Towards a Context Management Framework for MobiLife

Patrik Floréen*, Michael Przybilski*, Petteri Nurmi*, Johan Koolwaaij†, Anthony Tarlano¶, Matthias Wagner¶, Marko Luther¶, Fabien Bataille‖, Mathieu Boussard‖, Bernd Mrohs§, SianLun Lau‡

Abstract— **Recent advances in the field of context reasoning and in wireless information devices, which allow mobile access and provide a multitude of context information, have enabled the provision of new, context-aware applications and services. The architecture framework described in this paper provides a very flexible approach to implementing different reasoning tasks and combining it with application logic, in order to provide context-aware applications and services. We are describing the initial version of this architecture, which will be used within the MobiLife project.**

I. Introduction

Context-aware systems play an increasingly important role in modern software systems, especially in software for wireless information devices. Context, according to Dey and Abowd [1], 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 application themselves."

The MobiLife project (IST-2004-511607) aims to bring advances in mobile applications and services within the reach of users in their everyday life. In doing so, it will provide new context-aware and proactive services and applications.

For the handling of context information, from context modeling to context reasoning, and for the provisioning of services that are relevant to an end-user in a given context, MobiLife is designing an architecture, which we call the Context Management Framework.

In this paper, after a short review of some existing context-aware architectures in Section II, we describe our approach in Section III, and finalize with a discussion in Section IV.

II. Architectures of Context-Aware Systems

Different approaches have been taken to provide a common architecture for context-aware applications and the structuring of functionality of context-reasoning mechanisms. According to Moran and Dourish [2], current research focuses on either different versions of a blackboard-based approach, or on widget-based approaches. Usually these approaches are implemented in the form of middleware, or in the form of application frameworks. Another possibility that Moran and Dourish describe is the implementation int he form of interacting agents, which are distributed over a network. Also Mayrhofer [3] describes an architecture for context-aware systems, which is however focused on devices with limited resources, but which does not take into account the distribution of the reasoning mechanisms.

1) Middleware: Examples of middleware approaches include, e.g., the *Reconfigurable Context-Sensitive Middleware* (RCSM) [4], [5] and the *CORTEX* middleware [6]. RCSM is adaptive in the sense that, depending on the context-sensitive behavior of the applications, it adapts its object discovery and connection management mechanisms. CORTEX, on the other hand, introduces special entities, called *sentient objects*, which are responsible for receiving,

*University of Helsinki,
Email: {firstname.lastname}@cs.helsinki.fi;
†Telematica Instituut, Email: johan.koolwaaij@telin.nl;
‡University of Kassel, Email: slau@comtec.e-technik.uni-kassel.de;
§Fraunhofer FOKUS, Email: bernd.mrohs@fokus.fraunhofer.de;
¶DoCoMo Euro-Labs,
Email: {tarlano, luther, wagner}@docomolab-euro.com;
‖Alcatel CIT, Email: {firstname.lastname}@alcatel.fr

processing, and providing context-related information. Sentient objects are defined as autonomous objects that are able to sense their environment and act accordingly [6]. The advantage of this approach is the possibility to re-organize them, for instance depending on their primary task.
Also Siegemund [7] has presented a very interesting approach of a communication platform for smart objects. Focusing on changing the communication mechanisms and the inter-object collaboration depending on the situation, the presented platform allows for the specification and implementation of context-aware communication services and adapts the networking structures according to the context of an object.

2) Frameworks: Also some application frameworks that support context-awareness have been proposed. For example, Korpipää et al. [8] describe the implementation of a framework that supports the management of context information on mobile terminals. The structure of this framework is centered around the blackboard paradigm for communication, which is handled by a context manager. Most components that use this framework, including the applications, act as clients for the context management system on the device itself. Other services can run not only on the device itself, but potentially also in a distributed environment.
Another framework approach is *The Context Toolkit* [9]. The framework separates acquisition and presentation of context information from the application that requires it, by using so-called widgets. The focus of this work lays in the automatic inference of higher level context information from lower-level sensory data.
The Hydrogen project [10] describes a three-layered architecture, designed to overcome existing problems, found in context-aware mobile systems. The framework consists of adaptor, management and application layer. For the communication between the different layers, the described framework utilizes an XML-based protocol. The aim of the framework is the provision of an architecture that is lightweight, extensible, robust, and which also enables the possibility of adding further meta-information to the system.

III. The Context Management Framework

The process of inferring higher level context, as well as the learning of new inference models, can be very complex and can be achieved in different ways. For this reason we are structuring essential functionalities into different components, which can be configured for different tasks and reasoning methods. Figure 1 shows the key functions of our Context Management Framework (CMF). We describe each key function in detail below.

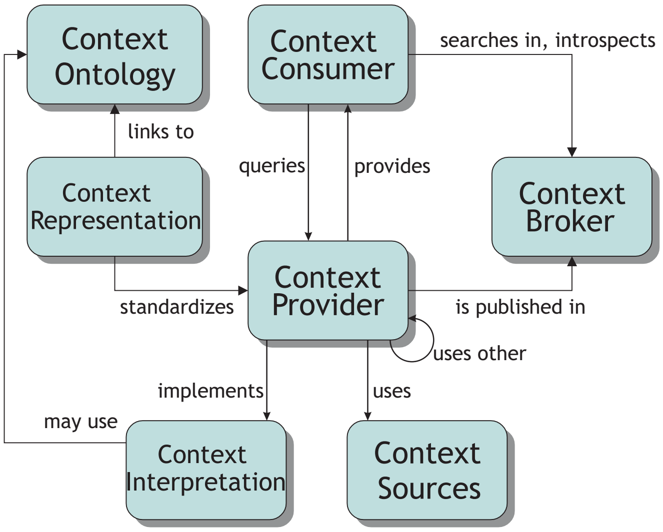

Fig. 1. Key functions of the CMF.

A. Context Provider

The Context Provider is a software entity that produces new context information from internal or external (context) information. A Context Provider exposes interfaces to provide context information to Context Consumers. These interfaces adhere to the MobiLife Context Representation standards. A Context provides is registered in the Context Broker, so that Context Consumers can discover and introspect it. The internal working of a Context Provider is usually hidden, but may include context aggregation, caching, prediction, reasoning etc.
Furthermore, in order to enable easy reconfiguration of different Context Providers and other components, so-called *Super Distribution Objects (SDO)* [11] may be used. They provide a base-technology, enabling the distributed interoperability between the different components of the system. For the Context Management Framework, SDOs

encapsulate on the one hand hardware and software entities that provide raw data, and on the other hand context providers, which are responsible for the essential context-reasoning tasks.
The SDO approach applies not only to the Context Provider Components, but also to the Context Sources, Context Consumers and Context Broker, which all have to implement the basic SDO interfaces.

B. Context Sources

The data sources (e.g. GPS receiver or a rain sensor) are usually not under MobiLife control and do not necessarily adhere to the Context Representation. For this reason they are wrapped in the form of Context Sources, which simply provide their information in a way that adheres to the Context Representation. A more complex Context Source would derive higher-order information, e.g. country instead of raw GPS coordinates, or implement some pre-processing on its own, for instance to clean-up outliers.

C. Context Consumer

A Context Consumer is a software entity that uses the Context Provider interface as communication endpoint to obtain contextual data. Typically, Context Consumers are implemented in the form of either middleware services or applications that use context information in combination with their application logic.

D. Context Broker

The main goal of the Context Broker is to provide a network addressable infrastructure service, in a *Service-oriented Architecture (SOA)* style architecture, which permits Context Providers to publish interface contracts and allows Context Consumers to find services and later on use the published interfaces. In order for the Context Broker to perform the lookup, all Context Providers have to register themselves to the Context Broker.
During context discovery the Context Broker may actively assist the discovery of Context Providers and Context Consumers by broadcasting advertisements. However, in order to decrease the amount of network overhead, the default behavior is to passively await connection requests from Context Consumers or Context Providers. The Broker discovery itself can be done by several means, such as broadcast or shoutcast, depending on the implementation.

E. Context Representation

In order to achieve interoperability between Context Providers from diverse domains, the context management framework standardizes a meta model for Context Representation that all Context Providers should adhere to, in order to register themselves in a Context Broker and to enable potential Context Consumers to discover the context information they need.
The design of the meta-model is based on a few simple design rules. The first is that the canonical form for the meta-model will be XML-based. All later formats can be derived from this canonical form and a new format may only be introduced if proper tooling is provided to convert from and to the canonical form. The reasons to start with XML are easy human inspection, extension, validation and integration with development platforms and tooling. Other design rules include rules on versioning, security, naming schemes and identities.
The initial version of the MobiLife Context Representation meta-model prescribes a standard for a Context Provider advertisement, a context query that could be requested from a content provider, and a context element, the elementary piece of context information.
With the prior knowledge of this meta-model a Context Consumer can discover the capabilities of a Context Provider, request relevant context information from that Context Provider and parse the response, in order to use the context information.

F. Context Ontology

Ontologies promise to play a pivotal role for different semantic-based and semantic-aware applications, not only in the area of the Semantic Web [12], but also in next generation mobile communication systems.
In MobiLife, we refer to an ontology as a logical theory, accounting for the intended meaning of a formal vocabulary, i.e. its ontological commitment to a particular conceptualization of MobiLife

applications and domains. Reasoning about such logical theories requires logic-based inference systems which have been studied within the field of knowledge representation in the AI community in the past. Description Logics (DLs), as a decidable fragment of first-order predicate logic, turned out to be an adequate formalism for the representation and the reasoning about expressive ontologies [13]. As a consequence, DLs form the formal foundation of W3C's Web Ontology Language (OWL) [14].
The MobiLife CMF aims at leveraging OWL ontologies for context management, in terms of so-called upper ontologies. These high-level, logical representations will support the representation of context data at different levels of abstraction, enable qualitative contextual logic reasoning and connect MobiLife context vocabularies to external standard or non-standard Context Representations.

G. Context Interpretation

The goal of Context Interpretation is to deduce semantically flavored descriptions of the context of the user or another relevant entity, such as a group. The technical implementation of the Context Interpretation, as part of the CMF, consists of a set of container components, which encapsulate within them communication, representation, privacy and security related functionalities.
Furthermore, different kinds of containers allow for different kinds of processing functionalities and the container components can subscribe to events (data) from other components. A typical configuration of these components is depicted in Fig. 2.
In the following, a short description of the individual components is given.

- A *Context Source* encapsulates the gathering of data.
- The *Preprocessor* handles possible inconsistencies with the raw data from an individual source. The processing that is applied serves only as a preceding step for the actual processing.
- The *Feature Extractor* attempts to reduce the amount of data by aggregating signals over time or by performing transformations, such as deriving pitch from speech.
- A *Feature Selector* selects the relevant features, derived by the feature extractor, and is responsible for communicating the data further. These first four components typically reside on the terminal side, as they do not require major resources.
- The *Source Selector* selects the most relevant sources, which may have been already further processed by the preprocessing, by the feature extraction or by the feature selection functionalities. This step further attempts to reduce communication and computational overhead.
- The *Classifier* itself attempts for instance to recognize the activity of the user from data. Additionally, the classifiers may provide confidence levels or full probability distributions on the values. Typically this process requires the most processing resources and is thus often implemented remotely.

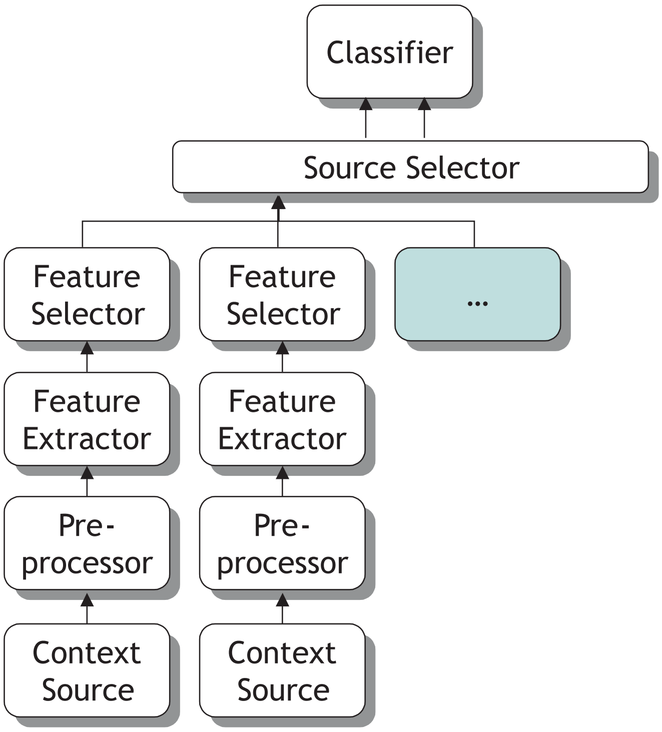

Fig. 2. A typical configuration of the Context Interpretation key function.

IV. Discussion

The previously described architecture has the advantage that it is very flexible and allows the configuration of the different Context Providers according to the different reasoning tasks and mechanisms used. it is furthermore suitable for the implementa-

tion in a distributed manner, using a combination of wireless information devices with limited resources, and server mechanisms with high level of resources. Besides the context reasoning in the steps described above, context reasoning can also be used to predict future context. In order to achieve this task, it is necessary to extend the timing information, used in the different reasoning steps and to include also references to probable future events and context information, as well as their confidence levels. This is another goal we are currently working on.
The CMF described in this paper is an initial version that has resulted after the first months' work in the MobiLife project. The CMF will be further developed and refined during the course of the project.

ACKNOWLEDGMENT

This work has been performed in the framework of the IST project IST-2004-511607 MobiLife, which is partly funded by the European Union. The authors would like to acknowledge the contributions of their colleagues.

REFERENCES

[1] A. K. Dey and G. D. Abowd, "Towards a Better Understanding of Context and Context-Awareness," College of Computing, Georgia Institute of Technology, Tech. Rep. GIT-GVU-99-22, 1999.

[2] T. P. Moran and P. Dourish, "Introduction to Special Issue on Context-Aware Computing," *Human-Computer Interaction (HCI)*, vol. 16, no. 2-3, pp. 87 – 96, 2001.

[3] R. Mayrhofer, "An Architecture for Context Prediction," *Advances in Pervasive Computing, part of the Second International Conference on Pervasive Computing (PERVASIVE 2004)*, vol. 176, pp. 65 – 72, Apr. 2004.

[4] S. S. Yau, F. Karim, Y. Wang, B. Wang, and S. K. Gupta, "Reconfigurable Context-Sensitive Middleware for Pervasive Computing," *Pervasive Computing*, vol. 1, no. 3, pp. 33–40, Jul.- Sep. 2002.

[5] S. S. Yau and F. Karim, "An Adaptive Middleware for Context-Sensitive Communications for Real-Time Applications in Ubiquitous Computing Environments," *Real-Time Systems*, vol. 26, no. 1, pp. 29–61, 2004.

[6] H. A. Duran-Limon, G. S. Blair, A. Friday, P. Grace, G. Samartzidis, T. Sivaharan, and M. WU, "Context-aware middleware for pervasive and ad hoc environments," Context, Tech. Rep., 2003.

[7] F. Siegemund, "A Context-Aware Communication Platform for Smart Objects," in *Proceedings of the 2nd International Conference on Pervasive Computing (PERVASIVE)*, ser. LNCS, A. Ferscha and F. Mattern, Eds., no. 3001. Springer-Verlag, Apr. 2004, pp. 69 – 86.

[8] P. Korpipää, J. Mäntyjärvi, J. Kela, H. Keränen, and E.-J. Malm, "Managing Context Information in Mobile Devices," *Pervasive Computing*, vol. 2, no. 3, pp. 42 – 51, Jul. - Sep. 2003.

[9] A. K. Dey, G. D. Abowd, and D. Salber, "A conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications," *Human-Computer Interaction (HCI)*, vol. 16, no. 2,3&4, pp. 97 – 166, 2001.

[10] T. Hofer, W. Schwinger, M. Pichler, G. Leonhartsberger, and J. Altmann, "Context-Awareness on Mobile Devices - the Hydrogen Approach," in *Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS'03)*, 2002.

[11] OMG, "OMG FTF Convenience Doc: Platform Independent Model (PIM) and Platform Specific Model (PSM) for Super Distributed Objects (SDO) Specification," Apr. 2004.

[12] T. Berner-Lee, J. Hendler, and O. Lassila, "The Semantic Web," *Scientific American*, vol. 284, no. 5, pp. 34 – 43, 2001.

[13] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, *The Description Logic Handbook*. Cambridge University Press, 2003.

[14] W3C, "OWL Web Ontology Language Reference. W3C Recommendation," http://www.w3.org/ TR/owl-ref.