Journal of Mobile Multimedia, Vol. 4, No. 3&4 (2008) 210-226


# SALSA – A FRAMEWORK FOR CONTEXT-SENSITIVE SERVICE DISCOVERY IN MOBILE COMMERCE APPLICATIONS

COLIN ATKINSON, PHILIPP BOSTAN, THOMAS BUTTER

*University of Mannheim, Germany*
*{atkinson, bostan, butter}@uni-mannheim.de*

Received April 20, 2008
Revised July 7, 2008

In an effort to increase the average revenue per user, over the past few years mobile network operators have introduced many new services for mobile devices connected to the internet. However, none of them has proven to be the desired killer application, and many have failed completely. Most of the attention to date has focused on location-aware applications, but the increasingly powerful miniaturized sensors that will probably be embedded in future mobile devices offer many more types of context information than just location. When these additional types of context information are used as well, applications and services can be made much more context-sensitive, and service providers can offer more value to end users. This, in turn, will lead to higher acceptance and adoption of Mobile Commerce services. However, creating context-sensitive applications and services for mobile devices is not a trivial task and developers' experience with these new kinds of technologies is low. Thus, the development of such services and applications, or the migration of conventional services to context-sensitive services, would be significantly boosted by framework support for client and service development. In this paper we introduce a framework for context-sensitive service discovery that offers systematic development support and reduces the effort involved in migrating from pure information services to context-sensitive services.

*Key words*: Mobile Commerce, Context-Sensitive Service Discovery, Service Frameworks, Service-Oriented Architectures (SOA)

## 1 Introduction

Although services for mobile devices like navigation, mobile TV and a few location-based services have gained a lot of attention in the last few years, applications that exploit more contextual information in service delivery are not yet supported or widely used. There are various historical reasons for this including the restricted capabilities of mobile devices, the low level of support for miniaturized and cheap sensor technology to access context information, cumbersome human-computer interaction via small keypads, low-speed mobile networks and high costs for data transmission. However, the growing power of mobile devices, networks and sensors has significantly changed the situation and new service delivery models such as data "flat rates" and wireless hotspots now allow mobile services to be accessed anywhere and anytime with less restrictions. Moreover, the introduction of software platforms like Java ME, the .NET compact framework, Symbian OS and the growing support for programmatically accessing features of mobile devices (including hardware access), now make it possible for any software developer to implement impressive (commercial and even context-sensitive) applications and services that are no longer dependent on device manufacturers

or mobile network providers and can be easily distributed to mobile users. In combination, these developments provide an excellent opportunity to build more complex and convincing applications and services.

Services and applications for mobile devices are different to traditional applications used on desktop computers. The main distinguishing characteristics of mobile applications are mobility, heterogeneity, resource limitations, personalization, and different usability requirements and user patterns. Since the location of a user has a major bearing on the type of services he or she is likely to be interested in, mobility is particularly important and requires a new service interaction paradigm to be developed which can deliver added value to the mobile user. In particular, services that use implicit context information, included as part of a service request without the knowledge of the user, can obviously deliver significant advantages compared to traditional services used from desktop computers. Especially in changing mobile environments where people are in transit, the long-winded definition of large, explicit parameter sets to find suitable services of interest is cumbersome. Thus the kind of approaches used to drive search engines from the desktop based on explicit keywords does not work well for mobile clients. With context-sensitive service discovery, human-computer interaction can be minimized, allowing users to concentrate on describing their key requirements.

Although the basic technological ingredients for building context-sensitive mobile applications are now available to mainstream developers, using them effectively together is still a major challenge. Thus, while it is possible in certain limited domains like navigation to enhance existing applications to exploit automatically sensed context information, for more general context-sensitive services or applications a more comprehensive framework is needed which offers the basic capabilities in a flexible and easy-to-use way. Above all, such a framework needs to seamlessly integrate technologies to manage the collection of context information using sensor technologies, the processing of context into higher order information and the matching of suitable services against given requirements and context information.

Enhancing the ability of mobile application developers to introduce context-sensitive (Mobile Commerce) services will not only deliver benefit to mobile users who will obtain easy-to-use, personalized, context-sensitive services, but also to network and service providers who will be able to offer new services that they can easily differentiate from “ordinary” services and derive new revenue streams based on different business models. Also, a whole new role is generated in the value chain – namely, the role of context providers. These can derive revenue by delivering basic context data or by processing raw context data to obtain higher-level context information. Hardware manufacturers can also participate in the new value chain since the demand for sensor technology will increase.

In the SALSA (Software Architectures for Location-Specific Transactions in Mobile Commerce) project at the University of Mannheim we have addressed this challenge by developing a generic architecture for context-sensitive service discovery with a focus on Mobile Commerce services. The goal of the framework is to support the creation of context-sensitive services and client applications able to deal with context in the process of service discovery, using mainstream, widely available development technologies. To encourage wide adoption, the requirements and concerns of mobile users, elucidated via questionnaires and surveys, were explicitly taken into account during the design of the framework.

In this paper we introduce our framework and its underlying technologies. In section 2 we consider some current practices and research on frameworks and application platforms for context-sensitive service discovery. Section 3 then describes the service types that are supported for our context-sensitive service discovery approach and their description schema. In section 4 we refer to the basic context processing steps, while section 5 focuses on context matching. In section 6 we introduce the basic SALSA client and server framework that has been developed to provide generic support for context-sensitive service discovery. Following that, in section 7 our prototype and a typical application scenario for Mobile Commerce are introduced before we finally give some concluding remarks and an outlook on further work in section 8.

## 2 Related Work

Since Schilit et al. initiated research on context-aware computing in 1993 starting with their PARCTab project [1], various researchers have focused on the subject of context-sensitive (context-aware) service discovery and service frameworks. However, as described in [12], in the early days many research projects focused exclusively on context-sensitive service discovery related to hardware, like near-by printers or other low-level services. In this section we give a brief overview of some related work with regard to service discovery approaches using contextual information and frameworks for context-aware services. One of the first research efforts on context processing was performed in connection with the Context Toolkit [13] which offers a framework for context processing at different levels like acquiring and handling context. It also introduces framework support for aggregation and inference.

The CB-Sec project [9] developed an architecture that focuses on the discovery of Web Services using contextual information. To this end, a service description schema was developed that includes a set of constraints, requirements and context functions that are used by a brokering agent to evaluate, filter and rank services that best fit conditions represented by a specific context. Context is collected by the context gatherer that receives contextual information from software and hardware sensors and is stored over time in the context data base that is available to the whole system and applications.

In [18], Kuck and Reichartz present an approach for context-sensitive discovery of Web Services based on matching the current context of the user with enhanced service descriptions stored in a UDDI repository. Their service description includes inferred information about syntactical and textual contents of a WSDL description as well as feedback information, e.g. the context at the time of service recommendation.

In the COSS approach [10], ontologies are used for the description of context and services. Service advertisement and service requests are represented as documents, while service requests include attributes defined by the user. An attribute like “nearby” is enhanced by rules that are evaluated during the matching process, for example to determine whether the user’s location is within a certain distance to the service’s location. In the COSS approach, context providers deliver the necessary contextual information needed for the matching process. Related to the WASP project a service platform for mobile context-aware applications was presented in [11] and [19]. The service platform is built on the principles of service-oriented architectures and includes several roles, like service and context providers. It offers a framework for the rapid creation and deployment of context-aware applications which is based on architectural support for context-aware applications providing context models and a

context interpreter for context processing. Furthermore it contains a registry that supports event-driven reactions to context changes and storage of data entities. A so called monitor that is responsible for interpreting and managing application subscriptions is also realized using the WASP Subscription Language (WSL). Subscriptions can be changed dynamically and are triggers for context-sensitive behaviour of applications and services.

The framework described in this paper focuses on accelerating the development of client applications and services for context-sensitive service discovery of information or real-world (business) services using the current context of the mobile user in a simple and practical way. This is realized by a client framework that supports the rapid creation of mobile client applications for context-sensitive service discovery by eased integration of context sensors, support for the management of context, support for security and reconfiguration of the mobile client application with regard to pluggable components and an adaptive user interface. The server framework supports the development of context-sensitive service discovery services including pre-processing of context and matching context against descriptions of services. This is emphasized by a rule-based transformation approach that offers dynamic service discovery of services related to the current context and is easy to configure for the provider of a context-sensitive service.

## 3 Services in SALSA

One of the goals of the SALSA framework is to support context-sensitive service discovery in the most generic way, independent of the type of service to be discovered. The basic ingredient therefore is a well-defined description of services. Although people use many different types of services on a day-to-day basis, they rarely think of them explicitly as services. In general, one can identify two main kinds of services: electronic and non-electronic services. An electronic service is a service that exclusively delivers its value by electronic means using IT technology (e.g. internet-based applications or Web Services, which are a special type of electronic services that make use of standard Web Service technologies). A non-electronic service, on the other hand, is an activity or service offered by an organization to improve the state of the user in some way (e.g. a cinema, a shop, a tourist site, public transportation and many more). We can also identify hybrid services which are composed of both electronic and non-electronic services (e.g. the Amazon selling service, which is offered electronically for ordering but its full value includes physical delivery by a transportation company). Due to the complexity of hybrid services, in the following we will only consider electronic and non-electronic services.

For the description of services in the SALSA project, a basic XML schema has been developed which separates the descriptions into a core part and an extension that adds domain-specific information if necessary. The core service description schema is based on OWL-S [2] and separates the service description into the three main parts, **ServiceProfile**, **ServiceProperties** and **ServiceGrounding** as depicted in Figure 1. As also shown in this figure, in SALSA there are four different types of services supported in the ServiceGrounding node - SALSAService, Web Service, Website and BusinessService. The first three are electronic services while the last one is a non-electronic service. A SALSAService is realized as a service with a Web Service interface together with embedded adaptable components which can be dynamically downloaded to mobile clients for local execution. A Web Service is a service executable on a remote server which delivers some kind of value

to the requestor. A Website is a service that delivers information and is used by a client web browser. Furthermore, a BusinessService in SALSA is a real-world (business) service that offers some physical or visible value to the user.

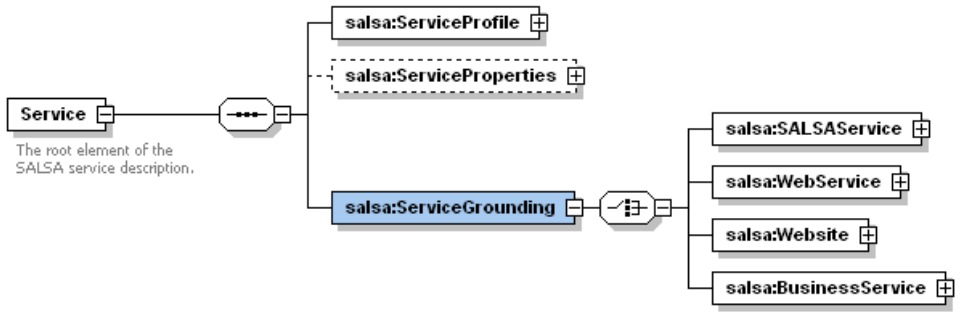


Figure 1 Service Core Description

The *ServiceProfile* of our service description schema describes general information about the service and its service provider. A general text description of the service and the categorization of the service, e.g. with NAICS [15] or a self-defined categorization schema can be used by the service provider to deliver general information about the service. The category description consists of a category name, the taxonomy it is contained in, a value and a code. The information about the service provider consists of a representative contact person and an informative part that contains several contact persons as well as a potential web address of the service provider or the service homepage. The description of a contact person includes his or her name, position, address, phone number and e-mail address. Furthermore, the *ServiceProfile* contains a subpart which represents the domain-specific business profile. Since a generic schema for all different types of possible services cannot be defined upfront, domain-specific extensions, e.g. the description of tourist sites for the domain of tourist services could be included in the service description based on its own extended schema. The domain-specific business profile and its attributes are mainly important for our context-sensitive service discovery approach as we will consider in section 5 when context matching is introduced.

The *ServiceProperties* describe several aspects of a service that are related to capabilities and restrictions. This includes the spatial and temporal availability as well as aspects like payment and security. This section could possibly be extended with further aspects of interest like quality, rankings and many more. The spatial and temporal availability can be assigned to both non-electronic and electronic services. While the value of including spatial and temporal availability seems to be obvious for non-electronic services, it is not so obvious for electronic services. For a non-electronic service the temporal availability clearly relates to its opening hours, e.g. a tourist site's or a shop's opening hours. For the support of context-sensitive service discovery the temporal availability is matched against the time a search request is issued. Similarly a non-electronic service is able to describe its delivery area. For an electronic service the description of the temporal and spatial availability can also be very useful in context-sensitive service discovery. For example, the temporal availability can be used to prohibit the delivery of an electronic service at times when it is not useful, e.g. a shopping or tourist guide is not useful at night when shops and tourist sites are closed, while the spatial availability can also be

used when a service covers only a certain spatial area where service delivery is meaningful. The delivery of an electronic tourist guide for New York to a mobile user in Paris who issues a request for suitable services is useless for example.

The last section of the service description schema is represented by the *ServiceGrounding* that includes basic information about how a service can be accessed. The type of information that is included is dependent on the type of service. While a non-electronic service requires the description of its physical location in the form of geographical coordinates, an electronic service requires information about access data in the form of a web or service endpoint address. In the ServiceGrounding the various types of services introduced in this section are reflected. The SALSAService defines its WSDL grounding as well as descriptions and addresses for its downloadable components. A Web Service defines its WSDL Grounding, a website defines its address and finally a BusinessService defines its physical and geographical location.

## 4 Context enabling technologies

In this section we describe the context processing mechanisms supported in our framework for context-sensitive service discovery. To this end, we first introduce a definition of the notion of context and explain how we represent it. We then consider the various supported context processing approaches including the retrieval of context information from various context sensors, its post-processing and the matching of sets of contextual information against potential service descriptions.

### *4.1 Definition of Context and Context Representation in SALSA*

Perhaps the most widely used definition of context is that of Dey et al. [1]:

*"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 applications themselves."*

This definition is easily applicable when new context-sensitive systems are built. In the SALSA project we use this definition, but with one key enhancement - namely that context is delivered implicitly, and is not explicitly delivered by the user in a service request.

In an approach for context-sensitive service discovery which is based on the principles of service-oriented architectures, several participants are involved in the discovery process, service providers, service brokers and mobile clients as service requestors. Since all of these participants need a common understanding of context for processing and evaluation a common representation of context is needed. Some context-aware systems introduced in past research projects, described in [12], apply a pre-defined, fixed model for a set of context types. Furthermore they propose complex models, compared in [14], which are not only difficult to understand but also difficult to apply in practice. Therefore in our approach we have chosen a simple representation based on the markup scheme language XML. Data types for representing context elements are fixed in an XML schema while new context types can be added arbitrarily based on data types similar to those used in common programming languages. For the unique identification of a context type, a namespace is added to a context element which is used within evaluation steps in context processing activities like context matching. In principle, the basic defined context types can be combined to complex context types and new context data types can be

defined as long as type-dependant matching routines for context matching, which is described in section 5, are provided for those defined types to the framework manually.

### *4.2 Context Processing in SALSA*

The processing and evaluation of context in the SALSA framework is performed both on mobile clients and on servers on which services are hosted. Since network connections are expensive and mobile clients have restricted resources compared to servers, the client framework in the SALSA framework supports only a subset of the context processing capabilities provided in the server framework. In general the capabilities of the overall framework can be summarized as sensing and acquiring context, enhancing context and finally matching context against service descriptions. The first level in context processing involves the collection of the context information from several sensors which are mostly located at the mobile client, e.g. a built-in GPS sensor. Sensors can also be integrated into the services delivered by service providers, e.g. an outdoor temperature sensor that is used by a local tourist guide to estimate weather conditions. The second level of context processing has the goal of enhancing the context information delivered by the previous level with more expressiveness and to collect additional context information that can be derived from first-level context information. This can be achieved using first Context Provisioning Services (CPS) and then aggregation and inference, which will be explained in the following sections. The result of context processing is a set of contextual information which is used for matching service descriptions.

#### *4.2.1 Context Sensors in SALSA*

In the SALSA project, several context sensors have been implemented for prototype applications. The main focus of sensor technology has been the development of enhanced, high-precision positioning technologies that also use a sensor fusion approach [7]. Therefore, enhanced algorithms were developed for both indoors and outdoors, using GPS, Bluetooth and wireless LAN technology and including an integrated digital compass that can be used to determine a mobile device's alignment. Estimations about current speed of movement can be made using data collected over a period of time. Other sensors that have been implemented in the SALSA project for usage in client applications are related to device information, general and personal information that can be accessed via APIs programmatically or profiles specified by the mobile user.

#### *4.2.2 Context Provisioning Services in SALSA*

Context Provisioning Services (CPS) are used in the SALSA framework to deliver additional contextual information and thus add value to the context matching process. Once context information has been obtained from sensors, the mobile client framework first searches for locally registered CPSs that can provide context enhancement data before service or search requests are issued. Since mobile network communication is expensive, in the normal case CPSs are used and accessed during the server-side processing of context. A CPS can potentially be a service provided by a so called context provider and generally requires some kind of context input to access and provide additional context information. For example, a weather CPS needs the current location to return information about current weather conditions. Therefore a CPS needs to describe its required input and provided output according to the defined context representation schema and registers its service in a special registry.

This CPS registry is used in the process of context resolution to build a chain of services that can provide as much additional context information as possible, as will be explained in the following section.

### *4.2.3 Context Enhancement*

As mentioned before, one of the goals of context processing in our framework is to enhance the sensor-delivered context data as much as possible for higher precision context matching. To this end we used the previously introduced CPSs as well as aggregation and inference technology.

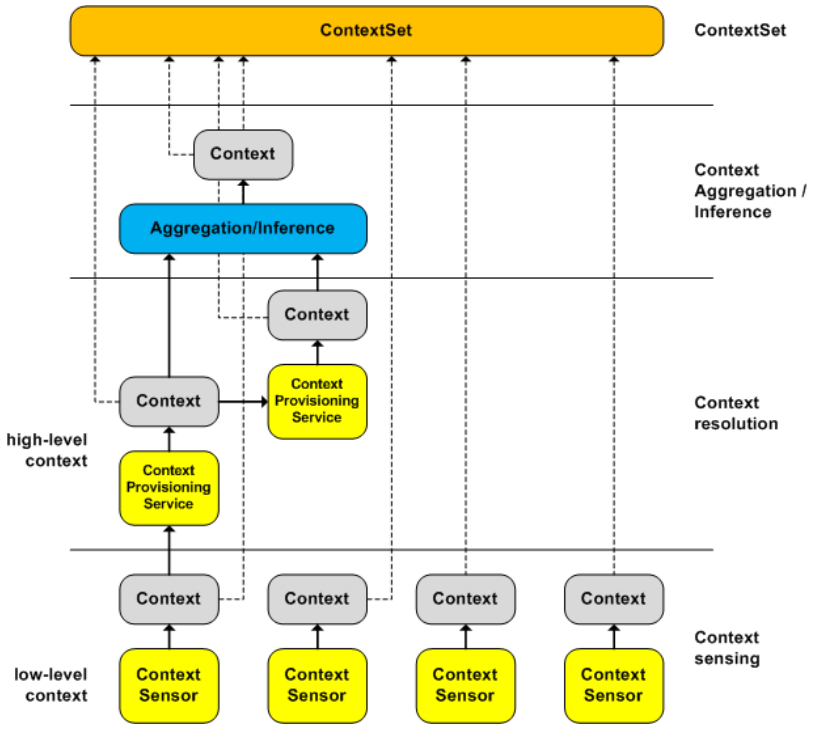


Figure 2 Context Processing Layers

In Figure 2 the different layers of context processing in the SALSA framework are shown. In the first layer, context is sensed and raw sensor data is converted into a representation of a SALSA context type. In the second layer, context resolution is applied using the CPSs to add higher-level context. In this step the chaining process touched upon in the previous subsection is used to derive as much new contextual information as possible. As an example, we consider a weather CPS that requires the current location in the form of a city name as input to deliver detailed weather information as output. At the beginning of the context resolution process the system might only possess the user's current location in the form of GPS coordinates delivered by the mobile client. One CPS first resolves these GPS coordinates into a city name which can then be passed to a weather CPS to obtain the weather data. Therefore, in the context resolution process we apply an approach for chaining the services in an

optimal way using the CPS registry. In the third layer, high-level context information is generated using aggregation and inference. Aggregation and inference are based on rules that indicate when several combined context types lead to new context types under certain assumptions and conditions. As also shown in Figure 2, single context types need not necessarily be processed through all layers and can be put directly into the final context set that is used for context matching as explained in the following section.

## 5 Matching Context and Service Descriptions

In context-sensitive service discovery the most critical step is the matching of a set of context information with potential services represented by service descriptions. Once the context set has been completely resolved and processed to an optimal extent, another step of pre-processing is needed before the matching mechanism can be applied. Potential service descriptions are transformed into a contextual representation that can be matched against a given context set on the server or even locally on the mobile client. In this section we describe this approach, first explaining the basic idea, the transformation approach and finally the context matching mechanism itself.

### *5.1 Basic idea of service transformation and context matching*

To provide a basis for matching the context information that has been gathered in the various steps of context processing against service descriptions, the approach needs to be extended by pre-processing the service descriptions. The main idea is based on the transformation of the service descriptions into contextual representations that contain the optimal context set for the service under consideration. The transformation is driven by rules based on certain facts in service descriptions and their corresponding effects which specify the transformation into contextual representations. The transformed result set is then used for context matching against the client-delivered context.

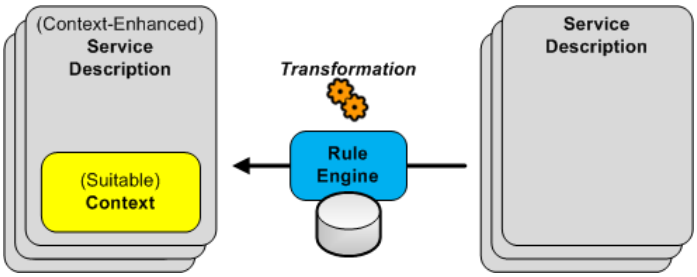


Figure 3 Rule-based Transformation of Service Descriptions

Figure 3 shows the basic principle of our approach where the result of the transformation process for a service description is a context-enhanced service description that contains an embedded optimal context set for the considered service. The main reasons for choosing such an approach stemmed from studies of user requirements [8] and security challenges [6] for context-sensitive services that were performed as part of the SALSA project. The studies revealed that one of the primary reasons why location-based and context-sensitive services for mobile users have not yet been widely adopted is the concern for privacy. Many mobile users expressed fear about misuse of their private data and most of

them were not willing to submit all of the information about their current context to unknown services. Our approach makes it possible to use the transformed service description for context matching locally on the mobile client based on the set of context information that is private and thus only locally available [3]. Different context attributes collected locally by the mobile client's context manager can be declared by the user as public, private or blurred. While public attributes are submitted to service brokers or providers when a search request is issued, private context attributes always remain local and are used only for local context matching to filter and personalize search result sets independently of remote services. To provide a better balance between keeping information private and supporting context-sensitive service discovery with acceptable precision, we introduced another type of context - namely blurred context, which artificially degrades the value of the context attribute (e.g. the location of the mobile user). Search requests using blurred context attributes still deliver a valuable result while maintaining the privacy of the user.

This practical approach has another advantage – it allows rules to be added, changed or removed from the inference engine's rule base dynamically. Furthermore, the fact that rules can be defined in various ways eases their description by service providers without the need to change the implementation of services programmatically.

### *5.2 Rule-based Transformation of Service Descriptions*

As previously explained, the transformation of service descriptions is a pre-processing step for the context matching process realized by a rule engine. A potential set of service descriptions is used to detect the occurrence of certain facts and to apply the corresponding effects to create a suitable contextual representation. For each fact to be transformed, a single rule is defined. Before discussing this approach in more detail we consider an example that illustrates this process. Using the scenario of a context-sensitive tourist guide, we define some reasonable rules for the transformation process. As introduced in section 3, the description of a service is in general divided into the core part and the domain-specific extension. We now consider some example facts appearing in the domain-specific extension of the description of tourist sites and how these are mapped into a contextual representation for context matching. The description of a tourist site is likely to indicate whether it is an indoor or outdoor attraction. The service provider of the tourist guide probably does not want to propose a tour of outdoor tourist sites to the mobile user when the weather conditions are very bad and there are multiple indoor tourist sites available at the same time and place. Therefore, the fact describing whether a tourist attraction is indoor or outdoor is transformed into context attributes representing the weather conditions that best fit. Furthermore the tourist sites can be categorized into several types that could be matched against the user's available free time; a museum for example requires a significant amount of time while a monument can be visited in less time. The rules to be applied use the type of tourist site to define the minimum required free time and compare this to the time likely to be available to the requesting user.

In the SALSA framework the Drools rule engine is used to realize the presented transformation approach. In Drools, rules can be described in the form of XML-documents composed of two parts – a left-hand-side which evaluates the occurrence of a certain fact and a right-hand-side which defines the conclusion or effect followed by some action, in our case the transformation. Since Drools allows the integration of Java statements within the left- and right-hand-side part of rules, Drools was well suited

for the SALSA framework which is implemented using Java technology. Instead of using Java, XSLT style sheets could also have been used to define the transformation. Generally, in the transformation approach, two different types of rules can be applied that we classify into dynamic and static rules. Elements of the core service description are transformed using static rules, for example the geographical location of a service description is always mapped the same way into a contextual representation. Therefore, a set of standard, static rules exists. Obviously elements to be transformed which appear in the domain-specific extensions of the service description are bound to dynamic rules which need to be specified by the service provider and can be changed or removed over time. For this kind of rule a few standard transformations to specific context data types exist that can be applied with a condition and an effect in a Drools rule.

### *5.3 Matching in Context-Sensitive Service Discovery*

The process of service discovery in the SALSA framework is in general a two-step process. Since each search request in SALSA consists of an implicitly delivered set of context data and explicit user-defined parameters, the first step is to pre-filter service descriptions, described in XML documents, that match the explicit defined search parameters. Mostly, these are facts contained in the service description, e.g. for a context-sensitive gastronomy guide, the explicit parameter "Italian" kitchen, pre-selects all service descriptions that contain this fact. In SALSA, we therefore apply the XPath query language which is supported in a very efficient way by the Natix XML Database [16], developed by a member of the SALSA team at the University of Mannheim. Afterwards, the implicit context set is enhanced using the previously described processing mechanisms and then used for context matching after applying the transformation approach. This task is performed by the *ContextMatcher* that can be part of a service and that optionally can also be included within the mobile client application for local context matching using private context information as already described in section 5.1. Since the result of the service transformation process is a representation of context which is based on pre-defined shared context data types, the matching process directly compares context types that share the same namespace. For each pre-defined context data type a matching routine is available or needs to be provided when extending the set of context data types. The result of the context matching process is an ordered set of suitable services based on the degree of matching. The result set is returned to the mobile client where it can be further processed by local context matching using the private context attributes. Other mechanisms that can be applied locally for service filtering and ordering are also part of the SALSA client framework and have been presented in [3]. In particular, preferences and history data can be used to order the services in a much more personalized way beyond context-sensitive service discovery.

## 6 The SALSA Framework

In this section we introduce the framework developed during the SALSA project to support our approach for context-sensitive service discovery used from mobile clients. It includes a client framework to build generic applications for service discovery as well as a component-based server framework to support the rapid implementation of context-sensitive service brokers/providers that are able to deliver useful services and information depending on the mobile user's current context.

*6.1 The Client Framework*

The SALSA client framework offers the basic capabilities needed to develop mobile client applications that make use of context-sensitive service discovery. In this section we will introduce the main components offered by the client framework. These are also depicted in Figure 4.

The *ContextManager* is mainly responsible for the administration of context. Arbitrary context sensors can be included and registered at the *ContextManager* which uses both "pull" and "push" mechanisms to receive context information. A further responsibility is the administration of context attributes that can be declared by the user as public, private or blurred as mentioned in section 5.1. When search requests are initiated by the mobile user, the *ContextManager* automatically collects all available contextual information and creates an XML search request. The *ContextManager* is also involved when search results are received by the client and local context matching needs to be applied to deal with private context attributes.

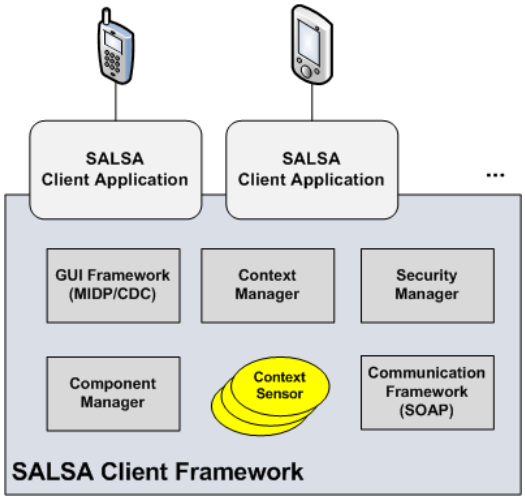


Figure 4 The SALSA Client Framework

The *ComponentManager* is responsible for the reconfiguration of the mobile client application. An important capability of the SALSA framework is to download software components from service providers that can include graphical user interfaces, business logic and also security components to support customized usage of the service. The *ComponentManager* is responsible for downloading and managing access to these components.

The *CommunicationManager* is responsible for the submission of requests to remote services. Since the whole framework is based on the principles of service-oriented architectures, it therefore uses SOAP. The *SecurityManager*, which sits on top, is responsible for ensuring a secure communication. It optionally also offers the possibility of anonymous communication using a Tor (The Onion Router) anonymity network approach, so that service brokers/providers are not able to infer information about certain clients over time. To support the storage of local sensitive data, it also enables encrypted data storage mechanisms on the mobile client.

The *GUI framework* [5] implemented in the SALSA framework is based on the XML User Interface Language (XUL) and offers the development of context-sensitive user interfaces. It separates the UI presentation and application logic whilst offering portability over the different Java ME platform configurations. While context is useful for service discovery, it can also be useful for reconfiguration and adaptation of the user interface. With the developed framework, user interfaces can adapt themselves to changing context and to different screen resolutions or orientations avoiding and reducing extensive programming effort for developers of mobile client applications.

### *6.2 The Server Framework*

In this section we introduce the core of the server framework. This is based on the notion of a Service Discovery Service (SDS) that acts as a context-sensitive service broker and offers components using mainstream development technology to realize context-sensitive service discovery based on the principles of service-oriented architectures. An SDS can be used in several ways as also described in section 7 when the SALSA prototype configuration is introduced. Figure 5 shows the basic framework architecture of an SDS.

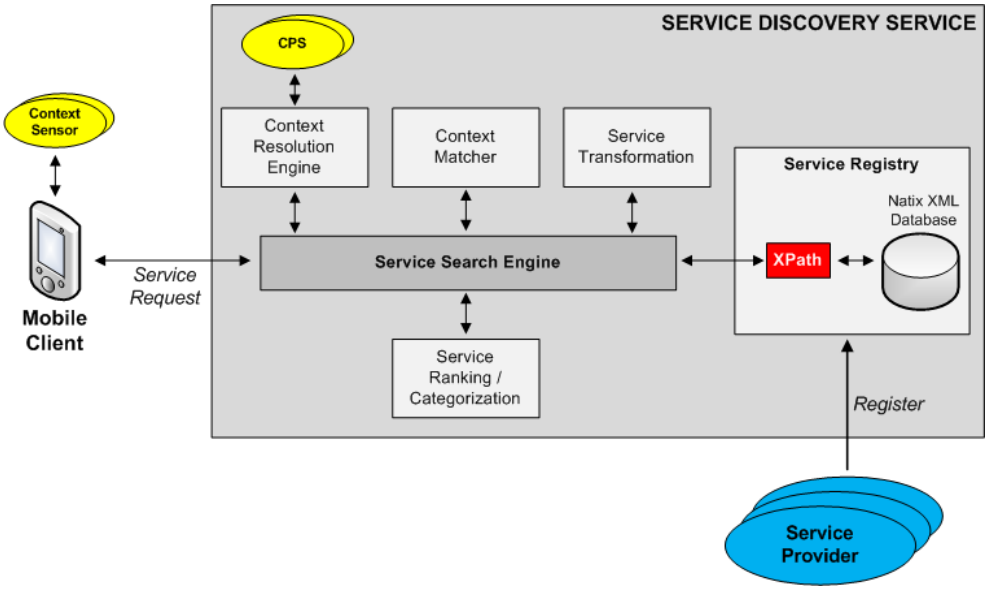


Figure 5 The SALSA Server Framework

The key part of the SDS is the *ServiceRegistry* which is needed to maintain descriptions of services available for discovery. For the SALSA prototype, this has been realized using the Natix XML database [17] for storage and retrieval of XML documents using optimized query mechanisms with XPath [16]. Service providers can register their services in the form of XML documents, according to the schema presented in section 3. This can be realized in several ways, using online registration forms provided by the service provider and customized to the domain of the service or by automated transformation of standard service descriptions.

The *ServiceSearchEngine* is responsible for coordinating incoming search requests. It first uses the implicit contextual information and the *ContextResolutionEngine* to extend the context set as far as possible as described in section 4. Then it queries the *ServiceRegistry* for potential service descriptions using XPath expressions built on the explicitly specified parameters of the delivered search request. Using the *ServiceTransformation* component, the result set of service descriptions is transformed and enhanced with the suitable contextual descriptions for each service and is then used by the *ContextMatcher* in the matching process as described in section 5.3. Finally, the result set is optionally categorized and/or ranked and returned to the mobile client that issued the search request.

## 7 The SALSA Prototype Scenario

In this section we briefly introduce the specific scenario and configuration that was chosen for the initial SALSA prototype. The basic configuration contains a mobile client, a Universal Service Discovery Service (USDS) and various additional Service Discovery Services (SDSs) registered as services at the USDS. It is important to note that the USDS and SDSs rely on the same basic technology described in section 6.

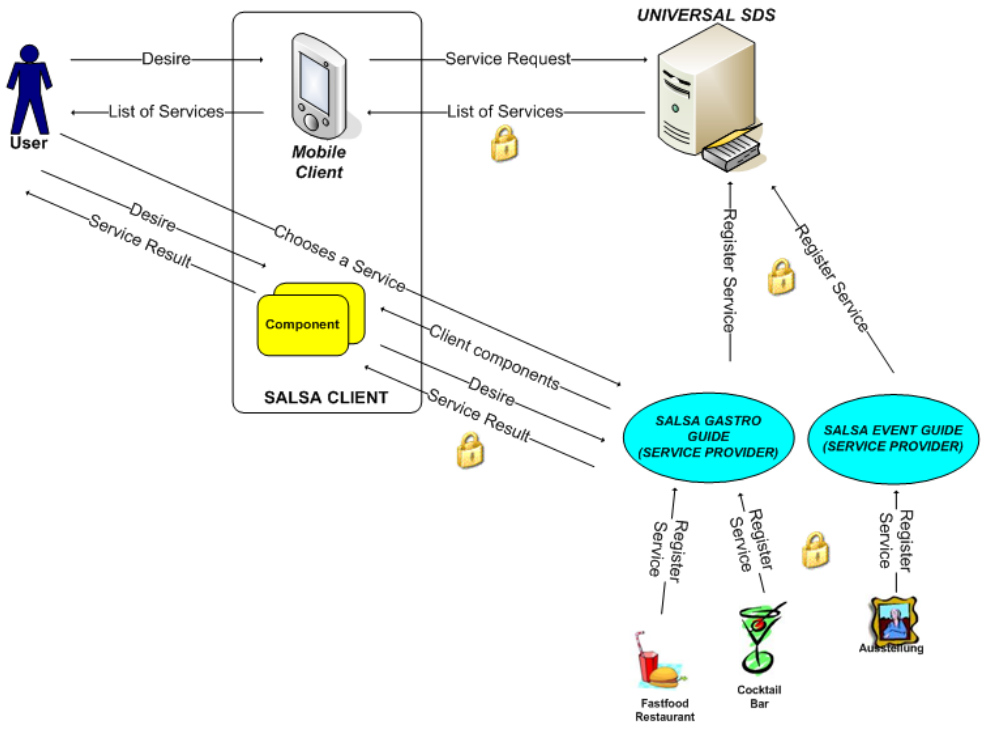


Figure 6 SALSA Prototype Scenario

In the user-managed linear configuration shown in Figure 6, the user sends an initial service request that contains all the available context information to the USDS which acts as a first-level service broker. The USDS then processes the context and pre-selects specialized context-sensitive service brokers that are suitable for the given context and returns a result list to the mobile client. Optionally, the mobile client can further refine the results using local private context information. The

resulting list of services is then presented to the mobile user. When the user selects a service from the result list, the mobile client first analyzes the service description. If the selected service requires additional components, the *ComponentManager* downloads these from the service provider (e.g. a graphical user interface or security components) and executes them. Then the service can be used by submitting optionally explicit request parameters if required (e.g. an event guide could require the category of the event, or the geographical search range) and the implicit context set. The service then processes the input parameters of the request using the same basic SDS processing technology to provide context-sensitive service discovery as also provided by the USDS.

Within the SALSA project at the University of Mannheim we implemented several kinds of services that follow this principle using the presented server framework. Examples include a context-sensitive gastronomy guide, a bargain hunter, an event guide and a tourist guide. All of these services have different requirements and use different kinds of context, as well as different rules that are applied for context matching. This shows that the generic technology and approach developed in the SALSA project is applicable to many different kinds of service-discovery scenarios for Mobile Commerce.

## 8 Conclusion and final remarks

In this paper we have introduced the SALSA framework for the development of mobile client applications and services that make use of context-sensitive service discovery using widely available, mainstream development technologies. The SALSA framework and its flexible architecture allow dynamic integration of new services in multiple configurations as also explained in another paper [4]. The framework not only supports the rapid development and deployment of context-sensitive services, it also simplifies the enhancement of conventional services with context-sensitive mobile commerce services. It distinguishes itself from other approaches by offering a simple context model that is easy to understand with the possibility of dynamic rule definition to influence the contextual behaviour of services. The context model and the rule base for realization of context-sensitive service discovery can be extended and applied in a simple way. Furthermore, privacy and trust are also guaranteed in the SALSA framework since the distinction between private, public and blurred context can be decided by the user himself.

By integrating mainstream technologies we believe the SALSA approach represents a pragmatic balance between simplicity and capability. It does what is needed in as straightforward a way as possible. By developing a range of different example applications and services using the framework we have demonstrated its usability. What remains to be evaluated is whether the framework does in fact speed up and reduce the cost of developing context-sensitive mobile applications, and whether the types of applications that can be developed using the framework are indeed more attractive to users. There are also a number of technical enhancements that would be desirable for an infrastructure for context-sensitive services in a Mobile Commerce scenario, like the integration of a payment or commission infrastructure for example. Other enhancements relate to higher-level support of context using a more complex context model including quality of context and better support for adaptation of multi-media content for applications and services. We are currently performing these investigations and enhancements within the Mobile Business Research Group at the University of Mannheim within the context of the GEM project funded by the DFG.

**Acknowledgement**

The authors wish to acknowledge the funding support from the “Landesstiftung Baden-Wuerttemberg” for the SALSA project at the University of Mannheim.

**References**

1. A. K. Dey and G. D. Abowd. Towards a Better Understanding of Context and Context-Awareness. In Proceedings of The Workshop on the What, Who, Where, When, and How of Context-Awareness held in conjunction with CHI’2000, The Hague, The Netherlands, 2000.
2. The OWL Service Coalition: OWL-S: Semantic Markup for Web Services. http://www.daml.org/services/owl-s/1.1/overview, 2004.
3. T. Butter, S. Deibert and F. Rothlauf, “Using Private and Public Context – An Approch for Mobile Discovery and Search Services”, In: Kirste, Thomas and König-Ries, Birgitta and Pousttchi, Key and Turowski, Klaus (eds): Mobile Informationssysteme - Potentiale, Hindernisse, Einsatz, pp 144-155, Bonner Köllen Verlag, Bonn, 2006.
4. M. Aleksy, C. Atkinson, P. Bostan, T. Butter, M. Schader. Interaction Styles for Service Discovery in Mobile Business Applications. In Proceedings of 9th International Workshop on Network-based Information Systems, DEXA, Krakau, Poland, 2006
5. T. Butter, M. Aleksy, P. Bostan and M. Schader. Context-aware user interface framework for mobile applications. In Proceedings of the 27th International Conference on Distributed Computing Systems - Workshops (ICDCS Workshops 2007), Toronto, Ontario, Canada, 2007.
6. E. Tatli, D. Stegemann, S. Lucks. Security Challenges of Location-Aware Mobile Business. In the second IEEE International Workshop on Mobile Commerce and Services (WMCS'05), München-Germany. 2005.
7. T. King et al. COMPASS: A Probabilistic Indoor Positioning System Based on 802.11 and Digital Compasses. In Proc. of the First ACM International Workshop on Wireless Network Testbeds, Experimental evaluation and CHaracterization (WiNTECH 2006), pp. 34-40, Los Angeles, CA, USA, September 2006
8. H.H. Bauer, T. Reichardt, A. Schüle. User Requirements for Location-Based Services - An analysis on the basis of literature. Wissenschaftliches Arbeitspapier Nr. W94, Institut für Marktorientierte Unternehmensführung, Universität Mannheim, 2005 .
9. S. K. Mostefaoui and B. Hirsbrunner. Context Aware Service Provisioning. In Proceedings of the IEEE/ACS International Conference on Pervasive Services (ICPS 04), 2004.
10. T. Broens et al.. Context-aware, ontology-based, service discovery. In Proc. of the Second European Symposium on Ambient Intelligence (EUSAI 2004), Eindhoven, The Netherlands, November 8-10, 2004
11. S. Pokraev et al.. Service Platform for Rapid development and Deployment of Context-Aware, Mobile Applications. In Proceedings of International Conference on Webservices (ICWS'05), Orlando, Florida, USA, 2005.
12. M. Baldauf, S. Dustdar and F. Rosenberg. A survey on context-aware systems. Int. Journal of Ad Hoc and Ubiquitous Computing, Vol. 2, No. 4, pp.263–277, 2007.
13. A.K. Dey, D. Salber and G.D. Abowd, G.D.: A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. Human-Computer Interaction 16, pp. 97–166, 2001.
14. T. Strang and C. Linnhoff-Popien. A Context Modeling Survey. In Processings of Workshop on Advanced Context Modelling, Reasoning and Management as part of UbiComp 2004, Nottingham/England, September 2004.

15. NAICS – North American Industry Classification Standard, http://www.census.gov/epcd/www/naics.html
16. M. Brantner, C.-C. Kanne, S. Helmer, G. Moerkotte. Full-fledged Algebraic XPath Processing in Natix. In Proc. of the 21th ICDE Conference, pp. 705-716, Tokyo, Japan, 2005.
17. Natix – A native database system for XML, http://pi3.informatik.uni-mannheim.de/~moer/natix.html
18. J. Kuck and F. Reichartz. A collaborative and feature-based approach to Context-Sensitive Service Discovery. In Proceedings of 5th WWW Workshop on Emerging Applications for Wireless and Mobile Access (MobEA'07), Banff, Alberta, Canada, 2007.
19. P. Dockhorn Costa, I. Ferreira Pires, M. van Sinderen, D. Rios. Services Platforms for Context-Aware Applications. In Proc. of the Second European Symposium on Ambient Intelligence (EUSAI 2004), Eindhoven, The Netherlands, November 2004.