© Springer-Verlag Berlin Heidelberg 2008. This is a preprint version of the article.
-
Pederson T., Ardito C., Bottoni P., Costabile M. (2008). A General-purpose Context Modeling Architecture for Adaptive Mobile Services. In: I.-Y. Song et al. eds. ER 2008. (vol. LNCS 5232, pp. 208-217). Heidelberg: Springer-Verlag (Germany). ISBN: 978-3-540-87990-9.

A General-purpose Context Modeling Architecture for Adaptive Mobile Services

Thomas Pederson[1], Carmelo Ardito[1], Paolo Bottoni[2], Maria Francesca Costabile[1]

[1]Dipartimento di Informatica, Università degli Studi di Bari, 70125 Bari, Italy
{pederson, ardito, costabile}@di.uniba.it
[2]Dipartimento di Informatica, Università di Roma La Sapienza, Rome, Italy
bottoni@di.uniroma1.it

Abstract. Mobile context-aware computing aims at providing services that are optimally adapted to the situation in which a given human actor is. An open problem is that not all mobile services need contextual information at the same level of abstraction, or care for all aspects of the user's situation. It is therefore impossible to create a unique context model that is useful and valid for all possible mobile services. In this paper we present a compromise: a three-tiered context modeling architecture that offers high-level mobile services a certain freedom in choosing what contextual parameters they are interested in, and on what abstraction level. We believe the proposal offers context modeling power to a wide range of high-level mobile services, thus eliminating the need for each service to maintain complete context models (which would result in severe modeling redundancy if many services run in parallell). Each mobile service must only maintain those parts of the context model that are application-dependent and specific to the mobile service in question. We exemplify the use of the context model by discussing its application to a mobile learning system.

Keywords: Context Model, Context Awareness, Mobile Services, Human-Computer Interaction.

1 Introduction

Because of the very nature of mobile devices, human interaction with them is strongly related to their context of use. Recent applications for mobile use try to take advantage of contextual information to offer better services to users. Pervasive, or ubiquitous, computing [5] calls for the deployment of a wide variety of smart devices and sensors throughout our working and living spaces, which not only can offer a more "intelligent" local environmental behaviour but also provide important contextual cues to mobile devices operating in the environment. The overall goal with these infrastructures combining wearable and instrumented computation power is to provide users with immediate access to relevant information and to transparently support them in their current tasks. As Human-Computer Interaction (HCI) systems expand beyond the virtual environment presented on a computer screen and start to encompass also real-world objects and places, the need to better conceptualize these

new components of the system, as well as the intentions of the human agents currently operating the system, becomes pressing. Context-aware systems differ from traditional HCI systems not only because they utilize the state of the physical world as part of interaction, but also because they do it implicitly [4]. One might say that context-aware systems provide computational functionality directly or indirectly tied to real-world events without adding input devices but by gathering information in other ways (typically through sensors, of which the human is not necessarily aware).

The work presented in this paper is part of the CHAT project ("Cultural Heritage fruition & e-learning applications of new Advanced (multimodal) Technologies"), which aims at developing a general-purpose client-server infrastructure for multimodal situation-adaptive user assistance. In such architectures, dialogue management is typically based on the integration of independent components that execute specific tasks (sometimes, these components are "out-of-the-box", e.g. components for voice recognition). CHAT intends to develop and evaluate an architectural framework that facilitates implementation of such multimodal services.

In this paper, we use the term "context" as defined by Dey et al. [2]: "any information that can be used to characterize the situation of entities (i.e. whether a person, place or object) that are considered relevant to the interaction between a user and an application, including the user and the application themselves." An open problem is that not all mobile services need user-related contextual information at the same level of abstraction or care for all aspects of the user's situation. It is therefore impossible to create a unique context model that is useful and valid for all possible mobile services. In this paper we present a compromise: a three-tiered context modeling architecture that offers higher-level mobile services freedom in choosing what contextual parameters they are interested in, and on what abstraction level.

After a brief introduction to the Adaptive Dalogue Manager module, the rest of the paper focuses on the context model and its application to a mobile learning system.

2 The Adaptive Dialogue Manager

The framework proposed in the CHAT project aims at adapting the "dialogue" with the user according to several factors: the service provided, the task currently executed, the environment in which the user acts ("context"), the user him/herself and her/his device. These factors are measured and managed by a set of specific software components that together make up the the Adaptive Dialogue Manager as shown in Fig. 1. One of these software components is the Context Reasoner, pictured in the lower right corner, that creates and maintains the context model.

The Adaptive Dialogue Manager is the CHAT framework element in charge of a) identifying the most appropriate content to be returned to the client in order to satisfy user's request and b) determining the next system state by updating the models describing the different interaction factors. The Adaptive Dialogue Manager receives its input from a software module, called *Fusion*, which recognizes and combines low-level user input events from different channels (tap or sketch on the screen, voice, gesture, RFID or visual tag scan, etc.) in order to build an overall meaningful input. In a specular way, the output of the Adaptive Dialogue Manager, indicating the most

suitable content to be delivered to the user on the basis of the overall interaction state, is refined by a *Fission* module. This retrieves or generates suitable forms of the (possibly multimedia-based) content and takes care of their delivery and synchronization aspects. In CHAT, the Adaptive Dialogue Manager runs on a server that communicates through wireless networks with mobile thin clients. However, nothing prevents (parts or the whole) of the manager to run locally on the mobile device in the future as the computing power of mobile devices increases.

As shown in Fig. 1, the Adaptive Dialogue Manager results from the interaction of different reasoners, each managing a specific model. In the next section we focus in particular on the context model, maintained by the Context Reasoner.

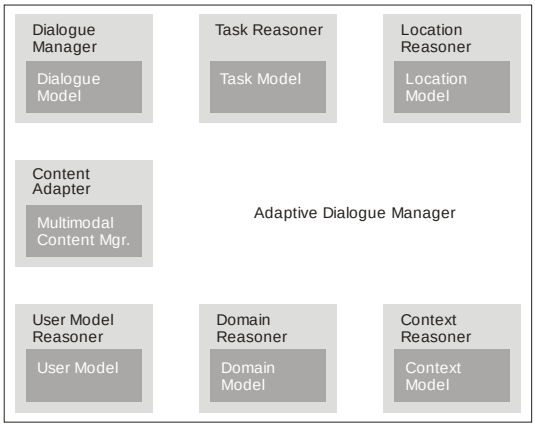

Fig. 1. The components of the Adaptive Dialogue Manager and their associated models.

3 The Context Model

The context model is an aggregated body of information about a) environmental parameters that can be used to determine the user's current situation, and b) computation and interaction characteristics of the mobile device of the specific user.

In our view, an ideal context model should 1) provide information on context aspects relevant for the given service application; 2) hide irrelevant context details; 3) offer a high-level interpretation of lower-level context details if requested. The context model has to cope with different types of requirements. A low-level general-purpose context modeling (e.g. the identification of absolute geographical coordinates for a specific mobile device) has to serve both service applications and other Adaptive Dialogue Manager components (e.g. Dialogue Manager, Content Adapter, User Model Reasoner, Domain Reasoner, Task Reasoner) At a high, application-specific level, the system could exploit the previous information with respect to specific semantics (e.g. the fact that the user of the device has successfully passed all exhibition rooms in a specific museum and is heading for the exit). In order to deal with this variety of needs, we propose that the context model be distributed over three

levels of abstraction, similar to [3], of which the Context Reasoner maintains the first two (low- and medium-level) as part of the Adaptive Dialogue Manager (see Fig. 2).

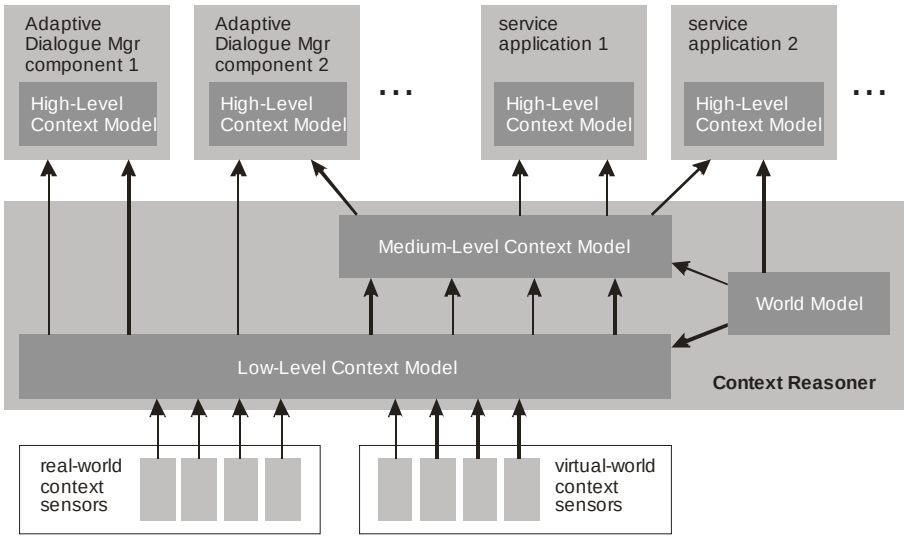

Fig. 2. A three-tiered context modeling architecture where the Context Reasoner maintains the two lower-level context models and leaves higher-level context modeling to other Adaptive Dialogue Manager components and service applications.

"Real-world context sensors" are system entities providing information about real-world phenomena such as geographical position, temperature, etc. "Virtual-world context sensors" provide information about events happening inside computing devices such as cellular phones, as well as information about the devices' technical capabilities themselves. Higher-level models are maintained by components and services outside the Context Reasoner. Fig. 2 illustrates how high-level contextual information can be derived by service applications and Adaptive Dialogue Manager (ADM) components by selecting and combining attributes from the three resources made available by the Context Reasoner, namely:

1. the Low-Level Context Model;
2. the Medium-Level Context Model;
3. the World Model.

In Fig. 2, the ADM component 1 chooses to create and maintain its internal high-level context model on the basis of low-level context attributes only (e.g. the time of day, the GPS coordinates of a mobile device). ADM component 2 makes use of both low and medium-level context attributes (e.g. the time of the day and the semantic location of a mobile device). Service application 1 uses only medium-level context attributes, while service application 2 combines medium-level context attributes (e.g. geographical speed) with information from the World Model database (e.g. the maximum speed allowed on a specific road). Each of the three Context Reasoner components is described in more detail below.

Low-Level Context Model

The Low-Level Context Model (LLCM) is continously updated by the Context Reasoner with the status of available real-world context sensors as well as status information about the software environment running on the mobile device (denoted as "virtual-world context sensors" in Fig. 2). The functional requirement of the LLCM is to be able to capture all aspects of context which can be useful to improve context-aware performance of the specific service applications targeted within a given development project. In the case of the CHAT project we have decided to let the LLCM capture and provide the following low-level context properties:

- *time*
- *absolute position* of a device, given directly by sensor
- *relative position* (e.g. given by proximity to objects included in the world model carrying Bluetooth position transmitters or visual tags)
- *device capabilities* (e.g. CPU power, client-server bandwidth; battery, etc.)
- *status information of available sensors* (both wearable – e.g. accelerometers, electronic compass – and environment-based ones.)

Medium-Level Context Model

The Medium-Level Context Model (MLCM) incorporates a set of context attributes created by the Context Reasoner by combining low-level context attributes provided by the LLCM and information from the World Model according to rules which are universally applicable for all foreseeable services that the CHAT architecture is imagined to host. We propose the MLCM to contain the following set of attributes:

- *approximate absolute position* derived by combining relative position + World Model
- *approximate relative position* derived by combining absolute position + World Model
- *semantic location* derived by combining (potentially approximated) relative position + World Model, e.g. "at home", "in the car"
- *absolute position history* derived by combining (potentially approximated) absolute position + time
- *relative position history* derived by combining (potentially approximated) relative position + time; e.g. includes series of scanned visual tags or proximity events caused by "Bluetooth tags"
- *geographical speed* (derived by combining a specific past time interval + absolute position history.

World Model

In order to generate some of the medium-level context attributes, the Context Reasoner needs to know about objects situated in the real world. For this purpose, the Context Reasoner needs to maintain a simple database. Information about objects of interest to the service applications and the Adaptive Dialogue Manager, such as the

semantic label of a specific object (e.g. “car”), an object’s current absolute location (e.g. long. x, lat. y), an object’s internal state (e.g. “engine off”), have to be defined and kept up-to-date by the owner component of the specific object, i.e. a specific service application or Adaptive Dialogue Manager component.
For reasons of computation efficiency as well as for user privacy, the objects in the World Model database should be governed by at least a simple authorization schema. A world model object marked as “public” by its owner will be accessible by all other service applications, while “private” would mean that only the specific service application, which defined and inserted the object into the world model, is allowed to access the object. In this way, the service applications and the various components making up the Adaptive Dialogue Manager can share real-world contextual knowledge and reduce redundancy, while at the same time maintaining security among service applications.

In addition to the information about real-world objects, the World Model also contains information about all sensor instances known to the system.

High-Level Context Models

There is a huge body of evidence within the area of Context Awareness, well aligned with previous related findings in Artificial Intelligence research, showing that high-level interpretations of contextual data (for example, a higher temperature in a room inevitably implies that there are more people in the room; if the person speed is higher than 10 km/h, that person is running) is very risky unless the application domain is extremely well specified. For a general-purpose architecture like the one intended to be developed in CHAT, only the application services themselves (which by definition are targeted to specific application areas) can draw high-level conclusions from context data. It is therefore left to each individual service application to construct its own, more or less complex, high-level context model if the execution of the application is improved by such a model. For this purpose, the Context Reasoner offers the service applications access to three context-related resources: the LLCM, MLCM and the World Model.

4 A Service Application Example: The “Gaus’ Day” Game

The proposed context modeling architecture is currently deployed in various mobile application scenarios within the CHAT project. Here we briefly describe the use of this architecture in *one* such application: the current version of the educational mobile (m-learning) game “Gaius’ Day” [1] designed to support students’ school visits at the archaeological park of Egnathia, Italy. The next section describes how we are currently enhancing the game using the proposed context modeling architecture.

The “Gaius’ Day” game is structured like a treasure hunt to be played by groups of 3-5 students: it combines the excitement of chase and solving the case with the joy of freely exploring a place and discovering its hidden secrets. The students’ challenge is to discover meaningful places in the park following some indications, and mark the position of the place on a map. Each group is given a cell phone and the map of the

park on paper. The cell phone is used as an instrument to communicate the challenge and to store data about how the game progresses. Due to display size limitations, the challenge is divided into separate units, corresponding to missions.

The group walks around the ruins trying to identify the place the mission refers to (Fig. 3 left). When students believe they have identified the target place of the mission, the leader digits on the phone the numerical code of the place, indicated in the park by small signs, on the phone. A sound signals that the current mission is concluded and the next mission is beginning. It visualizes the text of the new mission and reads it out aloud to attract the attention of all group members.

Fig. 3. A group performing the current version of the game (left), the 3D reconstruction of the public baths visualized on the phone (center), and the existing remains (right).

After completing the last mission, the group can explore on the phone the 3D reconstructions of the places correctly identified, so that they can visually compare the possible ancient look with the existing remains (Fig. 3 center and right).

5 Deploying the Context Model

In the first version of the Gaius' Day mobile game architecture, all data was stored and handled locally on the cellular phone during the game phase, and the mobile system did not consider contextual aspects. On the contrary, the current version lets the cellular phone be in constant connection with a game server and takes several aspects of context into consideration to enhance the user experience both during the game phase and during debriefing. In this section we account for how the context model proposed in Section 2 and illustrated in Fig. 2 is being deployed. Apart from providing each group with one cellular phone (featuring technolgies such as GPS, Bluetooth, WLAN), we also let all students carry small limited-range Bluetooth transceivers to be able to get an approximate measure of the relative position between each student and the mobile device carried by one of the students in the same group.
In particular, the *high-level service application* "Gaius' Day" inserts the following entities into the *World Model* (see Fig. 2), described in pseudo code:

```
student:
   INTEGER studentID
   STRING student_name
```

```
   INTEGER BluetoothDeviceID

student_group:
   INTEGER groupID
   INTEGER LIST group_members (studentID)
   INTEGER game_play_status (# of missions completed)
   INTEGER mobile_deviceID

mobile_device:
   INTEGER student_owner_group
   GPS_LOCATION absolute_geographical_position

location_of_interest:
   STRING location_name
   GPS_LOCATION absolute_geographical_position

sound_source:
   STRING sound_name
   SOUND_DATATYPE sound
   GPS_LOCATION absolute_geographical_position
```

Fig. 4 illustrates the world model and some contextual properties for the Gaius' Day service application.

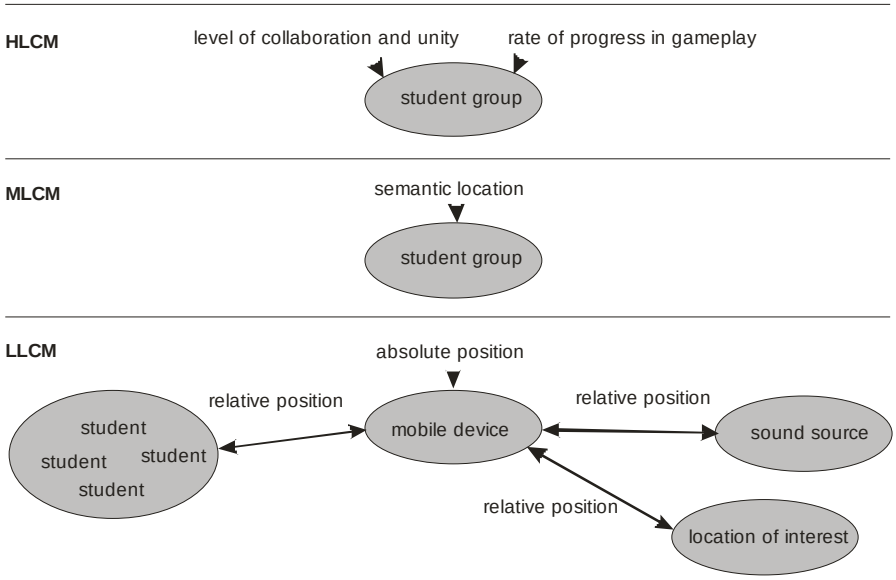

Fig. 4. World model entities used by the Context Reasoner module to provide low-level and medium-level context properties to the Gaius' Day service application.

Specifically, the Gaius' Day service application makes use of the following LLCM properties provided by the Context Reasoner:

- *Time.*
- *Absolute position* of group's mobile device, provided by inbuilt GPS, modeled as *real-world context sensor* in Fig. 2.

- *Relative position* of students with respect to the group's mobile device, approximated through the presence or absence of a Bluetooth connection between the mobile device and the transceivers carried by all students and modeled as *real-world context sensors.*
- *Relative position* of the group's mobile device with respect to the locations of interest stored in the *World Model*.
- *Relative position* of the group's mobile device with respect to the locations of the sound sources placed at strategical positions at the site.

From the MLCM, the Gaius' Day service application makes use of the following deducted contextual properties:

- *Approximate absolute position* of each student derived by combining the relative position of each student in relation to the group's mobile device.
- *Approximate relative position* of the group's mobile device with respect to locations of interests derived by combining absolute position of the device and absolute positions of locations of interest as stored in the *World Model.*
- *Semantic location* derived by combining relative position mobile device<->location of interest.
- *Absolute position history* derived by combining absolute position of mobile device + time.
- *Relative position history* derived by combining relative position of mobile device with respect to locations of interest + time.
- *Geographical speed* of a student group derived by combining the time interval spent on a specific game mission and the absolute position history of the mobile device during that time interval.

Based on these properties given by the LLCM and MLCM, the Gaius' Day service application maintains its own high-level context model, deducing properties such as:

- *The rate of progress in gameplay* for a specific group, based on how much the group's path at the site for each mission deviates from the ideal path, how much time it takes for the group to complete missions, and how much help from the "Oracle" is requested by the group. The rate of progress could be used for adapting the Oracle help to become more elaborate for slower groups and brief for faster groups, so that the challenge faced by each group is on the right level for optimal learning efficiency and motivation.
- *The level of collaboration and unity* within a specific group, based on movement patterns of group members in relation to each other.

In addition, the three-tiered context model architecture allows for additional functional and experiental value to the Gaius' Day gameplay as follows:

- It replaces the need for entering a location code on the mobile device as the final answer to each mission, with a simple press of a button, letting the GPS coordinates gathered in the background provide the answer.
- It makes it possible for the service application to wait with audio and visual game information until the whole group is gathered (determined by relative position between students and the device).

- It enables service applications to easily create a geo-spatial sound environment by placing virtual sound sources at various locations, perceivable through headphones, and whose volume depends on the distance between GPS device and sound source. In the case of the Gaius' Day game, we are placing noise from people at the market location, crackling of fire at the furnace location, etc.), determined by the GPS coordinates of the groups' cell phones. The intention is to increase the number of cues available for problem solving as well as to enhance the overall user experience.

6 Conclusions

We have presented a three-tiered context modeling approach to mobile services intended to simplify design, reduce computation needs on the client side (until mobile hardware catches up), and to reduce redundancy in general. The real value and power of the proposed context modeling approach for mobile human-computer interaction can only be assessed by successfully deploying it in a wide set of real-world scenarios. We are currently performing a field study with children of a middle school in Bari playing the Gaius' Day game based on the presented context model architecture. It will allow us to better assess both the technical aspects (which has been the focus of this paper) as well as usability and user experience dimensions.

We plan to deploy the model also in other mobile scenarios to determine its generality and utility. Although this empirical validation is not yet concluded, some benefits of the approach, as presented in this paper, have been already proven in formative evaluations carried out in our laboratory.

References

1. Costabile, M. F., De Angeli, A., Lanzilotti, R., Ardito, C., Buono, P., Pederson, T.: Explore! Possibilities and Challenges of Mobile Learning. In: Proceedings of CHI 2008 Conference on Human Factors in Computing Systems, pp. 145-154. ACM Press (2008)
2. Dey, A. K., Salber, D., & Abowd, G. D.: A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. Human-Computer Interaction, 16(2-4), 97-166. (2001)
3. Hong D., Schmidtke H. R., Woo W.: Linking Context Modelling and Contextual Reasoning. In: Kofod-Petersen, A., Cassens, J., Leake, D. B., Schulz, S. (eds) 4th International Workshop on Modeling and Reasoning in Context (MRC). Roskilde University, pp. 37-48. (2007)
4. Schmidt, A.: Ubiquitous Computing – Computing in Context. PhD Thesis, Lancaster University. (2002)
5. Weiser, M.: The Computer for the Twenty-First Century. *Scientific American*, pp. 94-10, September 1991.