An open Context Information Management Infrastructure
the IST-Amigo Project

Fano Ramparany\textsuperscript{1}, Remco Poortinga\textsuperscript{2}, Maja Stikic\textsuperscript{3}, Jörg Schmalenströer\textsuperscript{4}, Thorsten Prante\textsuperscript{3}

\textsuperscript{1}Orange Labs
\textsuperscript{2}Telematica Institute
\textsuperscript{3}Fraunhofer Institute
\textsuperscript{4}University of Paderborn
fano.ramparany@orange-ftgroup.com

Abstract

Amigo Context Management Service (CMS) is an open infrastructure for managing context information. The role of the CMS is to acquire information coming from various sources, such as physical sensors, user activities, applications in process or internet applications and to subsequently combine or abstract these pieces of information into "context information" to be provided to context aware services. This article introduces the basic principles behind underlying the CMS design and implementation and illustrates its application to implementing an ambient intelligence enabled home environment, supporting users in handling daily routine tasks such as the morning waking up. The most salient features of the CMS include:

- Its compliance to the web service architecture both for interfacing to context consuming applications and for integrating its sub components
- The modeling of context information using a high level language with much expressiveness
- State of the art context sources such as a context history manager and an audio based positioning system.

1. Introduction

Despite substantial progress in wireless communication technology and home networking, despite continuous fall in prices and miniaturization of computing hardware components, the promises of Pervasive Computing (PerComp for short) to improve people's lives are yet to be brought to life. One particularly ambitious approach and sub-discipline of PerComp is Ambient Intelligence (AmI for short). AmI objective is to adopt a fully user centric view of PerComp and to gear the physical environment of our homes towards our needs, activities and preferences without requiring our explicit attention.

Unfortunately, the complexity of our everyday device installation procedures, the lack of interoperability between different manufacturer's equipment and the lack of compelling user services has hampered the mass deployment of PerCom and AmI.

The Amigo project aims at overcoming these obstacles by developing open, standardized interoperable middleware and attractive user services that will improve end-user usability, attractiveness and experience.

One salient feature of AmI services is to be sensitive to context [7] either through adapting its modalities or behaviour to specific context information (such as the time in the day, the age of the user), through triggering specific actions in response to particular change of context, or simply through informing users about some context information that are specially relevant to achieve their current activities. For example, if my late visitor knew if my young daughter
were sleeping, he/she would know if she could knock the door softly or simply ring. Or even wait up until to morrow if he/she's been informed that all of us were already in bed. Such context information could be displayed at the doorstep, for example using a red lamp just like at the doorstep of a movie studio when a scene is being shot. Such services have been coined context-aware services because they could exploit context information for the benefit of users. In this paper, we present the Context Management Service (CMS for short) developed within the Amigo project. This service provides timely and relevant context information to context aware applications or services.

2. Scenario

We introduce here a short scenario of ordinary people engaged in an everyday life activity in an home environment augmented with ambient intelligence. This scenario will illustrate the benefit of making applications and services context aware. John wakes up every weekday at 06:30 AM. He usually goes to the kitchen to prepare the tea and toast some slices of bread, then he goes to the bathroom to have a quick wash while the tea is infusing. He sometimes departs from this routine plan. There are several reasons for this to happen:

- John is late because he went to bed very late the day before and prefers to stay some extra time in bed, or because his alarm-clock was unable to wake him up.
- John has taken a occasional day off (vacation or public holiday)

Using very simple sensors in the physical environment and deploying very simple services that takes these sensors information into account to control some devices that are involved in this routine activity can make John's life a lot easier, and relieve his cognitive load. As we've mentioned in the introduction, this is indeed the main objective of ambient intelligence. At this point we can already notice that some device mentioned in our description, such as the alarm clock, already integrate a sensor and a context aware service. For example, the alarm clock senses the time as it flows and triggers an alarm when this time reaches a deadline set by John. If we now take into consideration less common and nevertheless very simple sensors, from measures yield by a pressure or contact sensor placed in John bed we can detect if John is in the bed or not. From detecting the exact moment the bed shifts from "occupied" to "empty", we can infer that John is awake and leaving his bed. Placing a movement detection sensor at the door of John room will enable to infer that John is leaving his room. If we now compare the waking time set on the alarm-clock and the time is leaving his room we can know that he is going to engage in the routine morning plan.

We can note that with only a few set of very simple sensors, for instance, the alarm-clock, the bed contact sensor and the movement detection sensor we were able to discriminate between several situations, namely:

- John is sleeping
- John is waking up
- John will be taking is breakfast
- John is exiting his bedroom

If we are able to identify these situations, there are already few actions that we can anticipate John is likely to carry out. For example:

- If John has just waked up, he will probably exit his bedroom.
- If John has just waked up and is exiting his bedroom, he will probably go to the kitchen and prepare his breakfast

The point is that if we are able to identify some meaningful contexts, i.e. typical situations best described by the user's activities or states, we will be able to anticipate the user immediate actions and design the environment in such a way that it can either help him/her
carry out these actions or simply do these actions on the behalf of the user. Examples of actions that could be easily controlled by an instrumented environment include:
- Lighting the corridor in case it is dark
- Switching the tea pan or the toaster in case the ingredients have been prepared the day before

Enabling such context aware environment requires an appropriate infrastructure for acquiring information coming from sensors, for processing this information to encode it in a machine understandable format and for making it available to applications or services that can adapt their functionality or behaviour on the basis of this information.

In the following, we introduce a context management service (CMS) that has been developed in the framework of the Amigo project with exactly this goal in mind.

3. CMS architecture

The role of the CMS is to acquire information coming from various sources, such as physical sensors, user activities, applications in process or internet applications, and to subsequently combine or abstract these pieces of information into "context information". It then makes this information available to context aware services.

Our design principle is to provide a well-defined interface to context aware applications. This interface enables applications to query context information or subscribe to context change events. The main functional entities in the CMS architecture are depicted in the following figure:

The CMS architecture prescribes the encapsulation of physical sensors or any devices that could provide context information within a software layer that implements this interface. Pure data processing systems such as a context history management system or a dedicated context reasoning system that abstracts or aggregates context information could also implement this interface. Entities implementing this interface are called Context Sources because they constitute the primary source of context information. The CMS also uses a high level language for exchanging context information between context consumers and context sources, irrespective of the context's nature, be it user context, environmental context or device context. A comprehensive analysis of Ambient Intelligence applications has been conducted within the Amigo project, and has resulted into a context ontology which identifies the types of context information that is relevant to context aware applications and relationships among these types.
Context information is represented using the Resource Description Framework (RDF). RDF is a language that has originally been designed for representing information about resources in the World Wide Web[9]. In our modelling approach, a piece of context information is a RDF fragment which relates entities that are instances of concepts of the context ontology to other entities or to literate values (strings, numerical values or URIs). An very simple example of context information telling that "Jerry is located in the Kitchen" is represented graphically in RDF as:

While transmitting this information between context sources and context aware applications a textual serialized format based on XML is used. In RDF/XML notation, the same information will look like:

```xml
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns="http://www.owl-ontologies.com/unnamed.owl#"
    xml:base="http://www.owl-ontologies.com/unnamed.owl">
  <User rdf:ID="Jerry">  
    <locatedIn>  
      <Room rdf:ID="Kitchen"/>  
    </locatedIn>  
  </User>
</rdf:RDF>
```

The context broker is a directory service which is aware of context sources that are actually available. Based on this information the context broker is able to connect context consumers to context sources. Context sources register to the context broker by providing a description of their capabilities as soon as they are deployed or come into the environment.

An interesting function of the CMS makes it possible to store and exploit recorded histories of the users' interactions in context – context histories for short [5]. A specific component of the CMS manages context histories.

Context history data are stored in a central MySQL database. Logging of context history data is provided for full fledged PCs. Implementation of this component has been done in .NET C# and it is based on a client-server logging infrastructure with a SOAP-based protocol that copes with connection timeouts. The data are gathered from context sources that provide access to the context data defined in the context history data model. The context history component of the context management service acts as a context source itself. Therefore, access to the context history data is provided by a Web Service implementing the IContextSource interface.

Context information stored in the context history database is timestamped, where the timestamps are captured as a standard attribute. Each timestamp represents the time when context entry was last updated and is a prerequisite for inference mechanisms that should resulting an enhanced understanding of the users' activities and interactions as they expand and develop over time. Therefore, the context history data enables time-based reasoning and has a strong potential for supporting dynamic adaptations in ambient intelligence networked home environments.

The acoustic position estimation sensor (APE) is part of the acoustic scene analysis [1], which intends to collect all kinds of available information from acoustic signals. In contrast to the
diarization task of estimating “who spoke when”, in the acoustic scene analysis additional information like position or emotional state of the speaker are considered.

The APE sensor is based on microphones linearly arranged in microphone arrays. With each array it is possible to estimate the Direction-of-Arrival information (DoA), i.e. the angle towards the speaker. In [2] an adaptive Filter-and-Sum Beamformer (FSB) was presented whose coefficients are blindly adapted without DoA information. Hereby the filter coefficients form the principal component of the power spectral density matrix of the microphone signals. The FSB is intended to compensate the detrimental effects of reverberation and distance from speaker to microphones on the speech quality. Additionally the blind adaptation offers the opportunity to obtain the DoA information by calculating the cross-correlation of the FSB filter impulse responses. Position estimation in Cartesian coordinates is done by combining DoA information from multiple arrays. Tracking the position of a moving speaker can be significantly improved by particle filtering, as it is shown in [3].

Acoustic position estimation must be seen as a building block of a location management system, offering position information of different accuracy’s by integrating different sensing techniques. The accuracy of the APE sensor is approximately below 0.5m with an average adaptation time of 0.4s, depending on the reverberation of the room, the number of arrays and the amount of microphones per array. The speaker identity can be acquired from the enhanced audio signal of the beamformers.

Furthermore it is possible to use the position information to significantly improve other components of the acoustic scene analysis, e.g. speaker change detection [4]. As a context source for the Amigo system, the APE sensor uses a context wrapper to offer its contextual information to context consumers. After registering at the APE wrapper the context consumers are asynchronously notified about context changes.

4. Discussion

Compared to existing context management systems, the CMS design is unique in its adopting a Service Oriented Distributed approach. Context sources are designed as web services providing a standard context source interface. Compared to existing work, this Service Oriented Approach to designing a context management system is new and fits the dynamic nature of ambient intelligence environments or more generally pervasive computing environments, where objects and humans appears and disappears and interact in a loosely and ad-hoc fashion.

The integration of context histories in a context management service is a novel and promising approach. It will overcome the limitation often found with the current generation of context-aware applications that they only consider the present context, and neglect the history of context and its value in enabling to infer patterns from users' behavior [6]. These histories can be used to support applications with an enhanced understanding of the users' activities and interactions as they expand and develop over time, including their relationship to other interactions happening in parallel, before or afterwards.

For example, in the scenario introduced earlier, where John was involved in the routine tasks of the morning waking up, storing time stamped records of John's location over a number of days and month in the context histories database and analyzing the content of this database enables frequent temporal patterns to be identified such as the sequencing of moves John does every week days after he has waked up (get out of bed, exit the bedroom, enter the kitchen,...),
or situations/actions correlation to be detected such as John switching on the light when entering the corridor and when the corridor is dark.

The current CMS includes a set of ready to deploy and use context sources. More precisely, in addition to the audio positioning and context history manager, other locations sensors based on BlueTooth connectivity, active Radio Frequency tagging, a topic recognizer (keyword spotting using speech recognition technology), household appliances sensors have been implemented.

From the implementation standpoint, special care has been paid for ensuring the possibility to run any component of the CMS on a constrained execution platform. For instance, the Context Broker is implemented as a OSGi bundle and has a mere 20Kb memory footprint (current version). This makes it possible to run most of the CMS components on light devices such as a Java enabled PDA. The current version of the CMS has been deployed on Oscar [13], an open source implementation of the OSGi framework.

The most prominent advantages of the Amigo Solution compared to existing context management infrastructure (Salber et al [14], Khedr and Karmouch [15]) are its scalability and openness. Our approach promotes:
- Scalability through a distributed architecture
- Dynamicity as context sources can appear and disappear in the environment with minimal impact on services currently running.

The CMS owes its openness and interoperability from its relying on standard technologies for:
- Binding applications to context sources: applications interact with context sources using the SOAP [8] protocol which is a well established standard in the web service domain.
- Modeling context information: context information exchanged between context sources and context consumers are expressed in RDF [9] which is the standard adopted by the semantic web community for describing resources. In our approach, pieces of context information are considered as resources. The semantics of these resources is made explicit by relating them to an extensible context ontology expressed using OWL [10].

Furthermore the CMS is an easy to extend and deploy infrastructure as it is implemented on top of the OSGi and .NET component based framework. This component based approach made it reasonably easy to integrate the context history management component and to envisage context interpreters components.

5. Conclusion

The work presented in this paper is ongoing, but results achieved so far are well encouraging. The CMS is currently being used in the Amigo project by application developers to deploy context aware services. As application development will progress, CMS functionality will extend. Future extensions already planned include simple context sources (comfort sensors) as well as new context interpreters such as social context. During the forthcoming year, an extensive exploitation of the context history component and the use of the security framework developed in WP3 will be carried out.
Acknowledgements

This work was partially supported by the European Commission as part of the IST-IP Amigo project under contract IST-004182.

References

[8] SOAP Specifications: http://www.w3.org/TR/soap/