

A System Architecture for Context-Aware Service Discovery

Christos Doulkeridis¹ Nikos Loutas² Michalis Vazirgiannis³

*Department of Informatics
Athens University of Economics and Business (AUEB)
Athens, Greece*

Abstract

Recent technological advances have enabled both the consumption and provision of mobile services (m-services) by small, portable, handheld devices. However, mobile devices still have restricted capabilities with respect to processing, storage space, energy consumption, stable connectivity, bandwidth availability. In order to address these shortcomings, a potential solution is context-awareness (by context we refer to the implicit information related both to the requesting user and service provider that can affect the usefulness of the returned results). Context plays the role of a filtering mechanism, allowing only transmission of relevant data and services back to the device, thus saving bandwidth and reducing processing costs. In this paper, we present an architecture for context-aware service discovery. We describe in detail the system implementation and we present the system evaluation as a tradeoff between a) the increase of the quality of service discovery when context-awareness is taken into account and b) the extra cost/burden imposed by context management.

Key words: Context-aware Service Directory, Web Services, Mobility, System Architecture

1 Introduction & Motivation

Recent technological advances in the areas of mobile and service-oriented computing have enabled novel scenarios of use that involve mobile devices (laptops, PDAs, cellular phones, etc) that can both consume and provide mobile services [21]. We adopt a data-centric and service-oriented approach: data is encapsulated as services, and access to data is enabled through service calls [16].

¹ Email:cdoulk@aueb.gr

² Email:n_loutas@yahoo.com

³ Email:mvazirg@aueb.gr

This approach hides the heterogenous nature of mobile data scattered in diverse devices and provides a standard, well-defined and globally accepted way for data access.

Any autonomous, portable, handheld device can be considered as a candidate for participating in our scenarios of use that involve innovative applications, either as a service requestor or as a service provider ⁴. However, handheld devices still have restricted capabilities that hinder the design of straightforward communication protocols and prevent the management of data or services in an easy way. Limitations of mobile devices involve restricted processing capabilities and storage space, increased power consumption, limited display capabilities, variable bandwidth availability, sudden disconnections.

Moreover, mobile users usually tend to prefer using mobile services: a) from nearby locations, b) that return fresh data ⁵, c) provided by trusted users, and d) returning results displayable in the requesting device. In other words, several parameters such as location, time, user identity and profile, device capabilities, that involve both the requesting user as well as the service provider, have great influence in the search for the most suitable service. This fact highlights the need for context management to support the efficient deployment of applications that strongly depend on mobility of users and data sources. Context plays the role of a filtering mechanism, allowing only transmission of relevant data and services back to the device, thus saving bandwidth and reducing processing costs.

According to traditional web service architectures, service discovery is based on the use of service directories. However, existing service directories specifications [20,8] do not address issues related with mobility of services nor with support of context-awareness in a straightforward and unambiguous manner. In previous work [4,5,6], we have presented our approach for managing a context-aware service directory, i.e. construction, search, update and merge algorithms. In this paper, we capitalize on this theoretical foundation and describe in detail the overall architecture and implementation of a system that supports context-aware service discovery by means of enhanced service directories. In particular, our contributions are:

- an architecture for enhanced service discovery based on the use of context-aware service directories
- a prototype system implementation that supports mobile service discovery based on various contextual parameters
- the evaluation of the system with respect to the tradeoff between the increase of the quality of service discovery and the overhead imposed by man-

⁴ A typical scenario of use involves tourists that carry mobile devices and walk near monuments and local attractions, sharing data and resources (documents, photos, videos) with each other

⁵ Freshness is related to the temporal dimension and implies that newly created shared data and resources (images, files, etc) are more popular than outdated ones

aging contextual information.

The rest of this paper is organized as follows: Section 2 reviews the related work, while Section 3 discusses the significant role of context for mobile services. In Section 4, the overall architecture of our context-aware service discovery system is presented in detail, Section 5 describes the system implementation and Section 6 evaluates the implemented system in terms of advantages versus overhead incurred by the use of contextual information. Finally, Section 7 concludes the paper and presents future research directions.

2 Related Work

Context-aware service discovery has been addressed lately by several research initiatives that have proposed enhanced discovery mechanisms. The CB-SeC framework [14] attempts to enable more sophisticated discovery and composition of services, by combining agent-oriented and context-aware computing. Its notion of context is similar to our approach, but the representation is different. Modelling the contextual information is achieved by means of a recursive function, whereas our approach is more data-centric and focuses on providing a context-based index for service discovery.

Lee *et al.* [11] also recognize the limitations of existing service discovery approaches and try to exploit context-awareness in this respect. Moreover, they also argue in favor of providing support for context-awareness by means of service registries. The authors introduce the notion of context attributes, as part of the service description, which allow dynamic evaluation of contextual parameters to enhance service discovery.

The WASP project [17] extends the functionality of UDDI by introducing UDDI+, an attempt to improve the existing service discovery mechanisms regarding semantic and contextual features. However, this approach is technology-specific, depending on previous work concerning UDDI [20] and DAML-S [15], whereas our approach is an external and independent model that enables context-based indexing within any service directory. While we both stress the importance of the context of a service and focus on improving the performance of service discovery based on contextual parameters, Pokraev *et al.* [17] provide support for exact and inexact service matching based on their types, capabilities and models, through the use of a context ontology. We expect to deal with inexact matching in our future work.

Current standardization efforts regarding service directories like UDDI [20] and ebXML [8] present certain limitations. For instance, in [10], the authors achieve service categorization based on their WSDL descriptions, which proves restrictive, when trying to address issues like mobility of services, semantic discovery and context-awareness. Therefore, research initiatives have tried either to enhance the functionality of service directories [1,10,18] or to identify new research challenges [9].

Maamar *et al* [12] present an approach for Web service composition based on the use of agents and context. Their context model (I/W/C-Contexts) is service-centric and comprises three types of context: a) I-context refers to a Web service instance context b) W-context is the Web service context that is defined by means of I-contexts and c) C-context is the context of the composite service and is defined by the respective W-contexts.

Constantinescu *et al* [3] also deal with service composition and discovery in their work. They allow partial matching of services and user-defined ranking functions for decreasing directory response times. As in our case, the importance of indexing within the service directory is recognized, however our focus is on context representation for services and on context-based service retrieval. Moreover our approach also focuses on mobility of services.

Distributed service discovery approaches (for instance JXTA) are also relevant to mobile service discovery (and consequently to our approach). Furthermore, distribution of service registries is considered in our work [4,7]. Compared to [13], our registries rely on a wired infrastructure, communicate and inter-operate directly, in order to support discovery of services hosted by mobile devices that communicate in a wireless way. In fact, we have used the JXTA framework in recent work [7], in order to study mobile service discovery in a distributed architecture of context-aware service directories.

3 Context for Mobile Services

Limitations of both mobile devices and current wireless communication technology impose careful resource management. Applications running on mobile devices cannot afford to waste power nor consume available bandwidth by transmission of irrelevant data, not to mention unnecessary retransmissions. Furthermore, screen size and, more generally, device capabilities impose restrictions on displayed data, for example presentation of high quality images is usually not supported. Returning long lists (i.e. results of queries posed at Google) to mobile users leads to user annoyance, besides excessive bandwidth consumption. Finally, since timely responses are required for successful mobile applications, it is often necessary to exploit user location in order to provide resources from nearby locations rather than distinct ones.

We define context in service discovery for mobile computing as *the implicit information related both to the requesting user and service provider that can affect the usefulness of the returned results* [5]. Two kinds of context are identified, namely *service context* and *user context*. Service context can be the location of the service, its version, the providers identity, the type of the returned results and possibly its cost of use. On the other hand, each user is characterized by a user context that defines her current situation. This includes several parameters like location, time, temporal constraints, as well as device capabilities and user preferences.

During service discovery user context is matched against service context

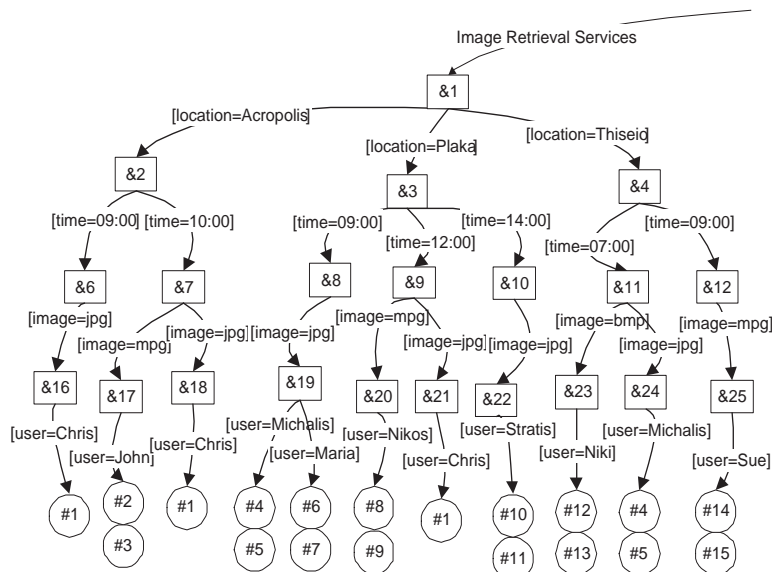


Fig. 1. Part of a context-aware service directory for image retrieval services

in order to retrieve relevant services with respect to context-awareness. Context is very important in mobile service discovery, since it can play the role of a filtering mechanism returning only the subset of retrieved services that conform to the user's current context.

In our approach, we focus on representation of context for web services and we choose a context representation formalism adopted from previous work on context-dependent representations [19]: context is represented by a set of dimensions that take one or more discrete values or range over a specific domain. Combining different dimensions of context that specify the conditions under which a service exists, results in a graph that represents a specific service category within the service directory. A set of such service categories form the service directory. Note that each service category may contain different contextual dimensions, based on the type of services it includes. Obviously, the combination of different dimensions within a service category results in more costly updates, but improves greatly searching for services based on contextual information.

The graph (see Figure 1) comprises two kinds of nodes (following the notation of [19]): rectangular nodes, called *multidimensional nodes*, represent multidimensional entities that present different facets under different contexts, whereas circular nodes, representing entities, are called *atomic* or *context nodes*, depending on whether they are leaves of the graph or not. Atomic nodes are able to hold some kind of data and in our approach they represent web services. In our approach, only *atomic* nodes exist, no *context* nodes. There are also two kinds of edges: *context* and *entity edges*. Context edges are represented by thick lines and they define the contexts (represented by labels called *explicit contexts*) under which the services hold. Entity edges are plain edges that represent relations between entities, and in this approach they are

used to define the service category.

In Figure 1, a service category within the service directory is depicted, namely image retrieval services. As an example, consider service with identifier: #1. This service is provided by tourist Chris, returns images of type jpg and is available at different locations at different times of the day. As the user walks, the service is provided near Acropolis at 9 and 10 a.m., while two hours later it is provided at Plaka (the oldest section of Athens and arguably the nicest neighborhood in central Athens). Given this representation, context-based service retrieval is performed efficiently by means of a breadth first search algorithm [4]. For example, consider another tourist who also visits Acropolis shortly after 10a.m. and wants to find pictures of Acropolis to store in her PDA. She enters the keywords:

$$Q_{usr} = (\text{photos}, \text{Acropolis})$$

to initiate a search for suitable services and at the same time a contextual query is formulated:

$$Q_{ctx} = (\text{location}=\text{Acropolis}, \text{time}=10:00, \text{image}=(\text{jpg}, \text{gif}))$$

that captures her location, the current time and the capabilities of her device. The evaluation of Q_{ctx} on the service directory focuses on the left subtree of the graph (for location Acropolis) and returns service with id: #1, which belongs to Chris, and ignores services with ids: #2 and #3 that return mpg files. As you see, even for a simple example, the precision of the search is increased and the results are refined in terms of the relevant context.

4 System Architecture

In order to realize context-aware service provision in a mobile environment, there is a need for introducing context within the core of the service discovery process. As already mentioned, we consider a context-aware service directory, i.e. a service directory enhanced with contextual descriptions, as the cornerstone of an architecture for mobile service discovery. In this section, we present the overall architecture for context-aware service discovery and discuss in detail its design.

4.1 Context Management

Our notion of context within a service directory resembles an index, like the ones used in traditional relational databases. Intuitively, parallels can be drawn between service retrieval based on contextual attributes and tuple selection based on a table's indexed column. In this sense, a context-based index enables service discovery within a service directory based on contextual attributes. This is an alternative and complementary approach to standard searching mechanisms provided by directory specifications, for example search by organization or by service category.

In Figure 2, the overall system architecture for context-aware service dis-

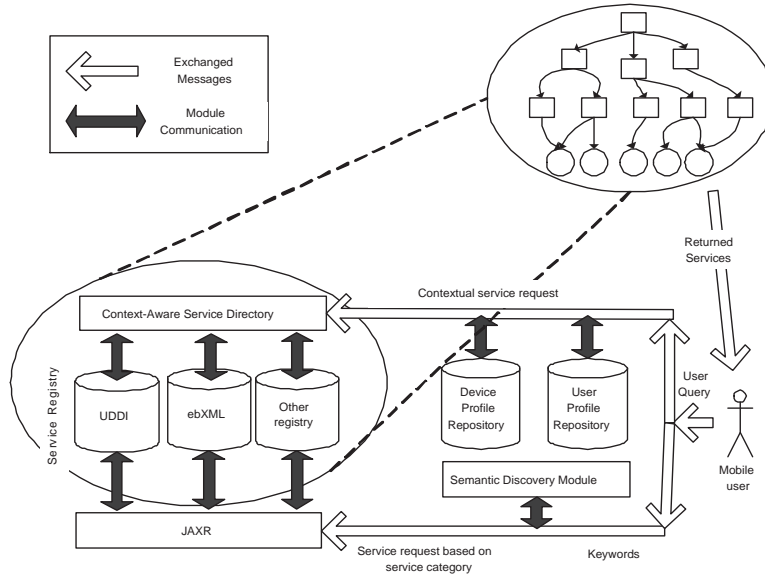


Fig. 2. Overall system architecture for context-aware service discovery

covery is depicted. In a typical web service architecture, a variety of service registries (UDDI, ebXML, other variant) is generally available for publishing web services. Access to this information is provided by means of a standard interface, namely JAXR. JAXR achieves to abstract the particular underlying service registry and allows transparent access to service descriptions. Similarly to JAXR, the context-aware service directory plays the role of an interface to all aforementioned registries - using functions provided by JAXR (e.g. `findServices`, that returns all services in the Registry Server that belong to a specified category given as parameter)- but additionally supports service retrieval based on a contextual query. However, a context-aware service directory requires to be updated, whenever a change of a service's context occurs. We have addressed such updates in [6].

As mentioned in Section 3, device capabilities and user preferences are important features that are incorporated in user context. In order to manage this kind of information a set of repositories is maintained, namely a device repository and a user profile repository.

- *User Profile Repository*: During user registration in the system, a user may submit his/her preferences to form the corresponding user profile. This can be updated later, based on user's behavior, for example using the invoked web services or the history of use. User preferences provide a personalization mechanism, which can be seen as part of the overall context, and enable service discovery in a way that matches best the explicit or implicit user requirements.
- *Device Profile Repository*: Since a user may have access to the system using more than one device, during each log on, the mobile device's profile is sent to the device repository. Notice that uploading the device profile

usually takes place at first log on, and when subsequent changes of the device characteristics will be detected, only the modified part of the profile will be sent to the repository, thus saving valuable bandwidth. Taking into account device profiles avoids services with device requirements that exceed the current device’s capabilities.

4.2 Service Discovery

In our application scenarios, mobile users issue service requests (or queries) for mobile or stationary web services, using handheld devices. Due to mobility of both requestors and providers, the surrounding environment is exceptionally dynamic and volatile. Service discovery plays an eminent role in enabling intercommunication of mobile users as well as data sharing. In the following, we describe the service discovery process in the context of our architecture.

Mobile users are rarely aware of available web services that fulfill their current needs. Therefore, enhanced service discovery mechanisms are required, besides traditional search by category or organization (the latter is actually useless in mobile environments). Our system allows users to issue keyword-based requests for services, so that a user can describe to the best of his knowledge his needs. This strongly resembles web search engine, where users search for documents based on a set of submitted keywords. A semantic discovery module is responsible for determining the service categories that are related to the user’s request, based on the use of a domain ontology. For example, if the keywords were ”University Master Programs”, then after the semantic analysis the service category would be Educational Services. However, semantic service discovery is out of the scope of this work, so the interested reader may refer to [21] for a more detailed view of our approach.

A separate query (henceforth represented by Q_{cxt}) capturing the user’s current context is attached to the user-defined keyword-based query Q_{usr} , in order to present context-dependent results to the user. Q_{usr} is semantically disambiguated and issued against the service registry through the JAXR interface (see Figure 2). This query finds all services that potentially fulfill the user’s needs. That is, all the Services belonging to the specified category, that are stored in the various registries, are retrieved and sent to the context-aware service directory. At the same time, Q_{cxt} is formulated using the device profile and the user profile repositories. Additional contextual information is contained in Q_{cxt} , such as the user’s location and the time of the request. The contextual query is sent to the context-aware service directory and filters the set of services. The remaining subset of services, namely those that match the user’s context, along with short descriptions is sent back to the user.

Summarizing, Q_{cxt} acts as a filter on Q_{usr} ’s results, thus, a significantly smaller set of services that both meet the user’s needs and his/her context is returned to the user.

5 Implementation Status

As described in Section 4, the system has a server-side and a client-side part. The server-side part implements the context-aware service directory, while the client-side part provides the user interface. In this section we will present an overall description of the system's implementation and of the tools that were used to implement the system. In the server-side part we used:

- *UDDI (Universal Description, Discovery and Integration)*: UDDI is an XML registry that provides standard mechanisms for businesses to describe and publish their services, discover other businesses that offer desired services, and integrate with them. UDDI provides two types of service discovery a) by organization and b) by category. We have used version 2 of the Universal Description, Discovery and Integration project implemented in the Registry Server's release of Java Web Services Developer Pack (JWSDP v.1.4). The Registry Server is a service registry where organizations and services are stored and implements both types of service discovery provided by UDDI.
- *JAXR (Java API for XML Registries)*: JAXR provides a standard Java API for performing registry operations over a diverse set of registries. It also provides a unified information model for describing registry contents. So regardless of the registry provider used, all programs deal with the same information model. JAXR provides also the Registry Browser Tool. This tool connects to a Registry Server and enables operations such as:
 - insert/delete services or organizations from the Registry Server
 - discover organizations in the Registry Server
- *Programming environment*: To develop the various modules of our system we used J2EE 1.4 and Apache Tomcat 1.4 for JWSDP (Java Web Services Developer Pack) as web and application server.

The systems client-side part consists of three basic modules:

- *The system administrator's interface*: This is a web interface, where the administrator can delete an organization and all its services or bulk insert organizations from a text file with a certain format (this option extends the Registry Servers insertion, where only one organization or service can be inserted at a time).
- *The client's (user) web interface*: Gives the user two options, either search by organization or by service category. The user also has to fill his contextual parameters manually. This is a simple prototype that we developed for querying the context-aware service directory and it is not used by users, since in our approach context is transparently managed without requiring user intervention. Both the administrator's and client's web interfaces are implemented using JSP and Servlet technology.
- *The user's mobile interface (m-interface)*: The system purpose is to improve

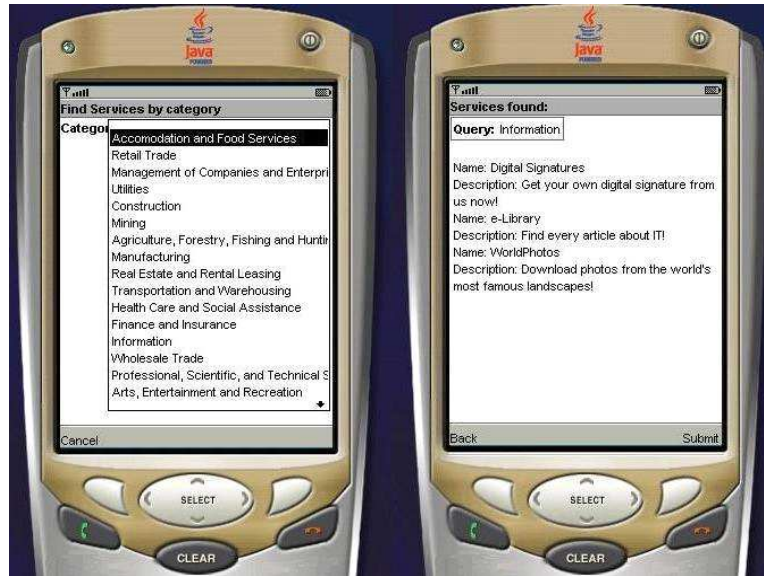


Fig. 3. A mobile device issuing service requests, as shown in the emulator. At the left part, the user searches by service category, while at the right part, a keyword-based query and the returned service descriptions are depicted.

the search results for mobile users; this is why we developed one more user interface, this time for a mobile device. This interface has been developed using the J2ME technology. The communication between the user's mobile device and our system is implemented using the IEEE 802.11 protocol.

In Figure 3, the mobile user's interface is depicted, as shown by the Java Wireless Toolkit emulator. A mobile user has two options for submitting requests for services: a) search by service category and b) keyword-based search. In the former case, a set of available service categories is presented to the user, who selects the one closer to his/her current needs. However, sometimes users prefer submitting keywords for discovering resources (a typical example is Web search) and this describes the latter case. The left picture shows how a set of available service categories is presented to the user, who then chooses one of them, and at a further step all services in the selected category that match the user's current context are retrieved. The right picture shows the results of a keyword-based search request, again based on the user's context. This comprises the name and a short description for each retrieved service. In this example three services seem to match the user's contextual request.

In our system implementation we have adopted a useful and extendable representation for device profiles, namely the CC/PP specification [2], so as to manipulate device characteristics. A CC/PP profile is a description of device capabilities and user preferences that can be used to guide the adaptation of content presented to that device. We use device profiles (a subset of context) mainly as a filtering mechanism that guides the search, in order to return only services that are within the device capabilities, with respect to

processing, display, etc. We have implemented a CC/PP repository on top of SQLServer2000, through the construction of stored procedures that enable storing CC/PP files (which are XML-based) to a relational database, as well as subsequent querying based on the device identifier.

Regarding user profiles, we adopt a simple model based on hierarchical representation of user preferences. The user can submit his profile at registration time and also update it later. This enables only basic personalized service discovery, however we intend to extend this module in the future by integrating more sophisticated user profiles that can express the user preferences in a better way.

6 System Evaluation

Evaluation of the proposed system is a complex procedure due to various characteristics that affect the overall system performance. We focus on what we think is a representative subset of indices to support the feasibility and the applicability of our approach.

We run all our experiments on a Pentium IV (2,8GHz) equipped with 512 MB RAM, which hosted the registry server and the context-aware service directory (CASD). We used the 1.4.2 version of Java and Tomcat 1.4 for JWSDP.

Our experimental setup consists of two evaluation scenarios. We created 250 (scenario 1) and 1000 (scenario 2) web service descriptions and inserted them into the registry server. The first three columns of Table 1 show the service distribution in 10 service categories. The latter three columns show the query selectivity for 10 random contextual queries that we used in our experiments (the same queries are used in Figure 4). Note that the total number of different contexts is 100 in both scenarios (in other words there exist 100 different paths in the CASD, or 3 contextual dimensions, each having 4 or 5 distinct values). For instance, category 1 is *Accommodation and Food Services* with 25 and 105 services in each scenario, and the contextual query 1 is:

[device=palmtop,os=winCE,location=France]

which matches a total of 2 and 4 services (that belong to this category) respectively.

Regarding search quality, it is rather self-evident that using contextual information improves the results of searches. Especially the precision of search is increased, since the returned results do not contain services that are considered useless, as they do not match the user's context. Precision is defined as the proportion of retrieved services that are relevant. We consider as relevant those services that conform both semantically and contextually to the user's request. In other words, relevant services are those that belong to the appropriate service category and at the same time return results that can be processed by the mobile device; these are the services that are really useful to the requesting user. Context-aware service discovery achieves to reduce

	Scenario 1	Scenario 2		Scenario 1	Scenario 2
Total services	250	1000	Contexts	100	100
Category 1	25	105	Query 1	2	4
Category 2	20	100	Query 2	4	6
Category 3	25	100	Query 3	1	3
Category 4	20	100	Query 4	3	7
Category 5	20	70	Query 5	1	2
Category 6	25	100	Query 6	3	4
Category 7	20	100	Query 7	4	6
Category 8	25	125	Query 8	4	10
Category 9	20	100	Query 9	1	2
Category 10	20	105	Query 10	2	5

Table 1

Experimental setup parameters: The first 3 columns show the service distribution in 10 service categories within the service directory, while the last 3 columns show the query selectivity for each of the 10 queries used in our two evaluation scenarios.

the number of retrieved services, by returning only the subset of potentially relevant services that conforms to the context of the request. In the example of scenario 1, a typical search for *Accommodation and Food Services* would return all 25 services in this service category, while the context-based search would only return the 2 services that actually conform to the context of the request.

Then, we measure the extra burden imposed by context management, in terms of processing times for serving requests for services. We also calculate the associated cost of transmitting service descriptions with and without context management.

Our initial evaluation results show that the use of context within the service directory increases processing time during searching only by 14 % (scenario 1) and 2,5 % (scenario 2) on average (see Figure 4) and this improves when the number of services increases. It is clear that CASD processing times are small (less than half a second) and increase slightly with the number of services. Of course processing times of registry server and CASD are not directly comparable for several reasons, i.e. the number of retrieved services is not equal, however the setup simulates a realistic situation where only a small subset of services within a particular service category is expected to match the context of the query.

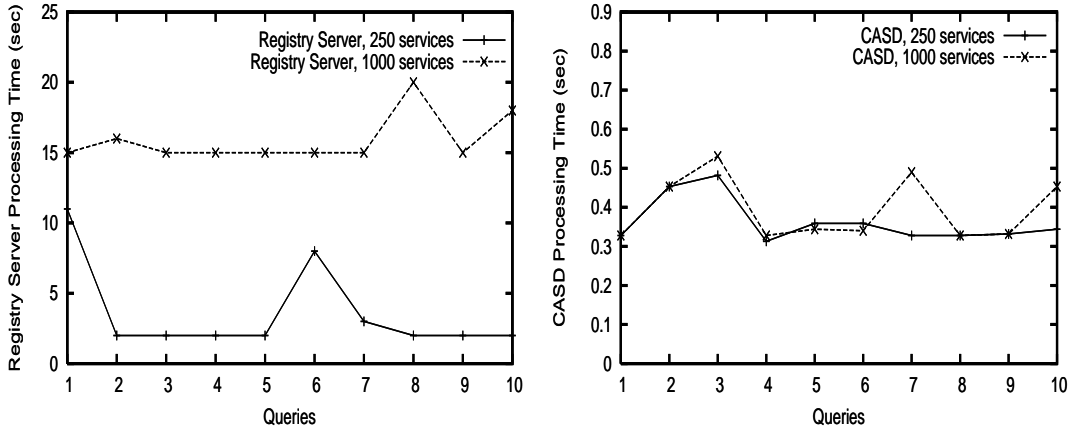


Fig. 4. Processing time for service registry (left) and CASD (right) for the 10 distinct queries of Table 1.

Given an average size of retrieved service description: SD_{avg} bytes, the amount of transmitted data is SD_{avg} multiplied by the number of matched services. Searching by service category without taking context into account results in all services in the category, while context filters out those services that do not conform to the user’s current situation. The average amount of transmitted bytes is reduced by a factor of 8,8 in scenario 1 and 20,5 in scenario 2, due to the increased query selectivity when context is taken into account.

Concluding, the extra cost paid by managing context is considered minimal, when compared to the achieved increase in precision and the reduced number of bytes transmitted to the requestor. Updating and searching the CASD has a low cost in principal [4,5]. Even when the number of contextual dimensions increases, these operations show good performance. Actually, the only drawback of our approach is the CASD construction cost, especially for large values of contextual dimensions and associated values, but a) this cost is paid only once and b) the number of dimensions refers to those within each service category and not all dimensions over all categories, so it is normally bounded. After all, service directories usually receive many search requests and relatively few updates, and this approach deals effectively with search requests.

7 Conclusions & Future Work

This paper proposes a system architecture for service discovery, based on a novel context-aware service directory. We motivated the use of context for service discovery (esp. when mobility of both consumer and provider is involved), we presented the implementation of our system and we described the experimental results of two evaluation scenarios. The results show that our approach reduces significantly the amount of transmitted data at the cost

of a relatively small processing overhead, while at the same time the quality of the search is improved.

As for future research directions, we intend to focus mainly in extending the context-aware service directory by means of contextual ontologies, in order to allow serving semantically enriched contextual requests for services, without the restriction of predefined contextual types or values.

References

- [1] Akkiraju, R., R. Goodwin, P. Doshi and S. Roeder, *A Method for Semantically Enhancing the Service Discovery Capabilities of UDDI*, in: *Proceedings of IJCAI-2003 Workshop on Information Integration on the Web (IIWeb-03)*, 2003.
- [2] *Composite Capabilities/Preference Profiles (CC/PP)*, <http://www.w3.org/Mobile/CCPP/>.
- [3] Constantinescu, I., W. Binder and B. Faltings, *An Extensible Directory Enabling Efficient Semantic Web Service Integration*, in: *International Semantic Web Conference*, 2004, pp. 605–619.
- [4] Doulkeridis, C., E. Valavanis and M. Vazirgiannis, *Towards a Context-Aware Service Directory*, in: *Proceedings of the 4th VLDB Workshop on Technologies on E-Services (TES'03)*, 2003, pp. 54–65.
- [5] Doulkeridis, C. and M. Vazirgiannis, *Querying and Updating a Context-Aware Service Directory in Mobile Environments*, in: *Proceedings of the 2004 IEEE/WIC/ACM Web Intelligence Conference (WI'04)*, 2004, pp. 562–565.
- [6] Doulkeridis, C. and M. Vazirgiannis, *Updating a Context-Aware Service Directory for M-services*, in: *3rd Hellenic Data Management Symposium (HDMS'04)*, 2004.
- [7] Doulkeridis, C., V. Zafeiris and M. Vazirgiannis, *The Role of Caching and Context-Awareness in P2P Service Discovery*, to appear in the 6th International Conference on Mobile Data Management (MDM'05), 2005.
- [8] *ebXML*, <http://www.ebxml.org>.
- [9] Hofreiter, B., C. Huemer and W. Klas, *ebXML Status, Research Issues, and Obstacles*, in: *12th International Workshop on Research Issues on Data Engineering (RIDE'02)*, 2002.
- [10] Jeckle, M. and B. Zengler, *Active UDDI - an Extension to UDDI for Dynamic and Fault-tolerant Service Invocation*, in: *International Workshop Web Services - Research, Standardization, and Deployment (WS-RSD'02)*, 2002.
- [11] Lee, C. and S. Helal, *Context Attributes: An Approach to Enable Context-awareness for Service Discovery*, in: *Proceedings of the 2003 Symposium on Applications and the Internet (SAINT'03)*, 2003.

- [12] Maamar, Z., S. K. Mostefaoui and H. Yahyaoui, *Toward an Agent-Based and Context-Oriented Approach for Web Services Composition*, IEEE Transactions on Knowledge and Data Engineering **17** (2005), pp. 686–697.
- [13] Maamar, Z., H. Yahyaoui, Q. H. Mahmoud and F. Akhter, *Dynamic Management of UDDI Registries in a Wireless Environment of Web Services*, in: *Proceedings of the 2nd International Conference on Wired/Wireless Internet Communications (WWIC'04)*, 2004, pp. 284–294.
- [14] Mostefaoui, S. K. and B. Hirsbrunner, *Towards a Context Based Service Composition Framework*, in: *Proceedings of the 1st International Conference on Web Services (ICWS'03)*, 2003.
- [15] Paolucci, M., T. Kawamura, T. Payne and K. Sycara, *Semantic Matching of Web Services Capabilities*, in: *Proceedings of the 1st International Semantic Web Conference on The Semantic Web*, 2002.
- [16] Pitoura, E., S. Abiteboul, D. Pfoser, G. Samaras and M. Vazirgiannis, *DBGlobe: a Service-oriented P2P System for Global Computing*, SIGMOD Record **32** (2003), pp. 77–82.
- [17] Pokraev, S., J. Koolwaaij and M. Wibbels, *Extending UDDI with Context-aware Features Based on Semantic Service Descriptions*, in: *Proceedings of the 1st International Conference on Web Services (ICWS'03)*, 2003.
- [18] ShaikhAli, A., O. Rana, R. Al-Ali and D. Walker, *UDDIe: An Extended Registry for Web Services*, in: *2003 Symposium on Applications and the Internet Workshops (SAINT-w03)*, 2003.
- [19] Stavrakas, Y. and M. Gergatsoulis, *Multidimensional Semistructured Data: Representing Context-dependent Information on the Web*, in: *Proceedings of the 14th International Conference on Advanced Information Systems Engineering (CAISE'02)*, 2002.
- [20] *The UDDI specification version 3.0*, <http://www.uddi.org>.
- [21] Valavanis, E., C. Ververidis, M. Vazirgiannis, G. C. Polyzos and K. Norvag, *Mobishare: Sharing Context-dependent Data and Services from Mobile Sources*, in: *Proceedings of IEEE/WIC International Conference on Web Intelligence (WI'03)*, 2003.