A Framework for Building and Operating Context-Aware Mobile Applications

Aaratee Shrestha, Bettina Biel, Tobias Griebe, and Volker Gruhn

University of Duisburg-Essen, Gerlingstrasse 16, 45127 Essen, Germany
{aaratee.shrestha,bettina.biel,
tobias.griebe,volker.gruhn}@paluno.uni-due.de

Abstract. A context-aware mobile framework must support and handle complex context data which is dynamically manipulated in the distributed mobile network. Research in this area has focused on the efficient design of such a framework. However, there are still key problems such as dynamic adaptation, reusability, interoperability, high energy and memory consumption. Our approach to solve the problems of Context-Aware Mobile Applications (CAMA) is to design a framework architecture by using Service Oriented Architecture (SOA). The reusable, loosely-coupled local and external services allow CAMA to communicate with the CAMA Framework, OS and external service providers using minimum interfaces. The framework supports interoperability, dynamic adaptability and context handling in a frequently changing environment. In this work-in-progress paper, we define SOA, usability and testing requirements for a prototype CAMA and the CAMA Framework. We conclude that our approach will enhance mobile framework architecture to provide solutions to the key problems of CAMA.

Keywords: context-awareness, context-aware mobile applications (CAMA), service oriented architecture (SOA).

1 Introduction

Mobile technology has been broadly adapted in business and entertainment domains with increasing demand. Users can access context data, workflows and systems anywhere anytime. A robust mobile framework that can handle different context data, process it and make it accessible to lightweight applications for users without time, place and network restrictions is an important requirement.

Context is the "situational information of entities such as person, places or objects, that are relevant to the interactions between a user and an application" [6]. We define CAMA as context-aware mobile applications where context data and services are manipulated in dynamic environments. We agree with Pauty [15], that "the services are aware of the current context of the user and self-adapt to context changes". Thanh [19] states, that mobile services are realized by combining different SOA services. SOA features are essential in an environment depending on non-robust connections and multi-device user access to services [18]. Also, SOA provides the advantages of flexibility, implementation abstraction and interoperability [13,17,2], that are very important in dynamic mobile computing platforms. SOA can address multiple client devices in distributed environments where the client heavily relies on the interaction with a server system. The server exposes only one interface to various clients and acts as provider of autonomous interfaces [16]. Because of these advantages over traditional mobile web services, our framework is motivated to implement SOA.

In this work-in-progress paper, our first step towards a solution is to introduce the example of a Calendar-Location-Weather (CLW) application and then summarize its requirements. We explain how our framework is designed to overcome the general challenges and problems of context-aware frameworks. Our framework supports key SOA features such as reusability, interoperability, loose coupling as well as dynamic adaptability and context handling in dynamic environments. The contribution of this paper is an architecture design by using the CLW example of a CAMA, usability and testing requirements. The structure of this paper is as follows. Section 2 covers the CLW application scenario, followed by a summary of requirements of the CAMA framework in section 3. We present the current architecture design in section 4 and compare it to related work in section 5. A conclusion and future work is found in section 6.

2 The Calendar-Location-Weather (CLW) Application

Imagine a user wants to be woken up in the morning to travel from Essen to Berlin to attend a meeting. He opens the CLW application, enters the event's name, date, time, location, means of transportation (train) and saves it. Weather forecast and related services are enabled by him as well. The CLW will wake him up on time, give weather information and, if it rains, it reminds him to take an umbrella. In case the train is late, it will inform him and provide a list of alternatives (e.g., other trains, buses, renting a car). The CLW application will alarm the user when it is time to go to catch the train at Essen main station.

The CAMA framework stores the event data in the Context Database and communicates with external provider's databases. The framework checks the

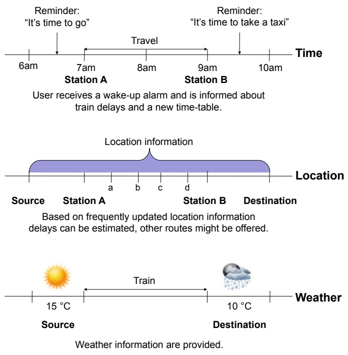
Reminder: "It's time to go"
Reminder: "It's time to take a taxi"
Travel
Time
6am
7am
8am
9am
10am
Station A
Station B
User receives a wake-up alarm and is informed about train delays and a new time-table.
Location information
Location
a
b
c
d
Source
Station A
Station B
Destination
Based on frequently updated location information delays can be estimated, other routes might be offered.
Train
Weather
15 °C
10 °C
Source
Destination
Weather information are provided.

Fig. 1. The CLW application integrates three types of context information

date whether it matches the calendar date in the Context Database using the Inference Rule Set. The Inference Rule Set supplies commands to update the data of the CLW application and queries location and weather updates, e.g., the German railways or a weather forecast.When the context changes, it is displayed at the current location map and the weather information.

3 Requirements of the CAMA Framework

Mobile SOA. Using SOA for mobile architectures is an emerging area and several researches have been done in different aspects of context-awareness. In this research, we analyzed the specific problems related to CAMA while using SOA, which was considered during the design of framework architecture. High level requirements based on business goals and the usage context are: instant access to relevant applications, services and data required by mobile device users through a highly usable unified user interface of composite CAMA. These are not supported by traditional framework approaches as explained by IDC Research Report [11]. Monitoring of the mobile context of user and environment is frequently needed, as it changes continuously. In traditional frameworks, this is difficult for the reasons of tight coupling, non-interoperability and modes of operation as described by Ennai et al. [7]. Interoperability and reusability of local and remote context data and services are often required by mobile device users such that a large number of CAMA are using the same context data and services multiple times. Transferring large amounts local context data is difficult when there arise any problems with the device such as network, memory, battery. In present mobile devices, storage space and computing capability are increased. Yet, problems exist to deploy large complex applications and data on a mobile device as explained by Tergujeff et al. [18].

Usability of CAMA. CAMA running in our proposed framework should be highly usable. Usability is closely related to other qualities, such as performance, robustness, fault-tolerance, security, modifiability and adaptability. Hence, we decided to design the framework architecture using the Software ArchiTecture analysis of Usability Requirements realizatioN (SATURN) method [1]. During this research, the method is applied in an iterative architecture-based development process with alternating analysis and design activities.

In SATURN, requirements are expressed through scenarios, which are described and/or selected from pattern-based knowledge base of generic scenarios. Regarding usability, we selected the scenarios *Canceling Commands*, *Feedback*, *Context-Aware Interaction*, *Positioning* and added *Evaluating the application.*

Evaluating the scenarios, the open questions are (1) which structure and behavior of components and interfaces can realize the use cases of a scenario; and (2) how responsibilities can be distributed between the CLW application, the CAMA framework and the mobile platform.

Testing CAMA. Due to the high dynamics of changing context and service topology, testing CAMA imposes a novel set of requirements to the design of the

CAMA framework, the design of test cases and the testing process. CAMA depend on multiple internal and external services whose availability may change at any moment, either inducing a service discovery process or rendering the application inoperative. Test engineers must design functional test cases and sufficiently control the CAMA's execution environment including context data. Rather than concentrating on the system under test, the testing of a CAMA requires the test environment to produce and manipulate context data on demand.

As it is complicated to artificially reproduce context information for testing purposes, the design of the CAMA framework needs to implement features to facilitate testing of CAMA. The key questions regarding an integrated testbed in the architecture design are: (1) how context information can be generated; (2) how test cases need to be designed to react properly to changing context information; and (3) what test coverage criteria need to be applied.

Test cases require the artificial setting of position data and the creation of weather, traffic and travel information. Besides the local sensor readings, most of this information is derived from the context-aware system of the CAMA framework which in turn consumes data from external sources. Hence, the CAMA framework needs to employ a testing mode which allows the artificial provisioning of context information, overriding the actual context data, to create a valid and reproducible test environment.

4 Mobile Framework

The framework architecture is designed to support the customized CAMA development for different mobile platforms and features service-oriented functionality. Its design is focused on the dynamic adaptability to frequent and unpredictable changes in context and user requirements to provide continuously available services. The integration of context-aware data and services is executed in real-time. Therefore, our framework architecture as seen in Fig. 2 is designed to fulfill some of the requirements to solve the above addressed problems and challenges. Mobile applications act as clients and may heavily rely on context data and context-aware services. Different components of the architecture play a vital role for accessing the requested data. The components of our mobile framework are briefly discussed as follows.

The *Service Provider* creates a mobile service and publishes its interface in Fig. 2 label (2) and access information to the Service Registry. The Service Provider sends the context data (21) received from the Context Interpreter to the Service Consumer. The sub-components of Service Provider such as *Publishing Manager, Service Interface Handler and Service Manager* helps to publish service interfaces. The Publishing Manager is responsible for publishing the service interface of different services. Service Manager decides in which category the service should be listed in the Service Registry and list all potential service recipients. It decides which service to expose, regarding the service charges and how to make trade-offs between security and availability. It is responsible for what Service Level Agreements (SLAs) are required to use the service. Service

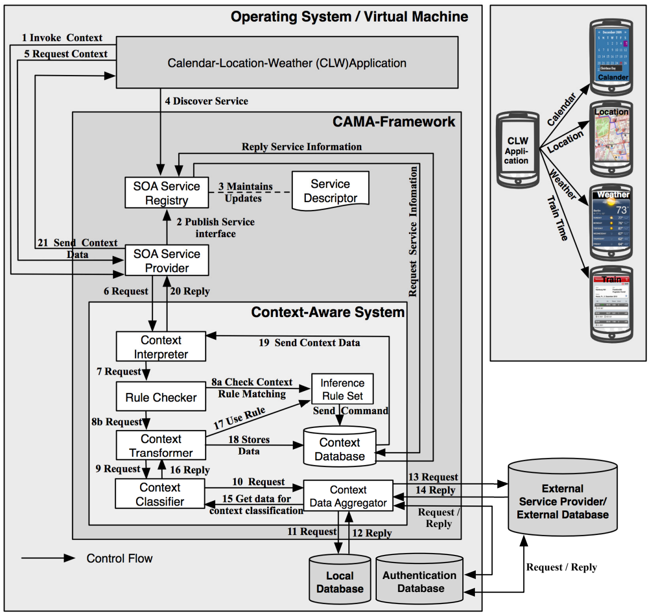

Fig. 2. CAMA Framework Architecture and CLW Application

Interface Handler acts as an interface for the service contract request between service provider and service consumer using the SLA.

The *Service Consumer* (e.g., CLW Application) of a client locates entries in the Service Registry with the help of a find-bind mechanism (1,2,3,4) and requests the Service Provider to invoke one of its services. After the discovery of a service in the Service Registry, the Service Consumer sends a service request (5) that is executed by the Service Provider. The *Service Registry* is a reference database containing information about services, service definitions, interfaces and parameters. The *Service Descriptor* is a database containing data and metadata maintained and updated (3) by the Service Registry.

The Service Provider, Service Consumer and Service Registry utilize SOA for using the services from the Context Database. For the communication and the request/response mechanism of SOA, the framework uses HTTP based Representational State Transfer (REST) protocol, which has advantages on the client side in mobile communication compared to the SOAP/WSDL protocol like in traditional frameworks as discussed by [7] where communication is more focused between multiple backend systems. According to [14] REST implementation of the data transmission proved to be more efficient compared to SOAP. Our CAMA framework utilizes the REST implementation as it features lightweight, flexible contracts and interfaces – a uniform way to interconnect with web resources and a balance between security and usage of resources.

Context Interpreter sends the request (6) to the Rule Checker, and checks for valid context when the Service Provider requests context data. It collects data from the Context Database (19) and makes it available to the Service Provider (20). The *Rule Checker* checks in (8a) the Inference Rule Set whether the rule for the specific context matches with the predefined rule. The Rule Checker sends requests to the Context Transformer (8b). The *Context Transformer* sends a request (9) to the Context Classifier to get data (15) from the Context Data Aggregator. It uses the rule (17) from the Inference Rule Set and applies it on the received data (16) from the Context Classifier. The *Inference Rule Set* is a part of the Context Database, which consists of a set of rules and facts and sends context update commands. It defines rules for each context. Then the Context Transformer finally stores the data (18) in the Context Database. The *Context Database* makes records about what context data the Context Interpreter request to it. It sends (19) the context data to the Context Interpreter and updated context information to the Service Registry. The *Context Classifier* classifies the context in different categories so that the Context Database can store and manage it in different databases, e.g., location, weather. The *Context Data Aggregator* aggregates the context data (12) from the local or the (14) external database. The main purpose of this aggregation is to get detail information about particular groups based on context parameters. When the context data is requested, the Context Data Aggregator first checks if it is available in the Local Database and then External Database. The *Local Database* provides a small set of generalized data which is frequently used by CAMA, e.g., calendar database located on the mobile device.

CAMA framework retrieves context data at first from the Context Database. The data is processed and stored in the Context Database from the Local Database. If data is not found in the Context Database, it searches the External Database and the External Service Provider as they provide more detailed context information. The Authentication Database holds the user credentials and manages secure login and access. The context services originate across various channels. The CAMA framework provides automatic session management such as session-time out, secure access of external service provider and multiple service providers.

5 Related Work

The traditional SOA framework on the client has been proposed by Tergujeff et al. [18] does not address context-awareness. Gehlen et al. [8] present a client proxy that executes requests and routes responses, but it does not support real-time change of context. A mobile middleware for CAMA on rule-based data monitoring is discussed by Costa et al. [5] but specific concerns (e.g., lack of user context, integration of services, etc.) have not been addressed. A mobile web service framework is described by Kim et al. [12]. However, it does not support hosting and migrating services in a context-aware environment. A traditional SOA framework that moves all the processing to the server and leaves only user interactions and user interface on the client side is mentioned by Kozel and Slaby [13]. In contrast, our solution is more focused on lightweight client-side processing. A reflective middleware for dynamic adaptability proposed by

Ghim et al. [9] does not address SOA features and uses mobile agents instead. A dynamic framework for context-aware mobile services by Chang et al. [4] describes a problem classification and a complexity model of context-aware mobile services into 3D dynamic problems. Our approach uses another classification of context, that is less complex.

CARISMA [3] mainly deals with the conflict between profiles of applications, that are kept as meta-data of the middleware and consist of passive and active parts. Policies are used to provide context services. CARISMA addresses issues related to the usage of context for dynamic adaptation of applications, but there are no application interfaces with local and external services and it mainly focuses on policy conflicts.

SOCAM [10] middleware is built on top of the OSGi service platform, and its architecture presents a formal context model based on ontology. It provides support for the tasks of dealing with context by context reasoning. External or internal context can be used by services directly or by Context Providers. The Context Interpreter consists of the Context Reasoner and the Context Database, which contains instances of the current ontology. The context is updated by a triggering mechanism. SOCAM has been deployed mainly for intelligent vehicle environment which may not be suitable for other mobile environments.

Conventional mobile distributed systems using web services have proven successful for the design of SOA in mobility, yet they could not overcome the challenges of context-awareness. Therefore, our SOA framework for lightweight and flexible CAMA is a more suitable solution.

6 Conclusions and Future Work

In this work-in-progress paper, we defined mobile SOA, usability and testing requirements for our mobile framework implementing a SOA based on a specific CAMA, i.e. the CLW application expressed through a scenario. Our CAMA framework architecture design provides a number of solutions to the problems introduced by context-awareness. We showed the conceptual design of the SOA and the context-aware mobile framework architecture. By using a SOA-architecture, we met requirements regarding instant access to applications, services and data and frequent monitoring of the mobile context of user and environment.

In our current work, we focus on the architecture design of the context adaptation and aim at realizing the usability and test requirements striving for a highly usable unified user interfaces of CAMA. Open questions and motivational future work comprise the data transfer on p2p interactions with near-by mobile devices, managing network problems and processing requests without context data in the Context Database and metadata in the Service Registry.

References

1. Biel, B., Grill, T., Gruhn, V.: Exploring the benefits of the combination of a software architecture analysis and a usability evaluation of a mobile application. Journal of Systems and Software 83(11), 2031–2044 (2010)

2. Bosch, J., Friedrichs, S., Jung, S., Helbig, J., Scherding, A.: Service orientation in the enterprise. Computer, 51–56 (2007)
3. Capra, L., Emmerich, W., Mascolo, C.: CARISMA: context-aware reflective middleware system for mobile applications. IEEE Transactions on Software Engineering 29(10), 929–945 (2003)
4. Chang, C.-C., Tseng, J.C.R., Lin, K.-J.: A dynamic capability framework for context-aware mobile services. In: Proc. of the 10th IEEE Conf. on E-Commerce Tech. and the 5th IEEE Conf. on Enterprise Computing, pp. 183–189 (2008)
5. Costa, P., Pires, L.F., Sinderen, M.V., Filho, J.P.: Towards a services platform for mobile context-aware applications. In: Proc. of 1st Intl. Workshop on Ubiquitous Computing, pp. 48–61 (2004)
6. Dey, A.K.: Understanding and using context. Personal Ubiquitous Comput. 5(1), 4–7 (2001)
7. Ennai, A., Bose, S.: MobileSOA: a service oriented web 2.0 framework for context-aware, lightweight and flexible mobile applications. In: EDOCW (2008)
8. Gehlen, G., Mavromatis, G.: Mobile web services based middleware for context-aware applications. In: Proc. of 11th European Wireless Conference 2005, pp. 784–790 (2005)
9. Ghim, S.-J., Yoon, Y.-I., Choe, J.-W.: A Reflective Approach to Dynamic Adaptation in Ubiquitous Computing Environment. In: Kahng, H.-K., Goto, S. (eds.) ICOIN 2004. LNCS, vol. 3090, pp. 75–82. Springer, Heidelberg (2004)
10. Gu, T., Pung, H.K., Zhang, D.Q.: A service-oriented middleware for building context-aware services. J. Netw. Comput. Appl. 28, 1–18 (2005)
11. IDC. Worldwide mobile enterprise applications 2006-2010 forecast and analysis. IDC Research (2006)
12. Kim, Y.-S., Lee, K.-H.: A lightweight framework for mobile web services. Journal Computer Science – Research and Development 24, 199–209 (2009)
13. Kozel, T., Slaby, A.: Mobile devices and web services. In: Proc. of 7th WSEAS Intl. Conference on Applied Computer Science, pp. 322–326 (2007)
14. Mulligan, G., Gracanin, D.: A comparison of soap and rest implementations of a service based interaction independence middleware framework. In: Proc. of the 2009 Winter Simulation Conference, pp. 1423–1432 (2009)
15. Pauty, J., Preuveneers, D., Rigole, P., Berbers, Y.: Research challenges in mobile and context-aware service development. In: Proc. of Future Research Challenges in Software and Services (2006)
16. Praher, C.P.: Mobile service oriented architecture in the context of information retrieval. Master's thesis, University of Linz (2008)
17. Schroth, C., Janner, T.: Web 2.0 and SOA: Converging concepts enabling the internet of services. IT Professional, 36–41 (2007)
18. Tergujeff, R., Haajanen, J., Leppanen, J., Toivonen, S.: Mobile SOA: Service orientation on lightweight mobile devices. In: Proc. of IEEE Intl Conf. on Web Services, pp. 1224–1225 (2007)
19. Thanh, D., Jorstad, I.: A service-oriented architecture framework for mobile services. Special issue on Situated Interaction and Ubiquitous Computing, 65–70 (2005)