Journal of Mobile Multimedia, Vol.6, No.1 (2010) 049-062

AN EXTENDIBLE CONTEXT-AWARE SERVICE SYSTEM FOR MOBILE COMPUTING

BEEN-CHIAN CHIEN SHIANG-YI HE HSIN-CHAN TSAI YUEN-KUEI HSUEH

Department of Computer Science and Information Engineering
National University of Tainan, Tainan, Taiwan, R. O. C.
bcchien@mail.nutn.edu.tw tkuchris@hotmail.com johnkenkae@gmail.com ph.study@msa.hinet.net

Received August 12, 2009
Revised January 14, 2010

Context-aware computing is one of the research fields in pervasive computing and mobile computing. Context-aware systems can react to the user's preference according to the circumstances called context including location, time and other environment conditions. Many context-aware computing paradigms were developed in recent year; however, it is difficult for the present systems to extend the application domains and interoperate with other service systems due to the problem of heterogeneity among systems. In this paper, an extendible context-aware service system importing the concept of context independence is proposed and designed. The main idea of context independence includes the physical context independence and the logical context independence. The approaches of constructing context independence are also provided for the proposed system architecture. The context-aware system thus can easily integrate different service domains and diverse devices into a unifying environment.

Key words: Context-aware, mobile computing, ambient intelligence, context independent

1 Introduction

Ubiquitous computing was the vision initiated by M. Weiser from Xerox PARC (Palo Alto Research Center) in 1991 [21]. It is also known as pervasive computing for some researchers. The vision of constructing pervasive computing environment collects and integrates significant hardware, system, communication, and information technologies to provide appropriate service for our lives in a vanishing way. The goal is to make users and their service tasks the aim rather than the computing devices. In traditional computation environment, like PC-based working style, is hard to achieve the purpose of embedding in everyday living environments. With the rapid growth of mobile devices such as PDAs and smart phones, the techniques of mobile computing are becoming important and popular in recent years. The applications of mobile computing conduct the research issues to pervasive computing and ubiquitous computing. Context-aware computing is also one of the research fields and domains in mobile computing. A context-aware system is a mobile environment in which applications can discover and make use of context information including user location, time, date, nearby devices and other environmental activities to adapt their operations and behavior [6]. Due to the fast development of wireless communication on mobile devices, different kinds of context-aware architectures were designed and employed for a wide spectrum of applications [2]. However, since the

individual focus of each framework on its specific application domain, the current context-aware systems are heterogeneous in all aspects, such as hardware, mobile resources, operating systems, application software, and platforms. The serious heterogeneous characteristics of context-aware computing are especially important and become significant drawbacks while developing or integrating context-aware services for the applications in mobile computing environment.

In this paper a framework for context-aware service and the approaches of achieving context independence are proposed and developed. In this framework, we introduce the idea of ANSI/SPARC architecture for database management to context-aware computing. The concept of *context independence* is revealed in the system framework. Two types of context independence, *the physical context independence* and *the logical context independence*, are presented. The purpose of the physical context independence is to prevent misinterpreting raw data from mobile devices with various standards; whereas the logical context independence is to allow context to be understood and applied by applications. As a result of context independence, cross-domain applications will be able to be built by developers for mobile devices used in the existing service domains regardless of the heterogeneity of mobile computing environment. The context-aware architecture thus can integrate different service applications and diverse mobile devices into a unifying context-aware system.

The remainder of this paper is structured as follows. Section 2 reviews the related researches of context-aware systems. The framework for context-aware service and the system architecture are introduced in Section 3. The detailed functions of system components are also discussed in this section. Section 4 describes the approaches of completing context independence. Consecutively, we construct an application developed under the architecture to demonstrate the feasibility of the system in Section 5. Section 6 summarizes the work presented here and expresses the future work.

2 Related Researches

The early applications of context-aware systems started when Want et al. proposed the Active Badge Location System in 1992 [20]. The term *context-aware* first appeared in 1994 mentioned by Schilit and Theimer [18]. Then, various architectures of context-aware systems based on the framework of context management [14] were proposed. The *Context Toolkit* (Dey *et al*., 2000) [9, 10] provided context interpretation using widgets and a set of object-oriented API to offer the creation of service components. The *Hydrogen* (Hofer *et al*, 2002) [13] is a framework based on layered architecture in which contains the adaptor layer, the management layer and the application layer. The *Gaia* project (Roman *et al*., 2002) [17] is a middle-ware based architecture; the system consists of *Gaia* kernel and application framework to support the development and execution of mobile applications. Another middle-ware system, *SOCAM* (Gu *et al.* 2004) [11], uses a central server called context interpreter to obtain context data for building and prototyping of context-aware services. The *CORTEX* system (Biegel *et al*., 2004) [4] is also a middle-ware structure based on sentient object model which supports the context-aware services in an ad-hoc mobile environment. *CoBrA* (Chen et al. 2003) [7] is an agent based architecture especially designed and used for pervasive computing of an intelligent space. The system manages and shares a context model using a community of agents.

Each context-aware framework discussed above is generally utilized on its individual specific domain of employment. The problem of heterogeneity issues were characterized not only on the physical devices but also on the logical context interoperability. To bridging the communication

between heterogeneous devices, Nakazawa *et al*. [16] presented a framework for heterogeneity handling. Bartelt *et al.* [3] proposed a system that can integrate devices dynamically and enable interoperability between them. Schmohl and Baumgarten [19] further derived a generalized context-aware architecture for mobile computing environments to resolve the heterogeneity issues in both context-awareness and interoperability domains.

3 The Framework and System Architecture

The proposed context-aware system architecture follows the general framework presented in [1] and [2] but makes some modification. The five-layer model in [1] consists of *the physical*, *the data*, *the semantic*, *the inference* and *the application* layers. These layers in this model focus on the descriptions of functions in a context-aware system instead of a framework. The abstract five-layer proposed in [2] presents a conceptual framework of a context-aware system containing *the sensors*, *the raw data*, *the preprocessing*, *the storage/management*, and *the application* layers. The preprocessing layer in this framework emphasized the interpreting and reasoning of context from raw data. The access of context is controlled by the storage/management layer. However, context information is generally complex and diverse for context-aware systems in mobile computing environments. The management of context information is an important task. The framework thus was modified to reveal the role of context manipulation. The modified conceptual framework consists of five layers: the device layer, the resource layer, the context layer, the storage layer, and the application layer; as shown in Figure 1.

The contents of each layer are described as follows.

1) *The device layer*: This layer contains the physical equipments and devices operated and used in the context-aware systems including sensors, identifiers, mobile devices, and actuators, etc.
2) *The resource layer*: Entire resources of the context-aware computing environment including places, persons, devices, and objects are constructed and managed in the resource layer. The resources of the environment are generally called the domain knowledge of the system and the knowledge can be represented as ontology or semantic networks. A mapping component between the device layer and the resource layer is also needed and used to bridge the device layer and the context layer for archiving physical context independence.
3) *The context layer*: Context processing is the core of a context-aware system. Context information is generated and managed in this layer. The interactive activities in the resource layer are presented as context and used to adapt behavior of the system. Context model is needed here to define and represent context data. Effective context extraction and efficient context management are the main two components for processing context.
4) *The storage layer*: The storage layer stores not only the context data of the current status but also the historical context data in the context-aware system. The context data produced in the context layer are used to provide the services of applications in the application layer. In order to easily access context data, an effective context database is important. The context data access mechanism generally includes the storage of context schema and context query. The usage of a context database system has to match the representing structure of context model in the context layer.
5) *The application layer*: In this layer, application can be built and executed by querying the current status of context and the related historical context data from the context database in the storage

layer. Since the contents of context-awareness are accessed by context queries from context databases, the various applications can be constructed under diverse application developing platforms.

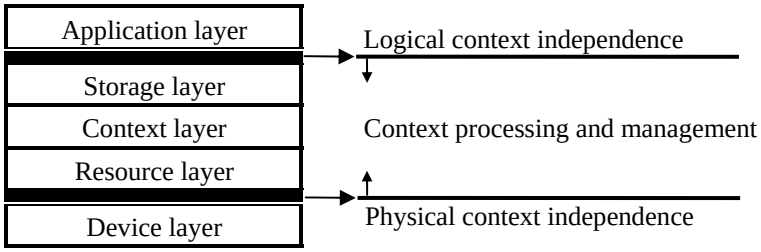


Figure 1 The five-layer conceptual framework.

In the five-layer framework, the main task of the resource layer, the context layer, and the storage layer is to process and manage the context from the different devices and support various kinds of application services. Hence, the context independence is considered between these three layers and the others layers. Two kinds of context independence are revealed in this framework: physical context independence and logical context independence.

The physical context independence is defined as the immunity of context to the change and update of physical mobile devices. A traditional context interpreter is dedicated to interpreting the context for a specific device in an application. Although such a tightly coupled methodology results in good performance of fast reaction for system, the drawback is that the application does not work and the service cannot be extended to other devices without completely packing the context function. The physical context independence tries to solve this problem. It means that the devices and platforms used in context-aware service can be changed, reconstructed and replaced anytime regardless of updating the software or middleware for context interpretation.

The logical context independence is considered between the application layer and the storage layer. In previous context-aware systems, applications use context to reason and trigger the actuator in a predefined service program. It is lacking in flexibility of constructing or updating new services. Further, it is difficult for the context-aware system to manage and reuse a great deal of context. The logical context independence separates application services from context process. The context is managed and stored by the context layer and the storage layer, respectively. The application layer employs context to detect circumstances and develop guest services instead of direct programming by middleware.

For accomplishing this framework, the architecture of context-aware system based on context database management (CADBA) is designed. The architecture is shown as Figure 2. The functions of each component are described as follows:

- *Mobile devices*: This component is the device layer in the framework of Figure 1. The possible hardware and the drivers used in the domain that could be mobile devices like PDAs, smart cell phones, or detecting sensors like thermometers, RFIDs, or actuators like alarms, temperature regulators.

- *Mobile devices*: This component is the device layer in the framework of Figure 1. The possible hardware and the drivers used in the domain that could be mobile devices like PDAs, smart cell phones, or detecting sensors like thermometers, RFIDs, or actuators like alarms, temperature regulators.

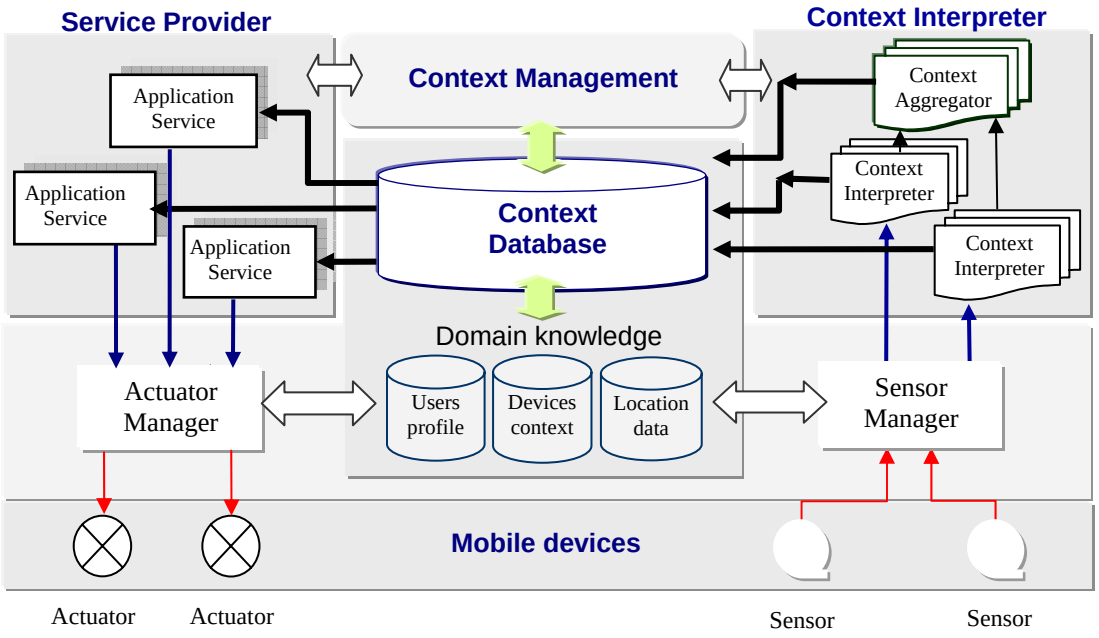


Figure 2 The architecture.

- *Device manager*: The devices in the device layer have to be registered and named before they are used. The device manager containing the sensor manager and the actuator manager supports a flexible tool for collecting and monitoring the status of all current devices in the system. The devices can be classified into different types according to their properties for the sake of managing easily.
- *Domain knowledge*: It contains the entire resources of the context-aware computing including places, persons, devices, and objects. The resources are described by knowledge representations like ontology or semantic networks. The domain knowledge can be generally divided into three categories.
  1) Places: All possible locations where activities will occur in the environment of applications.
  2) Persons: The possible people including the application users and other persons who interact in the system.
  3) Devices: The description for all existing devices that support the device manager to record the status of the hardware.

Except the above individual domain knowledge, the mapping of the relationships between places and devices, and the relationships between persons and devices are also needed to be depicted in the domain knowledge.

- *Context interpreter and context aggregator*: The raw signals from sensors or mobile devices cannot work as their original format. They have to be transformed to context information in context-aware system. The context interpreter is used to interpret the structures of raw data from sensors and represent the information as low-level context called *sensor context*. The context aggregator then gathers the related low-level sensor context data to form a higher-level context. As soon as the context model is provided, context can be generated by context interpreters and context aggregators. The context model is discussed as next description.
- *Context model*: The *context model* is essential for a context-aware system to progress the context processing. Ontology implemented in OWL is a suggested representation for representing and sharing context knowledge. To archive the physical context independence of the architecture, the context data are divided into three classes. The approaches of constructing context model and managing devices as a context are described in next section.
- *Context management*: Context-aware computing implies the interconnection of people, devices, and environments. It makes possibility of sharing the users' context among each other in heterogeneous systems. As a consequence, the issues on context management are getting complicated and important in cross-domain mobile computing. The challenges include context configuration, device fault, context conflict, context inconsistency and security [12]. These problems will be discovered and discussed as new research topics while more context-aware systems are developed and more interactions are needed in the wide-spread mobile applications.
- *Context database*: In order to conduct the context independence to the context-aware applications, a powerful context database is the kernel of the system. The purpose of context database is to provide the integration of current system context and the storage of historical context data. As a traditional database system operates, the context database system must store context data with well-defined context data schema. As usual, the storage schema is strongly connected and matched to the context model of the system. The main function of the context database is to provide an efficient context access mechanism to store and retrieve context data using context queries.
- *Service provider*: Generally, the mobile services and applications are dependent on the context of the context-aware system. That is, the location, the user, date, time and the surrounding objects may determine the response of user's requirements. The source of the context comes from the context database. The functions of the service provider consist of context querying, service construction, and service coordination. To satisfy the user's need and complete the application, the service provider starts context process by context querying. Then, the returning context data from the context database are collected to determine and prepare the necessary services. Finally, the service coordinator is used to arrange the sequence of services and perform the services in the application. The logical context independence is accomplished by accessing context data to adapt system for satisfying applications under distinct platforms using unifying context queries.

4 The Enforcement of Context Independence

The core of executing context independence in the proposed architecture is the components of the context interpreter and the service provider. However, the context model is essential for a context-aware system and the representation of context must be determined before performing the context interpreter and the service provider. In general, the context model can be constructed by any proper knowledge representation. As the previous mention, ontology represented in OWL is a suitable context model for the proposed architecture. The reasons are:

1) The structure of an ontology is easy to describe, and is flexible to extend. Further, ontology construction tools are popular.
2) The techniques of ontology fusion can be applied to integrate heterogeneous context models.
3) The representation, like OWL, is XML based scripts. It is formatted, unified, and standardized.
4) There exist native XML database systems [15] and query tools - Xquery [5] for accessing XML data and structures.

Hence, designing context model and domain knowledge as ontology in XML based representation can take advantage of not only the characteristics on XML storage structures but also the well-designed query language. While importing data from sensor devices, the XML format is also easy to convey the information extracted by data acquisition interface to the context interpreter. In additional to XML-based context model, the context used in the system can be divided into three levels:

- *Sensor context* ($Context_{sensor}$): This type of context is the essential raw information triggered by sensors and interpreted by context interpreters directly. The definition is as follows:

$$Context_{sensor} = Interpret\{sensor_i | sensor_i \in D_j \text{ and } D_j \subseteq Dev\},$$

  where $sensor_i$ is a kind of sensors in the type of devices D_j, and Dev is the set of all types of devices in the context aware system.

- *Event context* ($Context_{event}$): This type of context is aggregated by different sensor context. We define the behavior of aggregating various actions of sensors as an event context.

$$Context_{event} = Aggregate\{\bigcup_{j=1}^{m} Context_{sensorj}\}.$$

- *Scenario Context* ($Context_{scenario}$): The scenario context is another high-lever context containing not only sensor context but also at least one event context. The scenario context is defined as follows:

$$Context_{scenario} = Aggregate\{\bigcup_{i=1}^{n} Context_{eventi}\} \cup \{\bigcup_{i=1}^{m} Context_{senosri}\},$$

  where $1 \le |Context_{event}| \le n$ and $1 \le |Context_{sensor}| \le m$, or $2 \le |Context_{event}| \le n$ and $0 \le |Context_{sensor}| \le m$.

Based on the above definitions of distinct context, the physical context independence can be accomplished by an XSL (eXtensible Stylesheet Language) scripts based context interpreter. As shown in Figure 3, the context interpreter consists of context editor and XSLT (XSL Transformation) processor. Mobile sensors prepared circumstances data in XML format and delivered them to the context interpreter. After sensor data were received by the context interpreter, the context editor is used

to confirm the relationship between sensor data and sensor context according to the definitions in context model. An XSL script will be generated for interpreting the sensor context if the confirmation is done. Then, the XSLT processor is used to translate original XML sensor data into XML formatted sensor context in context model.

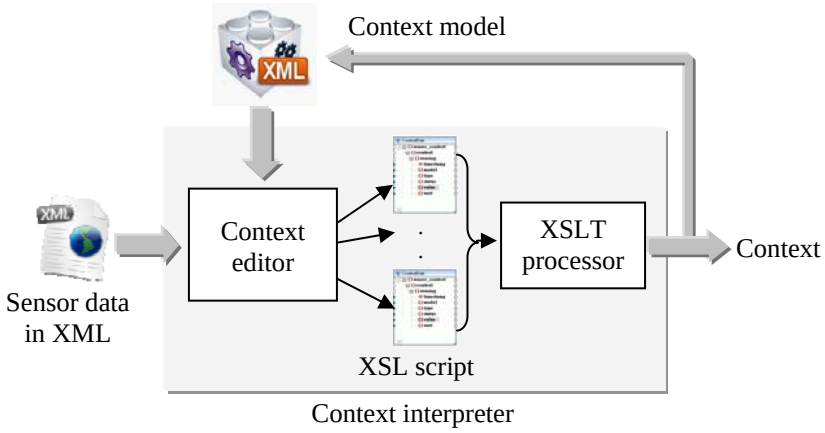
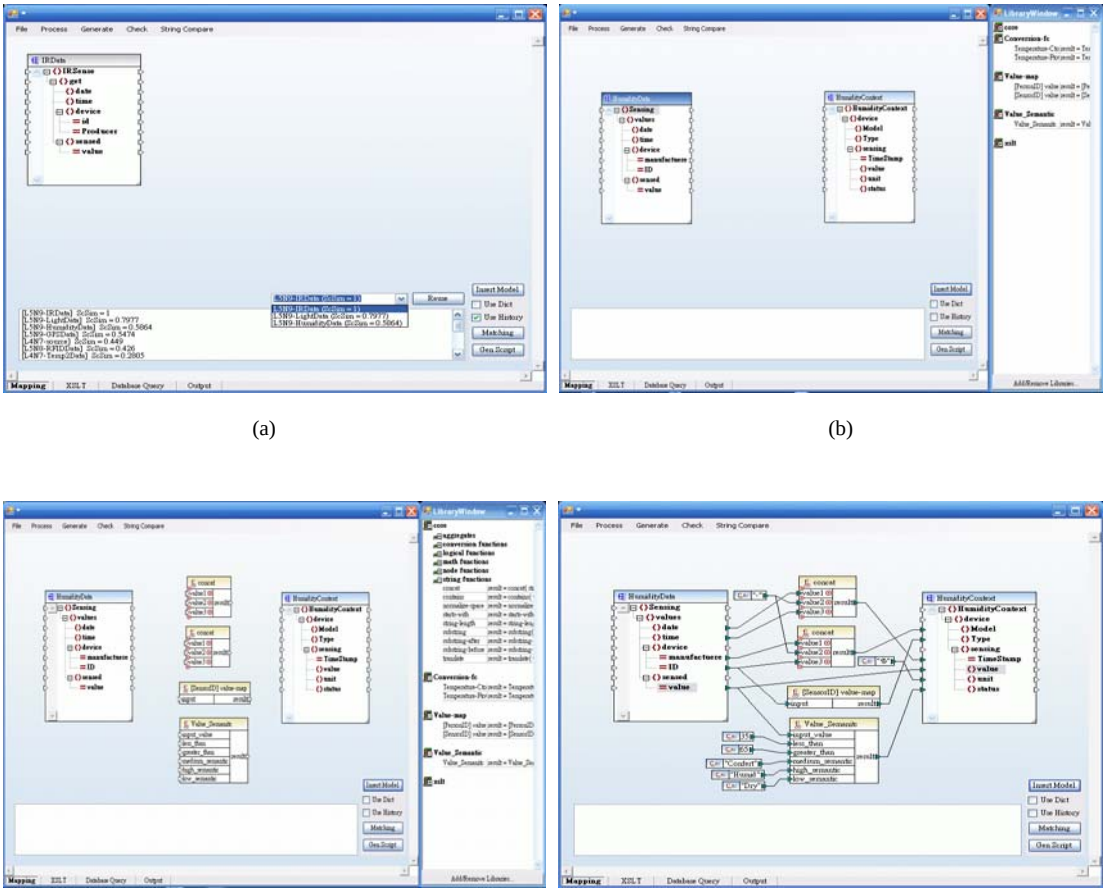


Figure 3 The context interpreter for physical context independence.

While completing the sensor context, the context editor can aggregate sensor context into event context and scenario context. The context editor is demonstrated as Figure 4. First, the editor reads the XML sensor data or others context from context model, as shown in Figure 4(a). Second, the corresponding context in the context-aware system is selected from context model, as Figure 4(b) shows. The next step shown in Figure 4(c) is to arrange the interpreting operations. Some predefined transformation functions and user defined operations can be added into the system for transferring data values to meaningful context. At last, the finished context mapping, as Figure 4(d), can be stored in XSL script to interpret the context in the context-aware system. Since the context interpreter is done by interpretation scripts, the context interpreter only builds a new context interpretation script instead of full-function program while mobile devices are updated or changed. The goal of the physical context independence can be enforced.

A context-aware system perceives context generated from environment. The service provider takes charge of staring reactions for users. The context describing events of applications has to be collected first, and then the service provider determines if the action will be fired. The entire generated context in the CADBA architecture is stored in the context database in XML format. Xquery [5] can be used to access context easily. As shown in Figure 5, a new service is built by applying XQuery API to check the existing of context from context database and provide the message of reaction through web service. In this strategy, services are not concern with the detailed context generation and management at the lower layers of the system. Services do not need to be rebuilt if the context is modified. New services can also be built and applied by any mobile application directly regardless of the hardware types and operating systems. The logical context independence thus is accomplished.



(a) (b)

(c) (d)

Figure 4 The context editor.

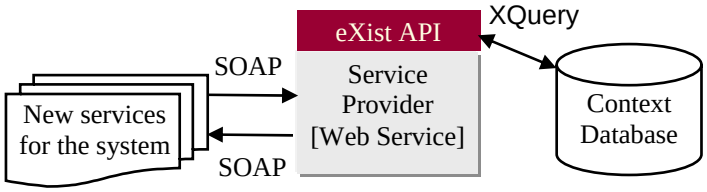


Figure 5 The service provider for logical context independence

The problem of heterogeneity is a bottleneck while developing and extending context-aware systems in mobile computing environment. The enforcement of context independence resolves dependency of devices and improves interoperability of applications.

5 An Application of Home Safety based on CADBA

We had constructed an application of Home Safety context-aware system based on the proposed CADBA architecture [8]. The environment of the system is shown in Figure 6. The home space is separated into a garage, a living room, a dining room, a bathroom, a kitchen, a yard and a balcony. The members of family consist of the couple (George and Catherine), their parents, their child (Lesten) and a pet (a dog called Small). Figure 7 shows their relationships. The house equips a lot of devices including sensors on each doors and windows, smoke detectors and infrared object detectors, actuators like alarms, and RFID keys detectors.

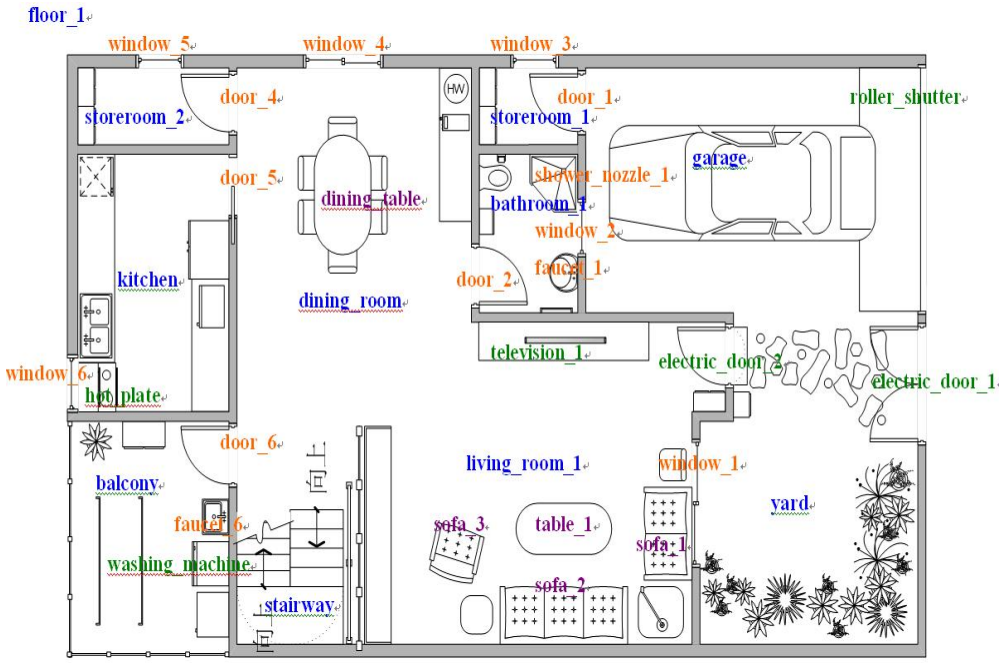


Figure 6 The environment of the Home Safety.

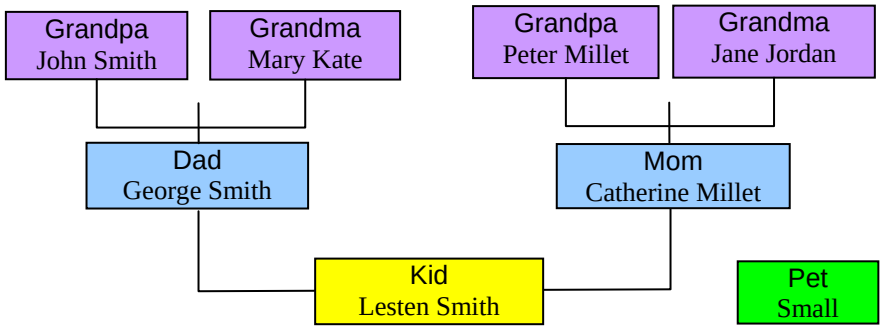


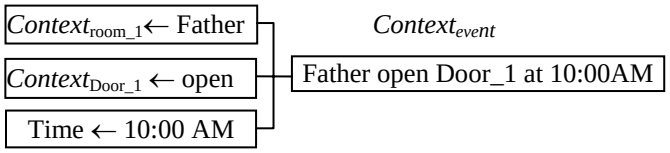
Figure 7 The members of Family.

All above resources are described and built as the context model in a OWL ontology form by Protégé. The context represented in $Context_{sensor}$, $Context_{event}$ and $Context_{scenario}$ are shown as the following examples.

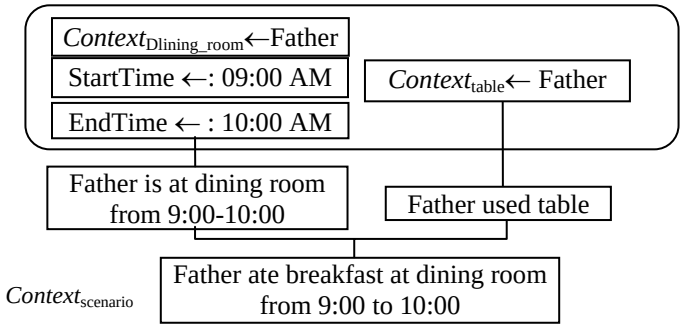
1) The sensor context described the sensor of Door_1 being detected in the "open" status, the corresponding $Context_{sensor}$:

"The Door_1 opened" ($Context_{Door\_1} \leftarrow$ open)

2) More sensors detected different sensor context that were aggregated by context interpreter with detecting father in the room and Door_1 was opened at 10:00 AM. The resulting event context, $Context_{event}$:



3) The event context "father is at dining room from 9:00 to 10:00" and the sensor context "father used table" can be aggregated into a higher level scenario context, $Context_{scenario}$:



The above three classes of context are defined by the context editor of context interpreter. While the context-aware system operates successfully, the context interpreter will generate context as the XLS definitions. The interpreted context are represented as XML descriptions and stored in the context database. The context schema is important for efficient context management when a great deal of context is generated. The storage structure is shown in Figure 8 as a tree-like structure of XML. In this case, the top level of the structure is location since the place is well-marked and is almost static. Such a design is helpful for the maintenance of the sensor devices. It is also convenient for engineering to extend new devices and update sensor context. The lower level of the structure is date and time. It is designed for recording each room's activities everyday. Since the users in this application are mobile targets, the context model has them be stored to the right place and the right time when an activity occurs. A native XML database eXist [15] is used to be the context database in this application. The main advantage of applying native XML database is that the system can store the XML description directly and provides efficient XML search engine for query language such as XQuery.

While the current awareness context and the historical context are maintained in the context database, the service provider then tries to detect the context data using context queries. If the scenario context or the event context contained in the target service is detected and all the conditions are satisfied, the target service will be triggered. It is clear that the service works on any device in which can inquire and accept context. Hence, the heterogeneous application domains can be integrated smoothly after the fusion of context models belonging to different systems. The advantage of context independence obviously improves the ability of system extension in this case.

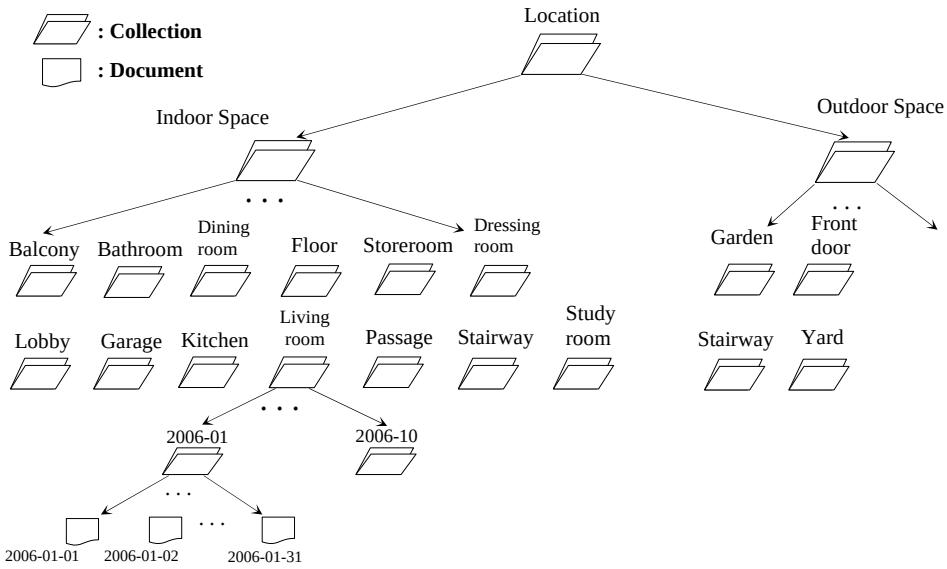


Figure 8. The structure of XML in context database.

6 Conclusion

The main contribution of this paper is to reveal the concept of *context independence*. For accomplishing the proposed idea, a framework for context-aware service and a context-aware architecture CADBA based on context database for mobile computing are developed. The proposed architecture employs the ANSI/SPARC database management architecture into the context-aware computing. We also design the approaches of enforcing the physical context independence and the logical context independence to demonstrate the feasibility of the architecture. Based on the context database, context independence is accomplished and the system can be extended easily. The heterogeneity environments in mobile computing will gain a graceful solution.

This work is intended as a starting point of future research on context-aware computing. A generalized context interpreter is being investigated and the problems of context management for context-aware computing will be paid more attention in the future.

Acknowledgements

This research was supported in part by the National Science Council of Taiwan, R.O.C. under contract NSC 96-2221-E-024-011-MY2 and the Industry Technology Research Institute, Taiwan under contract B200-95N001.

References

1. Ailisto, H., Alahuhta, P., Hataja, V., Kylloenen, V. and Lindholm, M., Structuring context aware applications: Five-layer model and example case, *Proceedings of the Workshop on Concepts and Models for Ubiquitous Computing (UbiComp 2002)*, Goteborg, Sweden, 2002.
2. Baldauf, M., Dustdar, S. and Rosenberg, F., A survey on context-aware systems, International Journal of Ad Hoc and Ubiquitous Computing, Vol. 2, No. 4, 2007, 263-277.
3. Bartelt, C., Fischer, T., Niebuhr, D., Rausch, A., Seidl, F. and Trapp, M., Dynamic integration of heterogeneous mobile devices, in *Proceedings of the 2005 workshop on Design and evolution of autonomic application software*, New York, NY, 2005, pp. 1-7.
4. Biegel G. and Cahill V., A framework for developing mobile, context-aware applications, in *Proceedings of the 2nd IEEE Conference on Pervasive Computing and Communication*, 2004, pp.361 – 365.
5. Chamberlin, D., Clark, J., Florescu, D., Robie, J., Siméon, J. and Stefanescu, M., *XQuery 1.0: An XML Query Language, W3C Working Draft*, W3C Consortium, December 2001 http://www.w3.org/TR/xquery..
6. Chen, G. and Kotz, D., A survey of context-aware mobile computing research, *Technical report* TR2000-381, Dept. of Computer Science, Dartmouth College, 2000.
7. Chen, H., Finin T. and Joshi, A., An ontology for context-aware pervasive computing environments, in *Proceedings of the Workshop on Ontologies in Agent Systems* (AAMAS), 2003.
8. Chien, B. C., Cho, H. P., Chang, Y. S.,. Tsai, H. C and Lai, Y. S., A design of framework for context-aware database management, in *Proceedings of Symposium on Digital Life Technologies: Building a Safe, Secured and Sound Living Environment*, June 6-7, 2007, Tainan, Taiwan.
9. Dey, A. K. and Abowd, G..D., The Context Toolkit: Aiding the Development of Context-Aware Applications, in *Proceedings of the 22nd International Conference on Software Engineering*, Limerick, Ireland, June 2000.
10. Dey, A. K., Abowd, G. D. and Salber, D. , A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications, *Human- Computer Interaction*, 16, 2001, 97 – 166.
11. Gu, T., Wang, X H., Pung, H. K. and Zhang, D. Q., A middleware for context-aware mobile services, *IEEE Vehicular Technology Conference*, Milan, Italy, 2004.
12. Hegering, H. G., Küpper, A., Linnhoff-Poien and C. ,Reiser, H., Management challenges of context-aware services in ubiquitous environments, in *Proceedings of 14th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management, Heidelberg, Germany, LNCS No. 2867*, 2003, pp. 246-259.
13. Hofer, T., Schwinger, W., Pichler, M., Leonhartsberger and Altmann,G., J., Context-awareness on mobile devices – the hydrogen approach, in *Proceedings of the 36th Annual Hawaii International Conference on System Sciences*, 2002, pp.292-302.
14. Korpipää, P., Mantyjarvi, J., Kela, J., Keranen, H. and Malm, E-J., Managing context information in mobile devices, *IEEE Pervasive Computing*, Vol. 2, No. 3, 2003, 42-51.
15. Meier, W., "eXist: An Open Source Native XML Database", in*:Proceedings of Workshops on Web, Web-Services, and Database Systems*, Erfurt, Germany, *LNCS No. 2593*, 2002.

16. Nakazawa, J., Tokuda, H., Edwards, W. K. and Ramachandran, U., A Bridging Framework for Universal Interoperability in Pervasive Systems, in *Proceedings of the 26th IEEE International Conference on Distributed Computing Systems*, 2006, pp. 3-3.
17. Roman, M., Hess, C., Cerqueira, R., Ranganathan ,A., Campbell, R. H. and Nahrstedt, K., A middle-ware infrastructure for active spaces, *IEEE Pervasive Computing*, Vol. 1, No.4, 2002, 74-83.
18. Schilit A. and Theimer, M., Disseminating Active Map Information to Mobile Hosts, *IEEE Network*, Vol. 8, No. 5, 1994, 22-32.
19. Schmohl, R. and Baumgarten, U., A generalized context-aware architecture in heterogeneous mobile computing environments, in *Proceedings of the 2008 The Fourth International Conference on Wireless and Mobile Communications*, July 27-Aug. 1 2008, pp. 118-124.
20. Want, R., A. Hopper, Falcao, V. and Gibbons, J., The active badge location system, *ACM Transactions on Information Systems*, Vol. 10, Vol. 1, 1992, pp. 91-102.
21. Weiser, M., The Computer for the 21st Century, Scientific American, Vol. 265, Sept. 1991, pp.94-104.