Service Selection based on Non-Functional Properties

Stephan Reiff-Marganiec[1], Hong Qing Yu[1], Marcel Tilly[2]

[1] Department of Computer Science, University of Leicester, UK
email: {srm13,hqy1}@le.ac.uk
[2] European Microsoft Innovation Centre, Aachen, Germany
email: marcel.tilly@microsoft.com

Abstract. Service-oriented Architecture supports software to be composed from services dynamically. Selecting and composing appropriate services according to business process, policies and non-functional constraints is an essential challenge. This paper proposes a method for automatic selection of the most relevant service for composition based on non-functional properties and the user's context. In doing this we also propose a method of obtaining and evaluating non-functional aspects.

1 Introduction and Motivation

Service-oriented Architecture (SOA) is by now widely used in the industry for solving B2B problems due to their ability to deliver flexible software systems that support the changing nature of business co-operations. The predominant implementation of SOA is called Web Services (WS). Considering WS, the fundamental standards are SOAP, WSDL and UDDI – together they address the fundamental paradigm of SOA: publish-find-bind.

Services are made available via the internet by a service provider, and their description is published (using WSDL descriptor files with details stored in UDDI repositories); a service consumer will query the UDDI repository to find an appropriate service and then use SOAP to invoke that service (this involves very late binding, essentially taking place at execution time). Currently this process is largely based on a human user making the decisions as to which service is suitable for their purpose. Furthermore, currently the matching is mostly based on functional requirements while non-functional aspects are not formally considered. However, in order to decide which service is most suited for a particular user in their current situation clearly depends on the functionality, but also on non-functional properties such as cost or reliability. Of course a UDDI repository might contain information about the cost of using a service or the service level agreements provided, but again this mostly for human consideration and hence not suitably formalised

for automatic selection. There has been some effort in the Semantic Web Services Community to address these issues, however adopting this requires much more fundamental changes than our suggestion and hence might not be as readily available in the short term.

The complexity of business processes and the dynamic nature of the co-operations make it difficult for the business modeller to select appropriate services, manage the compositions efficiently and understand requirements within a dynamic context correctly. In this paper we present the service management layer developed as part of the inContext project[1] which is aimed at addressing the above issue, in particular considering that a service's suitability depends largely on the user's context. We will focus on a specific aspect of this management layer: namely the service lookup and relevance ranking. What is special about this lookup is that in addition to the functional aspects of a service non-functional aspects are considered both when looking up a service as well as when finding the most suitable service.

The remainder of this paper is structured as follows: section 2 introduces the reader to relevant background and related work, section 3 presents the service management layer of the inContext platform and its position in the wider platform. Section 4 discusses how data concerning the non-functional aspects is obtained, while section 5 discusses how it is quantified. Section 6 describes the ranking mechanism and section 7 shows an example. Finally we round the paper off with a summary and discussion of future work in section 8.

2 Background

Most of the related work on using non-functional properties for service selection concentrates on defining QoS (Quality of Service) ontology languages and vocabularies and identification of various QoS metrics and their measurements with respect to semantic services.

In [1] and [2], QoS ontology models are defined, which propose QoS ontology frameworks aimed at formally describing QoS attributes. To our understanding, these works have not considered non-functional property based service matching and neither how to quantify the attributes.

[1] Interaction and Context Based Technologies for Collaborative Teams; EU-IST-2006-034718; www.in-context.eu

Ran [3] enumerates a large number of non-functional properties and organizes them into several categories, such as runtime-related, transaction support related, configuration management, cost-related QoS, and security-related QoS. However, the work fails to illustrate the quantifiable measurements as it simply assumes that all measured values are available somewhere.

The work in [4], [5], and [6] attempts to conduct a detailed evaluation and proposes QoS-based service selection. However, it does not explicit where the criteria come from. Additionally, all the current works does not consider the logic relations between criteria, using only the average of all individual values of the criteria as the final score.

Compared to the existing work, our selection approach has 5 major advantages. (1) Our process combines evaluation and selection activities in contrast to [4] and [7] which only address selection issues. (2) Our three measurement functions can deal with most types of criteria. This makes the measurement functions reusable and applicable to a wide range of non-functional attributes. Other work only focuses on criteria specific metrics and does not provide generic functions for all kinds of criteria. (3) Our method is more dynamic in that it automatically applies the correct metric while other work requires a manual association, or at least predefined maps, relating metrics to attributes. (4) We separate different non-functional criteria into different service categories. This is more sensible than ranking all kinds of services by using the same predefined criteria and hence not considering the different attributes that occur with specific services. (5) The key feature of incorporating the Logic Scoring of Preferences (LSP) method into our approach is that it captures the logic relations between criteria rather than just simply using average weight mechanism. LSP has been successfully used in manual multi criteria decision making.

3 Service Management

The inContext platform provides means of integrating services to support collaborative teams. In that sense it is a quite a complex structure and not all of it is relevant for this paper. An overview is provided in Figure 1.

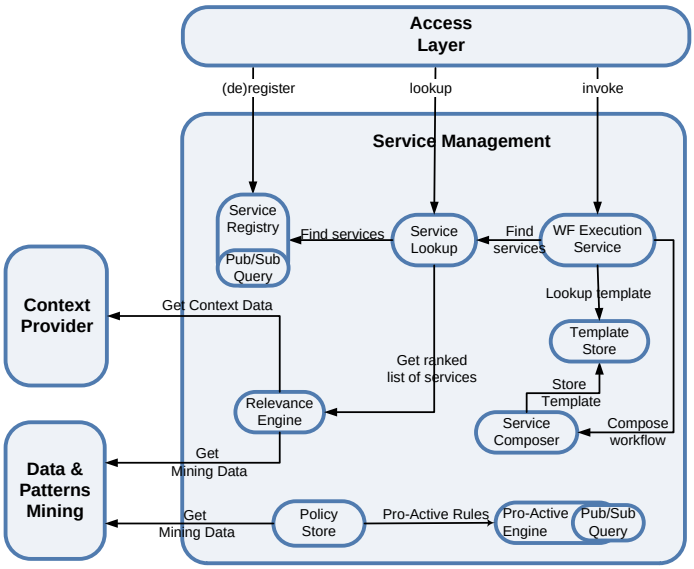

Figure 1: The Service Management Subsystem of the inContext Platform

The most interesting part is the service management core. This is located inside the platform, and provides access to service lookup, registration and invocation through the access layer to users. We will consider some aspects of the core in more detail, as is relevant for this paper – other aspects will remain unexplained here.

The other parts of the architecture that it interacts with are the context provider and the data and pattern mining subsystems. The context provider can be queried to gain insight into a user's current context which includes location, but more interestingly information about their activity and its relation to other collaborators. The data mining system provides information about past use of services and the situations that they have proven useful in.

A typical invocation starts with a request being submitted via the access layer to the service lookup. This queries the service registry to obtain all relevant services – that is services in the category that the user requires: e.g. a user might be looking for printing services. The list of services thus obtained is passed to the relevance engine, which conducts 2 tasks: filtering out services that do not meet minimal requirements (one might require a colour printer, and not all printing services will fulfil that need) and then rank the remaining ones based on the evaluation of additional, mostly non-functional, criteria (e.g. print cost or print quality).

An additional way of invocation is through the workflow execution engine, which when executing workflows will encounter the need to instantiate tasks with services. Again, this uses the lookup mechanism and the process described above, but it provides additional context in the form of which step is being executed next and which steps have already been executed thus allowing for a more fine grained service ranking taking the larger execution frame into account.

The next few sections will discuss how the data of non-functional properties is stored and used in the lookup and ranking steps described above.

4 Obtaining Relevant Non-functional Properties

Essential questions that have not been addressed by previous work include how one obtains the non-functional requirements (that is the requirements of the service user) and how the non-functional properties of the relevant services can be captured. We address this by assigning each operation offered by a service to one or more categories at registration time, essentially storing the extra information in the service repository. Capturing the data would be step in the design of individual services. We deliberately assign categories at an operation level, rather than service level, as a single service might offer quite diverse functionality. A category associates to a certain set of criteria (non-functional properties) which are defined during generation of the categories. As Figure 2 shows, a criterion is defined as a tuple *{Name, Type, Weight and Value}*. Name is the unique string for identifying criteria, the same string is used for the CriterionData in the semantic description of services. Type is the data type of each criterion such as Boolean, String, Integer, etc. The weights have an initial value created at the same time as the criteria. However, the values of the weights can be modified by end-users at invocation time. The meaning of the weight is as follows: if the weight is equal to 1, then the criterion is *hard requirement*, which means that services not satisfying this criterion should be discarded. If the weight is less than 1 and larger than 0, then criterion is considered a *soft requirement* which impacts on the final ranking of the service. When the criterion is of numerical type, the weight can be less than 0 but bigger than -1. In this special case, it means that a smaller numerical value is desired (as e.g. for bandwidth usage or price). The value attribute is to specify the parameter constrains for the criterion. For instance, a value for cost can be defined as 100 Pounds.

However, the value can be empty since if we simply want to have the lowest or highest one.

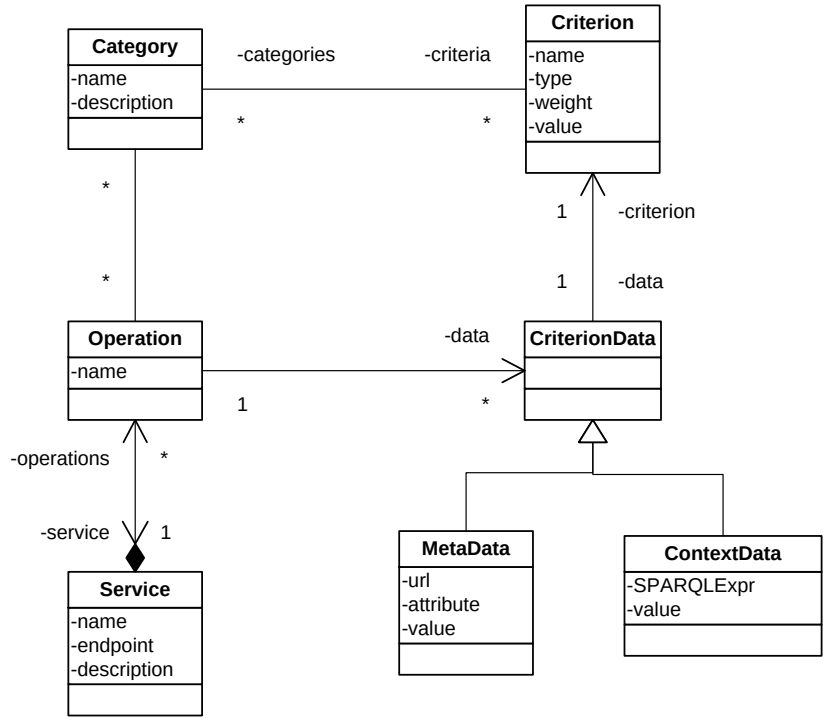

Figure 2: Generic model for non-functional service aspects

Furthermore, each criterion also has associated a CriterionData class; this is subclassed in two ways allowing for two different methods to obtain the related information from relevant services descriptions. Metadata allows gaining information through an URL, querying by id or attribute. ContextData describes how to get data from context management system by using the SPARQL query language. Figure 3 shows a concrete example in terms of "Send SMS" service category.

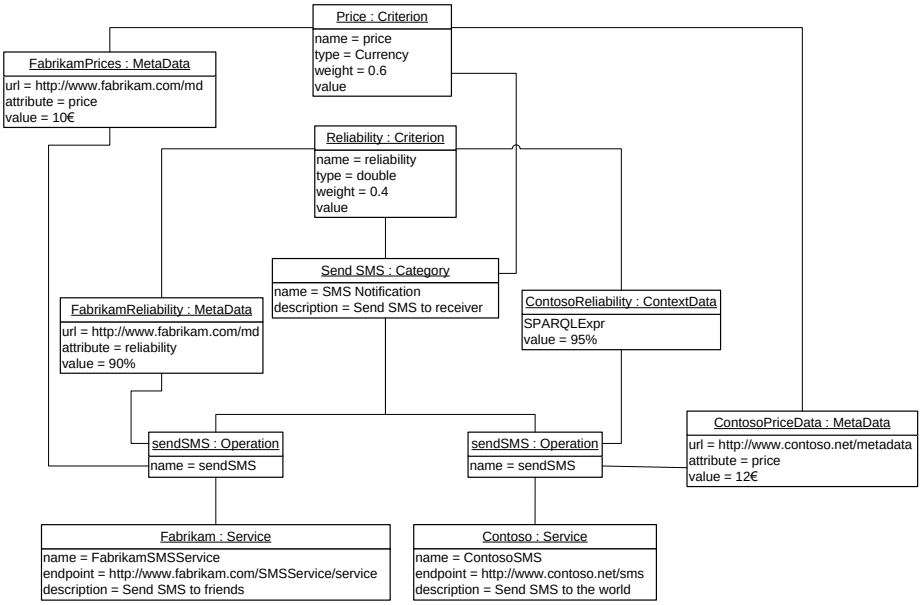

Figure 3: the "send-SMS" service class

5 Metrics for Non-functional Properties

As we discussed in the previous section, the values gained from context information are not only numerical types, but also text and Boolean types. For automated ranking and service selection it is crucial to invoke the correct evaluation functions dependent on the data type of the criteria; we found that we require three metrics (one for each of the three above types).

If the value of the criterion can be expressed by a numerical data type, then the numerical metric is used. The numerical type includes all the types that can be shown as number, such as “integer”, “double”, “time” and “currency”. For example, if the cost criterion can be defined as {Name=“cost”, type=“Currency”, weight=“-0.5”, value=“”}, then formula (1) will be invoked as evaluation function.

If the criterion is Boolean type, then the exact match will be used as Boolean metric (formula (2)). Moreover, the Boolean type is used not only to

capture 0/1 values, but also matching of text. For example the location criterion can be defined as {Name="location", type="Boolean", weight="1", value="UK"}. This definition implies that the service's location must be in UK.

However, not all text criteria can be defined as Boolean type. For example, the payment criterion can be defined as {Name="payment", type="string", weight="0.3", value="credit card, debit card, pay pal"} we use the set type. If the type is set, then it means that a subset of the provided string can be matched. Of course, a larger subset match is preferable. This contrast to the Boolean type in that a partial match will already provide a score, rather than requiring a full match. Formula (3) is designed for the set type, and will be selected in this case.

Because of the link to the data type, it can be automatically determined which function should be used. The respective formula to compute values for these three types are as follows:

1. Given v_{min} and v_{max} being the minimum and maximum value of all services. v is the value for the current service, we calculate (note that this calculation takes the weight W of the aspect in the aggregation into account):

$$E = \begin{cases} 1 - \left(\dfrac{v_{\max} - v}{v_{\max} - v_{\min}} \right) & \textit{iff } W \geq 0 \\ \left(\dfrac{v_{\max} - v}{v_{\max} - v_{\min}} \right) & \textit{otherwise} \end{cases}$$

2. $$E = \begin{cases} 1 & \textit{if criteria is met} \\ 0 & \textit{otherwise} \end{cases}$$

3. $$E = (e_1 + e_2 + ... + e_n)/n \quad \textit{with } e_i \textit{ being a score for each element of the set}$$

The big advantage of our metric is that we have designed evaluation rules dependent on the attributes' type. As the result, we can reuse one metric rule for different metric aspects.

6 Ranking services

Normally, more than one non-functional criterion is desired for a required service in the category. Therefore, we need a global preference calculation function $L(E_1(a_1),..,E_n(a_n))$ to calculate all aspects of criteria. The function itself must reflect specific requirements and logic conditions, such as simultaneity, replaceability and others [8]. The logic conditions can be reflected by the value of the weight. The function E_i is one of the individual evaluation methods which were discussed in the previous section. The formula (4) is defined as the function L:

$$L = (|\omega_1|E_1^r + |\omega_2|E_2^r + \ldots + |\omega_n|E_n^r)^{1/r} \text{ with } 0 \le E \le 1, \sum_{i=1}^{n} |\omega_i| = 1 \quad ,$$

where the ω is the weight of each criteria. *r* is the logic power value adopted from the LSP method [8], however we obtain the value of *r* automatically by using the method introduced in [9].

Additionally, we use the conjunctive partial absorption function as global aggregation structure (see Fig. 4) [10]. Preferences are separated into a hard criteria group (EP_i^h) and a soft criteria group (EP_j^S). The hard criteria group presents all mandatory requirements; the soft criteria group takes all other preferences. The function L is applied twice using two different values for *r*. The first *r* value is automatically calculated and is called DAC. The second *r* value is statically defined as CA; CA acts as a filter to weed out the services not satisfying one or more hard criteria. Note that DAC and CA are two of the typically offered LSP GCD operators, and details can be found in [11], but are immaterial here.

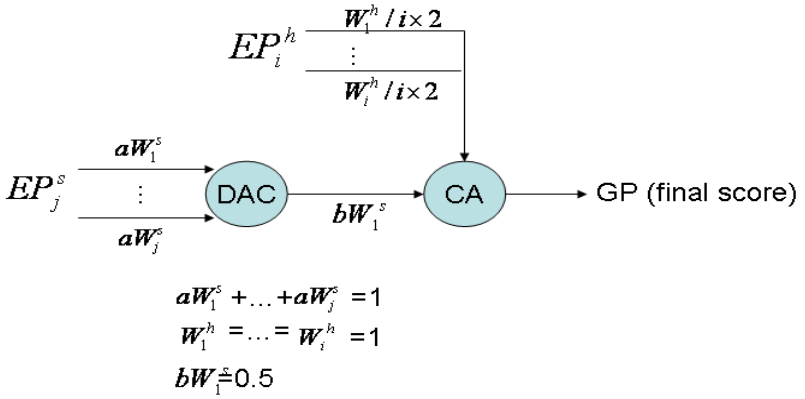

Figure 4: The structure of the *conjunctive partial absorption* aggregation function

Behaviours of the conjunctive partial absorption function are such that the global preference value (denoted by GP) will be 0 when any of the critical preferences are not satisfied, in which case the service will be discarded. On the other hand, a web service that satisfies all critical preferences will be valuated to a non-zero value, from which the degree of satisfaction of the desired preferences determines the final global preference.

7 Example

Let us now consider a scenario (depicted in Fig. 5), where a business organization needs a payment service to complete an online product selling business process. The payment services category includes several non-functional criteria. Firstly, the market aspect, the target customers might be at home or travelling. In the case that they are travelling, the customers could make use of several devices, such as a laptop, PDA, landline, desktop or mobile phone. Secondly, the QoS aspects, such as security must be high; performance rate should be reasonable and privacy should be respected. Thirdly, the policies include that customers are supposed to understand one language out of English, Spanish and French. Moreover, the service provider must be located in UK, Spain and France. A lower transaction fee is better. Finally, being able to accept more types of bank card such as Visa, MasterCard, Solo and Switch is preferable. Table 1 shows the definition of all criteria.

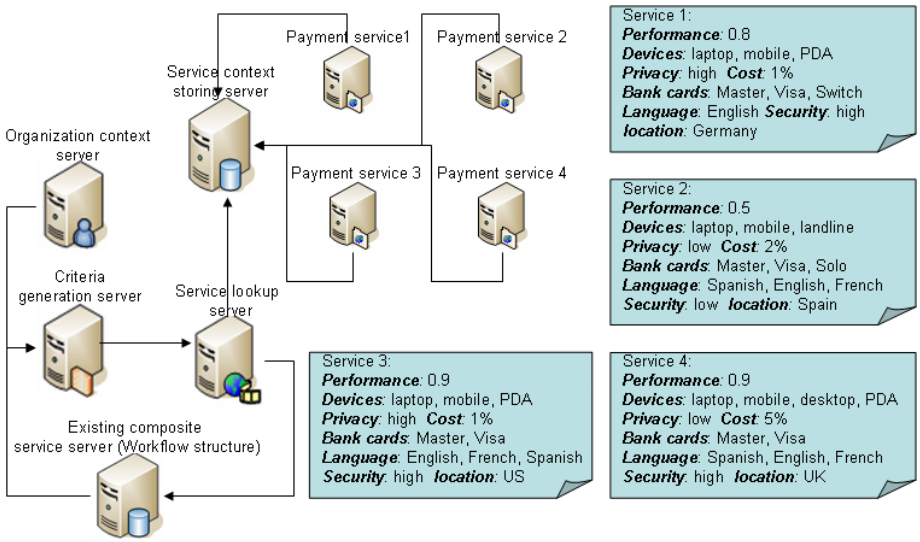

Figure 5: Four registered payment services

Soft Criteria	Data type	Weight	Value
Performance rate	Integer	0.1	
Devices	String	0.1	laptop, PDA, landline, desktop, mobile phone
Privacy	Boolean	0.1	High
Cost	Percentage	-0.5	
Bank cards	String	0.2	Visa, MasterCard
Hard Criteria	**Data type**	**Weight**	**Value**
Security	Boolean	1	High
Location	String	1	UK, Spain, France, US
Language	Boolean	1	English, Spanish, French

Table 1: Weights and types of criteria

There are four services available, which can functionally fulfil the payment task shown in Figure 5. To obtain a ranking result, we first calculate the value of the power r to be 3 by using the automated calculation method introduced in [9]. The global soft criteria evaluation results for each service are shown in Table 2.

	Performance	Devices	Privacy	Cost	Bank Cards	Global soft

	w	v	E	W	E	w	E	w	v	E	w	v	E	result L(1)
Service 1	0,1	0,8	0,75	0,1	0,6	0,1	1	0,5	1	1	0,2	M,V, Sw	0,67	0,8975
Service 2	0,1	0,5	0,00	0,1	0,6	0,1	0	0,5	2	0,75	0,2	M,V, So	1,00	0,7563
Service 3	0,1	0,9	1,00	0,1	0,6	0,1	1	0,5	1	1	0,2	M,V	0,67	0,9209
Service 4	0,1	0,9	1,00	0,1	0,8	0,1	0	0,5	5	0	0,2	M,V	0,67	0,5948

Table 2: Evaluation of soft criteria

This is not the final result as we also need to consider the hard criteria. We integrate soft and hard criterion and obtain the final results shown in Table 3.

	Security		Location			Language		Soft value		Final scores
	W	E	w	v	E	w	E	W	L(1)	L(2)
Service1	0,1667	1	0,1667	GER	0	0,16667	0	0,5	0,8975	Discard
Service2	0,1667	0	0,1667	SPN	0,25	0,16667	1	0,5	0,7563	Discard
Service3	**0,1667**	**1**	**0,1667**	**UK**	**0,25**	**0,16667**	**1**	**0,5**	**0,9209**	**0,6853**
Service4	0,1667	1	0,1667	US	0,25	0,16667	1	0,5	0,5948	0,5644

Table 3: Evaluation of hard criteria and overall result

The result means that for the given situation service 3 is the best one, second best is service 4 and services 1 and 2 are discarded as they do not satisfy the hard criteria.

8 Conclusion and Further Work

We have presented a method for selecting services based on non-functional requirements adding to ongoing work on selection of services in general which mostly concentrates on functional aspects.

Our method consists of a number of elements, notably a generic model for capturing non-functional attributes which are defined for categories of services with equal (or at least similar) functional behaviour and a method of automatically ranking services. The latter takes into account that some services might not fulfil what we termed hard selection criteria – essential requirements by the user – and filters these out while ranking the remaining ones using soft criteria (essentially preferences). The method is embedded in

the relevance engine of the inContext project, but is generic and hence can be used to enhance any service lookup process. In the framework of the inContext project, the engine makes use of user context to obtain requirements.

Future work includes enhancing the ranking mechanism to also include ranking based on information available through workflows: services are usually not executed on their own but in the context of other services and hence one might make different choices depending on the usage environment (a cheaper product buying service might become less preferential if high shipping costs occur).

Another aspect for future investigation is enhancement of the model for non-functional properties, especially capturing user requirements in other forms and hence depending less on the context system.

Acknowledgement

This work is supported by the inContext (Interaction and Context Based Technologies for Collaborative Teams) project IST-2006-034718.

References

1. I.V. Papaioannou, D.T. Tsesmetzis, I.G. Roussaki, and E.A. Miltiades, A QoS Ontology Language for Web-Services, aina, pp. 101-106, 20th International Conference on Advanced Information Networking and Applications - Volume 1 (AINA'06), 2006.
2. D.T. Tsesmetzis, I.G. Roussaki, I.V. Papaioannou and M.E. Anagnostou, QoS awareness support in Web-Service semantics, aict-iciw, p. 128, Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services (AICT-ICIW'06), 2006.
3. S.P. Ran, A Model for Web Services Discovery with QoS,. ACM SIGecom Exchanges, v.4 n.1, p.1-10, Spring, 2003.
4. Y. Liu, A.H.H. Ngu and L. Zeng, QoS Computation and Policing in Dynamic Web Service Selection, In Proceeding 13th International Conference, World Wide Web, 2004.
5. Y. Mou, J. Cao, S.S. Zhang, J.H. Zhang, Interactive Web Service Choice-Making Based on Extended QoS Model, CIT 2005, pp.1130-1134.
6. D.A. Menasce, QoS Issues in Web Services. IEEE Internet Computing, 2002, 6(6).
7. X. Wang , T. Vitvar, M. Kerrigan and I. Toma, A QoS-aware Selection Model for Semantic Web Services, ICSOC 2006.

8. J.J. Dujmovic, Continuous Preference Logic for System Evaluation, In Proceedings of Eurofuse 2005, edited by B. De Baets, J. Fodor, and D. Radojevic, ISBN 86-7172-022-5, Institute "MihajloPupin", Belgrade, 2005, pp. 56-80.
9. H.Q. Yu and H. Molina, A Modified LSP method for services evaluation and selection, In S. Gorton, M. Solanki and S. Reiff-Marganiec (eds) Proceedings of the 2nd European Young Researchers Workshop on Service Oriented Computing. June 2007.
10. J.J. Dujmovic, A Method for Evaluation and Selection of Complex Hardware and Software Systems. The 22nd International Conference for the Resource Management and Performance Evaluation of Enterprise Computing Systems. CMG 96 Proceedings, Vol. 1, 1996, pp. 368-378.
11. S. Y. W. Su, J. Dujmovic, D. S. Batory, S. B. Navathe, R. Elnicki. A Cost-Benefit Decision Model: Analysis, Comparison, and Selection of Data Management Systems. ACM Transactions on Database Systems, Vol. 12, No. 3, September 1987, pp. 472-520.