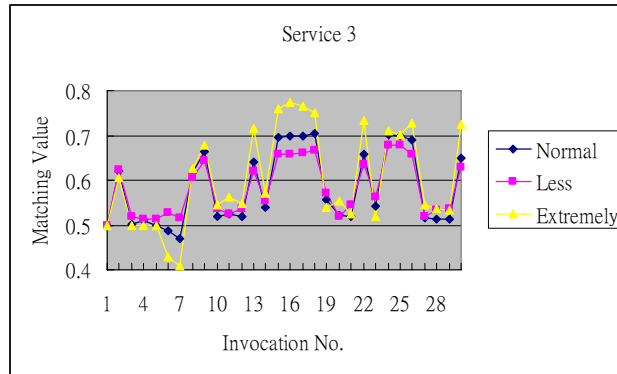


Fig. 13. Result of the service 3.

**Table 7. Service selection strategies.**

	Service 0	Service 1	Service 2	Service 3	Service 4
Less	0	0	34.50%	27.60%	44%
Normal	3.40%	0	24.10%	37.90%	34.50%
Extremely	0	0	20.60%	41.40%	37.90%

'Response time' along with the value 'Extremely' are used. The table shows that Service 1 is never chosen and Service 3 is most chosen when 'Extreme' is of concern, while Service 4 is chosen when 'Less' is the primary concern. More simulation was conducted according to different preferences, and the results show that our definition of reputation with the concept of reformation is informative.

## 6. CONCLUSION

This paper proposed the usage of concept categorization from the domain and user point of view. Personal ontology is used in representing difference in users by recording their preferences and other information. The attribute of 'reputation' is used for evaluating service quality by adding a concept of 'reformation'. A reasoning method based on fuzzy theory is also introduced along with the strategies of setting policies. A prototype system is built and a simulation was executed. The results showed that the attributes discussed in the paper can be beneficial to service selection based on user preferences.

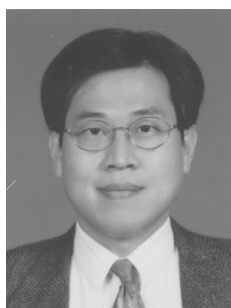
## REFERENCES

1. H. Wang, J. Z. Huang, Y. Qu, and J. Xie, "Web services: Problem and future directions," *Elsevier Journal of Web Semantics*, Vol. 1, 2004, pp. 309-320.
2. <http://www-306.ibm.com/software/solutions/>.
3. <http://www-128.ibm.com/developerworks/library/specification/ws-bpel/>.
4. M. Paolucci and K. Sycara, "Autonomous semantic web services," *IEEE Internet Computing*, Vol. 7, 2003, pp. 34-41.
5. <http://www.w3.org/Submission/OWL-S/>.
6. S. Ran, "A model for web services discovery with QoS," *ACM SIGecom Exchanges*, Vol. 4, 2003, pp. 1-10.
7. M. N. Huhns and L. M. Stephens, "Personal ontologies," *IEEE Internet Computer*, 1999, pp. 85-87.
8. C. H. L. Lee and A. Liu, "Model the query intention with goals," in *Proceedings of the 1st International Workshop on Ubiquitous Smart Worlds*, 2005, pp. 535-540.
9. J. Chaffee and S. Gauch, "Personal ontologies for web navigation," in *Proceedings of the 9th International Conference on Information and Knowledge Management*, 2000, pp. 227-234.
10. C. S. Lee, Z. W. Jian, and L. K. Huang, "A fuzzy ontology and its application to news summarization," *IEEE Transactions on Systems, Man and Cybernetics Part B*, Vol. 35, 2005, pp. 859-880.
11. Q. Sheng, B. Benatallah, L. Zeng, M. Dumas, and J. Kalagnanam, "Quality driven web services composition," in *Proceedings of the 12th ACM International Conference on World Wide Web*, 2003, pp. 411-421.
12. E. M. Maximilien and M. P. Singh, "Toward autonomic web services trust and selection," in *Proceedings of the 2nd ACM International Conference on Service Oriented Computing*, 2004, pp. 212-221.
13. E. M. Maximilien and M. P. Singh, "A framework and ontology for dynamic web services selection," *IEEE Internet Computing*, Vol. 8, 2004, pp. 84-93.

14. IBM, "Guarantee your web service with a SLA," <http://www-128.ibm.com/developerworks/library/ws-sla/>, 2002.
15. P. Resnick, R. Zeckhauser, R. Friedman, and K. Kuwabara, "Reputation systems," *Communications of the ACM*, Vol. 43, 2000, pp. 45-48.
16. S. Kalepu, S. Krishnaswamy, and S. W. Loke, "Reputation = f(user ranking, compliance, verity)," in *Proceedings of IEEE International Conference on Web Services*, 2004, pp. 200-207.
17. R. M. Sreenath and M. P. Singh, "Agent-based service selection," *Journal on Web Semantics*, Vol. 1, 2004, pp. 261-279.
18. Protégé, <http://protege.stanford.edu/>.
19. Jena, <http://jena.sourceforge.net/>.
20. FuzzyJ ToolKit, [http://www.iit.nrc.ca/IR\\_public/fuzzy/fuzzyJToolkit2.html](http://www.iit.nrc.ca/IR_public/fuzzy/fuzzyJToolkit2.html).
21. K. H. Huang and A. Liu, "Service composition using planning and case-based reasoning," in *Proceedings of International Conference on Software Engineering and Knowledge Engineering*, 2007, pp. 610-615.
22. M. Paolucci, T. Kawamura, T. R. Payne, and K. Sycara, "Semantic matching of web services capabilities," in *Proceedings of the 1st International Semantic Web Conference*, 2002, pp. 333-347.



**Chung-Hon Leon Lee (李俊宏)** received his Ph.D. degree in Electronic Engineering from the National Chung Cheng University in Taiwan in 2006. He is an assistant professor at Department of Computer Science and Information Engineering, Nanhua University, Taiwan. His research interests are in agentbased software engineering, web services, knowledge representation, and fuzzy time series.



**Alan Liu (劉立頌)** received the Ph.D. degree in Electrical Engineering and Computer Science from the University of Illinois at Chicago in 1994. He is an associate professor at Department of Electrical Engineering, National Chung Cheng University in Taiwan. His research interests in artificial intelligence and software engineering include knowledge acquisition, requirements analysis, intelligent agents, and applications in embedded systems and robotic systems. He is also a member of IEEE, ACM, and TAAI.

**Jeng-Shin Hung (洪正鑫)** received his master degree in Electronic Engineering from the National Chung Cheng University in Taiwan in 2006. His research interests are in web services, software agent and personal ontology.