Managing Advertising Context

Martin Strohbach, Martin Bauer, Miquel Martin, Benjamin Hebgen

NEC Europe Ltd, NEC Laboratories Europe, Kurfürsten-Anlage 36,
69115 Heidelberg, Germany
{strohbach, bauer, martin, hebgen}@neclab.eu

Abstract. This technological chapter provides an overview of how real-world knowledge and context information can be integrated in pervasive advertising applications. Often developers of such applications integrate sensing technologies directly into their application. This has two consequences. First, developers need to learn how to interface, use and manage potentially distributed sensors for each individual sensing technology. Second, applications are hard to evolve, e.g. when a better sensing technology is available, the application must be modified so that it can interface with the new technology. As a solution to this problem we describe a methodology that facilitates the integration of sensing technologies and provides an overview of context management tools that are suitable for pervasive advertising applications. The presented methodology is extracted from previous research in context-aware systems and our own research in pervasive advertising. For a case study based on our Context Management Framework (CMF) we describe an application using public digital displays. We illustrate how the methodology can be applied for more effective development of pervasive advertising applications.

1 The Challenge: Using Sensor and Context Information

It is an intrinsic property of pervasive advertising applications that they rely on knowledge about the physical environment in which they operate. For instance in research, public displays are often used to show personalized advertisements with the objective of making advertising more effective and show more relevant information to potential customers (Müller 2007). Such applications require sensors to detect the proximity of passers-by. Obtaining additional information, e.g. about the current activity of nearby people allows even further customization. Other examples include interactive display installations that engage potential customers with advertised products (Intel 2010) or contextualized advertisements on mobile phones (Nguyen 2010).

A common problem of such applications is the effective integration of appropriate sensing technologies in the application. Ideally an application developer should be able to simply specify what information is required and get some notifications about changes in the real world to which the application needs to adapt.

Caring about individual interfaces and specific technologies or solving distribution and communication problems are practical problems that have to be solved, but are typically of secondary importance to integrators and researchers that are interested in the application itself.

From previous research (Chen 2004, Dey 2001, Van Sinderen 2006) it is well known how common distributed middleware abstractions can effectively speed up development and provide a better degree of reusability. Nevertheless, pervasive advertising applications are often developed as vertical applications in which sensing technologies are directly integrated in the application. As a consequence developers need to learn how to interface, use and especially manage potentially distributed sensors for each individual sensing technology. Also, applications are hard to evolve, e.g. when a better sensing technology is available, the application must be modified so that it can interface with the new technology.

The objective of this chapter is to provide an overview of state-of-the art in context management and illustrate in a concrete case study how pervasive advertising applications can be developed more effectively by using this knowledge. This chapter is intended for

- **Practitioners in industry** that seek to integrate sensors in a cost effective way requiring a minimal level of expertise with the sensing technologies,
- **Researchers** that study pervasive advertising applications, e.g. usability, acceptance, privacy, novel ways of applications and interactions, and
- **Sensor Technology Providers,** i.e. researchers and businesses that develop sensors and sensing technologies and are interested in making their technologies available for easy use by pervasive advertising application developers.

Section 2 defines our terminology. In section 3 we describe a targeted advertisement scenario with public displays. Based on this scenario we describe challenges associated to managing context and provide an overview of suitable solutions in section 4. In section 5 we describe a methodology that guides developers in integrating context in their applications. We describe our own context management framework (CMF) in section 6, and in section 7 we present a case study in which we realize the application scenario. Finally, we discuss our design decisions in section 8 and conclude the chapter with a summary in section 9.

2 What is Context?

Our definition of context is based and closely related to Dey's definition that relates context to an entity such as people, places and things (Dey 2000). Furthermore, we use the term *context information* to explicitly relate to a concrete structured data set instead of the high level concept.

We consider this relationship to a real-world entity as an important property of context information: rather than describing raw sensor data and having knowledge

about the related entity implicitly in the sensor and application, context information carries the relationship to the entity in the data itself. This way, applications know what the data relates to without making potentially wrong assumptions. For instance, a context information instance directly embeds the information that the noise level is related to a particular shop.

Context is typically related to information obtained from sensors that often provide potentially fast changing information about the physical world. In general we consider context information to be independent of the nature and usage of the data. For instance, static user profile data can also be considered as context information. When integrating context information in the application, it is however necessary to examine its origin, nature and use in more detail. This analysis and realization for pervasive advertising applications is the purpose of this chapter.

3 Guiding Scenario: Context-aware Digital Signage

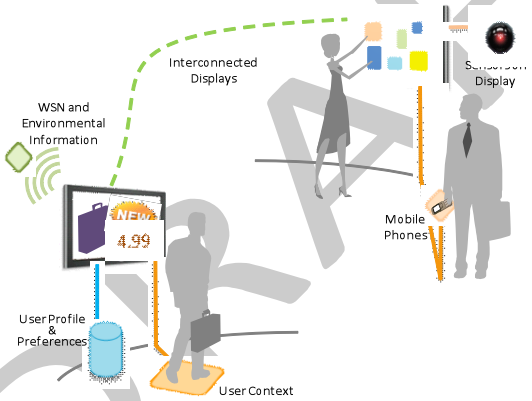

Fig. 1: Context-aware Digital Signage Scenario using various context information sources

Based on our own experience (Martin 2010, Strohbach 2009) with building pervasive advertising applications prototypes, collaborations with stakeholders and previous academic work in this area (Mueller 2007, Ribeiro 2010), we choose a Digital Signage scenario to illustrate the importance of context management. In Digital Signage electronic displays are used for communicating to a potential audience. In connection with real-world knowledge they can help advertisers to better communicate and engage (Michelis 2009) with potential customers, e.g. by better targeting and measuring the impact of delivering advertising content and offer interactive applications such as games.

The tools and methods described in this chapter can easily be transferred to other forms of advertising, e.g. using other devices such as mobile phones

(Nguyen 2010), or modalities such as sound (Beyer 2010) or even smell (CrunchGear 2008, Emsenhuber 2009).

For our scenario as depicted in Fig. 1, we assume that customers have opted in to the services described below and provide personal information such as preferences and privacy settings. We also assume that the services are able to learn preferences and detect situations in which content and services are most likely to be accepted both based on anonymous statistical data and on an individual basis, and by correlating other user data.

Displays in our scenario show targeted content depending on nearby individuals or groups of individuals. The content may be based on users' profile information, learned preferences, dynamic knowledge about users' current situation such as activity, environmental context of the display (weather, temperature, brightness, noise, etc.), or items on a public shopping list. The presentation of content is further influenced by application specific knowledge that provides information about the success rates of certain content for a specific situation, time and location (Müller 2007). For instance younger people may be more receptive to an interactive game than older people, or good weather may increase the likelihood of successfully advertising ice-cream, while rain may increase the likelihood for advertising umbrellas.

Content is augmented with QR codes, i.e. two dimensional bar codes, or Near Field Communication (NFC) tags (NFC Forum 2011) that allow customers to remember the content or access a secondary web page offering additional product information and services. Remembered content can later be accessed in an experience log that, among other functions, supports the purchase of the advertised product. Other services include buying the product on the spot, retrieving personalized discount coupons in the form of barcodes, and a navigation service that uses displays and mobile phones (Rukzio 2009) to navigate customers to a destination, e.g. the place where they can buy the product. Such navigation itself may be weather dependent, e.g. to guide users in such a way that rain is avoided. The interaction with the display and later actions, e.g. purchasing the product, provide valuable audience measurement data for advertisers.

In addition to normal content, displays also offer interactive, product-related games. Players at different locations can interact by touch with a nearby display showing a collection of mobile phone models. The display prompts for a phone model that both players must touch at the same time in order to increase their scores. At the end of the game players receive a – potentially score-related – discount coupon. The game performance is used as a form of audience measurement, i.e. for determining which models are identified fastest and can provide valuable feedback about customer's product knowledge.

4 Context Management

In the last decade, after Dey et al. raised an awareness on the importance of context middleware (Dey 2001), a myriad of systems have been developed that seek to simplify development of applications that rely on sensor and context information.

In this section we provide an overview of some recent approaches and relevant systems. In particular we point out that current research, as for instance carried out in the EU project SENSEI (SENSEI 2011), shows that providing sensor and context information from distributed, heterogeneous sources is still a challenging task. Further relevant systems are for instance surveyed by Kjær (Kjær 2007).

4.1 Challenges of Developing with Context

The scenario described in the previous section raises questions regarding the acquisition, access and management of context information. We consider the following aspects of key importance for effectively supporting application developers:

- **Discovery of context information,** related to people, places and objects, e.g. identifying the passers-by
- **Discovery of mobile context sources** such as a mobile phone sensor close to a display
- **Providing unified, high level abstractions** for accessing heterogeneous context sources providing static and dynamic information about people, places and things. These abstractions make it easier to adapt and extend the pervasive advertising application to different environments.
- **Appropriate handling of distributed context sources**, e.g. sensors associated with displays, mobile phones or wireless sensor networks deployed in the environment
- **Efficient monitoring of changing context conditions,** such as passers-by approaching or leaving the vicinity of a display or display environment in order to display the right content

4.2 Context Management and Access Approaches

There are different approaches for gathering, managing and accessing context. These approaches range from libraries offering APIs to toolkits providing building blocks, to complete infrastructures through which local as well as distributed context information can be accessed (Hong 2001).

Based on the challenges identified above, we provide an overview of frameworks and their characteristics that can in principle be applied to the domain of pervasive advertising (cf. Table 1).

Modern sensor APIs such as the Android Sensor API (Android 2011) or the Microsoft Sensor API (Microsoft 2011) typically provide low-level access to local sensors but also provide a certain level of abstraction to the developer. Both systems provide a *SensorManager* class which allows requesting a sensor by its type. This is possible via a common driver model which has to be implemented by device manufactures.

A *LocationManager* class which uses the *SensorManager* to get location information from the available location sensor is provided by both APIs. Depending on the requested quality of the information, the *LocationManager* uses a predefined type of location sensor or tries to get the best result. For the discovery of new hardware sensors, the Microsoft Sensor API can use the driver model and the device discovery of Windows.

The Xensor System is a research targeted context system (ter Hofte 2007). The main goal of Xensor is to allow researchers to acquire logged data about persons' behaviours without requiring them to attend a research lab. It provides clients for Windows Mobile 5.0 that gather sensor data and a logging repository to store the data. The Xensor client uses three types of *Xensor Modules* to gather context information that is provided as .NET objects to the developer. Module types include (1) *Experience Samplers* that are used to get direct input from a user, (2) *Usage Sensors* that are used to gather application usage data, and (3) *Context Sensors* that are used to gather information from the actual sensors of the device such as GPS, Bluetooth, etc. These modules are controlled by the *Xensor Engine* which loads and unloads modules dynamically on request.

Toolkits provide components that can be re-used when building applications. The Context Toolkit (Dey 2001) offers widgets, aggregators and interpreters as components. Widgets provide access to context information, hiding the low-level details of sensor access. Interpreters are used to derive higher level context information. Components can be distributed and re-used by multiple applications, but applications need to be statically configured to use them. Information is modelled as programming language objects, but there is no underlying semantic model like an ontology.

Context infrastructures are intended to be used by multiple client applications. They can be local or distributed and differ regarding their scalability. In the Context Broker Architecture (CoBrA) (Chen 2004) one broker maintains a context model for a certain space. This simplifies local reasoning as all information relevant to a certain space is available centrally, e.g. hosted by a powerful server. Clients can dynamically discover context information, but they need to find the right broker first and any processing or reasoning about information from different brokers needs to be handled by the client. Context information is modelled using the Web Ontology Language (OWL).

The Context Management Service (CMS) developed in the Awareness project (Van Sinderen 2006) is based on context brokers and context sources. The CMS is structured according to domains consisting of one context broker and multiple context sources. Context sources can be sensor-based, context reasoners, or context storage services. Context brokers can be peered through an inter context broker discovery protocol. The CMS assumes a context model with entities and context associated to these entities. As representations both an extended Presence Information Data Format (PIDF) and RDF are supported. Context sources support queries based on SQL and RDQL and subscriptions based on the Session Initiation Protocol (SIP).

Table 1: Overview of Approaches for Context Management and Access

Approach	Structure and Distribution	Discovery	Modelling abstractions	Processing and Inference
Microsoft Sensor API (Microsoft 2011)	Local API	type-based sensor discovery	C struct records	-
Android Sensor API	Local API	Type-based sensor discovery	Java Objects	-
Xensor (ter Hofte 2007)	Mobile clients synching to a repository	-	.NET objects	-
Context Toolkit (Dey 2001)	Distributed widgets	Static configuration	Programming language objects	Interpreters, aggregators
Context Management Service (CMS, Van Sinderen 2006)	Context domains with peered context brokers	Context broker as discovery service	UML or ontology model represented as extended PIDF or RDF	Context Reasoners: ontology reasoning and machine learning
IYOUIT (Böhm 2008)	mobile phone clients, coordinated by a central server	Static configuration	Ontology concept instances	Ontology reasoning and other classification algorithms
CoBrA (Chen 2004)	Multiple independent brokers	Broker directory, access through model kept by brokers	OWL ontology	Context Reasoning Engine in broker
C-CAST (Knappmeyer 2009)	P2P distributed brokers	Broker based discovery of context providers	entity-based context represented in XML	Context providers: aggregation, fusion, inference
Sensor Web (Botts 2007)	sensor access and management services	Supported by registry services	Sensor focused with semantic extensions	No specific support, can be provided as sensor
SENSEI (SENSEI 2011)	Distributed components for discovery and context access	Peered directories for resource discovery	Integrated two level ontology for sensor values and context entities	Resources providing aggregation, fusion and reasoning

IYOUIT (Böhm 2008) provides a social aspect to Context Management. It provides a client for Nokia S60 phones which collects information from the phone sensors (e.g. GPS), user input (e.g. mood) and derived data (e.g. weather at the location). The underlying context system is the CMS described in the previous section. The deployment, however, has a strong focus on semantically enriching information, making wide use of ontology reasoning and classification algorithms to ensure data consistency and infer new relationships among gathered information. For example, a day spent at home, work, a place of the type "restaurant", then work again and finally home, can be classified as a normal working day.

In C-CAST, a broker manages relationships between context providers and context consumers (Knappmeyer 2009). Aggregation, fusion and inference of context information is handled by context providers that have to manage their own inputs, possibly from other context providers. Clients can dynamically discover context providers through a broker and context information is modelled in ContextML. Context providers must support queries and may optionally support subscriptions.

The OGC Sensor Web Enablement (SWE) working group is defining a comprehensive set of web services for accessing sensor data (Botts 2007). SWE models sensors as processes providing geographically referenced observations and measurements. Sensors and their information can be discovered based on provided sensor descriptions rather than contextual information. Sensors can be freely distributed, but need to push their information to a Sensor Alert Service (SAS) in order to support subscriptions. The Sensor Web has been considered for use with pervasive advertising applications by Foerster et al. (Foerster 2009).

SENSEI (SENSEI 2011) integrates Wireless Sensor Networks into a sensor framework and provides a context framework on top of it. For clients, the Semantic Query Resolver (SQR) provides a single point of access for discovering resources as well as directly querying or subscribing to context information. The SQR includes a planner that can dynamically create processing trees based on the currently available sensor and processing resources, which are used for aggregating, fusing and inferring context information. Information is modelled as an integrated two level ontology representing sensor information as well as real world entity-based context information.

5 A Methodology for Developing Context-aware Advertising Applications

In this section we describe a simple methodology that helps researchers and practitioners to design and develop flexible context-aware advertising applications systematically and efficiently. The methodology presented in this section is generally applicable to context-aware applications and is extracted from the approaches described in the previous section and our experience in designing context-aware and

pervasive advertising applications. As depicted in Fig. 2 it is based on a simple system model that considers a pervasive advertising application as distributed application consisting of context consuming (sinks), producing (sources) and processing components (processors). A processor is both a sink and a source.

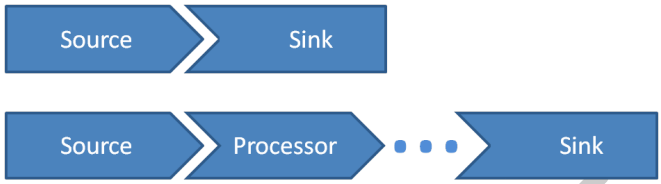

Fig. 2: Basic system model of a typical context management system. Sources produce context information, processor fuse or modify context and sinks consume information.

The methodology follows a top-down approach for developing the application and consists of the following questions that must be answered step-by-step:

- What are the context sinks in my application?
- What information does each sink consume?
- How is the information modelled and described?
- What information sources, i.e. sensors and services, are used?
- How are sensors and other information sources integrated?
- How can sinks access context information?

As illustrated in the previous section many choices exist in particular with respect to modelling and accessing the information as well as distributing and structuring the components. For the remainder of this chapter we use our own middleware for demonstrating how answering these questions helps to realize our scenario.

6 The Context Management Framework

We developed the Context Management Framework (CMF) with the purpose of simplifying the development of context-aware applications. The Context Management Framework originated from the MAGNET project (Nicolakopoulos 2009) and has been applied to many application areas – in particular to pervasive advertising and Digital Signage (Strohbach 2009).

As depicted in Fig. 3 the CMF consists of multiple *Context Agent*s. A collection of Context Agents is typically grouped in clusters and can further be organized in clusters of clusters. Each agent provides access to information stored locally and at connected agents. Agents also enforce policies that define which applications and other agents can access which entity/attribute pairs.

A *Context Agent* offers information to applications from any of the following three sources: sensors accessed by *retrievers*, persistent context available from the *storage* component and *processing units*. Indirectly applications are also context sources as they are able to modify information in the storage component. According to our system model in Fig. 2 processing units, storage components, and applications are also sinks.

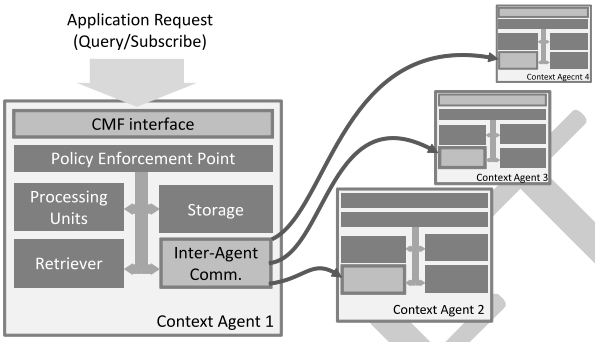

Fig. 3: High level architecture of the Context Management Framework consisting of coordinating Context Agents

Applications typically access the CMF using the *CMF Interface* on a local Context Agent, i.e. an agent deployed on the same computing node as the application. The CMF Interface offers access via a declarative *Context Access Language* (CALA) that is able to query distributed entity/attribute pairs. CALA provides query, subscribe, insert, delete, and update operations that operate on a cluster of *Context Agents*. Thus applications can access information in a fully declarative way and do not need to address individual agents directly. They are able to use a unified interface to access information generated by any sensor.

The CMF uses an entity/attribute based data model (see Fig. 4). Each piece of information is described as an entity that has an id and a type. An entity consists of one or multiple context attributes consisting of a name, type and value. Each attribute can be associated with metadata, e.g. for quality of information. In particular we use the metadata to annotate attributes with temporal information indicating in which time interval the attribute is valid.

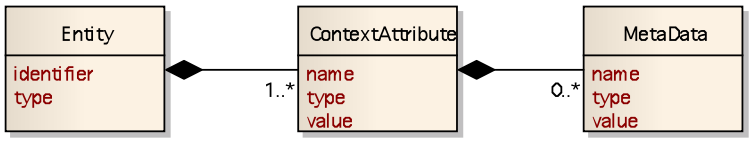

Fig. 4: CMF information model consisting of an Entity, ContextAttribute and MetaData (Bauer 2010)

The CMF is OSGi based (OSGi 2010) and runs on most devices that support a Java virtual machine, ranging from regular computers to embedded PCs – as used in professional Digital Signage displays – to Windows Mobile and Android mobile phones, home gateways and set top boxes. The programming interface is based on XML-RPC, and its main concepts are currently being standardized as part of the Context API in the Open Mobile Alliance (OMA) Next Generation Service Interface (NGSI) (Bauer 2010).

7 A Case Study: Developing with the CMF

In this section we apply the methodology described in section 5 to our Digital Signage scenario from section 3. We use our CMF middleware as a tool that supports researchers and practitioners in realizing their applications. Using the example of our guiding scenario we answer each of the questions presented in section 5 and use our CMF middleware as a tool to conceptualize and implement the scenario step by step.

7.1 What Are the Context Sinks?

Table 2 lists the identified sinks. The content adaptation component is responsible for showing the content based on the display context such as nearby users. Consequently it is realized as a CMF application.

Table 2: Sinks extracted from the guiding scenario and their realization with the CMF

Component Name	CMF Realization
Content Adaptation	Application
Preference Learning	Processing Unit
Preference Database	Storage
Secondary Content Website	Application
Navigation Service	Application
Mini Game	Application
Context Logging Service	Processing Unit
Context Log	Storage
Experience Log	Application

The preference learning component learns preferences about users and content. User preferences include preferred content for a user or user group in a given situation. In contrast, content preferences indicate the preferred presentation context (e.g. environmental context, time, location) of a certain content or content type. Such preferences can be learned based on previously recorded attention

times (how long has the audience looked at the content) or on actions resulting from watching the content (e.g. purchasing actions for advertisements). Learned preferences can thus be considered as context that is known before and constantly updated after the content is shown.

The preference learning component is realized as a processing unit that updates the preference database. The preference data base itself is realized with the storage component.

For a prototype realization selecting the CMF as preference database is the preferred choice as all information can uniformly be managed in one system using a single API. In a commercial deployment however, it is more likely that learned preferences are managed by an existing preference database external to the CMF and thus the preference learning component may be realized as application.

The secondary content website, navigation service and the mini game are all realized as applications using respective context information.

Finally, the context logging service records related context information, e.g. for audience measurement, later analysis and the experience log application. Similar to the preference learning service it is realized as processing unit that stores the result in the *Context Log* realized with the CMF storage component.

7.2 What Information Does Each Sink Consume?

The information consumed by each sink is listed in Table 3. The content adaptation sink component requires a considerable amount of context information in order to target the content to the current audience. Similarly, the preference learning component uses presentation context, audience measurement data (e.g. attention time, dwell time, interaction feedback, etc.) and other feedback in order to derive the user and content preferences. For instance the *opportunity to see (OTS)* is a common term to describe the number of passers-by that could have potentially looked at the content of the display (Spaeth 2008).

Table 3: Context Information consumed by each sink

Context Sink	Consumed Context Information
Content Adaptation	Identity/number of passers-by, User profiles and preferences, user activity, environmental context, shopping list items, content preferences
Preference Learning	Presentation context, audience measurement data, buying actions
Preference Database	User preferences
Secondary Content Webpage	Launch Trigger incl. Display
Navigation Service	User location, destination, display orientation
Mini Game	Touch event
Context Logging Service	All of the above, game performance
Context Log	All of the above, game performance

The secondary content webpage only requires a launch trigger, i.e. the information that should be displayed. In our scenario secondary information is being displayed on the mobile phone. In an extended scenario this information could however also be displayed on any other nearby display including the display on which the original advertisement is shown.

The navigation service requires information about the user's location, destination and display orientation in order to display the right directions.

7.3 How Is Information Modelled And Described?

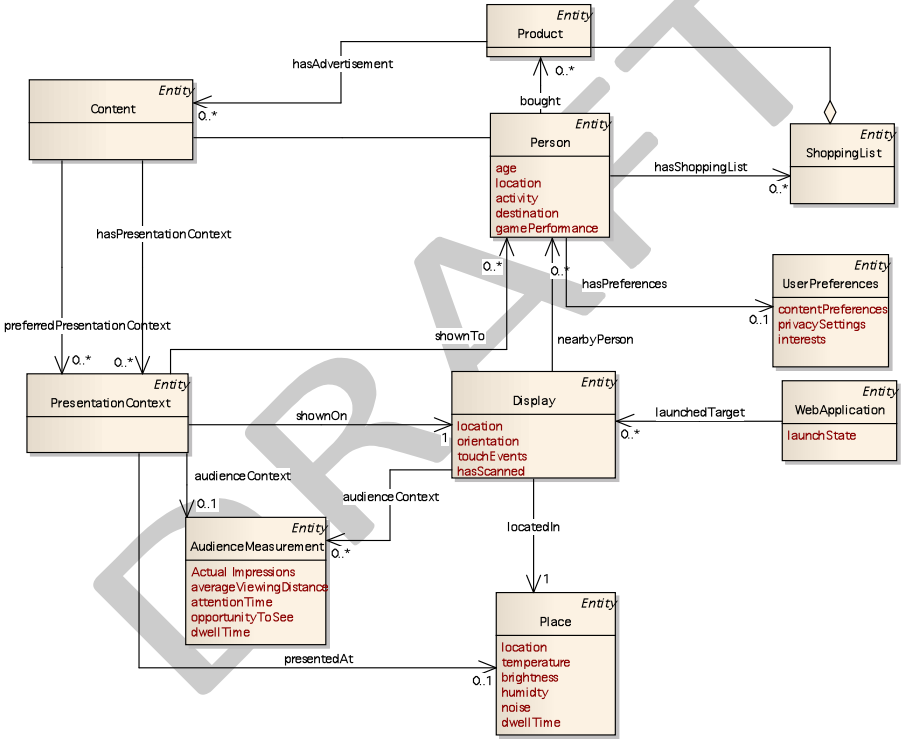

Fig. 5: An UML visualization of the domain ontology suitable for realizing the guiding scenario. Each class is a subclass of *Entity* and each attribute is an instance of *ContextAttribute*. Associations are also realized as *ContextAttributes*. The model is simplified only containing the most important entities and attributes. For instance it does not show basic user profile information.

Fig. 5 shows the domain ontology of the guiding scenario in form of a UML diagram. It captures the information produced and consumed by context sources and sinks as described in the previous section.

Audience Measurement information is captured in a separate entity and is linked to both the display and indirectly via *PresentationContext* to *Content* as at measurement time it is easier to relate audience measurement context statically to a display or a place rather than to dynamically changing content. This way it is easier to develop context sources as they do not rely on other context information. The relationship to content is inferred by processors such as the preference learning component or dedicated reasoners.

The relationships between *Product, ShoppingList, Person,* and *Content* further illustrate how the CMF can be used to capture, store and react to the context of a display. The shopping list contains a set of products that a user may eventually buy. The purchasing action is recorded in the *Person* entity. The *Product* entity is related to the content that advertised the product. The associated *PresentationContext* captures the context the advertisement was shown in. This way, user and content preferences (*preferredPresentationContext)* can be inferred and used to optimize the overall system behaviour.

Finally, we use a *WebApplication* entity to model our secondary web page. It uses the URL as identifier. The *launchState* attribute is inserted in the storage component to indicate that the application should be launched. This shows how the CMF can be used as distributed communication infrastructure.

7.4 What Context Sources Are Used?

Table 4: Context sources and their realization with the CMF.

Context Source	CMF Realization
User preferences	Storage, Application, Preference Learning Processor
Activity	Microsoft Outlook Calendar Retriever
Environmental context(humidity, temperature, brightness, noise)	SUN SPOT Retriever (SUN 2010)
Shopping list items	Application or Database Retriever
Age, gender, attention time, actual impressions, viewing distance, OTS, dwell time,	Camera Retriever
Launch Trigger	Barcode Retriever/NFC Retriever
Product purchased	Barcode Retriever, reward scheme Retreiver
Location	Bluetooth ping Retriever, barcode Retriever, NCF Retriever, GPS Retriever, Location System Retriever
Display orientation	Application, Location System Retriever
Touch event	Application
Mini game score	Mini Game Application
Context History	Context Logging Processor and Context Log Storage

Table 4 lists the context sources and their realization with the CMF. In order to fully implement the scenario, there must be a source for each CMF attribute. We only list the most important sources. Other sources are mostly CMF applications that provide static information, e.g. the location and orientation of the display or processors that infer new relationships, e.g. between the presentation context and audience measurement data.

We have only listed those sensors that we have integrated or are considering integrating in the CMF. Many other sensors could be used, e.g. body worn sensors for inferring user activities (Van Laerhoven 2009). Fig. 6 shows the resulting information flow between sources and sinks.

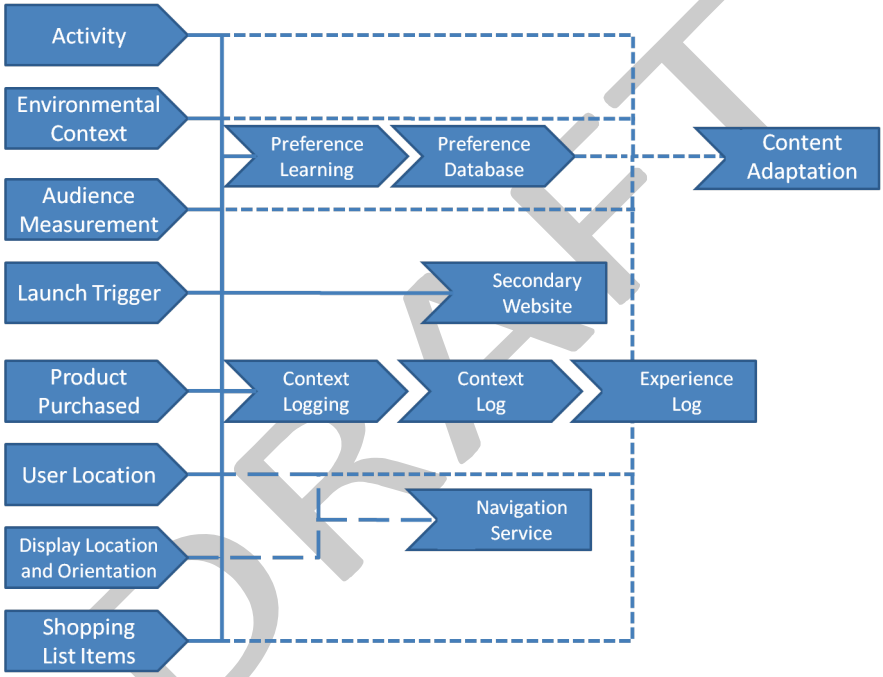

Fig. 6: Simplified view on the information flow between sources and sinks. Related information is summarized and some sources and sinks are left out for better readability. For instance the mini game application and related sources are not shown.

Shopping list items can in principle be provided by an application that uses the CMF as distributed database. If the shopping list application already exists it may be a better choice to access the shopping list by a retriever.

Most of the audience measurement data can be obtained by a camera attached to the display. Many commercial systems are available, e.g. the VISAPIX People Counter (VISAPIX 2010), Trumedia's audience measurement systems (Truemedia 2010), or the Field Analyst (NEC 2011). For obtaining opportunity to see meas-

urements and dwell time we implemented our own source using the OpenCV programming library (OpenCV 2010).

The launch trigger can be provided by an NFC or Barcode Retriever on a mobile phone. In our implementation (Strohbach 2009) we used the QR code reader QuickMark (Quickmark 2010) that is available for a vast range of mobile phones. While the above scenario does not require providing the launch trigger via the CMF, it is an easy way of extending the scenario with additional functionalities, e.g. launching the application on a different display than that of the device with which the barcode was scanned. The ZXing programming library (ZXing 2010) provides APIs for coding and decoding various barcodes that can be used as launch trigger retriever in our scenario.

The purchase of a product can best be monitored at the checkout. In our implementation we chose to integrate a retriever interfacing a commercial off-the-shelf barcode reader that is able to read personalized discount barcodes from a mobile phone. Such data collection requires the consent of users. Reward scheme infrastructures as offered by most larger European supermarkets could be used as alternative context source.

User location as required by the navigation service can easiest be obtained from any explicit interaction with the display, e.g. when scanning the barcode or NFC tag. In our own implementation we have used the scanning of the personal discount barcode to infer the position of the user.

For proactive content presentation it is necessary to obtain location information without explicit user interaction. Therefore we ping Bluetooth devices in a privacy preserving way (Strohbach 2009). For outdoor scenarios we have also integrated a GPS retriever for mobile phones. In indoor scenarios it would be necessary to equip customers with tags, e.g. integrated in loyalty cards or in trackable shopping carts (Krüger 2010, Metro 2011). More importantly it requires the deployment of usually expensive indoor positioning system (Ubisense 2010). If available, such a system can also be used to obtain dynamic orientation information. We have experimented with an ultrasound-based system (Zhao 2008) and integrated a retriever in our CMF.

7.5 How Are Sensors and Other Information Sources Integrated?

Sensor information or extracted features of a sensor are typically integrated into the CMF by implementing a *source* and a *mapper* that together constitute a *retriever*. A source accesses the actual data and provides its data set over a well defined interface to a mapper. A mapper then transforms data representations, e.g. extracting the actual data out of encoded Bluetooth information.

Sources and mappers are independent components that are wired in a configuration file. A configuration files defines mappings to entity/attribute pairs:

```
entityId=display1
```

```
entityType=Display
retrieverName=BtPingRetriever
attributeName=hasScanned
attributeType=MACAddress
sourceClass=eu.neclab.[...].BtPingSource
mapperClass=eu.[...].ToBluetoothAddressMapper
```

The mapper extracts the MAC address from the data set provided by the source. Additional retrievers can be provided with even less effort: for instance if one would be interested in the Bluetooth friendly name, one could define a mapper that creates a *BluetoothDevice* entity extracting the MAC address and the Bluetooth friendly name. In this case the configuration would not contain an *entityId* property but the mapper could be written to provide the MAC address as entity id.

Processing Units are used to fuse context information and use a CALA-based interface to access and provide context information. They provide context information using update and insert operations. For instance we use a processing unit that subscribes to MAC address sightings provided by the retriever above and provides the location of a user. It uses a simple mapping between display ids and location, and a mapping between MAC addresses and user Ids.

Applications provide context information simply by using insert, update and delete operations on entity/attribute pairs.

7.6 How Can Sinks Access Context Information?

With our Context Access Language, CALA, single one-time queries can be expressed or subscriptions on entity/attribute pairs can be defined. CALA provides similar concepts to the SQL query language, but has been deliberately designed with limited expressiveness so that it can easily be evaluated in a distributed way. For instance CALA does not support join operations.

CALA supports the following language constructs both for queries and subscriptions:

- **Selectors** – selects the entity and attributes to be returned. Entities can be selected either only by types or type and identity
- **Restrictions –** defines constraints on the returned attribute values
- **Scopes –** restricts the search space. In the current implementation, network scopes are supported that allow to specify whether only values available in the local CMF agent or a whole cluster should be returned

Subscriptions also allow the specification of a monitored attribute and a trigger condition defining when notifications should be sent to the sink. For instance one can specify that a notification should be sent every few milliseconds or whenever the attribute changes its value. The following example shows an XML representation of a CALA subscription used by the content adaptation component to access location information:

```
<Subscription>
    <EntityTypeSelector>
          <entityType>User</entityType>
          <attributeName>location</attributeName>
    </EntityTypeSelector>
    <OnChangeSubscriptionCondition>
          <attributeName>location</attributeName>
    </OnChangeSubscriptionCondition>
    <Scope>
          <networkScope>CLUSTER</networkScope>
    </Scope>
    <Options></Options>
</Subscription>
```

8 Discussion

In our case study we have made certain design and implementation decisions that impact the implementation effort, maintainability and extensibility of the realized scenario. These are in particular decisions with respect to

- Distributed vs. centralized context management,
- Semantic and declarative access vs. service-based access, and
- Contextualized information vs. low level sensor information.

We decided for a decentralized architecture that allows dynamic peering of context sources and sinks. An alternative approach could have been to use central servers that store all context information (cf. CoBrA (Chen 2004)). This approach works well for small scale installations and avoids the challenging task of distributed reasoning. But a major drawback is the required communication overhead as all information needs to be sent to the server, which may result in unacceptable latencies. Additionally, centralized approaches require a higher level of trust that context information is not misused.

Semantic and data driven context access as realized in CALA, enables the specification of *what* rather than *how* context information is accessed. This reduces the complexity for application developers as they do not need to care about which sources to access, nor use a directory service (cf. SensorWeb (Botts 2007)).

Our case study also shows that even a comparatively simple scenario involves a large number of sources, processors and sinks (cf. Fig 6). As the number of components increases it becomes increasingly hard to manage the relationship between the components. By its declarative approach the CMF takes away this burden both from application developers and developers of sources and processors.

Our data driven approach requires developers to carefully think about how they model the information. For instance, if we decided to configure a mapper to pro-

vide Bluetooth sightings directly as user location, we would not be able to derive other information easily. Likewise, application developers must be aware of what information is available in the system. We are considering providing an extensible ontology targeted at Digital Signage and Pervasive Advertising applications to reduce and simplify the data modelling effort.

By its entity/attribute model the CMF enforces contextualization of any provided data. This is in contrast to the sensor APIs described in section 4. The entity/attribute model forces developers to think about the semantics and as a result provides richer and more meaningful data.

9 Summary

In this chapter we have presented state-of-the art in context management and applied it to the specific problems that researchers and practitioners face when developing Pervasive Advertising applications. Based on a Digital Signage application scenario, we motivated the challenges faced by pervasive advertising applications that rely on context information. We have provided an overview of existing solutions for context management including our own framework, the CMF, and presented a methodology for developing context-aware advertising applications. We illustrated how the methodology can be followed by presenting a case study using the CMF. Finally we have discussed design choices in our case study.

Acknowledgments. The work described in this chapter has been partially funded by the SENSEI and Pervasive Display Networks (PDN) projects. SENSEI, contract number 215923 is partially funded by the European Union as part of the European 7th Framework Program. PDN is a joint project between NEC Europe Laboratories and NEC Display Solutions Europe GmbH.

References

1. Android Developers Site http://developer.android.com/ [accessed 12/01/2011]
2. Bauer M, Ito N, Kovacs E, Schülke A, Criminisi C, Goix L-W, Valla M (2010) The Context API in the OMA Next Generation Service Interface. In: ICIN 2010 Weaving Applications into the Network Fabric, Berlin
3. Beyer G, Meier M (2010) Interactive Advertising Jingles: Using Music Generation for Sound Branding. In: Proc. Pervasive Advertising and Shopping, Helsinki, Finland
4. Böhm S, Koolwaaij J, Luther M, Souville B, Wagner M, Wibbels M (2008) Introducing IYOUIT. In: Proc. International Semantic Web Conference (ISWC'08), October 27–29, Karlsruhe, Germany, vol. 5318 of LNCS, pp. 804–817, Springer Verlag
5. Botts M, Percivall G, Reed C, Davidson J (2007) OGC Sensor Web Enablement: Overview and High Level Architecture, available at http://www.opengeospatial.org/pressroom/papers

6. Chen H et al (2004) A Context Broker for Building Smart Meeting Rooms. In Proc. Knowl. Representation and Ontology for Autonomous Systems Symposium, AAAI Spring Symposium
7. CrunchGear (2008) NTT commercializes aroma-emitting Digital Signage system, http://www.crunchgear.com/2008/08/26/ntt-commercializes-aroma-emitting-digital-signage-system/ [accessed 12/01/2011]
8. Dey A K, Abowd G D (2000) Towards a Better Understanding of Context and Context-Awareness. In: Proc. CHI 2000 Workshop on the What, Who, Where, When, Why and How of Context-Awareness
9. Dey A K, Salber D, Abowd G D (2001) A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications, In Hum.-Computer Interact. (HCI) J., Volume 16 (2-4), pp. 97-166.
10. Emsenhuber B (2009) Scent Marketing: Subliminal Advertising Messages. In: Proc. 2nd Pervasive Advertising Workshop, Luebeck, Germany
11. Foerster T, Broering A, Jirka S, Mueller J (2009) Sensor Web and Geoprocessing Services for Pervasive Advertising. In: 2nd Pervasive Advertising Workshop, Lübeck, Germany
12. Hong J I, Landay J A (2001) An Infrastructure Approach to Context-Aware Computing. In Human Computer Interaction Volume 16, pp. 287-303, Lawrence Erlbaum Associates, Inc.
13. ter Hofte G H (2007) What's that Hot Thing in my Pocket? SocioXensor, a smartphone data collector. In: 3rd International Conference on e-Social Science
14. Intel (2010) Intel Unveils Digital Signage Concept, Intel News Release, available at http://www.intel.com/pressroom/archive/releases/2010/20100111corp.htm
15. Kjær K E (2007) A survey of context-aware middleware. In: SE'07 Proc. 25th Conference on IASTED International Multi-Conference: Software Engineering
16. Knappmeyer M, Baker N, Liaquat S, Tönjes R. (2009) A context provisioning framework to support pervasive and ubiquitous applications. In: 4th European Conference on Smart Sensing and Context, pp. 93-106, Springer, Berlin
17. Krüger A, Spassova L, Jung R (2010) Innovative Retail Laboratory - Investigating Future Shopping Technologies. In: it – Inf. Technology, vol. 52, nr. 2, Oldenbourg Verlag, pp. 114-118
18. Van Laerhoven K, Berlin E (2009) When Else Did This Happen? Efficient Subsequence Representation and Matching for Wearable Activity Data, In: Proc. 13th International Symposium on Wearable Computers (ISWC 2009), p.69-77
19. Martin M, Scott J (2010) NEC and Instoremedia present Sensor enhanced Digital Signage Solutions, http://www.youtube.com/watch?v=ibqBtF3PGwU [accessed 12/01/2011]
20. Metro Future Store Website, http://www.future-store.org/, [accessed 12/01/2011]
21. Michelis D, Send H (2009) Engaging Passers-by with Interactive Screens - A Marketing Perspective. In: Proc. 2nd Workshop of Pervasive Advertising, Lübeck 2009
22. Microsoft Sensor API, http://msdn.microsoft.com/en-us/library/dd318953(VS.85) [accessed 12/01/2011]
23. Müller J, Schlottmann A, Krüger A (2007) Self-optimizing Digital Signage Advertising. In: Adjunct Proc. Ubicomp 2007, Innsbruck
24. NEC Field Analyst Video, http://www.youtube.com/watch?v=eh5zL30iSA0 [accessed 21/01/2011]
25. NFC Forum, http://www.nfc-forum.org/home/ [accessed 12/01/2011]
26. Nguyen Q N, Hoang P. M. (2010) Push Delivery of Product Promotion Advertisements to Mobile Users. In: Proc. Pervasive Advertising and Shopping 2010 Workshop, Helsinki
27. Nikolakopoulos I G, Patrikakis C Z, Cimmino A, Bauer M., Olesen H. (2009) On the Personalization of Personal Networks - Service Provision Based on User Profiles. In: J. of Univers. Computer Science, Vol. 15, No. 12, pp. 2353-2372
28. Nishikawa H, Yamamoto S, Tamai M, Nishigaki K, Kitani T, Shibata N, Yasumoto K, Ito M (2006) UbiREAL: Realistic Smartspace Simulator for Systematic Testing. In: 8th Int'l Conf. on Ubiquitous Computing, pp. 459-476, Springer

29. OpenCV Website, http://opencv.willowgarage.com/ [accessed 06/11/2010]
30. OSGi Alliance, http://www.osgi.org, [accessed 10/06/2010]
31. Quickmark QR Reader web site, http://www.quickmark.com.tw [accessed 01/11/2010]
32. Ribeiro F R, José R (2010) Timely and keyword-based dynamic content selection for public displays, In: Int. Conf. On Complex, Intelligent and Software Intensive Systems
33. Rukzio E, Müller M, Hardy R (2009) Design, Implementation and Evaluation of a Novel Public Display for Pedestrian Navigation: The Rotating Compass. In: 37th International Conference on Human Factors in Computing Systems, pp. 113-122, ACM
34. SENSEI Consortium. The SENSEI Real World Internet Architecture. White Paper, available at http://www.sensei-project.eu/ [accessed 12/01/2011]
35. Van Sinderen M J, Van Halteren A T, Wegdam M, Meeuwissen H B, Eertink E H (2006) Supporting Context-aware Mobile Applications: an Infrastructure Approach. In: IEEE Communications Magazine, vol. 44, nr. 9, pp. 96-104
36. Spaeth J, Singer S, Hordeychuk S (2008) Audience Metric Guidelines, Digital Place-based advertising Association, available at http://www.dp-aa.org/guidelines.php
37. Steggles P, Gschwind S. (2005) The Ubisense Smart Space Platform, Advances in Pervasive Computing. In: Adj. Proc. 3rd International Conference on Pervasive Computing. Vol. 191
38. Strohbach M, Kovacs E, Martin M (2009) Towards Pervasively Adapting Display Networks. In: 1st Pervasive Advertising Workshop, Nara, Japan
39. SUN SPOT Sensor Nodes Web Site, http://www.sunspotworld.com/ [accessed 31/10/2010]
40. Trumedia Company Website, http://www.tru-media.com/ [accessed 31/10/2010]
41. Ubisense Website, http://www.ubisense.net [accessed 01/11/2010]
42. VISAPIX Company Website, http://www.visapix.com [accessed 01/11/2010]
43. Zhao J, Wang Y (2008) Autonomous Ultrasonic Indoor Tracking System, In: IEEE Int. Symposium on Parallel and Distributed Processing with Applications (ISPA08), pp.532-539
44. ZXing Website, http://code.google.com/p/zxing/ [accessed 01/11/2010]