J. Software Engineering & Applications, 2009, 2: 103-110
doi:10.4236/jsea.2009.22015 Published Online July 2009 (www.SciRP.org/journal/jsea)

Autonomic Software Component QoS Matchmaking Algorithm Based on Fidelity Factor in Agent-Based Autonomic Computing System

Kun Zhang[1,2], Manwu Xu[2], Hong Zhang[1]

[1]School of Computer Science & Technology, Nanjing University of Science & Technology, Nanjing, China; [2]State Key Laboratory for Novel Software, Nanjing University, Nanjing, China.
Email: zhangkun@mail.njust.edu.cn

Received December 8th, 2008; revised March 20th, 2009; accepted March 31st, 2009.

ABSTRACT

Autonomic software component (ASC) QoS matchmaking problem for autonomic element has been taken as one of the most important issue in field of autonomic computing based on agent. Aimed at overcoming drawbacks such as subjectiveness and unfairness, and improving the self-configuration capability for autonomic element, we introduce evaluation mechanism of confidence of individual QoS attributes during ASC QoS matchmaking, i.e., fidelity factor for each attribute, and propose an ASC QoS matchmaking algorithm based on fidelity factor. Simulation experiments demonstrate that our proposed algorithm performs best performance in terms of QoS than other existing algorithms, and has better compromise between attribute quality and users' evaluation when selecting ASC.

Keywords: *Autonomic Software Component, QoS Matchmaking, Fidelity Factor, Autonomic Computing, Autonomic Element*

1. Introduction

In mid-October 2001, aimed at the problem of looming software complexity crisis, IBM Company innovatively proposed autonomic computing [1] technology – computing systems that can manage themselves given high-level objectives from administrators. An autonomic software component (ASC, or element, in IBM parlance) [1] is—

"The fundamental atom of autonomic applications and systems—a modular unit of composition with contractually specified interfaces, explicit context dependencies, and mechanisms for self management, responsible for providing its own services, constraints (e.g., system resource requirements, performance requirements, etc.), managing its own behavior in accordance with context, rules, and policies, and interacting with other autonomic components. [1]"

Autonomic software components will have complex life cycles, continually carrying on multiple threads of activity, and continually sensing and responding to the environment in which they are situated. Autonomy, proactivity, and goal-directed interactivity with their environment are distinguishing characteristics of software agents. Viewing autonomic software components as agents and autonomic systems as multi-agent systems makes it clear that agent-oriented architectural concepts will be critically important [2].

In order to efficiently accomplish the self-configuration and self-management between ASCs, one of the most important issues is how to design an efficient autonomic software component matchmaking algorithm to find appropriate provider(s) for a consumer according to the providers' advertising description.

However, most of the existing algorithms are only concerned with functional matchmaking on software component. Few of them focused on non-functional factors, such as software component cost, time, reliability, satisfaction, i. e. quality-of-service (QoS) of software component. Therefore, when more ASCs than one can provide a functional component, these algorithms just randomly selected one from these ASCs, and could not be obtained optimal ASC with high QoS performance. Moreover, only a few existing algorithms are considered QoS factors during software component matchmaking. However, these algorithms were paid regard to QoS level as a whole, ignoring confidence of individual QoS attributes. Due to QoS data was often advertised by software component providers, it suffered from the drawbacks such as subjectiveness and unfairness. And these algorithms didn't be considered the user's feeling and satisfaction.

Aimed at overcoming the above difficulties, we introduce evaluation mechanism of confidence of individual QoS attributes during ASC QoS matchmaking, i.e., fidelity factor for each attribute, and propose an ASC QoS matchmaking algorithm based on fidelity factor, which will improve the facility and fairness concerned with QoS attributes, and objectively select optimal ASC with QoS performance.

The rest of the paper is organized as follows. Some of the existing work related to this paper is described in Section 2. Section 3 gives our proposed ASC matchmaking algorithm in detail. A case study and simulation results are presented in Section 4. Section 5 concludes the paper.

2. Related Work

In recent years, much research has been devoted to the development of software component matchmaking algorithms for agent or autonomic element [3,4,5,6,7,8]. Wickler [3] addressed the problem of capability brokering agent, and proposed a new capability description language (CDL) for the representation of agent capabilities. Sycara [4] defined a language called LARKS for agent advertisements and requests, and presented a flexible and efficient matchmaking process that used LARKS. LARKS performed both syntactic and semantic matching, and in addition allowed the specification of concepts (local ontologies) via ITL, a concept language. The establishment for semantic distance consumed much workload, so that their matching algorithm had limitations on practicability and reliability. Arisha *et al.* [5] provided approximate software agent service matchmaking by using semantic distance. Whereas, the algorithm could not support definition of data type and descript software component efficiently. However, neither of the above algorithms considered the factors of quality-of-service attribution, such as cost, time, and reliability. Zhang [6] and Jiang [7] analyzed the drawbacks in previous papers, i.e., the matchmaking was only based on the advertised capabilities of provider agents or software components. They considered the matchmaking influenced by QoS of software components, and individually presented software components matchmaking algorithm based on QoS. In [6], it was argued that the practical performance of provider agents had a significant impact on the matchmaking outcomes of middle agents. The authors' proposed algorithm could pick up the provider agents based on the history information in accomplishing similar tasks in the past rather than choosing randomly. At the launching of an agent system, the proposed algorithm provided initial values of the track records. With agents' history information and the initial values of the track records, the quality of matchmaking algorithms could be improved significantly, and the returned results were more accurate and reasonable. Jiang [7] pointed out that there were two drawbacks using the track records in [6], i.e., the value of track records was too subjective, and track record model is too simple to judge agent services' performance more accurately. So, the authors in [7] presented QoS-driven matchmaking algorithm, which aimed at matching the best satisfying agent for user. It is pity for the above two algorithms that the authors didn't consider the user's feeling and satisfaction when selecting agents or software components. The authors in [8] studied agent service selection and matchmaking in manufacturing industry field. They applied the idea of rough sets theory and fuzzy information filter to reclaim agent selection problem and realized reasonable evaluation for recycling quotient. Their algorithm was easy to select optimum reclaiming agent for manufacturer. However, their method belonged to specific field and non-universal.

Besides software component or agent service matchmaking, some researchers took software component or agent as web service, and proposed many QoS-aware web services matchmaking or selection algorithms [9,10,11,12,13,14]. Some theoretic methods or principle in these algorithms can be used for reference when matchmaking software component, due to many similarities between software component and web service. Ma [9] proposed a semantic QoS-aware framework for semantic web services discovery, and presented a selection algorithm to obtain the optimal offer from clients' viewpoint under complex QoS conditions through confirming the compatibility of concepts. Hu *et al.* [10] proposed a novel and extended web service QoS model (attributes include *time*, *cost*, *reliability* and *interest*) by adding an interest degree property, and then put forward to a QoS matchmaking algorithm based on this model. Aimed at resolving such conflicts to ensure consensus on the QoS characteristics in the selection of web services, Wei [11] proposed a QoS Consensus Moderation Approach (QCMA) in order to perform QoS consensus and to alleviate the differences on QoS characteristics in the selection of web services. QCMA enhanced the moderation for opinion similarity and preference on QoS attributes and was been designed a mechanism for providing group consensus moderation on QoS. Giallonardo *et al.* [12] used ontology to describe QoS-based WS specifications and reasoners to perform the matchmaking. They addressed QoS semantics to improve the recall of the matching process by exploiting ontology knowledge, and their QoS specification approach used metrics to understand, describe, and control the QoS in the matching process. However, they didn't offer useful results for over-constrained demands. In addition, although they were capable of, they did not perform semantic QoS metric matching. Guo *et al.* [13] proposed three-dimensional QoS model of web services, and designed a web services selection algorithm. In their algorithm, the concept and measure method of web service effectiveness were proposed, and then a web service ranking algorithm based on the effectiveness of web service was designed. The comprehensive experiment showed their proposed model possessed high precision, high response rate and better influence on the load balance of web service. Liang Kai-jian [14] studied QoS support problem in Service-oriented Grid system, and presented a new parameter called GSQN(Grid Service Quality Number) to depict the profile of qualitative characteristics of a grid service. Based on this idea, the author proposed a strategy of matchmaking with a price model to match grid service.

3. Our Proposed Matchmaking Algorithm

3.1 Problem Description

Definition 1. Autonomic software component matchmaking based on QoS problem. Given the set of ASC providers $PASC = \{pasc_1, pasc_2, ..., pasc_n\}$, an ASC requester *rasc*, autonomic software component matchmaking based on QoS problem is to find an ASC such that their QoS level is maximized.

3.2 QoS Model for Autonomic Software Component

Definition 2. Autonomic software component QoS model for autonomic element is a 6-tuple as follows:

$$qos\_model\_asc = (t, c, rel, m, rep, fid) \qquad (1)$$

where *t* is ASC response time from sending request to receiving result, including process time and transmission delay time, i.e., $q_{time}(asc) = T_{process}(asc) + T_{trans}(asc)$; *c* is ASC cost, and represents the expenses paid by users to ASC provider; *rel* is a metric of reliability, denoting the probability of ASC providing its registered software component; *rel* is a technical measure related to hardware and/or software configuration of ASCs and the network connections between the ASC requesters and providers. *m* denotes the probability of accurate maintenance when an exception occurs for an ASC; *rep* is a measure of ASCs' trustworthiness or satisfaction degree, and denotes users' (or requesters') satisfaction to ASC. *rep* mainly depends on end user's experiences of using the ASC. Different end users may have different opinions or satisfaction degree on the same ASC. Usually, the end users or requesters are given a range to rank or score ASCs, for example, in Amazon.com, the range is [0,5]. In this paper, the range is [0,1].

The first four QoS attributes are often published by ASC providers themselves, and describe the basic QoS performance of an ASC, so most of the authors took them as quality criteria for an ASC. However, such criterion is too subjective to reflect actual quality of ASC. In this paper, we introduce attribute "*reputation*" to measure users' satisfaction degree. On the other hand, *fidelity* vector is added to the QoS model to improve the impartiality and objectivity. Fidelity is treated as a vector composed of fidelity attributes. Each fidelity attribute refers to the confidence or fidelity of above first four QoS attributes, i.e.,

$$fid = \langle fid_t, fid_c, fid_{rel}, fid_m \rangle \qquad (2)$$

3.3 Matchmaking Algorithm Description

In our proposed QoS matchmaking algorithm, the drawbacks in [6] and [7] such as subjectiveness and unfairness are overcame through introducing evaluation mechanism of confidence of individual QoS attributes, i.e., fidelity factor for each attribute, which can improve the self-configuration capability for autonomic element.

The basic idea of our algorithm is as follows: firstly, normalizing each attribute in QoS description to range [0,1] for each ASC in initial set; then fidelity of each attribute is considered to evaluate QoS overall performance for every ASC comprehensively and objectively; finally, an ASC whose total QoS value is maximal is picked out from all candidates.

Concretely, suppose that there is a set of ASC providing a certain software function, i.e. $PASC = \{pasc_1, pasc_2, ..., pasc_n\}$. By merging the quality vectors of all these candidates, a matrix $\mathbf{Q}=(Q_{i,j}, 1 \le i \le n, 1 \le j \le 5)$ is built according with Definition 2, in which each row Q_j corresponds to an ASC $pasc_i$ while each column corresponds to a QoS attribute value. Let a matrix $\mathbf{F} = (fid_{i,j}, 1 \le i \le n, 1 \le j \le 4)$ denote confidence fidelity of the first four QoS attributes for $pasc_i$. There are two phases as follows:

1) Scaling phase.

Some of the QoS attributes could be negative, i.e., the higher the value, the lower the quality, such as *time*, *cost*. Other QoS attributes are positive, i.e., the higher the value, the higher the quality, such as *reliability*, *maintainability*, and *reputation*. For negative attributes, values are scaled according to (3). For positive criteria, values are scaled according to (4).

$$M_{i,j} = \begin{cases} \dfrac{Q_j^{\max} - Q_{i,j}}{Q_j^{\max} - Q_j^{\min}} & if\ Q_j^{\max} - Q_j^{\min} \neq 0 \\ 1 & if\ Q_j^{\max} - Q_j^{\min} = 0 \end{cases}, \quad j = 1, 2 \quad (3)$$

$$M_{i,j} = \begin{cases} \dfrac{Q_{i,j} - Q_j^{\min}}{Q_j^{\max} - Q_j^{\min}}, if\ Q_j^{\max} - Q_j^{\min} \neq 0 \\ 1 \qquad\qquad if\ Q_j^{\max} - Q_j^{\min} = 0 \end{cases}, \quad j = 3,4,5 \qquad (4)$$

In the above equations, $Q_j^{\max}$ and $Q_j^{\min}$ are the maximal and minimal value of the jth QoS attribute, respectively. Let $\mathbf{M} = \left(M_{i,j}, 1 \le i \le n, 1 \le j \le 5\right)$ be normalizing matrix according to **Q**, where $M_{i,j}$ denotes normalizing value of the *j*th QoS attribute for ASC provider *pasc$_i$*.

2) Weighting phase.

The following formula is used to calculate the overall quality score of ASC provider *pasc$_i$*.

$$qos(pasc_i) = \sum_{j=1}^{4}(w_j \cdot M_{i,j} \cdot fid_{i,j}) + w_5 \cdot M_{i,5} \qquad (5)$$

where, w_j is the weight value of the *j*th QoS attribute, $0 \le w_j \le 1, \sum_{j=1}^{5} w_j = 1$. End users express their preferences regarding QoS by providing values for the weight w_j.

3.4 Algorithm Analysis

Theorem 1. In worse case, the time complexity of our proposed algorithm is *O*(*n*), where n is the total number of ASC providers.

Proof. In scaling phase, the time complexity of getting normalizing matrix for all attributes is *O*(*kn*), where *k* is the number of attributes (constant), in this work, *k*=5. Then, weighting phase takes *O*(*n*). So, the worst time of algorithm is *O*(*n*).

4. Simulation Results

4.1 A Case Study

A case study of our proposed matchmaking algorithm will be given for explaining the effect of our proposed algorithm. The following experiment method will be used to select the most satisfactory ASC. Suppose the initial providers have 10 ASCs, i.e., $PASC = \{pasc_1, pasc_2, \ldots, pasc_{10}\}$. It means that these ten ASCs can provide the same or closely similar capabilities.

The QoS values of 10 ASCs are generated through the following simulation. Assume that there are 50 similar tasks (or ASC requests). For each request, we randomly delegate it to an ASC, *pasc$_i$*, from *PASC*, and randomly generate QoS attribute values and their fidelity values, denoting QoS value of *pasc$_i$* in this request. In these attributes, *t* is randomly distributed between 70 and 100 ms; *c* is uniformly distributed [10, 100] $; *rel*, *m*, *rep* is randomly generated in [0,1]. The fidelity for the first four attributes is in [0,1]. The QoS values of 10 ASCs for 50 requests are shown in Table 1, where *pn* denotes provided ASC number of *pasc$_i$* for all requests.

Table 1. The QoS values of 10 ASCs for 50 request

pasc	*pn*	QoS attribute value. (*time*, *cost*, *reliability*, *maintainability*, *reputation*)
pasc$_1$	4	(81.364, 94.46, 0.069, 0.614, 0.575), (73.451, 35.37, 0.838, 0.042, 0.187), (71.636, 99.96, 0.304, 0.106, 0.656), (99.294, 43.29, 0.384, 0.493, 0.011)
pasc$_2$	5	(84.663, 23.28, 0.740, 0.456, 0.716), (79.636, 70.50, 0.746, 0.767, 0.616), (91.45, 26.392, 0.427, 0.235, 0.245), (91.586, 92.89, 0.124, 0.888, 0.587), (73.615, 87.73, 0.738, 0.082, 0.804)
pasc$_3$	3	(99.547, 94.69, 0.690, 0.593, 0.930), (90.360, 86.49, 0.422, 0.235, 0.356), (90.825, 22.23, 0.678, 0.784, 0.763)
pasc$_4$	4	(77.087, 90.03, 0.947, 0.393, 0.697), (74.888, 44.23, 0.978, 0.834, 0.624), (80.114, 12.45, 0.502, 0.948, 0.990), (73.533, 35.89, 0.114, 0.325, 0.026)
pasc$_5$	9	(82.862, 32.69, 0.708, 0.043, 0.986), (73.595, 59.18, 0.611, 0.816, 0.219), (97.835, 69.63, 0.150, 0.017, 0.724), (91.110, 11.66, 0.540, 0.945, 0.221), (97.034, 54.32, 0.205, 0.832, 0.493), (93.738, 58.43, 0.659, 0.478, 0.182), (70.386, 36.56, 0.674, 0.919, 0.364), (70.894, 60.31, 0.288, 0.810, 0.637), (98.963, 65.05, 0.412, 0.013, 0.052)
pasc$_6$	4	(73.856, 41.60, 0.411, 0.647, 0.385), (89.247, 25.84, 0.409, 0.542, 0.114), (87.707, 16.07, 0.649, 0.706, 0.112), (88.005, 90.24, 0.991, 0.102, 0.045)
pasc$_7$	5	(91.256, 34.28, 0.566, 0.295, 0.087), (91.973, 61.65, 0.708, 0.659, 0.263), (89.541, 24.46, 0.096, 0.515, 0.247), (89.445, 27.47, 0.342, 0.484, 0.978), (85.628, 31.53, 0.692, 0.551, 0.706)
pasc$_8$	8	(82.750, 60.07, 0.224, 0.790, 0.624), (97.355, 89.42, 0.283, 0.307, 0.567), (71.577, 88.84, 0.643, 0.443, 0.804), (80.630, 51.27, 0.360, 0.116, 0.666), (97.848, 46.98, 0.243, 0.958, 0.529), (82.601, 50.14, 0.681, 0.785, 0.296), (75.847, 14.88, 0.266, 0.178, 0.547), (87.174, 71.81, 0.983, 0.453, 0.635)
pasc$_9$	4	(93.761, 60.72, 0.447, 0.177, 0.424), (75.342, 47.91, 0.730, 0.548, 0.783), (77.398, 70.89, 0.231, 0.295, 0.638), (78.867, 40.35, 0.624, 0.599, 0.457)
pasc$_{10}$	4	(88.814, 56.19, 0.883, 0.563, 0.029), (80.738, 64.78, 0.728, 0.648, 0.872), (75.470, 20.82, 0.637, 0.401, 0.752), (72.877, 20.72, 0.305, 0.750, 0.601)

The QoS values in matrix **Q** and the confidence fidelity matrix of the first four QoS attributes for ASC $pasc_i$ are the mean value of the results produced by its provided software component number, respectively. The QoS values matrix **Q**, confidence fidelity matrix **F**, and normalizing matrix **M** are as follows.

$$\mathbf{Q} = \begin{bmatrix} 81.44, 68.27, 0.40, 0.31, 0.36 \\ 84.19, 60.16, 0.56, 0.49, 0.59 \\ 93.58, 67.80, 0.60, 0.54, 0.68 \\ 76.41, 45.65, 0.64, 0.62, 0.58 \\ 86.27, 49.76, 0.47, 0.54, 0.43 \\ 84.70, 43.44, 0.62, 0.50, 0.16 \\ 89.57, 35.88, 0.48, 0.50, 0.46 \\ 84.47, 59.18, 0.46, 0.50, 0.58 \\ 81.34, 54.97, 0.51, 0.40, 0.58 \\ 79.47, 40.63, 0.64, 0.59, 0.56 \end{bmatrix} \quad (6)$$

$$\mathbf{F} = \begin{bmatrix} 0.34, 0.15, 0.41, 0.40 \\ 0.68, 0.47, 0.81, 0.36 \\ 0.54, 0.53, 0.41, 0.62 \\ 0.41, 0.76, 0.38, 0.26 \\ 0.51, 0.69, 0.43, 0.65 \\ 0.35, 0.30, 0.46, 0.48 \\ 0.60, 0.18, 0.66, 0.67 \\ 0.51, 0.42, 0.37, 0.46 \\ 0.42, 0.65, 0.50, 0.47 \\ 0.51, 0.65, 0.45, 0.47 \end{bmatrix} \quad (7)$$

$$\mathbf{M} = \begin{bmatrix} 0.71, 0.00, 0.00, 0.00, 0.37 \\ 0.55, 0.25, 0.65, 0.55, 0.83 \\ 0.00, 0.01, 0.83, 0.72, 1.00 \\ 1.00, 0.70, 0.99, 1.00, 0.81 \\ 0.43, 0.57, 0.31, 0.73, 0.51 \\ 0.52, 0.77, 0.90, 0.60, 0.00 \\ 0.23, 1.00, 0.34, 0.60, 0.56 \\ 0.53, 0.28, 0.26, 0.61, 0.81 \\ 0.71, 0.41, 0.46, 0.29, 0.79 \\ 0.82, 0.85, 1.00, 0.89, 0.77 \end{bmatrix} \quad (8)$$

Suppose that weight values of each QoS attribute defined by users are: w_1=0.15, w_2=0.2, w_3=0.2, w_4=0.15 and w_5=0.3. The overall quality scores of each ASC provider by using Formula (5) are shown in Table 2.

It is easy to see from Table 1 that pasc10 has maximal QoS score among all ASCs, and this ASC will be selected as optimal one with QoS performance.

The matchmaking result ASCs and their normalizing matrix are shown in Table 3 by using traditional algorithm, Zhang's algorithm [6] and Jiang's algorithm [7], respectively.

In traditional matchmaking algorithm, the first ASC is usually selected as match result, i.e., $pasc_1$. In Zhang's algorithm, the track records concept is equivalent to the fifth QoS attribute in our algorithm, i.e., *reputation*. So, the result QoS scores of each agent are: -1, 1.33, 1.33, 0.67, -1.33, -3.33, -1, 1.33, 0.67, 1. As a result, $pasc_2$, or $pasc_3$, or $pasc_8$ with maximal value (1.33) will be selected. Finally, Jiang's algorithm will be compared with our proposed algorithm, and fidelity factor is not considered by Jiang's algorithm. The overall QoS scores of each ASC by using their algorithm are: 0.218, 0.594, 0.576, 0.880, 0.503, 0.501, 0.562, 0.521, 0.562, 0.858. Obviously, $pasc_4$ (score is 0.880) is optimal ASC.

We can see from normalizing matrix **M** in Table 2, QoS performance (0.82, 0.85, 1.00, 0.89, 0.77) of $pasc_{10}$ gives slightly worse than performance (1.00, 0.70, 0.99, 1.00, 0.81) of $pasc_4$. But, fidelity performance of QoS attributes of pasc10 (0.51, 0.65, 0.45, 0.47) is evidently better than that of $pasc_4$ (0.41, 0.76, 0.38, 0.26). This phenomenon shows that some ASC providers usually claim their higher QoS attribute performance like pasc4, but if the confidence fidelity or users' feeling degree is also considered when selecting ASC, the overall quality or performance for these ASCs is not necessarily optimal. This point accords with the practical situation on Internet, such as e-business, online shopping, in which, people have always compromised selection need software function between quality and users' evaluation. Therefore, our proposed QoS matchmaking algorithm is effective and reasonable.

Table 2. The QoS score for each ASC provider

ASC	overall QoS	ASC	overall QoS
1	0.147	6	0.200
2	0.463	7	0.332
3	0.437	8	0.368
4	0.525	9	0.402
5	0.363	10	0.558

Table 3. Results by using different algorithms

Algorithm	ASC	QoS normalizing value
Traditional Alg.	1	(0.71, 0.00, 0.00, 0.00, 0.37)
Zhang's Alg.	2	(0.55, 0.25, 0.65, 0.55, 0.83)
	3	(0.00, 0.01, 0.83, 0.72, 1.00)
	8	(0.53, 0.28, 0.26, 0.61, 0.81)
Jiang's Alg.	4	(1.00, 0.70, 0.99, 1.00, 0.81)
Our proposed Alg.	10	(0.82, 0.85, 1.00, 0.89, 0.77)

4.2 Simulation Experiment

To evaluate the impact of fidelity on the final matchmaking results, we conduct the following simulations and compare the matchmaking results of our algorithm with Jiang's algorithm.

For the sake of simple, suppose that there are 10 initial ASCs. At each experiment, we randomly generate QoS attributes and fidelity values of each ASC according to the following strategy: *t* is randomly distributed between 10 and 100 ms; *c* is uniformly distributed [70, 100] $; *rel*, *m*, *rep* is randomly generated in [0,1]. The fidelity for the first four attributes is in [0,1]. At each experiment point, we record average values $\overline{M}$ of normalizing matrix **M** and average fidelity values $\overline{F}$ of matrix **F** of selected ASC with highest score by using Jiang's algorithm and our algorithm, respectively, i.e.,

$$\overline{M}(i)=\frac{\sum_{j=1}^{5}M_{i,j}}{5},\overline{F}(i)=\frac{\sum_{j=1}^{4}fid_{i,j}}{4} \tag{9}$$

where, *i* is numbering of ASC with highest QoS performance selected by using Jiang's algorithm or our algorithm. Let $Q_{ei}(i)$ in Formula (10) denote QoS evaluation index for current algorithm, in order to measure final QoS evaluation of selected ASC. The higher the Q_{ei}, the greater the comprehensive quality performance. In which, σ_1 and σ_2 are evaluation weight for QoS attribute and fidelity factor.

$$Q_{ei}(i)=\sigma_1\cdot\overline{M}(i)+\sigma_2\cdot\overline{F}(i) \tag{10}$$

Simulations are divided into three groups for different weight values of the QoS attributes in Formula (5) considering different users' quality preference.

Group 1: $w_1=w_2=w_3=w_4=w_5=0.2$, denoting unbiasedness;

Group 2: $w_1=w_2=0.35, w_3=w_4=w_5=0.1$, showing preference for *t* and *c*;

Group 3: $w_1=w_2=0.125,\quad w_3=w_4=w_5=0.25$, showing preference for other attributes.

For each group, we run the experiments for different evaluation weights pair, i.e., σ_1 from 0.9 down to 0.1, while σ_2 from 0.1 up to 0.9. We record average value of QoS evaluation index for 100 times in each point. The simulation results are shown in Figure 1, Figure 2, and Figure 3, respectively.

It can be seen from three figures that our algorithm gives higher QoS evaluation index performance than Jiang's algorithm in most cases for every group. Moreover, we observe from all figures that the performance of our algorithm is more stable, and the performance of Jiang's algorithm decreases as σ_1 decreases. This is because that Jiang's algorithm depends too much upon QoS values themselves, whereas QoS values and users' confidence fidelity are better compromised in our algorithm.

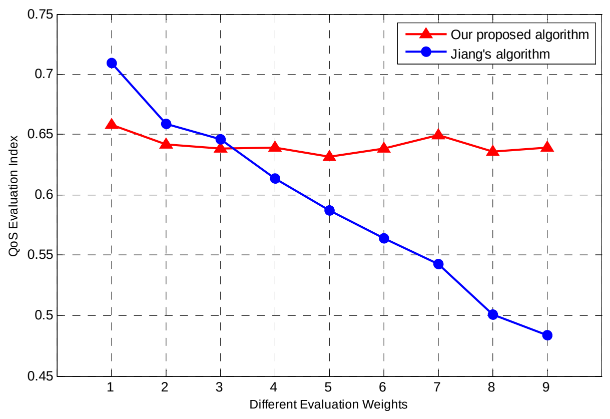

Figure 1. QoS evaluation index vs weights for Group1

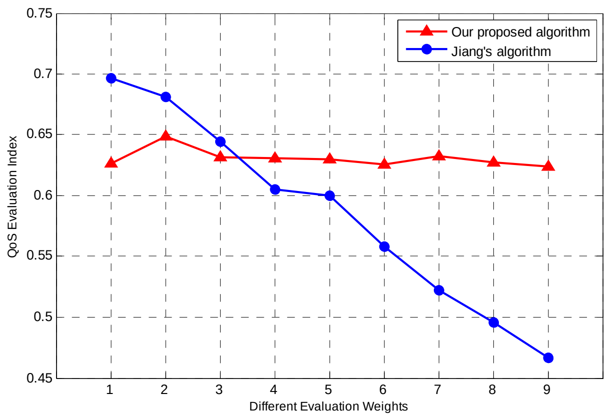

Figure 2. QoS evaluation index vs weights for Group2

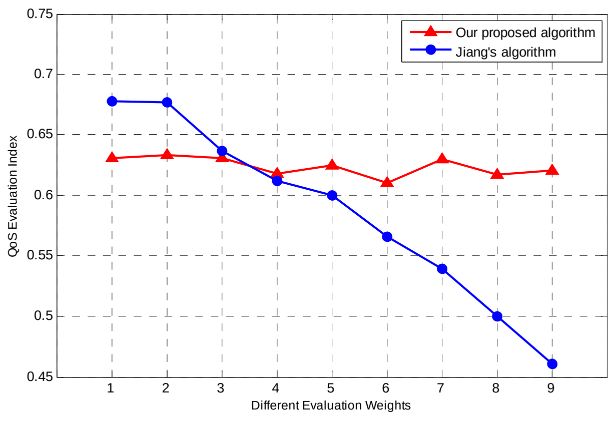

Figure 3. QoS evaluation index vs weights for Group3

To evaluate the effect of our proposed algorithm further, we conduct the following simulations and compare the QoS evaluation index of our algorithm with Jiang's algorithm, Traditional algorithm and Random algorithm. In traditional algorithm, the first ASC is selected as match result, i.e., *pasc*$_1$. Random algorithm randomly selects ASC. The simulations are also divided into three groups for different weight values of the QoS attributes as above. In each group, we assume there are 50 similar tasks (or ASC requests) and 10 ASC providers. At each experiment point, we record final QoS evaluation of selected ASC by using four algorithms, respectively.

The simulation results for three groups are shown in Figure 4, Figure 5, and Figure 6, respectively.

It can be seen from above three figures that our algorithm gives best QoS performance among compared four algorithms. In almost all experiment request point, the QoS evaluation index of selected optimal ASC by using our algorithm is higher than other three algorithms. The simulation results demonstrate better compromised selection between quality and users' evaluation in our algorithm.

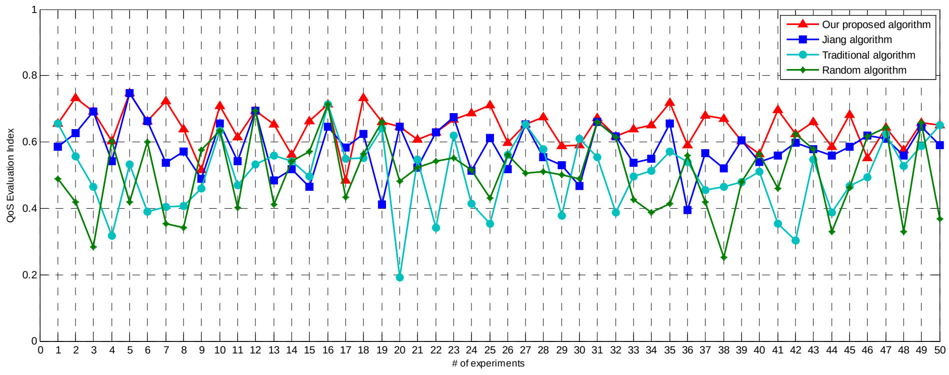

Figure 4. A comparison on QoS evaluation index of selected optimal ASC for Group1

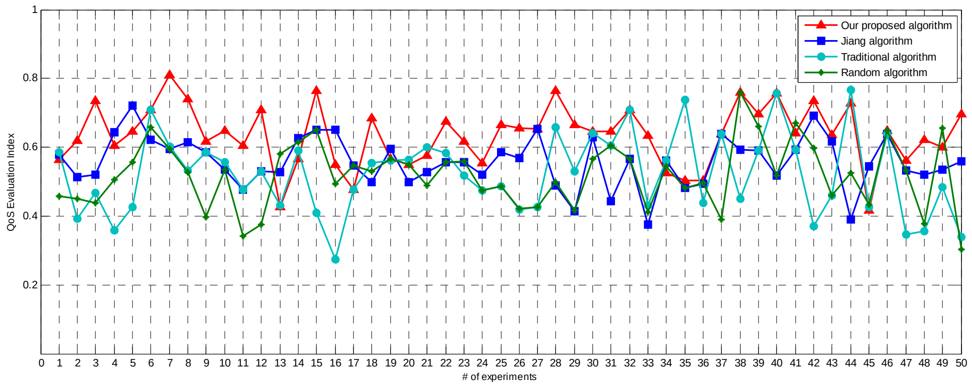

Figure 5. A comparison on QoS evaluation index of selected optimal ASC for Group2

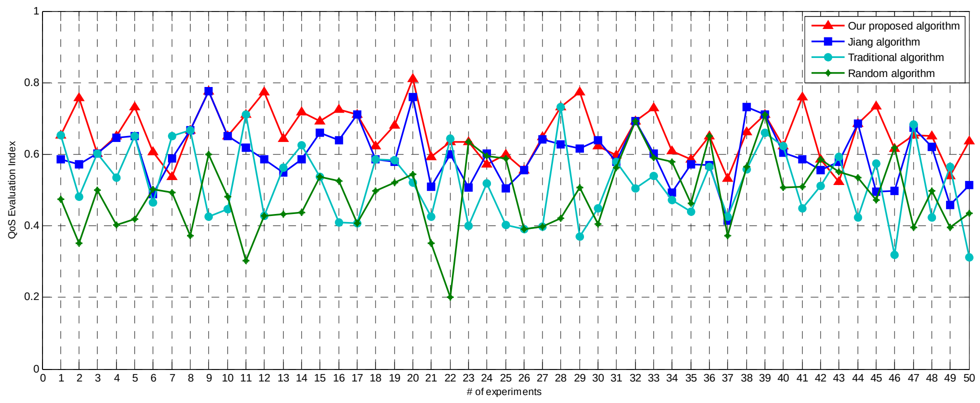

Figure 6. A comparison on QoS evaluation index of selected optimal ASC for Group3

5. Conclusions

In this paper, we discuss the autonomic software component QoS matchmaking problem for autonomic element, which has been taken as one of the most important issue in field of autonomic computing, especially in agent-based autonomic computing system. A QoS model for ASC is built, and on basis of which, an ASC QoS matchmaking algorithm based on fidelity factor is proposed. Main work in this paper has following characteristics:

1) Our proposed QoS model for ASC is simple and effective, and it does not limit type, amount, and value of QoS attributes, which shows better scalability and flexibility.

2) During ASC matchmaking process, we introduce evaluation mechanism of confidence of individual QoS attributes, i.e., fidelity factor for each attribute, which can overcome drawbacks such as subjectiveness and unfairness, and improve the self-configuration capability for autonomic element.

3) Simulation experiments demonstrate the effective and correction of our algorithm for matchmaking, and perform best performance in terms of QoS than other existing algorithms.

4) Simulations show also that our algorithm has better compromise between attribute quality and users' evaluation when selecting ASC. Our proposed algorithm is suitable for many situations on Internet, such as e-business, online shopping, in which, people have always compromised selection need software function between quality and users' evaluation.

6. Acknowledgements

This work is supported by the National Natural Science Foundation of China under Grant No. 90718021.

REFERENCES

[1] J. Kephart and D. Chess, "The vision of autonomic computing," IEEE Computer Society, No. 1, pp. 41–59, 2003.

[2] N. R. Jennings, "On agent-based software engineering," Artificial Intelligence, Vol. 177, No. 2, pp. 277–296, 2000.

[3] G. J. Wickler, "Using expressive and flexible action representations to reason about capabilities for intelligent agent cooperation," PhD Thesis, University of Edinburgh, Edinburgh, UK, 1999.

[4] K. Sycara, S. Widoff, M. Klusch, *et al.*, "LARKS: Dynamic matchmaking among heterogenous software agents in cyberspace," Autonomous Agents and MultiAgent Systems, Vol. 5, No. 2, pp. 173–203, 2002.

[5] K. Arisha, S. Kraus, F. Ozcan, *et al.*, "IMPACT: The interactive Maryland platform for agents collaborating together," IEEE Intelligent Systems, Vol. 14, No. 2, pp. 64–72, 1999.

[6] Z. L. Zhang and C. Q, Zhang, "An improvement to matchmaking algorithms for middle agents," in AAMAS'02: Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems, Bologna, Italy, pp. 1340–1347, 2002.

[7] Y. C. Jiang and Z. Z. Shi, "Quality of service driven agent service matchmaking," Mini-Micro Systems, Vol. 26, No. 4, pp. 687–692, 2005.

[8] R. J. Wang, Y. H. Ru, and X. X. Zhu, "Study on reclaiming agent selection models and approaches," in IEEE SOLI'08: Proceedings of IEEE Service Operations and Logistics, and Informatics, Beijing, China, Vol. 1, pp. 1262–1267, 2008.

[9] Q. Ma, H. Wang, Y. Li, *et al.*, "A semantic QoS-aware discovery framework for web services," in IEEE ICWS'08: Proceedings of IEEE International Conference on Web Services, Beijing, China, pp. 129–136, 2008.

[10] R. Hu, J. X. Liu, Z. H. Liao, *et al.*, "A web service matchmaking algorithm based on an extended QoS model," in IEEE ICNSC'08: Proceedings of IEEE International Conference on Networking, Sensing and Control, Sanya, China, pp. 1565–1570, 2008.

[11] L. L. Wei, C. L. Chi, M. C. Kuo, *et al.*, "Consumer-centric QoS-aware selection of web services," Journal of Computer and System Sciences archive, Vol. 74, No. 2, pp. 211–231, 2008.

[12] E. Giallonardo and E. Zimeo, "More semantics in QoS matching," in IEEE SOCA'07: In Proceedings of International Conference on Service-Oriented Computing and Applications, Newport Beach, USA, pp. 163–171, 2007.

[13] D. K. Guo, Y. Ren, H. H. Chen, *et al.*, "A web services selection and ranking model with QoS constraints," Journal of Shanghai JiaoTong University, Vol. 41, No. 6, pp. 870–875, 2007.

[14] K. J. Liang, Q. Liang, and Y. Yang, "The strategy and method of QoS parameter processing for grid service matchmaking," in ICICIC'06: In Proceedings of First International Conference on Innovative Computing, Information and Control, Vol. 1, pp. 381–384, 2006.