Electronic Communications of the EASST
Volume 11 (2008)

Proceedings of the First International DisCoTec Workshop on Context-aware Adaptation Mechanisms for Pervasive and Ubiquitous Services (CAMPUS 2008)

Share Whatever You Like

Sebastian Böhm, Johan Koolwaaij and Marko Luther

12 Pages

Guest Editors: Romain Rouvoy, Mauro Caporuscio, Michael Wagner
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Share Whatever You Like

Sebastian Böhm[1], Johan Koolwaaij[2] and Marko Luther[1]

[1]DoCoMo Euro-Labs, Munich, Germany
[2]Telematica Instituut, Enschede, The Netherlands

Abstract: To leverage proactive context-aware services for mobile handsets, an architecture for the management, aggregation and distribution of information is required. This work presents a framework that has been developed to realize an extensible infrastructure in which personal information can be shared with others while on the go. Access control mechanisms restrict the distribution of data based on social relationships and the validity of context conditions.

Keywords: Context management, distributed access control, mobile application, ontology reasoning, context aggregation

1 Introduction

The future of mobile services is meant to be context-aware. Context-aware services have long been seen as the natural advancement of location-based services, which were introduced more than five years ago. At last, today's mobile handsets are offering various features that place this progress within reach, being equipped with various sensors, sufficiently powerful computing resources and the possibility to manage all kinds of personal information. And yet, their most important asset may be their omnipresence. This circumstance quickens the interest in intelligent services, designed to adapt to the user's current situation. Here, a framework for context aggregation is needed to manage the distribution and combination of information from various sources.

Our context management framework (CMF) builds on initial work [1] accomplished within the IST-project MobiLife[1] [2]. Today, the CMF represents an open network of distributed and highly interconnected components to gather, aggregate and further exchange context information proactively. It enables the transformation of quantitative context information into qualitative statements about a user's given situation. *Share whatever you like* reflects some of the CMF's main requirements on context distribution and access control. Sharing personal memories and experiences clearly involves a number of privacy issues that need to be carefully considered from the start. The CMF architecture tackles these constraints without compromising its open and extensible nature. IYOUIT[2] is the reference implementation of the CMF and comes as a mobile client that allows users to share personal information while on the go and a Web-based portal to stay in touch with the online community. IYOUIT's mobile client runs on most Nokia Series 60 smartphones and succeeds its predecessor ContextWatcher [3][4] by adding authentication and privacy protection mechanisms as well as lifting its system architecture to a flexible, broker-centric one. The general aim is to make it easy for an end-user

[1] http://www.ist-mobilife.org
[2] http://www.iyouit.eu

to automatically record, store, and use context information, e.g. for personalization purposes, as input parameter to information services or simply to share this information with others.

The process of enriching gathered context data to qualitative information is exemplified in Section II. In Section III an overview of the underlying system architecture is provided and insights into several core components as well as the applied delegation-based authentication mechanism is given. Access control mechanisms are discussed in Section IV, focusing on the actual directive management, context filtering and the subsequent verification of the latter. IYOUIT is described in detail in Section V and related work in the field of context-aware systems, access control mechanisms and their applications is described in Section VI. Concluding remarks can be found in Section VII.

2 Context Aggregation

The main objective of the CMF is to allow for developing context-aware (mobile) applications that gather context data from various sources. Raw context data, e.g. sensor outputs like the currently visible cell ID on a mobile phone do often not provide meaningful (human understandable) information right away. Instead, the aggregation of such context data and the subsequent combination with other pieces of information is regarded as an essential process to enable the development of intelligent applications that make sense of the user's surrounding. Figure 1 illustrates a simplified context aggregation to resolve a user's current location and to reason about his location traces recorded over time.

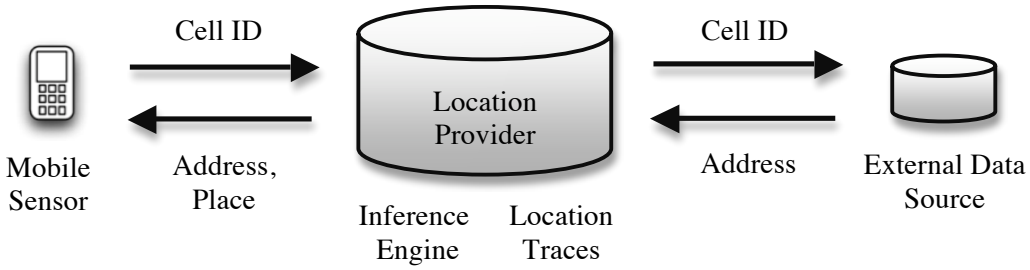

Figure 1: Simplified context aggregation

The cell ID obtained from the mobile phone is transmitted to the LocationProvider, a networked CMF component, to be resolved into an actual address record. It stores all recorded location traces and applies clustering techniques to identify frequently visited places of the user, e.g. his home or office [5], while remaining independent of any concrete mobile network infrastructure. This simplified example already implies the need for a context management infrastructure to also facilitate authentication, access control, easy extensibility as well as the combination of and the reasoning about different context information.

3 System Architecture

The CMF has been designed as an open, distributed and extensible framework to allow other parties to join the network and to enrich existing services for mobile handsets. Only a minimal set of components is regarded as being owned by the network operator and may have access to the users' credentials. Access control and authentication techniques allow for integrating

external partners or data sources without the threat of malicious data abuse. To do so, several essential requirements of the CMF were identified.

Instant revocation: The user as well as the network operator needs to have the ability to revoke access rights to context information at any time from certain components or distinct users.

Secure authorization and delegation: The user's credentials (his unique identifier and personal password) should not be propagated among uncontrolled components to prevent untrusted components from collecting personal information on behalf of the user without his explicit permission.

Abstraction and enrichment of context: Historic context information is stored within the respective component to allow for clustering and further data mining to abstract from lower level data, as exemplified in Figure 1.

Proactive requests: The user himself must not explicitly trigger the actual context retrieval. In addition, for answering a given context request, the number of components concerned cannot be predetermined. Computations on gathered context information to gain more qualitative information may require several components to autonomously request context from other components.

In general, the CMF architecture can be conceptually categorized in several groups of components that serve a distinct purpose. ContextConsumers mainly retrieve context information from various sources, the ContextBroker and the ContextHarvester provide means of easy context lookup and context gathering, whereas ContextProviders (CP) aggregate contextual data to provide meaningful information for specific services. Three management components are major constituent parts of the CMF to allocate basic functionalities that all components within the framework rely on. All other components that can be subsumed in a group of 3rd party applications act as external data source and may also provide additional context visualization techniques.

A. Management Components

The IdentityManager maintains the list of all registered entities (components and users), represented by uniquely named principles (`appID` and `entityID`). Each context retrieval request within the CMF is authorized by this central component to bar single components from collecting and distributing highly sensitive data without explicit permission. Therefore we introduced an authentication scheme based on ticket verification, similar to the Kerberos approach [6], to allow 3rd party integration without the need to disclose the users' credentials to each component within the framework. This authentication scheme is described in more detail in Figure 3, by means of a common context retrieval request.

The RelationManager enables the representation and exchange of qualitative social data. Explicit facts about social relationships between users (e.g. family members, friends or colleagues) are combined with the world knowledge encoded in an ontological model in order to apply logic based reasoning. These reasoning mechanisms ensure the consistency of the social data and allow for knowledge discovery techniques to complement the social network. As a result, groups of users can be formed dynamically, based on qualitative relationships managed by the RelationManager, further discussed in [7]. Explicit or implicit social connections are made available for other CPs to further enrich their context information. Apart from a subject and object entity as well as the actual relation predicate, each relationship definition has additional attributes. The status of a relationship, for instance, describes if a

relation has been *requested* (the initial value for all relations suggested by the user), *approved* (which means the object entity has already confirmed) or *inferred*, indicating that this relation has been deduced. The complete set of relationships, including deduced relations, is stored within a database to reduce query response times. Since the majority of requests are concerned with retrieving information rather than modifying data, the underlying knowledge base does not need to be involved. However, in case the set of approved relations has changed, either by adding a new relation or removing an existing one, all active relationships are transmitted to the inference engine to trigger the reasoning process. To model these qualitative social relationships we developed an ontology, specifically designed as a social knowledge base for the CMF. In contrast to most social networking services that only differentiate between buddy – or friendships, the social ontology with its 50 defined relationships allows for very detailed descriptions of all kinds of social connections between the users of the framework.

For providing personalized (mobile) services and applications, access control is always an issue that needs to be considered. The PrivacyManager (PM) enables the definition of relation and context-based access control directives to ease the process of managing ones personal directives. Our approach ensures both, the users' demand for privacy control and ease of use. Therefore directives do not have to be bound to concrete implementations or services in the framework but to abstract, general concepts of context. The user thus specifies a certain level of detail to be revealed per context category, without knowing which components are concerned with the actual execution and filtering of data. Social relationships allow for defining simple group access directives to restrict the distribution of context information. For instance, the user may specify that colleagues should only know the current city, whereas family members are granted access to the detailed set of location information, e.g. up to the street level. Further details on the proposed access control management can be found in Section IV.

B. Context-Broker, -Harvester, -Consumer and -Provider

The ContextBroker and the ContextHarvester provide access to a repository of system elements on the level of CPs (in the case of the ContextBroker) or on the level of context elements (in case of the ContextHarvester). The ContextBroker implements a registration and lookup service to enable the discovery of various CPs, based on the schema provided within the CP advertisement. The ContextHarvester can be seen as a specialized CP that collects context information across a range of other CPs. In supporting both, collections of context types and collections of entities, it can be used to obtain heterogeneous context information for various users or it may be used to obtain all personal information of a specific entity no matter where the information is stored or managed physically.

The majority of components within the framework surely belong to the category of CPs. At the time of writing more than ten CPs have been developed to provide services like local weather forecasts, picture sharing facilities or sound recording, to name only a few. In principle, a CP first gathers a certain type of information (e.g. the user's cell-id) from a sensor (for instance the mobile handset) or another CP to further process and enrich this information. This procedure of context aggregation is taking place within each CP and is essentially required for the subsequent (logical) combination of different context elements that, as a whole, describe the situation of an entity [8].

C. Basic Datatypes

Context aggregation and the later context combination across components is one important utilization of the CMF. Therefore numerous datatypes have been defined to leverage the exchange and assessment of context data within the framework.

Context Element: A context element is an elementary piece of context information that encapsulates the information obtained from a CP. A context element must at least include the CP's unique ID, an identifier of the owner of the respective piece of information as well as a number of parameters that hold the actual data values. Figure 2 shows a context element as rendered in XML.

```
<contextElement>
 <contextProviderAdvertisement id="1092"/>
  <entity id="1079" name="MaKoMo"/>
  <observer id="1083" name="Basti"/>
  <param name="weather" time="2008-01-20T14:37">
   <param name="temp" ontRef="#very_warm">
    <param name="tmin" value="3" accuracy="90"/>
    <param name="tmax" value="9" accuracy="90"/>
   </param>
   <param name="location" id="4468" accuracy="65">
    <param name="city" value="Munich"/>
    <param name="country" value="Germany"/>
   </param>
  </param>
</contextElement>
```

Figure 2: Context element

Parameter: A parameter object contains a name, a timestamp, a corresponding value and an extensible list of attributes further categorizing the parameter value (including the ontology reference and accuracy).

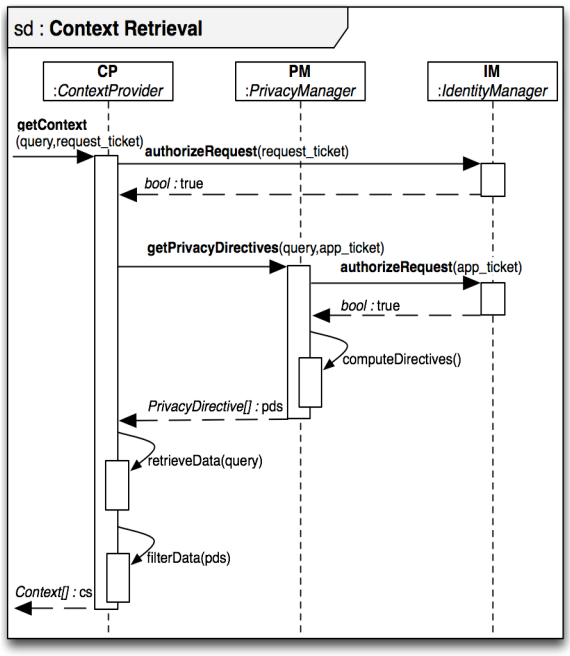

Figure 3: Context retrieval sequence diagram

Context Query: A context query request allows for retrieving distinct context elements that comply with certain constraints on the contents of context elements. The context query object contains a filter specified by simple and complex conditions. Simple conditions specify a comparison operator on values and attributes to select distinct parameters of interest, while complex conditions combine simple conditions recursively via logical operators. Applying a projection or summary operation further restricts the result set.

ContextProvider Advertisement: The advertisement includes a CP's unique id and a base URL to this service. Furthermore, it provides the data scheme in the form of a parameter hierarchy and a list of all featuring entities as well as options. Each parameter has been assigned a certain privacy level that indicates the sensitivity of the data and additional attributes such as the type or the unit of measurement. Since each CP may also provide software packages for the mobile client to make use of its public interface, the clientSoftware object indicates the availability of such software components. As soon as the mobile client has been identified and all constraints on downloading the package are fulfilled (e.g. the client's platform or version number), the client software can be delivered or updated.

D. Authentication and Component Interaction

To realize an open, distributed infrastructure and to allow 3rd parties to provide their own applications on top of the CMF's core infrastructure, a corresponding authentication mechanism needs to be in place. Two assumptions need to be considered that characterize some basic principles of how context is distributed and exchanged in a standard request-reply pattern, protected by common encryption techniques like https. All components that are not regarded as being controlled by the network operator (and thus entirely comply to the specification) are trusted in the sense that mandatory authentication and privacy mechanisms need to be implemented. Components that violate these constraints, be it on purpose or due to implementation flaws, are regarded as un-trusted. Furthermore, all communication paths need to be secure so that no request or reply messages could be intercepted and decrypted. In principle, two different types of authentication methods are applied within the CMF: user and application authentication. The request ticket that needs to be issued together with the actual context query is of the following form: `<entityID,appID,entityToken,appToken>`.

It includes an `entityID`, the requesting `appID` as well as two tokens. The `entityID` and the `entityToken` are used for the user authentication to ensure that applications retrieving context on behalf of the user are only granted access to those pieces of information the user is allowed to see. This means that for each entity and each application (aka CP), a corresponding `entityToken` exists. This token is issued by the IdentityManager (in case valid application credentials are given) and needs to be verified on each request as illustrated in Figure 3. Besides, the application authentication controls inter-component communication and requires a valid `appID` and `appToken`. So, for each combination of ContextConsumer and CP, a corresponding `appToken` exists, again issued and verified by the IM. All tokens are valid as long as either the user or the network operator revokes former access rights to any context retrieval on behalf of the user or disallows any component interaction. Instant revocation is given, since all context requests are explicitly authorized by the IdentityManager. Once the provided request ticket has been verified, the CP retrieves all appropriate access control directives from the PrivacyManager.

4 Access Control

The data flow in-between services or components allows for much richer services that build up on the knowledge made available. However, the network operator needs to be able to control the data flow in order to prevent others from unchecked data storage and general data abuse. In addition, the users themselves should be able to mark certain applications as trusted or disallow access to personal information for unknown or unwished service providers as well as other users. The proposed access control mechanisms are based on the social relationships managed by the RelationManager and the subsequent logical combination with other context data. In principle, three types of access control directives can be identified. Directives can either be defined for concrete users, groups of users (based on explicit or implicit social relationships) or certain context conditions. So besides role-based access control directives that build upon a social network of users and the underlying ontological model, context-dependent directives allow for directives that are bound to concrete context conditions (abstract descriptions of the user's surrounding). These conditions can be assigned to certain access control directives in order to restrict the validity of directives to concrete situations. Context-dependent access control directives are triggered in case certain conditions with respect to the

current context of a user are fulfilled. For instance, access to a business schedule may only be granted to colleagues that are nearby, but only during office hours. In this case, context information is actually the prime condition for deciding about access to, again, context information. To further refine the type of information that is made available, we differentiate between various granularities of context data with respect to their up-to-dateness. This means that a user could forbid access to the last known context state (e.g. the current location), but grants access to older or "out-dated" and therefore less critical information. The other way round, granting access to only the latest information, and therefore concealing the context history, prevents others from exploring issued data to analyze certain user behaviors.

The privacy decision point is realized in distributed components – the PrivacyManager (PM) and all concerned CPs. The actual filtering of context information according to the directives (specified by the user) has to be accomplished by each CP. Policy compliance according to the given access control directives is warranted because each component that processes a user's context information must be granted access by the user and authorized by the IdentityManager. Hence, the filtering of context information (see Figure 3) is accomplished by the respective component that knows the structure and the semantics of the data. Even though it would be in principle possible to establish a policy enforcement mechanism for access control in a thoroughly trusted environment, this was never our original intention. Instead, the CMF has been designed as a distributed, extensible framework that does provide certain means to verify the applied context filtering.

A. Directive Management

Access control directives are defined by concrete context specifications, a subject entity, an object entity or a certain relation predicate and the actual privacy level. The privacy level defines to what extend a certain type of context will be made available. A seven-staged granularity allows for fine-grained access control directives, ranging from no access to full access. The context specification contains a unique identifier, a CP's base URL, the parameter path, an entity's ID and an ontology reference (optional). However, the user does not have to provide a specific CP ID or parameter path as part of the request. General user-defined directives are automatically translated into concrete context specifications. To give an example, adding an access control directive to prevent disclosing one's current whereabouts, the PM first retrieves all appropriate CP advertisements from the ContextBroker that provide at least some spatial context information and computes all concrete parameter paths for the appropriate CPs.

The usage of context dependencies is restricted to non-dependent context data, which means that the validation of this condition must be directly answered by the corresponding CP. To do so, the PM gathers all context info exclusively via a specified interface (getContext()) of the appropriate CP since all context dependencies are modeled as context queries. Therefore, each CP has to implement this interface so that no additional requests to other CPs are needed to compute the last known state. In principle, this would result in a loop, since each CP would, in turn, have to consult the PM again. To break the resulting loop, a CP is able to detect a request that has been issued by the PM by its unique `appID` and the corresponding `appToken`. As highlighted in Figure 3, the normal authentication scheme requires two distinct requests to the IdentityManager. The first check authenticates the original request from a generic ContextConsumer to the CP, the second time the request from the PM to the CP is verified. In order to prevent the second authentication request, the IdentityManager automatically verifies

if the CP may also have access to the PM and eventually computes a secret token. This token is then passed to the PM to be verified in order to retrieve the appropriate access control directives.

Since a service infrastructure like the CMF undergoes frequent updates in terms of new services being introduced or existing services being modified, all existing access control directives must be adjusted dynamically. Once the ContextBroker has registered an update, the PM automatically verifies established directives and applies the corresponding modifications. In case a specific access control directive has been requested that has not been explicitly defined by the user, a more general directive will be returned that covers the given constraints. Similarly, requesting a directive for a specific entity might result in a directive that has been defined for a certain relationship, if the requested entity is the role-filler of that relationship. Also, the hierarchy of social relationships is considered, as represented within the social ontology. So, if no specific directive is given for the wife relationship, more general directives defined for all family members are returned.

B. User Interaction

The PrivacyManager is regarded as a controlled component, which means that only the user may modify access control directives. Applications (e.g. other CPs) may only retrieve directives that are concerned with their verified appID. To support the user in managing personal directives, a management interface has been developed and integrated into IYOUIT's Internet portal.

Web-based User Interface: This interface integrates most access control and social networking features into one compact and easy to use matrix-like interface, shown in Figure 4.

Figure 4: PrivacyManager Web-interface

Its principle design has been inspired by Almer's work on access control visualization techniques [9] during the MobiLife project. In general, two main views on access control directives can be distinguished. The BuddyView lists all approved contacts on the horizontal axis and the main context types on the vertical axis. Similarly, the RelationView allows for defining group directives and therefore shows all supported relationships in the horizontal menu. In case a certain context type on the left hand side is highlighted, more specific directives exist that can be revealed by clicking on the respective parameter, as shown in Figure 4 for the context type *weather* and its sub-parameter *weather/location*.

Each main category of context information has its own symbol, whose actual size indicates the respective context level assigned and therefore the amount of data revealed. So, defining access control directives for the most common context types is a matter of clicking through the respective icon sizes with hardly any effort. The controls on the right hand side provide filter

for the current view, e.g. to show only directives for a certain access level, to add distinct sub-categories of interest and to save or discard changes. Since access control directives can potentially overlap, the PM has to compute the most specific directive. For instance, having defined directives for all friends but also for one of your friends in particular would result in at least two overlapping directives (in this example the directive that has been assigned to this particular person would hold). Within the PM web interface, all directives that have been derived from other facts are shown in grayscale, whereas colored icons represent explicit definitions for concrete buddies.

Context Mirror: With the so-called context mirror, the user is able to verify the compliance with the given access control directives by inspecting (his own) personal data through the eye's of someone else. This way, directives can be revised and access to context can be withdrawn from applications that do not apply the filtering of data according to the directives. The context mirror is reflected in a Boolean value in the functional interface of a CP and an observer's entity ID, found in all context elements. Whenever the context mirror is enabled, the corresponding access control directives are retrieved from the PM and subsequently applied to the user's own context within the respective CP. The application of the context mirror within IYOUIT is described in the next Section.

5 Mobile Client Application

IYOUIT is a research prototype that represents our prime implementation of the CMF to apply context-aware technologies and methodologies in practice (including ontology-based reasoning and access control techniques). In short, IYOUIT facilitates the following usage scenarios:

Figure 5: IYOUIT's buddy tab

Real time context sharing for the exchange of qualitative information to keep track of friends and family members in an unobtrusive manner.

Contextual tagging of user generated media to describe the current context, to add automatically generated titles as well as descriptions to pictures, maps and sounds.

Storage and simplified retrieval of context data. Different views and context visualization techniques assist the user in finding useful information.

IYOUIT is a tab-based application, in which each tab either displays a certain type of context information or accumulates various pieces of information in a context overview. The *Me*-tab is IYOUIT's standard entry point and highlights all recently collected information. Clicking on one of the entries, a more detailed view is provided within the respective tab. The *Buddy* tab is the central place for all real-time context

information of approved buddies, including their current whereabouts, latest activities, shared photos and so on. Different views on context, various sorting orders and context-dependent ad-hoc groups help finding interesting or newly gathered information, as illustrated in the upper half of Figure 5. So besides grouping buddies with similar context (e.g. the same city or weather condition), the *Friends of friends* group lists all IYOUIT users that have been deduced as a rather close contact without being explicitly added by the user. This is a direct result of the applied social reasoning by the RelationManager, to support the user in completing his personal network of social contacts. Also part of the Buddy tab, the context mirror functionality has been implemented to verify the context filtering that has been applied in the respective CPs. Once the context mirror has been enabled (see lower screenshots in Figure 5), different views on your own context are displayed to reflect the way buddies see one's personal data. In general, IYOUIT is capable of sensing, aggregating, combining and distributing various types of context. Amongst others, this includes the user's location, photos, local weather data, nearby buddies or devices as well as personal experiences. Here, especially the way spatial data is processed can be taken as an example for the context aggregation that is taking place in the CMF. The location estimation can either be based on raw GPS coordinates, currently visible cell IDs or triangulation. The final step in the process of spatial aggregation is the application of a location-clustering algorithm to automatically detect frequently visited places [5]. Those places can be labeled as "Home" or "Office" and linked to a concept within the spatial ontology. Those places of interest have a qualitative meaning that allows others to easily classify this information.

6 Related Work

Several computing infrastructures for managing context have been proposed to support the rapid development of context-aware applications. All approaches generally provide a suitable abstraction mechanism to separate the process of context gathering and distribution from the core application logic. The Context Toolkit (CT) [10] is a prominent server-based platform, which uses XML structures to represent contextual data and to hide sensor details. A query and notification interface provides a standardized access to the data. Context providers and interpreters store historic context information and serve as a context abstraction layer. Context harvesters combine context information as well as repositories of services and components that are currently available within the system. Even though, in principle similar to our approach, the CT system only provides basic access control mechanisms for privacy protection. Foreseen as one potential solution, context-dependent privacy rules are mentioned, without giving any more details. Likewise, CT's use of standard public-private key technology for authentication does not allow for instantly revoking access rights in combination with delegation. In contrast to the closed CT system, our target is to create an open context infrastructure as also proposed in [11] to securely integrate external components using delegated credentials. Therefore, usage control [12] generally seems to conflict with our main aim to allow for further processing of context within trusted, external components. The aggregation and the logical combination of context is one of the main principles of the CMF to realize context-aware services.

Another related approach in terms of shifting resource intensive context manipulations from the mobile device to a service-oriented infrastructure is realized in the Context Distribution Framework (CDF) [13]. Similar to our parameter structure, a hierarchical representation of data elements is proposed to enable quality of context annotations. Furthermore, the CDF facilitates ontology-based context representation and reasoning techniques to derive higher-

level data. However, this approach implicates severe scalability issues for expressive ontology languages if applied as the main representation format [14]. Our attempt in making use of ontology technologies is considerably different. Distinct higher-level data elements are annotated with ontology references, making them available for further ontology reasoning. This way, the overall scalability is not affected, while at the same time valuable reasoning results can be achieved. Besides, the CDF does not address suitable mechanisms for privacy protection in its current version. The question whether or not the CDF can be regarded as a closed or extensible service infrastructure remains unclear, since no details on authentication are given.

A context-driven evaluation of policies that describe the behavior of agents in a pervasive computing environment is described in [15]. Similar to our context-dependent access control directives, context changes trigger the actual evaluation process of agent permissions. All policies are formalized in the Web Ontology Language (OWL) and consist of a context condition and a corresponding action. These policies do not allow for the specification of a subject, since policies are meant to be associated with context rather than subjects. Even though this assumption is legitimate in a pervasive environment where interacting components are unknown, our application area is different in this respect. Being able to identify components and users within the CMF is an axiomatic prerequisite. Therefore, we consider context in addition to subjects as attributes within our access control directives.

In [16], the authors focus on the exchange of spatial information and emphasize the distributed nature of access control when multiple context sources with different granularities are present. Access to location information is based on a concept called service trust. Trusted components may receive information only in case an authorized request has been forwarded and are thus required to sign all data returned to realize non-repudiation. However, it remains unclear how misbehaving services can actually be identified by an entity in practice.

Most context-aware mobile applications available today have a strong focus on uploading photos, including ZoneTag [17], Merkitys-Meaning[3] or Shozu [18]. Categorizing photos and other media content on the mobile handset is typically achieved by letting the user tag those items. Tag sharing or tag suggestions (as offered by Zonetag) simplify this otherwise rather time-consuming and solely manual process. However, in terms of automatically adding tags, titles and descriptions to any user-generated content (including photos) based on the current context (not just the user's location), IYOUIT is much more flexible and capable. Jaiku[4], which has recently been acquired by Google, is a mobile application with a similar approach compared to IYOUIT in sharing presence information with others. Through its connection to Twitter, a popular location-based micro blogging site, Jaiku became one of the most prominent mobile clients. With IYOUIT and the underlying CMF, our approach is to allow users to share personal information without the need to enter data manually, but to automatically sense, gather and process this information.

7 Conclusion

Initial usage statistics recorded during the 9 months development time and experiences that we made with the former ContextWatcher community point out the interconnected and lively nature of our system. Up to now, our 35 test users sent more than 250.000 location update

[3] http://meaning.3xi.org
[4] http://www.jaiku.com

requests (on average one every 15 mins), took 2300 photos, visited over 2000 cities in 23 countries and updated their local weather forecast about 3500 times. Based on those initial trials, the implementation of the CMF specification proved feasible in terms of scalability and practicability for managing context in a distributed architecture. More profound performance evaluations will follow as soon as a critical mass of users has been reached with the IYOUIT community. Focusing on access control and authentication to meet our users' obligations in terms of privacy protection without compromising the CMF's overall extensibility through the integration of 3rd party applications was deemed necessary but clearly entailed the complexity of the overall system architecture. However, shifting expensive tasks and complex computations to server components in the network made it possible to realize a lightweight mobile client on the one hand, and rich services centered on qualitative context information on the other hand. The aggregation and meaningful combination of context from various sources through lower-level clustering or higher-level reasoning techniques can only be accomplished with high-performance computing resources as provided by server components in the network layer. IYOUIT and the CMF architecture will be constantly enhanced with new components and services to underline IYOUIT's living test-bed character and to further verify research results with real context information. After all, sharing personal information in general and qualitative context information in particular is fun and helps you keeping track of your buddies. *Share whatever you like*, but only what you would like to share – with IYOUIT.

8 References

[1] P. Floréen, M. Przybilski, P. Nurmi, J. Koolwaaij, A. Tarlano, M. Wagner, M. Luther, F. Bataille, M. Boussard, B. Mrohs, and S. Lau, "Towards a context management framework for MobiLife," in *Mobile and Wireless Comm. Summit*, 2005.

[2] M. Klemettinen, ed., *Enabling Technologies for Mobile Services*. Chichester, England: Wiley & Sons Ltd., Sept. 2007.

[3] S. Böhm, M. Luther, J. Koolwaaij, and M. Wagner, "ContextWatcher – connecting to places, people, and the world," in *Demo Proc. of the 5th Int. Semantic Web Conference (ISWC'06)*, November 2006.

[4] J. Koolwaaij, A. Tarlano, M. Luther, P. Nurmi, B. Mrohs, A. Battestini, and R. Vaidya, "ContextWatcher – sharing context information in everyday life," in *Proc. of the Int. Conf. on Web Technologies, Applications, and Services (WTAS'06)*, 2006.

[5] P. Nurmi and J. Koolwaaij, "Identifying meaningful locations," in *Proc. of the 6th Int. Conf. on Pervasive Computing (Pervasive'08)*, pp. 1–8, IEEE Computer Society, July 2006.

[6] B. Neuman and T. Ts'o, "An authentication service for computer networks," *IEEE Comm.*, vol. 32, pp. 33–38, Sept. 1994.

[7] S. Böhm, M. Luther, and M. Wagner, "Smarter groups – reasoning on qualitative information from your desktop," in *Proc. of the Workshop on The Semantic Desktop*, pp. 276 –280, 2005.

[8] M. Luther, Y. Fukazawa, M. Wagner, and S. Kurakake, "Situational reasoning for task-oriented mobile service recommendation," *The Knowledge Engineering Review*, vol. 23, no. 1, 2008.

[9] D. Almer, "Mobilife – the privacy display widget," Master's thesis, Stockholm, Sweden, 2006.

[10] A. K. Dey and G. D. Abowd, "A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications," *HCI Journal*, vol. 16, no. 2-4, pp. 97–166, 2001.

[11] M. Wojciechowski and J. Xiong, "Towards an open context infrastructure," in *Proc. of the Workshop on Context Awareness for Proactive Systems (CAPS'06)*, pp. 125–136, 2006.

[12] R. Sandhu and J. Park, "Usage control: A vision for next generation access control," in *Proc. of the Workshop on Math. Methods, Models, and Architectures for Computer Network Security*, pp. 17–31, 2003.

[13] P. Pawar, A. T. van Halteren, and K. Sheikh, "Enabling context-aware computing for the nomadic mobile user: A service oriented and quality driven approach," in *IEEE Wireless Comm. & Networking Conf. (WCNC'07)*, pp. 2529–2534, 2007.

[14] T. Weithöner, T. Liebig, M. Luther, S. Böhm, F. W. von Henke, and O. Noppens, "Real-world reasoning with OWL," in *Proc. of the 4th European Semantic Web Conference (ESWC'07)*, pp. 296–310, 2007.

[15] R. Montanari, A. Toninelli, and J. M. Bradshaw, "Context-based security management for multi-agent systems," in *Proc. of the 2nd IEEE Sym. on Multi-Agent Security and Survivability (MAS&S'05)*, pp. 75–84, 2005.

[16] U. Hengartner and P. Steenkiste, "Implementing access control to people location information," in *Proc. of 9th ACM Sym. on Access Control Models and Technologies (SACMAT'04)*, pp. 11–20, 2004.

[17] S. Ahern, M. Davis, D. Eckles, S. King, M. Naaman, and M. Spasojevic, "ZoneTag: designing context-aware mobile media capture to increase participation," in *Proc. of the Pervasive Image Capture and Sharing Workshop (PICS'06)*, 2006.

[18] M. Ames and M. Naaman, "Why we tag: motivations for annotation in mobile and online media," in *Proc. Conf. on Human Factors in Computing Systems (CHI'07)*, pp. 971–980, 2007