

User Support for Formulating Complex Service Requests

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

B. König-Ries
University of Karlsruhe
Karlsruhe, Germany
e-mail: koenigri@in.tum.de

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

I.A. Vainerman
Ufa State Aviation Technical University
Ufa, Russia
e-mail: igor@ipd.uni-karlsruhe.de

Abstract¹

The potential to dynamically bind services at run time is one of the big advantages of service oriented computing. However, dynamic service binding requires rather sophisticated service requests. These requests need to be precise enough to allow for automatic invocation of the service without user intervention, while at the same time flexible enough to allow finding all possible candidate services. Obviously, it is not trivial to formulate such requests. In this paper, we examine how the user could be supported in this complex task.

1. Introduction

Service orientation is a promising paradigm for distributed resource usage. Service orientation has the potential to fully automate resource sharing. Full automation makes sense for repetitive business processes. Consider as an example a company that needs to buy 10,000 screws a month with a certain specification. Nowadays, even if web services are used for the ordering, a person in the purchasing department will manually select a supplier from those she finds using e.g. UDDI that might match the specification. Once this supplier is chosen, it is used regularly. While this supplier may be the best match for the request at the time it is chosen, this may well change over time: New suppliers might become available, other suppliers may lower their prices or offer screws with a higher quality. Thus, it would be better to have a system that allows to dynamically choose a

provider whenever new screws need to be ordered. Obviously this should not require human intervention each time. Therefore, what is needed is a system that allows for automatic service matching and binding. With such a system it would be possible to describe once which functionality should be provided and to then automatically find the best match for this request at each invocation.

Full automation of service matching and binding requires complex service requests. The reason for this is that in conventional systems, the search results are presented to the user who then filters them and selects the most appropriate service provider. In our example, currently, the user in the purchasing department will formulate a rather generic service request and will use this request to obtain a set of possibly matching service providers from the UDDI repository. The user will then look at the suggested providers and determine which of these providers matches best the requirements. This filtering and ranking is based on information that is either provided in human readable form in the UDDI or obtained offline. In a fully automated system, this manual filtering and selecting does not take place. This means that the initial request formulated by the user needs to be precise enough to strictly restrict the search result to services that offer exactly the functionality needed. Also, the request needs to contain all the information that is needed to rank among possible service providers. At the same time, service offers need to be described precisely, too, so that automatic matching becomes feasible. Within the DIANE project, suitable languages for service request and offer description have been developed [1]. They allow flexibly describing and processing the semantics of service offers and requests. If offers and requests are formulated in this language, automatic service matching and binding is possible.

Unfortunately, user requests become rather complex and nontrivial, if they are to match the requirements mentioned above. In our opinion, it is unlikely, that the user will succeed in formulating such a request without

¹ *Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CSIT copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Institute for Contemporary Education JMSUICE. To copy otherwise, or to republish, requires a fee and/or special permission from the JMSUICE.*

support by the system. The most likely scenario is that the user will formulate an initial request and will then check with a trial run whether the services found with this request match his expectations. If this is not the case, the user will adapt the request, rerun it, check the results once more and so on until, finally, the desired result set is retrieved. The request that produced this result set is then stored and used in future iterations of the process.

In this paper, we investigate where and how the user should be supported during request formulation. Explicit user support in this phase has two advantages: On the one hand, the user is guided and may reach the optimal request faster, on the other hand, it is possible to ensure that network load is kept to a minimum during this process.

The remainder of this paper is structured as follows: Section 2 contains a short overview of the DIANE request language. Section 3 describes where users need to be supported when formulating service requests, while Section 4 looks at one of the major issues (namely the relative weight different preferences should have in a user query) in more detail. Section 5 discusses related approaches and explains where we need to extend these to match our requirements. Section 6 concludes the paper with a summary and an outlook on our future work.

2. DIANE Request Language

Within the DIANE Project [3], the DIANE Service Descriptions have been developed. In [1] it was shown that DIANE Service Descriptions (DSD) are suitable for fully automatic service discovery and binding. In this approach user requests specify the desired state transition of the required service using ontologies. That is, services are characterized not by their inputs and outputs but by their preconditions and effects. This allows for semantic service matching. DSD is partly based on OWL-S (formerly DAML-S) [7] service description languages of the semantic web. This brings the possibilities of processing the service descriptions on semantic level. More details can be found in [9]. DIANE Service Descriptions can be represented in a number of different formats. Three important formats are: a graphical representation, java-code and a formal language. In this paper, we use the graphic representation to give a brief overview of the language. More information about DSD can be found in [2].

Figure 1 depicts an example of a service request². All the nodes in the request graph represent either object sets or variables. Object sets represent the containers of objects of one or more types. Variables are a special kind of object sets. They represent objects, which should be bound by the user or provider before, during or after service execution. Nodes are connected with arrows, which represent the object properties. The property value

² DSD also offers the possibility to describe service offers. We do not introduce this part of the language here.

can be a variable or another object set. Each node has a number of DSD parameters, such as: *Missing Strategy* (MS) and *Connection Strategy* (CS). Information about all the DSD parameters can be found in [2]. The missing and connection strategy can be used to express user priorities. Consider again Figure 1. Here, a service request for a printing service is depicted. The user asks for a service, that has a *profile*³ with a state after service execution (*effect* in Fig. 1) of type *Printed*. The object to be printed should have type *Document* with two properties *Format* and *URL*, which should be defined by the user before service execution. The document should be printed in colour and finished by 10 o'clock. After service execution the user wants to get the possible resolution and location of printout. The connection strategy specifies how the individual conditions should be combined. For instance, the user could specify that the color should be counted twice as important as the required output time. The missing strategy specifies what should happen, if an offer does not provide any information about a certain condition, e.g., the user should specify that, if an offer does not specify whether the printout will be in colour, this condition should be assumed failed, as it is most likely, that a colour printer would specify this.

Thus, these strategies define the performance of the matching. Correct matching results, that are results that match the implicit expectations of the user, will depend strongly on correctly specifying these conditions. Unfortunately, writing requests in DSD can be rather complex. The main problem is that the user will usually not be able to predict beforehand how a certain connection strategy will influence what result he receives. Hence, there is a pressing question of how to make the process of request formulation easier for the user. Our approach for the solution of this problem will be described in the next section.

3. Requirements for User Support

In the introduction, we have explained that automatic dynamic service binding is particularly attractive for repetitive business processes. We have introduced the DSD as a language that allows to formulate service requests that are precise enough to allow for this automation. In this section, we will take a closer look at an example to show where the user needs to be supported when formulating appropriate service requests using DSD.

As an example we will consider several steps of a business process on some mobile device (Fig. 2). Let us assume that we should create a report in PDF format, print it and then send it via a courier service to our headquarters. We are using some mobile device e.g. a

³ The service profile describes, what a service does. There are other parts of the service description describing how this functionality is provided that are not of interest here.

PDA or smartphone. At first, we create a report in .RTF format (e.g. in Pocket Word), but to make the further

operations we need a .PDF document.

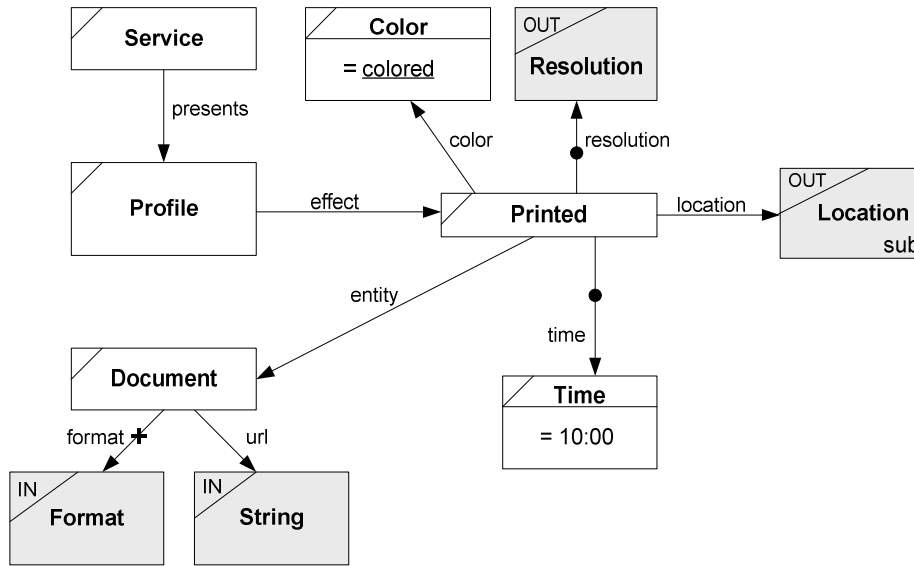


Figure 1. Example of a State Oriented Request Description for a Printing Service

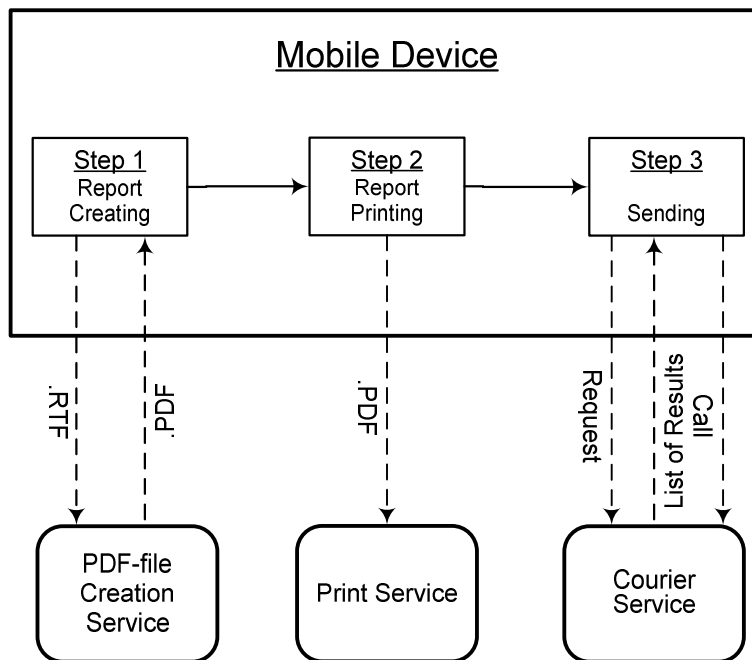


Figure 2. Business Process on a Mobile Device.

The process of converting into PDF format needs high processor performance and the execution on a mobile device can take very long. Therefore, it makes sense to use a suitable network service instead of using the mobile application. Thus, we have to find and call such a service. For this purpose it is necessary to make a request, get the list of results and choose the most suitable. But if we have to make a set of operation more than one time a day and the list of results can be rather big, it could be very difficult and requires a lot of user efforts. For instance, we

make several different reports a day. Each time we have to do all the operation to find and call the necessary converting-service and each time we use the same service. Obviously, the user needs to be supported. Converting - this, apparently, simple action requires a great number of efforts and makes the using of network service inconvenient and inefficient. The user needs to be provided with efficient, full-automatic methods of executing the routine and repeating actions.

So, after report converting the user needs to print it. He can do this in two ways. The first one is to use a rigidly determined printer. The user knows the name of this printer e.g. from previously using it, and sends the report to it. If our business process makes this step automatically we need to have 100% guarantees that the report will be printed. However, it could be that the user has gone from one place to another (i.e. from one building to another) and there is no connection with the remote printer, or it doesn't work. In this case, we should be able to choose the suitable printer each time before printing. Thus, it arises the second way to print the report: using a network printing service. Instead of calling the remote printer, we should send the request to print the report with some parameters to the service. Possible parameters would be, e.g., resolution, time or place, where the user can pick up the printed report. In an ideal case, the user should only send a request; the service chooses the appropriate printer and takes all necessary actions. The request can be generated by a client program on the user's machine - hence all further actions should be automatically executed by the service. This is also a point where the user should be supported in devising the correct request to obtain the report at the time, place and quality he needs it.

When the report is printed and picked up, we need to send it by courier. The main problem is to choose the appropriate courier company. Again, the user should be able to specify once what the characteristics of such a company are; it should then be ensured that during future iterations of the business process the request is processed and the optimal courier company for the task at hand chosen. The request can contain a lot of parameters and priorities, since the decision on what courier service to use may be a complex one. This makes the formulation of the query rather hard for the user. Also, if the list of the answers contains a lot of appropriate answers with similar characteristics, it should be possible to automatically rank them. Again, the parameters for the ranking need to be provided in the user query.

We mentioned above, that the user will probably not be able (even with support by the system) to come up with the "perfect" request, i.e. the request that produces exactly the intended output, from scratch. Rather, the formulation of the query will be an iterative process involving trial runs. These trial runs should be realized in a way that minimizes network load. In the next section, we take a closer look at the determination of preferences in the user request.

4. Priority Parameterization of the User Requests

In Section 3 we have shown that the user should be supported during the request formulation. Let us consider the example with the courier-service in more detail. The user needs to send the report and he is searching for the company, which offers the best conditions. For these purposes he uses the network searching-service. The user

defines the requirements for the courier-company. Now, it is necessary to determine the relations between the request parameters. These relations define the user's priorities, i.e. the hierarchy of the parameters. The priorities show what service aspects are more important than others. In other words the priorities define what request parameters are more critical for the user at service searching. They help the user to express his preferences in a request. During the processing of the request, it is necessary to separate the appropriate service offers from the not suitable. Also, the best matching service offer has to be determined. The DIANE Service Description includes methods and algorithms to perform the matching. However, for these methods to work, the user request needs to contain precise information. For instance, it may be necessary to specify how certain attributes of an object should be matched. That is, the user may need to tell the system when to consider two values to be similar or which value of two given values to prefer. For example, with price it is better when it is smaller, but with printer resolution this is not so. This information needs to be provided somewhere. There is also a question about the range of parameter value. If there is no exact answer, in what borders it is reasonable to search for it? Since the answers to these questions need to be provided as part of the user request, the system should guide the user in pointing out what additional information is needed to allow for a meaningful matching of the user's request to service offers.

Also, the user request needs to contain information about how to combine the different aspects of the request formulated. Which aspects absolutely need to be taken into consideration? Which aspects is the user willing to dispense with? How should the different aspects be weighted? Again, the system should guide the user through this process.

What we have described up to now are support mechanisms that will ensure that the user request is syntactically correct and contains all the information needed to perform service matching. However, while this is a necessary first step, it does not guarantee, that the user request does indeed describe the service the user is looking for. Therefore, the system should offer a mechanism that allows the user to perform trial runs of his requests. The mechanism should then display the results obtained and allow the user to adapt the request parameters. Even better would be a system that displays the results obtained, allows the user to indicate how close they are to the intended result and which then automatically adjusts the request parameters to obtain results closer to the intended ones.

So, we have shown that user support is an important part in providing of network services and priority parameterization of the user requests. There are many different aspects in user support, which make the realisation of user support and priority parameterization non-trivial and difficult.

5. Related Work

In the previous sections, we have shown that user support is required to enable users to successfully formulate complex requests. The main difficulty in formulating these requests is the correct prioritizing of user preferences. Therefore, in this section, we take a look at existing approaches that allow users to formulate preference based queries. This topic has enjoyed quite a lot of attention over the last few years. We will show that the existing approaches offer a valuable basis for our work. However, we will also show that they fall short of providing the kind of user support needed here.

Skyline Queries [8] are preference based queries that return as a result the “skyline” of best results. These are those results that are better than others along at least one dimension, i.e. pareto optimal. Moving along the skyline is equivalent to shifting the relative importance of the different dimensions. With respect to our problem, a skyline algorithm seems a good starting point to find a set of candidate services. However, in addition to what the skyline algorithm provides, we need a method to decide which point of the skyline to choose as the best match of the user’s expectations. Unfortunately, the existing algorithm does not offer any support here.

Preference SQL [6] is more flexible and allows the combination of different preferences not only as pareto preference, but also as prioritized preference or as a weighted sum of preferences. Thus, preference SQL allows expressing a wide variety of preferences. However, Preference SQL encounters the same problem that the formulation of requests with DSD poses: It is difficult for the user to judge beforehand, how a certain setting will influence the query result. Preference SQL offers an explanation component that at least allows comprehending how a result was produced once this has been done. Again, Preference SQL seems like an interesting basis for implementation of preference based user requests. It does not address the fundamental problem described in this paper, that is how to come up with the correct request in the first place. The same is true for the XPath counterpart of Preference SQL, Preference XPath [5].

Service Globe [4] is another example of a system that allows (at least to a certain degree) the formulation of preference based queries. Again, no support for designing correct queries is given.

6. Conclusion

In this paper, we have shown that automatic service matching and binding require user support in formulating appropriate service requests. These requests will have to be preference based. We have discussed existing approaches to preference based querying and have shown that while they offer methods to compute query results based on a given query, they do not support the user in asking the correct query in the first place. Given the

complexity of the queries required here, we believe that such a support is an absolute necessity. In our future work, we intend to develop a system that offers just this kind of support. This system should at the same time support the user efficiently and make economic use of sparse network resources.

The information about our previous works in the area of network services can be found in [10–13].

Acknowledgments

This investigation is partially supported by the following grants: 03-07-90242-B "Projects Implementation Support Internet-Ware for Complex Systems Fundamental Research Using Intelligent Technologies Based on Expert Systems" (Russian Foundation for Basic Research, section "Creation and Development of Informational, Computational, and Telecommunicational Resources for Fundamental Research", 2003-2005), and 03-07-00039 "Fundamental Research and New Technologies for Complex Technical Systems Design" (Russian Federal Target Program "Integration of Science and High Education in Russia for 2002-2006", section "International Partnership Activities Based on Mutual Research and Development of Integrated Scientific-Educational Institutions", 2002-2006).

References

1. Klein M., König-Ries B. “Combining Query and Preference - An Approach to Fully Automate Dynamic Service Binding”. In: *Proc. of IEEE International Conference on Web Services (ICWS 2004)*, 6.-9. Juli 2004, San Diego, CA, USA.
2. Klein M., König-Ries B. “Coupled Signature and Specification Matching for Automatic Service Binding”. In: *Proc. of European Conference on Web Services (ECOWS 2004)*, 27.-30. Sept. 2004, Erfurt.
3. DIANE - Projekt <http://www.ipd.uka.de/DIANE/>
4. Keidl M., Seltz S., Kemper A. “Reliable Web Service Execution and Deployment in Dynamic Environments”. *TES 2003*, pp. 104-118.
5. The official web-site of WWW Consortium <http://www.w3.org/TR/xpath>
6. Kießling W., Köstler G. “Preference SQL - Design, Implementation, Experiences”. *VLDB 2002*, pp. 990-1001.
7. Semantic Web Services <http://www.daml.org/services/>
8. Kossmann D., Ramsak F., Rost S. “Shooting Stars in the Sky: An Online Algorithm for Skyline Queries” In: *Proc. of VLDB 2002*, Hong Kong, China, 2002.
9. Klein M., König-Ries B. “A Process and a Tool for Creating Service Descriptions based on DAML-S B”.

- In: *Proc. of 4th VLDB Workshop on Technologies for E-Services (TES'03)*. Berlin, 8 September 2003.
10. Koenig-Ries B., Popov D., Muelle J., Plechova O. "Multidimensional Query Result Navigation for Mobile Users". In: *Proc. "Computer Science and Information Technologies CSIT'2002"*. University of Patras, Greece, 2002,. [http:// www. lar.ee.upatras.gr/ csit2002/](http://www.lar.ee.upatras.gr/csit2002/).
 11. Popov D.V., Vainerman I.A.: "A Prototype of the System to Provide Mobile Users with Services in Wireless Networks ". In: *Making Decisions under Indeterminacy Conditions*. USATU, Ufa, Russia, 2003, pp.14-22. (In Russian).
 12. Yussupova N.I., König-Ries B., Popov D.V., Vainerman I.A. "Data Structures for a System to Dynamically Provide Mobile Users with Information in Wireless Networks". In: *Proc. of the 5th International Workshop Computer Science and Information Technologies CSIT'2003*. Ufa, Russia, 2003, Vol. 1, pp. 100-107.
 13. Yussupova N.I., Popov D.V., Vainerman I.A., König-Ries B. "RDF-Technologies to Provide Mobile Users with Services in Wireless Networks". In: *Proc. of the 6th International Conference Complex Systems: Control and Modeling Problems*. Samara, Russia, 2004, pp. 203-208.