A Context-Aware Middleware for Interaction Device Deployment in AmI

Tao Xu, Huiliang Jin, Bertrand David, René Chalon, and Yun Zhou

Université de Lyon, CNRS,
Ecole Centrale de Lyon, LIRIS, UMR5205
{tao.xu,huiling.jin,bertrand.david,rene.chalon,
yun.zhou}@ec-lyon.fr

Abstract. Miniaturization of smart devices and sensors, as well as widespread use of new interaction modalities make Ambient Intelligence (AmI) not a prospect for the future but an impending reality of existence. This requires methods for solving the issues on how to integrate interaction devices into a context-aware environment. We thus designed a middleware to provide a promising approach. Our middleware adopts a two-layer structure. The low layer is the enterprise service bus, which is in charge of integrating context sensors and interaction devices, and of discovering context. The high layer is the versatile context interpreter, which is responsible for context inference, expressive query, and persistent storage. Finally, we implemented the prototype of this middleware on the street and store marketing scenario.

Keywords: Middleware, Context Awareness, Ambient Intelligence, Human Computer Interaction.

1 Introduction

20 years ago, Marc Weiser formulated the prospect of computers in the 21st century, and proposed the pioneering notion of ubiquitous computing. Many aspects of his visions have already become reality in the past two decades. Furthermore, one of his primary ideas has recently evolved to a more general paradigm known as Ambient Intelligence (AmI). This defines an interaction between users and a context-aware environment, which adapts its behaviors intelligently to users' preferences and habits so as to facilitate and enhance users' life [2]. Going one step further, the AmI focus augments ubiquitous computing with additional requirements for natural interaction and context-awareness [2].

Human-Computer interaction (HCI), and context awareness, as two standalone core concepts of AmI, play important roles in this research area respectively. Context awareness suggests that systems could adapt their functionality to a user's activity and situation in the environment [3]. Context-aware systems, on the other hand, are concerned with acquisition of context, abstraction and understanding of context, and application behaviors based on the recognized context [16].

N. Streitz and C. Stephanidis (Eds.): DAPI/HCII 2013, LNCS 8028, pp. 183–192, 2013.

HCI involves the study, planning, and design of interaction between people (users) and computers. The emergence of proxemic interaction provides a possibility for fusion of two technologies (HCI and Context Awareness), which takes spatial relationships into consideration and extends the traditional interaction from the binary relationship (computer and human) to the context-aware environment [4]. More and more researchers are taking into account the intersection part of HCI and Context awareness to enable users to interact more naturally in pervasive environments. Our team has been working on the user interface related to augmented reality in HCI for many years and has proposed three innovative user interfaces [7]. One of these is the in-environment interface (IEI) [21] [22], meaning that the environment provides all interaction support including input and output devices, as well as environment dependent information. Furthermore, more issues and challenges face researchers than ever before. Programmers need an enhanced platform integrating various sensors and actuators and interaction technologies in order to propose an appropriate and up-to-date HCI according to observed environment behavior. We propose a context-aware middleware for interaction device deployment (CMID) in AmI, based on MOCOCO principles (MObility, COntextualization and COoperation) [6]. CMID takes into account multi-modal interaction technologies (hand gestures, marks, large-scale body movements, etc.) in relation with users' AmI environment context.

In the remainder of this paper, an AmI situation scenario is presented allowing contextualizing following a technical explanation. Then, several aspects of our middleware are explained starting with a low layer based on an ESB (Enterprise Service Bus), in charge of communicating in standard manner with in-environment sensors and actuators and of managing their dynamic discovery. The next step is a description of a high layer called the VCI (Versatile Context Interpreter), providing higher and semantic data interpretations. The general structure and main components of the VCI are described and related to our scenario. Finally, the paper ends with the conclusion and future work.

2 Street and Store Marketing Scenario (SSM)

Up-to-date ubiquitous computing with in-environment distributed sensors, actuators and user interface devices are able to create an Ambient Intelligence environment in a shopping area. The main goal is to detect potential shoppers and propose them appropriate goods, i.e. goods in relation with their shopping profiles and identified present needs. To allow this, we need first to capture potential consumer presence by appropriate sensor(s), then to study his/ her profile and determine what kind of information it seems appropriate to present him/ her with in order to intercept his/ her attention. The first stage in this capture process occurs in the street. Data collected by the sensors provide the system with information about the potential consumer, data that can be more or less precise: a young man or woman at least, or a store regular client, etc. In relation with this profile, the system could display in the shop window an appropriate advertisement. In the shop window display, it seems important to display advertising information not only for one passer-by and potential client but for several. Display

strategy can be organized by applying the proxemic user interface policy i.e. give more information to shop windows near located client(s) and less to distant ones. It may prove interesting to take two additional behaviors into account: in-shop continuation of consumer tracking, in order to provide more precise information in relation to his/ her movement in the store and also in-the-street movement, in-the-street walking and shop window watching. In the first case, increasingly detailed information can be given to the potential consumer in relation with his/ her location in the store (suit, shirt, pants department, or kitchen furniture department) with detailed knowledge of his situation (just married, etc.). The system can collect, store and use the information collected from previous purchases in the store or elsewhere (his/ her Facebook profile). In the second situation (in-the-street walking) it could be interesting to propagate the discovered profile of the potential consumer to other stores to allow them to use this information to provide him/ her with increasingly detailed and appropriate advertising.

3 CMID

Context-aware applications are becoming increasingly prevalent and can be found in the areas of wearable computing, intelligent environments, context-sensitive interfaces, etc. [10]. A now generally accepted definition of context is given by Dey and Abowd [1]: "Any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and application themselves". The context-aware system is defined as the system that uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task.

Development of context-aware applications is inherently complex. These applications adapt to change context information: physical context, computational context, and user context/tasks [4]. To reach this aim, it needs to integrate all sensors, actuators, communication objects and computing devices into the system. Low-level mechanisms and drivers are necessary. Then, either we have to create only one standalone application within which high-level context reasoning takes place, or we propose a set of applications or a system and propose to create an application-independent common high-level of contextualization making it possible to collect, process, interpret and propagate information with the context model and reasoning mechanisms [20].

To implement this more in-depth approach of context-aware services in AmI, we designed a context-aware middleware, organized in two layers: the low layer is an Enterprise Service Bus, which provides a solution to integrate sensors and actuators with a standardized data representation and unified standard interface to achieve the core functions of service interaction: service registry, service discovery and service consumption.

The versatile context interpreter is our high layer, which is in charge of context inferences, expressive query, and persistent storage. Detailed information will be provided in the following sections.

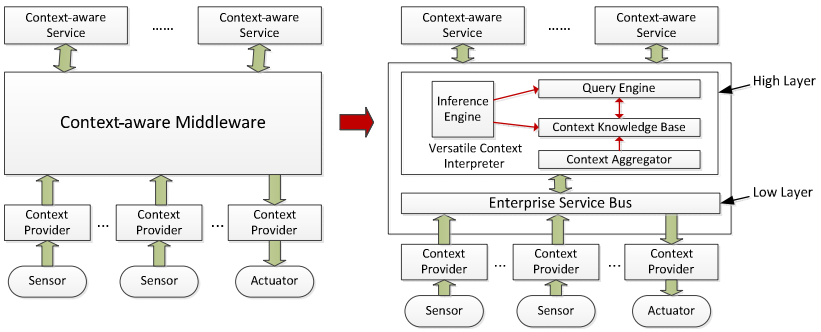

Fig. 1. Two context-aware architectures

3.1 Enterprise Service Bus

An enterprise service bus is a software architecture model used for designing and implementing interaction and communication between mutually interacting software applications in a service-oriented architecture (SOA). ESB allows services to be easily plugged in and out of the network without impact on other components and without the need to restart the system or even stop running applications. It is centered on a bus which provides a common backbone through which services can interoperate with a standardized format of data representation. In our paper, the context provider is the service used to obtain context from sensors, the web or other sources, and dispatch commands to actuators. ESB provides a unified standard interface to achieve the core functions of service interaction: service registry, dynamic service discovery, and service consumption. ESB also integrates interaction devices and a set of APIs for different interaction modalities to support development of interaction approaches.

3.2 Versatile Context Interpreter

The versatile context interpreter (VCI) is a high layer of context-aware middleware, as shown in Fig.1, made up of four parts: Context Aggregator, Inference Engine, Context Knowledge Base, and Query Engine. It leverages ESB basic services results to deliver and manage context-aware views and interpretations in order to deliver high-level information to the application. It adopts an ontology-based approach for context modeling and interpretation. Before detailing the description of other parts of the versatile context interpreter, the context model is introduced.

Ontology-Based Context Model

A context model, as a fundamental part of the context-aware system, aims at defining and storing context data in a machine processing form [3]. To develop a flexible and useable context model that covers the wide range of possible contexts is a challenging task [18]. We adopt the ontology-based context model to construct our context-aware middleware. For us, ontology is a reference model for components and behaviors of context [19]. The ontology-based model has a large number of good features for

developing the context-aware system, such as knowledge sharing, knowledge reuse, logic inference, etc. In particular logic inference enables the application using directly the deduced high-level context information.

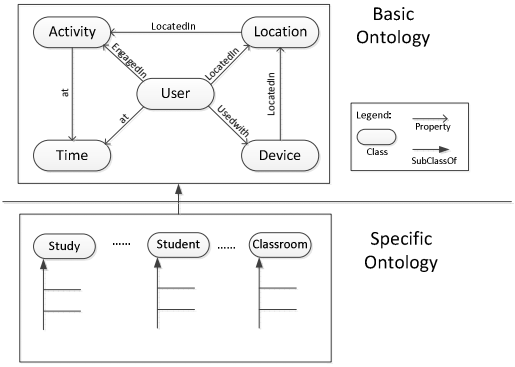

Fig. 2. Context expression ontology

We employ a hierarchical structure to describe the user's situation and circumstance based on Web Ontology Language (OWL) [9], which is an ontology markup language adopted by W3C as standard for semantic web. The structure is shown in Fig.2. The basic model defines generic conceptions and relationships in AmI, which come up with a basic context structure. It has five interrelated basic classes: user, location, time, activity, and device, which represent who, where, when, what activities, and devices; seven properties (relationships) between classes are identified. General context-aware ontology can be completed and upgraded by more precise information related to a particular application or application area. In our case, the general context-aware model for AmI context-aware systems is considered as the basic model. For a new application area such as the "Street and store marketing" application (SSM application), we propose a more precise and specific context-aware model. According to our build methodology, Fig.3, the general context-aware model is developed as the whole system by CMID designers and developers. The "Street and store marketing" application is developed in the scope of the CMID system by application developers. Most recent adaptation options can be implemented directly and dynamically during the application by the system (and its reasoning on collected data) or by the users (experienced users).

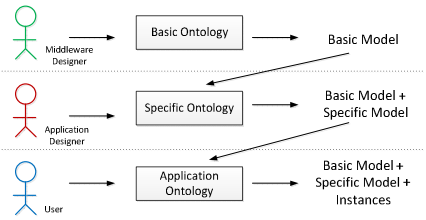

Fig. 3. Different contributors in the context-aware system

According to different context sources, we divide context into two categories: the low-level context and the high-level context. Context that can be extracted directly from sensors and devices such as location, time etc. is considered as low-level context. High-level context is issued from inference treatment based on low-level context data and semantic rules stored in the knowledge base and using the inference engine.

Context Aggregator

The context aggregator is responsible for working with basic contextual data collected by ESB to carry out fusion and fission of information:

1. The fusion service aims at integrating several basic data to discover high-level semantic information. In our SSM application, we can group the current date and the user's identification to discover that his birthday is today.
2. The fission service works in the opposite way, allowing in the SSM application a special discount for this user as well as the display on the shop window of the message "Happy birthday Mr. X", using the appropriate actuator.

Context Knowledge Base

The context knowledge base provides persistent storage for context through the use of relational databases, as well as supplying a set of library procedures for other components to query and modify context knowledge. We adopted the aforementioned ontology-based context model to build the environment and the user model. The ontology-based context model paves the way for the inference engine. The entire context is stored as the triple pattern.

In relation with different sources, the context is divided into three categories: predefined context, detected context and inferred context. Pre-defined context refers to context expressed in the application context model elaborated by application designers, such as user's profile context and specific environment context. Detected context is obtained from sensors as well as low-level context Inferred context is determined from collected data and knowledge rules by inference engine, and is considered as high-level context.

Context Query Engine

The context query engine has two main tasks:

1. Handle queries from the application: it supports SPARQL, which is an RDF query language, able to retrieve and manipulate data stored in OWL.
2. Invoke the context inference engine. When the application needs high-level context, it will invoke the context inference engine to generate the inferred context.

Context Inference Engine

The context inference engine is an important part of the context-aware system. It consists of the basic inference module and the predictive module.

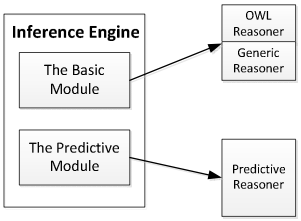

Fig. 4. Different inference engines and their use

The basic inference module is made up of Web Ontology language (OWL) Reasoner and Generic Reasoner. It focuses on checking context consistency and determining high-level context from low-level context. Consistency checking of the context model is an important action, activated by the CMID developer when he modifies the context-aware basic model. Generic Reasoner is activated by high-level middleware at each transaction between ESB and the context aggregator

OWL has a built-in reasoner based on the description logic. The reasoner can fulfill the essence of logical requirements, which comprises concept satisfaibility, class subsumption, class consistency, and instance checking.

Generic Reasoner offers a more flexible alternative. It adopts first order logic, which is more powerful than description logic. The application developer can freely define the rules for the specific situation. It is interpreted by an example in Fig. 5. Generic Reasoner is used to infer high-level context. Once the basic inference module is invoked, the knowledge base should be updated accordingly.

```
@prefix ex: <http://liris.cnrs.fr/smartspace.owl#>.
[Birthday:
   (?person ex:is ex:CurrentUser)
   (?person ex:hasBirthday ex:Feb.14)
   (?Date  ex:is ex:Feb.14)

->  (?person ex:isBirthday ex:Today) ]
```

Fig. 5. An example of a rule

Predictive Reasoner comes up with the available recommended information for the application according to analyzed users' previous activity. It has two main tasks: one is to provide users' recommended information based on users' previous behavior, while the other is to provide the proper interaction modality based on other users' selections.

We employ the classic decision tree algorithm (C4.5) to provide a recommended choice based on the training data set of users' activities [12]. This algorithm offers a fast and powerful method for different cases. In SSM, it can recommend the favorite style of clothes for clients approaching the shop window by analyzing this client's previous purchases.

We adopt the collaborative filtering algorithm to provide the proper interaction modality based on other users' previous selection. The motivation for collaborative

filtering is based on the idea that people often obtain the best recommendation from someone with similar tastes. The common process can be reduced to two steps:

1. Look for users who share the like-minded user's information with the active user (the user whom the prediction is for).
2. Use the interaction modality of those like-minded users found in step 1 to give a recommendation for the active user.

In SSM, this provides the appropriate interaction input modality (hand gesture) by user situation (distance between user and screen, number of users in front of the screen).

CMID Behavior Workflow

To summarize our proposal of a user-centric context-ware middleware for interaction device deployment (CMID system), we comment on the global workflow as shown in Fig.8: in the overall architecture we have 3 main components: an application layer with high level context-aware services related to the application, a high level middleware called VCI (Versatile Context Interpreter), and a low level middleware based on an ESB (Enterprise Service Bus). As stated earlier, two levels of behavior and their modeling are supported: a basic level related to general problems of AmI and context modeling, and a more specific level related to an application area. In particular, the application layer informs the VCI layer of the application area to take into account, while the ESB layer is mainly application area independent. With this in mind, the overall workflow functions as follows: (1) The application must inform the VCI of the context (specific model) to use. This model will be used by the Inference Engine, Query Engine, Context KB, and Context Aggregator. (2) Application Context Aware Services ask to receive contextual evolution from the VCI. (3) ESB collects the data from different sensors and propagates them to the context aggregator. (4) When the Context Aggregator is able to aggregate the received data, it does so, and places them in KDZ (Knowledge Data Zone). (5) Arrival of new data in KDZ generates the notification to the Context Query Engine. (6) The Query Engine calls on Context Inference to apply context inference. (7) The Context Inference engine introduces inferred data to KDZ. (8) When the Inference Engine terminates the inference process, the Context Query Engine collects new data from KDZ. (9) The Context Query Engine sends these data to the application. (10) The application context-aware services can also decide to update actuator states. They send new data to the Context aggregator which, using the Fission service, propagates the data to the appropriate actuator using ESB.

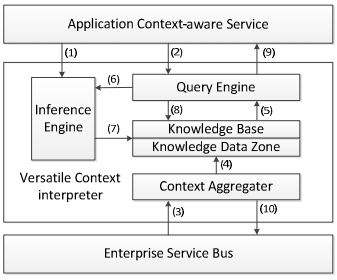

Fig. 6. CMID behavior workflow

4 Related Works

A large number of context-aware middleware has been proposed in the literature. Gaia [12] provides a distributed middleware infrastructure that coordinates software entities and heterogeneous networked devices contained in a physical space. Aura [17] comes up with services for managing tasks, and contexts for ubiquitous computing applications. SOCAM [8] has used an ontology-based model to represent the real environment, which provides efficient support for acquiring, discovering, interpreting and accessing various contexts to build context-aware services. MiddleWhere [13] is a distributed middleware infrastructure for location that separates applications from location detection technologies, utilizes probabilistic reasoning techniques to resolve conflicts, and determines the location of people given different sensor data. While these context-ware middleware provide several promising solutions for context awareness, they have not taken interaction modalities into account.

5 Conclusion and Future works

We have designed and implemented CMID, a user-centric context-ware middleware for interaction device deployment in AmI. This provides a platform associated with service discovery, mobility, environmental changes, and context retrieval. Besides the aforementioned features, it also integrates multi-modal interaction technologies (hand gestures, marks, large-scale body movements, etc.) and takes the user's context into consideration for extending the physical interactive environment to AmI. However, our context-aware middleware does not take into account the problem of users' privacy. When interacting in ubiquitous environments, protection of user's privacy is also a major problem. We will try to improve our middleware on this field to make it both intelligent and safe.

References

1. Abowd, G.D., et al.: Towards a Better Understanding of Context and Context-Awareness. In: Proceedings of the 1st International Symposium on Handheld and Ubiquitous Computing, pp. 304–307. Springer, London (1999)
2. Athanasopoulos, D., et al.: CoWSAMI: Interface-aware context gathering in ambient intelligence environments. Pervasive Mob. Comput. 4(3), 360–389 (2008)
3. Baldauf, M., et al.: A survey on context-aware systems. Int. J. Ad Hoc Ubiquitous Comput. 2(4), 263–277 (2007)
4. Ballendat, T., et al.: Proxemic interaction: designing for a proximity and orientation-aware environment. In: ACM International Conference on Interactive Tabletops and Surfaces, pp. 121–130. ACM, New York (2010)
5. Bettini, C., et al.: A survey of context modelling and reasoning techniques. Pervasive and Mobile Computing 6(2), 161–180 (2010)

6. David, B.T., Chalon, R.: IMERA: Experimentation Platform for Computer Augmented Environment for Mobile Actors. In: IEEE International Conference on Wireless and Mobile Computing, Networking and Communication, p. 51. IEEE Computer Society, Los Alamitos (2007)
7. López De Ipiña, D., et al.: EMI 2 lets: A Reflective Framework for Enabling AmI. J. UCS (2008)
8. Gu, T., et al.: A middleware for building context-aware mobile services. In: 2004 IEEE 59th Vehicular Technology Conference, VTC 2004, vol. 5, pp. 2656–2660 (Spring 2004)
9. Gu, T., et al.: Toward an OSGi-Based Infrastructure for Context-Aware Applications. IEEE Pervasive Computing 3(4), 66–74 (2004)
10. Krumm, J. (ed.): Ubiquitous Computing Fundamentals. Chapman and Hall/CRC (2009)
11. Lukowicz, P., et al.: From Context Awareness to Socially Aware Computing. IEEE Pervasive Computing 11(1), 32–41 (2012)
12. Quinlan, J.R.: C4.5: programs for machine learning. Morgan Kaufmann Publishers Inc., San Francisco (1993)
13. Ranganathan, A., et al.: MiddleWhere: a middleware for location awareness in ubiquitous computing applications. In: Proceedings of the 5th ACM/IFIP/USENIX International Conference on Middleware, pp. 397–416. Springer-Verlag New York, Inc., New York (2004)
14. Román, M., et al.: Gaia: A Middleware Infrastructure to Enable Active Spaces. IEEE Pervasive Computing 1, 74–83 (2002)
15. Schilit, B., Theimer, M.: Disseminating Active Map Information to Mobile Hosts. IEEE Network 8, 22–32 (1994)
16. Schmidt, A., et al.: There is more to Context than Location. Computers and Graphics 23, 893–901 (1998)
17. Sousa, J.P., et al.: Aura: an architectural framework for user mobility in ubiquitous computing environments. In: Proceedings of the 3rd Working IEEE/IFIP Conference on Software Architecture, pp. 29–43. Kluwer Academic Publishers (2002)
18. Stojanovic, D.: Context-Aware Mobile and Ubiquitous Computing for Enhanced Usability: Adaptive Technologies and Applications (Premier Reference Source). Information Science Reference (2009)
19. Wang, X.H., et al.: Ontology Based Context Modeling and Reasoning using OWL. In: Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops, pp. 18–23. IEEE Computer Society Press, Washington, DC (2004)
20. Xu, T., et al.: A context-aware middleware for ambient intelligence. In: Proceedings of the Workshop on Posters and Demos Track, pp. 10:1–10:2. ACM, New York (2011)
21. Zhou, Y., David, B., Chalon, R.: Innovative user interfaces for wearable computers in real augmented environment. In: Jacko, J.A., et al. (eds.) Human-Computer Interaction, Part II, HCII 2011. LNCS, vol. 6762, pp. 500–509. Springer, Heidelberg (2011)
22. Zhou, Y., et al.: PlayAllAround: Wearable One-hand Gesture Input and Scalable Projected Interfaces. Presented at the ERGO-IHM 2012, Biarritz (2012)