Personalizing the Usage of Complex Services

N.I. Yussupova
Ufa State Aviation Technical University
Ufa, Russia
e-mail: yussupova@ugatu.ac.ru

B. König-Ries
University of Jena
Jena, Germany
e-mail: koenig@informatik.uni-jena.de

D.V. Popov
Ufa State Aviation Technical University
Ufa, Russia
e-mail: popov@ugatu.ac.ru

I. A. Vaynerman
University of Jena
Jena, Germany
e-mail: igor@informatik.uni-jena.de

Abstract[1]

Granting access to functionality via services is a rather big part of the modern information technologies market. At the same time, the growing functionality of services makes their use more and more complex for the end user. This is especially pronounced when using mobile devices, such as a handheld computers (PDA) or smartphones with limited I/O-capabilities. In this paper we investigate how users can be supported in such environments by personalization of service interaction. We also discuss various aspects of how user priorities can be formalised and integrated into the service request. Such an approach allows for the reduction of the laboriousness of service usage by enabling its automation.

1. Introduction

The number of users which use mobile devices in daily life grows constantly. Recently, the functionality of such devices has changed significantly and their abilities are already comparable with those of usual PCs. This allows users to solve tasks once typical for desktop computers with handheld computers and smartphones". Examples for such activities are, e.g., an organizer, managing of files, using of multimedia abilities (viewing of video and listening to audio), working with documents (text documents, spreadsheets and presentations), using of e-mail and Internet browsing. Nevertheless, mobile devices still have limitations on resources and on abilities for information input-output. Besides, an obstacle for using „online" functionality is the high cost typically associated with it and the instability of mobile networks.

In our previous publications [1, 2] we showed the need to use services in such an environment in order to overcome the limitations mentioned above. The paradigm of service oriented computing is a promising and already quite widely applied approach to using distributed resources. Such resources are, e.g., information (for instance files or documents), processor time (distributed and remote calculations) or providing of remote functionality. Ideally, service oriented computing offers the means to automatically find appropriate resources for a given need. As discussed in [3] this requires rather complex descriptions of service offers and requests. While we can safely assume that service offers will be described by experts, this is not necessarily the case for service requests. Here, it is necessary to enable the end user to formulate appropriate requests. In this paper, we will show that this is a non-trivial task that requires sophisticated user support.

In the next section, DSD, our language for semantic descriptions of service offers and requests, will be presented. The differences from other existing languages for semantic service description will be briefly considered. A detailed example will show how complex even fairly simple requests become and why this is the case. In the remainder of the paper we will then consider the model of service usage personalization, which provides the user with means of support during formulation of the service request. This question is of great importance in view of increasing services functionality and as consequence of this the complication of service usage for the end-user. The paper ends with the conclusion and an overview of further work in the context of our research.

2. DSD – Language for Semantic Service Description

In this section we will briefly introduce the DIANE Service Description language (DSD) aimed at semantically describing services [4]. This technology is a

Proceedings of the 7th International Workshop on Computer Science and Information Technologies CSIT'2005
Ufa, Russia, 2005

language for the semantic modelling of service descriptions (requests and offers). Similar existing approaches are OWL-S (formerly DAML-S) [5] and WSMO [6]. The most important difference from these approaches with respect to this paper is the introduction of fuzzy declarative sets as a modelling entity between classes and instances. The concept "class" defines the set of all elements, which have some fixed properties, and a concrete object represents one element from this set. The declarative set can be defined as a set of objects of some class, which have some predefined properties with fixed values. That means this set can contain all objects, no objects, only one object or any quantity of objects of a certain class, depending on how the set of required properties and their values are defined. Since we define the sets to be fuzzy, it is possible to assign objects with degrees of membership in such a set by defining an appropriate membership function. In the case of an exact membership of an object, the result of this function is limited by logical "true" or "false". In the "fuzzy" case, the result is a degree of membership in the set. Below we consider this in more detail. In [7] we have shown that DSD is completely fit for fully automatic service matching and binding. The questions of offer and request description in DSD will be briefly considered below. More details about DSD can be found in [8].

2.1 An Abstract model of Service Description in DSD

There are several types of service description. Quite commonly, services are described by their input and output messages. The most prominent example of this class of descriptions is WSDL [9]. Another existing approach is based on state changes. The aggregation of all surrounding factors defines a current state, in other words a current environment. Two basic states are allocated: the state before service usage and the one after it. These states represent a precondition of service deploying and an effect of service usage made on the world around. If we describe these states and all their factors semantically, we obtain a semantic description of the service.

In Figure 1 the service description in DSD is represented. The formal description of a service is a graph, where each node is either a declarative set or a variable. Nodes have a number of properties, i.e. conditions and strategies. We will consider them in more detail below.

The two top elements of the service description are: the root node represents the service itself, and its direct descendant represents the service profile. Other children of the root include the service grounding and some non-functional properties. Since they are not of interest in the context of this paper, they are omitted here. The profile is an anonymous object of class *ServiceProfile*, it describes the functionality of the service. Direct descendants of the profile node are lists of service preconditions and effects. These and their descendants are either declarative sets (in Fig. 1 a rectangle with the inscription "Set" and cross in left top corner) or variables (in Fig. 1 a grey rectangle with the inscription "Variable"). In [8] a more detailed discussion of the rules, which determine the structure and the composition order of such trees, can be found. To find a service, a request tree should be compared with available offer trees.

2.2 An Example of Request in DSD

To understand how complex a service request can be, we consider an example of a train ticket booking. In Figure 2 the graphical description of the request to a service providing a booking of railway tickets is represented. The request means the following: "I want to book a ticket to a train which goes from Karlsruhe to Jena on the 24.06.2005. Desirable departure time is from 8 a.m. till 12 a.m. I need a reservation of a non-smoking seat in the second class for a ticket with a 50% discount (*bahncard50_2ndClass* in Fig. 2).

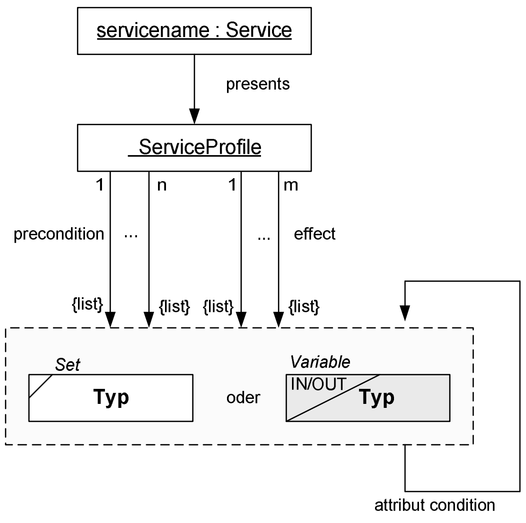

Figure 1. Common service description in DSD.

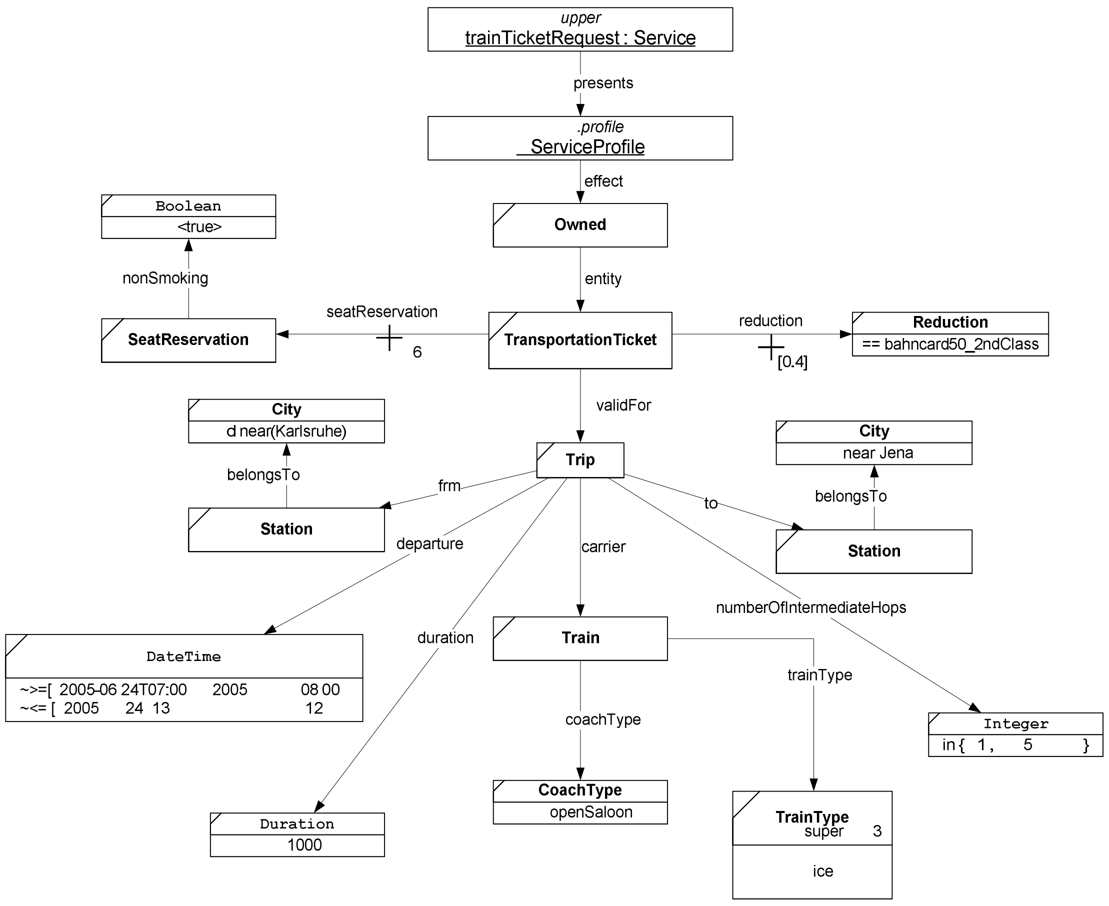

Figure 2. Example of request description in DSD.

Component name	Description
Type Condition	Defines the type (class) of set members
Direct Condition	Defines a condition which concerns directly to potential objects of set (but not to their attributes) It is possible to use the operators "==", "> = ", "<", etc. to define this condition. It can be set any number of such conditions, joined with logical "and", or no condition at all.
Property Condition	Intended for sets formed not by primitive types (classes). This condition defines a membership of objects in a set basing on values of their attributes.
Missing Strategy	Serves to define a membership of object in a set if not all of an object's attributes are set.
Connecting Strategy	Serves to define how separate property conditions will be combined
Type Check Strategy	Defines an opportunity of membership for objects of classes which are ancestors for a class of the given set

Table 1. Components of declarative sets.

The duration of the trip should be as short as possible; the desirable type of the train is the long-distance express (*ICE* in Fig. 2). The number of changes should be minimal“.

Before we discuss the request, it is necessary to consider the formal structure of a declarative set. In general a set may have three conditions and three strategies. These components define the membership of objects in the set, and relations between its attributes. In Table 1 a brief description of these elements is represented. There are two variants of objects membership in a set. The first variant is the exact membership of an object, i.e. an object is either a member of a set, or not. Another variant is a "fuzzy" membership, in other words, an object can be a member of a set to a certain degree. For example, the object *A* belongs to set *B* only to a degree of 80%. Such a fuzzy membership is only possible if conditions and strategies are also defined fuzzy. The exact set is a special case of the fuzzy set.

In Figure 2 the nodes *DateTime* and *Duration* have fuzzy direct conditions. In the node *Duration* the user expresses that the duration of the trip should be minimal. The direct condition is formulated as "<= [1000]0“. This means, that the duration is approximately less or equal to 0 minutes and the maximal admissible value is 1000 minutes. Certainly, the duration cannot be a negative number but the condition is set as "approximately", so such a formulation of the direct condition is semantically correct. The most appropriate value of this direct condition should be as close to 0 as possible, thus short trips gain higher membership values.

There is a second way to set a direct condition. In node *Trip,* the direct condition for the property *numberOfIntermediateHops* (number of changes) is set as the enumeration of possible values. The expression {0 [1], 1 [0.5], 2 [0.1]} means, that some object belongs to set with a degree of 100%, if its value of the "number of changes" property is 0; a value 1 implies a membership degree of 50%, the value 2 will result in a degree of 10 % only. The node *TransportationTicket* (ticket for transportation) has two properties *seatReservation* (reservation of places) and *reduction* (discount). For these properties, the fuzzy missing strategies are set. This means, that some object of class *TransportationTicket* belongs to the set with 60% if the attribute *seatReservation* is not defined and with 40%, if the attribute *reduction* is not defined. These values stand near the corresponding arrows representing the property conditions.

The fuzzy type check strategy can be set defining the maximal depth of inheritance and a number by which the membership degree will be multiplied at each step of inheritance. In the node *TrainType* (type of a train) this strategy is defined as "super [2, 0.3]". This means, that the set can include objects of classes which are a maximum second level ancestors of the class *TrainType*. By each step of an inheritance-level, the membership degree of an object will be multiplied by 0.3.

2.3 The Connecting Strategy

If you look at the request in Figure 3, you will notice that no information is given on how to combine the preferences on the individual properties. For instance the node *Trip* has 6 properties. In each case, an object can have different values of these properties and some values will be more suitable for the user than others. So, it is now unclear how these values should be combined. The problem is that we have the membership values for single properties, but we need to calculate the membership estimation for the whole node. Also, we should take into account how important the single properties are for the user. In other words, we need a function, which will express the importance of properties and will calculate the membership degree for the request node using the membership-degrees of its properties. The different importance of the properties defines the user preferences. For the reason of fuzzy sets usage, the values of request node properties should be also considered as the user preferences

The connecting strategy defines how property conditions will be combined. Actually, this strategy defines a membership function of the set. In case of an exact membership, this strategy is a logical expression containing all attributes of the set. In this expression the logical operators "and" and "or" can be used.

By default, property conditions of sets are combined by the logical operator "and". In the case of fuzzy sets, this operator will be replaced with multiplication. The expression "*A or B*" will be transformed to the expression "*A* + *B* - *A***B*" where *A* and *B* belong to an interval [0, 1]. When using fuzzy sets other functions can be used, too. In Table 2 the list of possible operations is represented. With the help of different combinations of properties users can formally express their priorities. By varying the connecting strategy it is possible to get different values of an object's membership degree.

Operation	Description
$min(x_1, x_2, \dots, x_n)$	Selection of a minimum value of properties x_1, x_2, … , x_n. Defines a case of an extreme conjunction.
$max(x_1, x_2, \dots, x_n)$	Selection of a maximum value of properties x_1, x_2, … , x_n. Defines a case of an extreme disjunction.
$(a_1*x_1 + a_2*x_2 + \dots + a_n*x_n)$	Weighted sum of property values x_1, x_2,…, x_n, where $a_1 + a_2 + \dots + a_n=1$.
$exp(x, a)$	Exponential transformation of property value x, is calculated as x^a.

Table 2. Operations for fuzzy connecting strategies.

Let us consider some possible connecting strategies for the nodes *TransportationTicket*, *Train* and *Trip* to show the influence the connecting strategy has on the desired result.

The node *Train* has only two properties: *coachType* (type of the seat) and *trainType* (type of the train). Assume the user has the following preferences: "To me, the type of train is a lot more important than the type of seat". These preferences can be mathematically expressed by a weighted sum, where the property *trainType* has a bigger weight than the property *coachType*. The following expression *(coachType*0.3 + trainType*0.7)* describes a connecting strategy, which reflects the user wishes expressed above.

The node *TransportationTicket* has three properties. They are: *seatReservation* (reservation of the seat), *reduction* (the discount for the ticket) and *validFor* (the property defines a trip for which the ticket is valid). Let us consider the following possible preferences of the user: "Of course, I want a ticket that is valid for the trip that I want to take. If at all possible, I also want to get the discount, even if that means that I won't be able to get a reservation." One possibility to encode these preferences is the following connecting strategy: *min (validFor, (validFor*0.9 + 0.1*(seatReservation*0.2 + reduction*0.8)))*. The user priorities concerning the seat reservation and the discount are expressed by the weighted sum. The weight of the property *seatReservatio*n is much less than the weight of the property *reduction*. With help of these weights the user expresses his attitude towards single properties. The trip itself is the most important parameter. This is expressed by the minimum-operation. If the value of *validFor*-property will be equal to 0, then the result value of the membership-function will also be 0. In the case of a non-zero value of the *validFor*-property, the value of the weighted sum consisting of *validFor* and a combination of the *seatReservatio*n-property and the *reduction*-property will be chosen. The *validFor*-property has in this sum the bigger weight (i.e. it is more important) than a combination of the two other properties.

The node *Trip* has six properties: *frm* (a city of departure), *to* (a city of arrival), *departure* (a date and a time of departure), *nOfIntermHops* (number of changes), *duration* (duration of a way), *carrier* (type of a vehicle). Let us consider two possible variants of the user preferences and the appropriate connecting strategies; we will compare how the various strategies influence the semantics of the request:

In the first case, the user states "It is very important that the trip starts and ends where I specified. Also, date and time of departure are pretty important. Among the remaining properties, it is as important that I have few changes as that duration and type of vehicle are as desired. If I have to choose between a short trip and one in the right kind of vehicle I prefer the one in the right type of vehicle The following expression is a suitable connecting strategy in this case: *0.5*exp((frm and to), 2) + 0.3*departure + 0.1*nOfIntermHops + 0.1*(0.4*duration + 0.6*carrier)*. The importance of single properties and their combinations is expressed through the weighted sum. The properties *frm* and *to* have in this case the equal importance and influence on the result equally, therefore they are connected with logical "and". The operator "and" in case of fuzzy sets will be treated as multiplication. Here it defines, that if one of the properties is equal 0, the value of whole item of the weighted sum will be also 0. The exponential transformation is used in the expression to emphasise the importance of the departure and arrival city.

In the second case, the user states:" I definitely want to depart in the city specified in the request. The city of arrival is also important, but not quite as important. It is followed in importance by the type of vehicle. The remaining three properties have the same importance. A connecting strategy, which expresses the user wishes, is: (*0.1* min(nOfIntermHops, duration, departure) + 0.3*carrier + 0.6* to) and frm*. The strategy consists of two parts, connecting with a logical „and". This expresses that the end result will be 0 if the value of the *frm*-property doesn't match the appropriate city at all, or the whole second multiplier is 0. By the fuzzy membership of an object, each part of the product will reduce (increase) the end result. Therefore the fuzzy membership-degree of the *frm*-property has the proportional effect on the overall membership-degree. The second multiplier of the product is the weighted sum of other properties. The city of arrival has the biggest weight that designates its importance concerning other properties. The next important property is the type of a vehicle; this is also underlined through its weight. Three other properties are connected by the minimum-operation, which expresses their identical importance to each other. The influence on the end result has only the lowest value of these properties. Concerning other properties, the last three have the lowest weights, in other words importance.

2.4 The need for Support

The examples should have made it obvious that the average user will not be able or willing to formulate requests like the one shown here. Support is needed with respect to a number of requirements:

- Filling out requests is cumbersome. Whenever possible, properties should be filled in automatically. For instance, most users will always want smoking or non-smoking seats in the same type of coach.
- Specifying property conditions is often unintuitive. Means of ranking possible different values should be provided.
- Specifying connecting strategies is highly complex. Users should be guided through this process.

3. Personalization of Service Usage

In the previous section of the paper we have motivated the need for user support. In our opinion the most suitable approach to provide this support is through personalisation of service usage. This process affects all elements of service usage: The *user interface* should be adapted to the hardware abilities of the device which is used to access a service. The functionality provided by service usage should meet the requirements and the expectances of the concrete user. Since our matcher guarantees to find only services that meet what is specified in the service request, this can be achieved by ensuring that the right service requests are posed. There are two possibilities to formulate such a request. Either the user should iteratively refine the request to get the most suitable one or he should formulate the extreme complex request, which will meet all the user requirements for the desired service. Another case for the complex request is a repeatedly usage of the service. It is worth to spend one time the efforts formulating such a request and then use the service without interaction of user. This in turn can only be achieved by sophisticated user support. To implement this concept, we have developed the following model of service personalisation. In Figure 3 the basic scheme of this model is represented. *DIANE Middleware* is the lowermost element of this structure, this is the system aimed to bind and manage services. In particular, here the user requests are compared with the available offers and the estimation of their match to the request is calculated. It is possible to allocate four basic parts of the personalisation model: Adaptable Graphical Interface, Communication Facade, User Semantic Profile and the block of calculation of mathematical representation of user priorities.

To accomplish the first requirement from Section 2.4 we have developed the mechanisms described below. The adaptable graphical interface represents a modular structure to transform information of the input and output streams into the form determined by the user device. To develop a new representation of the user interface it is necessary to implement the corresponding module which will communicate with the communication facade through the program interfaces.

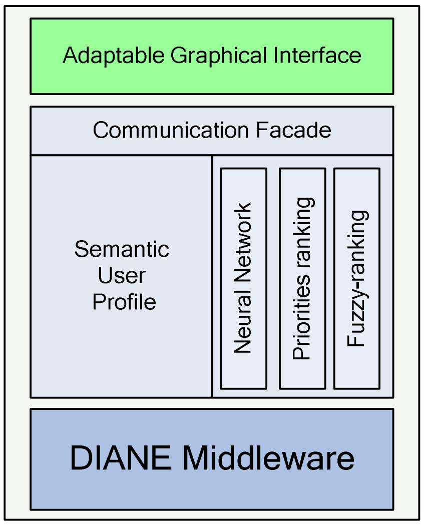

Figure 3. The model of system for service personalization.

The communication facade manages the in- and output data as well as other blocks of the system. The User Semantic Profile is intended for storage of meta-information about the user priorities. The information describing the user preferences in various areas of a human life is stored here. Actually, these are the mathematically expressed user priorities for a concrete ontology. This data is necessary for the dynamic creation of requests with the minimal participation of the user in this process.

The most complex element of the model is the block for calculation of mathematical representations of user priorities. It has a modular organization. At the moment we have developed three modules which differ from each other in the mathematical model used.

The first two approaches are based on the principle of ranking of the request parameters. They aim to provide the user with means corresponding to the second requirement from Section 2.4. Ranking in this case means that the user should grade the request parameters with respect to a scale of priorities. For example, this can be done by choosing for each parameter one value (rank) from the set of the predefined levels of importance.

The difference between the modules is the mathematical “background”. In the first case to transform the ranks into numerical form simple mathematical functions (parabola-function or exponent-function) are used. After ranking of the request parameters, the ranks will be converted into numbers with the help of different functions. For instance, the function $y=a*x^2$ will be used. y – is the calculated result match-value for a request parameter, which has the original match-value x. The rank, which defined by user for this parameter, will assign the factor of the transformation a. These factors would be precalculated and predefined for each scale of priorities. Then the functions of single parameters of one node can be connected in a connecting strategy with the help of the function of calculating the average value.

In the second case, functions from the theory of fuzzy logic are used. The user can set here not only the single ranks, but also the ranges of the parameter values. This defines the correspondence of the ranks with appropriate values of parameters. Two cases are possible. Either the user sets the ranks for the single parameters and then the result membership-degree is assigned through the fuzzy-transformation functions or the user defines the ranks for the ranges of the parameter values. In the first case the calculated result-values can be used in a connecting strategy, which will be also formulated with help of the function of calculating of average value. In the other case the ranks and the ranges of the values can be used by formulating of a direct condition. The single ranks assign the membership-degree for the possible parameter values through the fuzzy-transformation functions taking into account the location of this value concerning the predefined ranges.

The last approach is based on usage of the theory of neural networks. This approach corresponds to the third requirement from Section 2.4. On the scheme this module is indicated as *Neural Network*. The user request is transformed into a neural network. As explained above, the user request is a graph. In principle, we can imagine the nodes of this graph as neurons and its edges as connections between neurons. So, we need only the transformation function for the neurons and we will be able to construct a neuronal network representing the user request. When the network is constructed, it should be taught. To teach the neural network, the user should iteratively consider the possible suitable service offers and give his evaluation of these offers, i.e. tell the system how well this offer matches his wishes. This means, that the network needs a feedback from the user. During each step of the teaching phase suitable offers will be provided to the user. He should rate it by setting the estimate-values for the request parameters i.e. the neurons. These estimate-values can be not only numbers, but also some abstract quantity as rank. Then the ranks would be converted into numbers and these numbers will be used to calculate the new neuron weights. After several trails of such teaching steps the weights would be repeatedly optimized. These weights, calculated on a teaching phase, represent mathematical estimations of the user priorities. This module is used in complex cases, for instance when the request has a lot of nodes. The bottleneck of such approach is the phase of network teaching.

The last two modules can be applied for simple requests or separate sets of request nodes. It is necessary also to say, that the opportunity of combined use of all three approaches is now developed. We also work on the creation of new methods of mathematical calculating of the user priorities.

4. Conclusion and Outlook to Further Work

In this paper we have described and analysed different opportunities of usage of resources in various networks. We have shown the ways of using network services. Also we have presented the DSD language aimed to semantically describe the services. We have considered the components of this language and shown haw complex can be the service request. This language has the powerful means to express the abstract user priorities, but at the same time these means are pretty difficult to use for a normal user. Therefore, there is a need to support user with methods, which can help user to formulate his preferences in a form of formal parameters of the service request. So, we have presented the model of the system to provide the user with means of support by formation of the service request. The direction of the further researches is a practical realization and testing of this system. The other direction is the further development of means to formally represent the user priorities.

References

1. Yussupova N.I., König-Ries B., Popov D.V., Vaynerman I.A. “User Support for Formulating Complex Service Requests”. In: *Proc. of the 6th International Workshop on Computer Science and Information Technologies CSIT’2004.* Budapest, Hungary, 2004

2. I.Vaynerman “Service personalization for user support”. In: *Proc. of the 17th Workshop über Grundlagen von Datenbanken(GvD’2005).* Wörlitz, Deutschland, 2005

3. Michael Klein, Birgitta König-Ries, Michael Müssig ”What is needed for Semantic Service Descriptions - A Proposal for Suitable Language Constructs” *International Journal on Web and Grid Services, 2/2005*

4. D I A N E - Projekt http://hnsp.inf-bb.uni-jena.de/diane/

5. OWL-S Homepage http://www.daml.org/services/owl-s/

6. WSMO Homepage http://www.wsmo.org/

7. Klein M., König-Ries B. “Combining Query and Preference - An Approach to Fully Automatize Dynamic Service Binding”. In: *Proc. of IEEE International Conference on Web Services (ICWS 2004)*, 6.-9. Juli 2004, San Diego, CA, USA.

8. Handbuch zur DIANE Service Description. Technischer Bericht der Fakultät für Informatik; Universität Karlsruhe (TH). TR-2004-17, ISSN 1432-7864.

9. Web Services Description Language (WSDL) http://www.w3.org/TR/2002/WD-wsdl12-20020709