Context-aware service discovery, selection and usage

Friederike Klan
Institute of Computer Science
Friedrich-Schiller-University Jena
friederike.klan@informatik.uni-jena.de

Abstract

Ongoing advances in technology and proceeding miniaturization have led to growing popularity and widespread use of mobile devices. Apart from communication features and basic functionality like address books or organizers users expect of today's portable and handheld devices to provide more advanced features such as presenting relevant timetable information, a location plan or information about nearby shopping facilities. In spite of the advanced technological opportunities mobile devices are still restricted in terms of computing power, storage space, battery life, available network bandwidth and display size. One solution to overcome these shortcomings is by utilizing external functionality provided by services together with appropriate filtering mechanisms to obtain only relevant information. Since available and required services may vary due to the device's mobility, the process of service discovery, selection and comparison has to be context-aware. In this paper we present our general idea of a framework for context-aware service discovery, selection and usage. We analyze problems and challenges in conjunction with our approach and discuss related works in this area of research.

1 Introduction

Mobile devices are restricted in terms of computing power, storage space and periphery. Therefore they have to rely on external resources to obtain information or functionality. Since mobile devices operate in changing, dynamic environments, it does not make sense to have static connections to specific resources. Instead, it should be possible to dynamically bind appropriate resources at runtime. This should happen automatically, i.e. transparent to the user. Service-oriented computing is an appropriate approach to enable this. It allows to provide functionality or information as stand-alone services, that can be described by a service offer, then published, automatically discovered and selected by comparing a given service request with available offers (service matching), composed and executed.

However, automatic service matching, selection and composition is only feasible, if service offers can be described accurately and the wishes of a service requester can be expressed precisely within the service request. This results in fairly complex requests. It is unrealistic to assume that the user of a small mobile device would be able or willing to formulate such a request. In this paper, we argue that request formulation can be dramatically eased by automatically adapting requests to the current situation. A tourist looking for a city map, e.g., is most likely interested in one depicting his current location. The system could infer this without having the user to explicitly state it. More formally, the current situation can be regarded as context as defined by Dey [2]: "Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and the application themselves." In conjunction with our mobile service oriented scenario one could think for instance of a user, a mobile device or its surrounding as an entity, while contextual information that may be relevant to the process of service selection and binding could be preferences of a user, the current battery load or network

availability of a mobile device or information of the local environment like current brightness, noise level or weather conditions. A second difficulty with regard to using services in mobile environments that can also be overcome with the help of context information, is the fact that service offers may differ according to the current situation. Consider for instance a service located at a PDA that offers information about the weather conditions at its current position. Obviously the area for which meteorologic data is available changes with the PDA's location.

In the following sections we first introduce an approach for semantic service description, selection and invocation. Afterwards we analyze where contextual information could be helpful in this process and present our general idea of a framework for context-aware service discovery, selection and usage. We conclude with an examination of related work and a discussion of future work.

2 DIANE Service description and selection

Our approach for context-aware service discovery, selection and usage is based on the semantic service description language DSD (DIANE Service Description) [6] and its mechanisms for completely automatic semantic service matching, selection and binding which were developed within the DIANE project [1] and build up the DIANE middleware. We therefore shortly introduce the most important features of DSD to provide a foundation for our further discussion.

DSD is an ontology-based, purely state oriented description language, i.e. the functionality a service provides is described by means of the required state(s) of the world before (*precondition(s)*) and after (*effect(s)*) its execution. Preconditions and effects are represented by sets of possible states. A set is characterized by its *type condition*, which determines the concept in the ontology of which the members of the set have to be instances of and may be supplemented by *direct conditions* on its members and/or additional attributes (*property conditions*) (see Figure 1). Sets are configurable via variables symbolizing the inputs and outputs of a service. In doing so the influence of the parameters on the effects of a service is explicit. To prevent service descriptions from being unstructured, DSD provides an *upper service ontology* as well as some *category ontologies* to define the basic structure of service descriptions and to categorize possible states of the world. Additionally a wide range of *domain ontologies* is available or can be created to characterize services in the various fields of application. This results in both structured and flexible service descriptions (see Figure 1).

Service descriptions by the provider and the requester usually differ in their precision. Whereas a service provider is able to describe his offer exactly, a requester is often interested in a certain functionality, but not in a special service. DSD takes this into account by allowing to declaratively specify a fuzzy set of suitable effects along with preferences in a service request. Given a set of DSD offers the DIANE semantic matcher determines a service or a combination of services that best fits a given DSD request effectively and in a completely automatic fashion. Afterwards the service(s) is invoked by the DIANE middleware without requiring additional human intervention.

3 Where and how to use context?

Using some examples we identify where and how to use contextual information to ease the process of request formulation and make automatic service selection in mobile environments more efficient. Moreover we identify the main challenges that arise. In this conjunction we introduce the concepts of extending and refining a service request and motivate the necessity to enhance service offers with dynamic parts.

Context-aware request extension and refinement. Imagine a tourist, who is currently on the way in Jena as a pedestrian, looking for service(s) providing a map of her surrounding

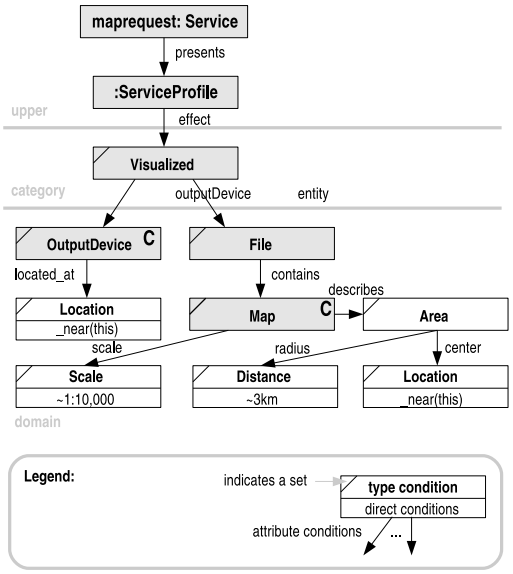

Figure 1: Original (shaded part) and extended service request

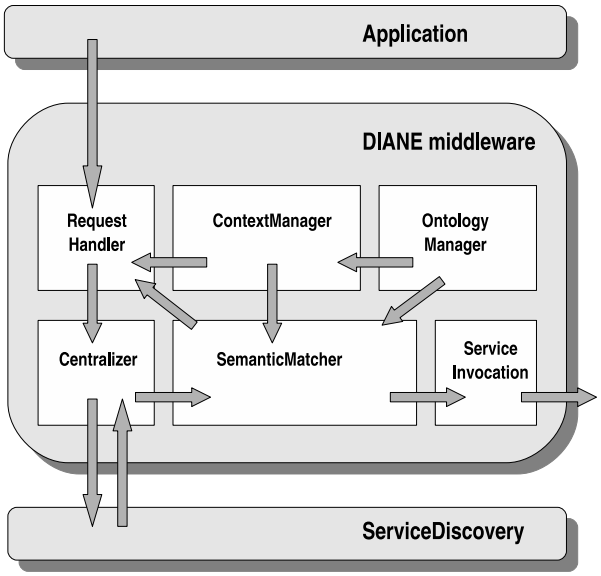

Figure 2: Architecture

area in an appropriate resolution and a device suitable to output it. Since proper requests tend to be complex (for an example see Figure 1), the process of their creation is difficult and time-consuming. For this reason it is desireable that the tourist simply has to ask for a city map and an output device instead. After that the original request is enriched automatically with additional information by means of the context in which it was given and thus expressing the tourists needs more exactly. Regarding our example we can restate the request, now containing additional attributes describing the area the map represents and the demanded scale (*request extension*). Additionally, we have to refine the request by replacing the more general concept `OutputDevice` by `Display` (*request refinement*), since simply asking for an output device wouldn't exclude devices that are not suitable for presenting a map. The above example demonstrates that request extension and refinement have the potential to specify service requests according to the requesters needs and thus making the matchmaking more selective and therefore more efficient.

The realization of a request adaption as pictured above requires to think about a lot of questions. The most essential topics are: What kind of changes to the structure and the content of a request due to available context information are useful, particularly how to realize extensions and refinements as depicted above? Which parts of a service request should be modified and which should remain unchanged? The most challenging question is how to describe the effect that available contextual information have on a given request.

Context-aware offer completion. In particular the functionality provided by mobile devices slightly differs depending on the present context. Consider as an example the PDA with meteorological information mentioned in the introduction. One can also imagine that the functionality provided by a service on a smart phone depends on the charging state of the device's battery. Since it would be very costly or even impossible to frequently update the corresponding service offer or to provide suitable offers for every situation, we have to keep some parts of the service offer open and complete them when needed by the matcher. Thus the process of service selection is more effective and also more efficient, since it is always based on up-to-date information.

With regard to offer completion we mainly focus on the questions of how to describe dynamic

parts in a service offer and how to complete them.

4 Context-aware service-oriented architecture

In the previous section we argued that service request extension resp. refinement and dynamic offer completion can be utilized for facilitating the process of request formulation and improve the efficiency of service selection and discovery in mobile environments. We now introduce our ideas of how one can realize the proposed request and offer adaptions. Afterwards we present our general idea of the overall framework for context-aware service discovery, selection and usage.

Context-aware request extension and refinement. The idea is to extend DSD service requests by additional property conditions provided by the category and domain ontologies and to refine them by either adding direct conditions to declarative sets already occuring in the offer or restricting the type of such sets to a suitable subconcept. The request designer is therefore asked to mark (with a **C** in the box symbolizing the set, see Figure 1) every set in her offer that may be modified by exploiting contextual information. In detail that means, if some attribute conditions are missing for a marked set, they may be added provided that appropriate contextual information is available. Similarly, if present sets are marked, they may be refined by adding direct conditions or restricting them to a suitable subclass.

Let us come back to the examples introduced in Section 3 to illustrate our ideas. First remember the tourist who asked for a service that provides a city map. To make the original request more suitable for the requester's needs, we add the two attributes `Scale` and `Area` to concretize the required precision and the topic of the map [1]. After adding these two attributes we may refine them for instance by stating the direct condition `~1:10,000` for `Scale`. Additionally we can refine the requests part refering to the output device by substituting the type `OutputDevice` by `Display` [2] (see Figure 1).

Context-aware offer completion. Service offers are composed of static and dynamic parts, whereby the first ones are ordinary DSD offer descriptions and the latter ones are attribute values or direct conditions that are typically subject to frequent updates and therefore remain open at the creation time of an offer. They are marked in the offer description and can be inquired for instance from the service provider via knowledge services when needed. As an example imagine a dynamic attribute `Area` in the offer description of the meteorological service introduced in Section 1, that can be completed when needed.

Framework. We give a summary of our overall architecture for context-aware service discovery, selection and usage (Figure 2) that extends the DIANE middleware for semantic service discovery (see Section 2).

DSD service requests transmitted to the middleware by an application are forwarded to the *RequestHandler*, which coordinates the process of extending and refining requests utilizing the information provided by the *ContextManager*. The *Centralizer* provides the extended service request to the low-level *ServiceDiscovery* that performs a first coarse prematching and returns the resulting set of service offers. Hereafter the *SemanticMatcher* compares these service offers with the extended service request and completes dynamic parts if needed by utilizing knowledge service requests that are also handled by the RequestHandler. Based on the resulting matching values the best fitting offer is chosen and the corresponding service can be invoked by the *ServiceInvocation*.

[1] We suppose that the motherset is marked and sets of the type `Scale` and `Area` are legal attributes according to the ontology.

[2] We assume that `OutputDevice` is marked and a subconcept `Display` exists.

5 Related work

Most of the existing approaches for context-aware service discovery and selection rely on a key-value-pair [8, 9] or keyword-based [4] service matching process. Context information are also represented by key-value-pairs and may be supplemented by simple if-then-rules. An exeption is the COSS approach [3] that utilizes an extensible set of ontologies for context and service description and therefore provides a common understanding of the represented information in contrast to the other solutions. The CAPEUS architecture of [11] also represents context information by key-value-pairs, but allows to describe entities an simple relations between these.

Instead of integrating context data into service offers the systems of [4, 8, 9] seperately add them to the offers. Consequently the context-based matching process in these architectures (except for the CAPEUS architecture) occurs subsequently to the ordinary service matching. Thus the potential for service selection provided by context information isn't used to full capacity. Besides [9] none of the matching algorithms takes user preferences into account. The approach of [3] also allows to integrate user-defined attributes like 'nearby' into an service request. Advanced concepts for context-aware request modification come from the field of database systems [5], but aren't directly adaptable to service oriented architectures.

Some approaches allow dynamic attributes within service offers. The process of updating their values takes place either by permanent monitoring [7, 8, 9] or on inquiry [10], but isn't coupled to the matching process, where we actually need it. Merely [8, 9] utilize dynamic service attributes within the matching process.

6 Conclusion and future work

Context-aware request extension and refinement as well as context-aware offer completion have the potential to make the process of service selection and usage more effective and efficient. In our future work we plan to further concretize these ideas and to implement the proposed architecture. Additionally we intend to utilize other types of context information like quality of service information or user profiles to further improve the matchmaking process. Concerning the process of context acquisition and provision we examine how service composition can be utilized for this process assuming that context information are provided by services.

References

[1] DIANE-Project, Services in ad hoc Networks. http://hnsp.inf-bb.uni-jena.de/DIANE, 2006.

[2] A. K. Dey. Understanding and Using Context. *Personal and Ubiquitous Comp. J.*, 5(1):4–7, 2001.

[3] T. Broens. Context-aware, Ontology based, Semantic Service Discovery. Master's thesis, University of Twente, Enschede, The Netherlands, 2004.

[4] C. Doulkeridis, N. Loutas, and M. Vazirgiannis. A System Architecture for Context-Aware Service Discovery. *Electr. Notes Theor. Comput. Sci.*, 146(1):101–116, 2006.

[5] A. Lubinski. Anfragen mobiler Benutzer. In *H. Höpfner, C. Türker, B. König-Ries: Mobile Datenbanken und Informationssysteme - Konzepte und Techniken*, pages 93–98. dpunkt.verlag, 2005.

[6] M. Klein, B. König-Ries, and M. Müssig. What is needed for semantic service descriptions? A proposal for suitable language constructs. *Int. J. of Web and Grid Services*, 1(3/4):328–364, 2005.

[7] S. Penz. SLP-based Service Management for Dynamic Adhoc Networks. In *6th International Middleware Conference, Grenoble, France, Workshop MPAC*, 28 2005.

[8] P.G. Raverdy, O. Riva, A. de La Chapelle, R. Chibout, and V. Issarny. Efficient Context-aware Service Discovery in Multi-Protocol Pervasive Environments (to app.). In *7th MDM, Nara, Japan*, 2006.

[9] S. Cuddy, M. Katchabaw, and H. Lutfiya. Context-Aware Service Selection Based on Dynamic and Static Service Attributes. In *IEEE WiMob, Montreal, Canada*, 2005.

[10] S. Marti and V. Krishnan. Carmen: A Dynamic Service Discovery Architecture. Technical Report HPL-2002-257, Hewlett Packard Laboratories, 23 2002.

[11] Samulowitz, M., Michahelles, F., and Linnhoff-Popien, C. Capeus: An architecture for context-aware selection and execution of services. In *DAIS 2001, Krakow, Poland*, pages 23–39, 2001.