Computer Communications 31 (2008) 4376–4382

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

iMuseum: A scalable context-aware intelligent museum system

Zhiyong Yu [a], Xingshe Zhou [a], Zhiwen Yu [a,b,*], Jong Hyuk Park [c], Jianhua Ma [d]

[a] School of Computer Science, Northwestern Polytechnical University, PR China
[b] Academic Center for Computing and Media Studies, Kyoto University, Yoshida-Honmachi, Sakyo-ku, Kyoto 606-8501, Japan
[c] Department of Computer Science and Engineering, Kyungnam University, Korea
[d] Faculty of Computer and Information Sciences, Hosei University, Japan

ARTICLE INFO

Article history:
Available online 16 May 2008

Keywords:
Intelligent museum
Context-aware
Scalability
Dynamic context knowledge

ABSTRACT

A context-aware intelligent museum system can capture the information of the visitor and surroundings, recognize the visitor's purpose, and then assist visiting in the museum. It is noted that not only the devices in the museum cannot be easily predicted, but the available applications may change over time. Therefore infinite entities and dynamic context knowledge are necessarily managed by a system with good scalability. This paper proposes a scalable context-aware intelligent museum system called **iMuseum**. The system is based on a new context model, **2*3CM** (**2** Sets and **3** Layers **C**ontext **M**odel) that integrates the advantages of ontology-based model and hierarchical model. The iMuseum system has two novel features: distributed acquisition of context knowledge on demand and centralized sharing of context knowledge with double-repository. With the two mechanisms, the system is able to support defining new concepts, synthesizing high-level context, and querying application-oriented context at run-time. As a result, the third parties can independently develop their own applications or context providers, and also add or end them optionally. The preliminary experimental results have demonstrated the scalability and acceptance of the system.

1. Introduction

The intelligent museum is the new generation museum following traditional and digital ones [1]. It not only helps visitors receive commentaries of cultural relics conveniently and accurately, but also provides personalization services toward visitors by mobile, multimedia or web technologies to enhance the tourist experience in the museum.

A context-aware system for intelligent museum treats the museum as a smart space. It collects information of visitors and surroundings, recognizes visitors' purposes, and then assists visiting, whiles striving to be minimally intrusive through this process [2]. The system manages all the entities (e.g., applications, sensors, and handheld devices) and context information (e.g., visitor location and profile, surrounding light and temperature, cultural relics, and device capabilities) in the museum. It should support the interoperation between entities and hide the complexity, heterogeneity, and dynamics of the computing environment.

The fundamental difference between an intelligent museum and other smart spaces is that visitors are ambulatory. A visitor may enter or exit at any time, which leads his/her profile and devices to be unpredictable. In addition, to attract visitors, the museum often needs to update tourist programs, and therefore new applications are developed and added into the system by third party service providers at all times. As a result, infinite entities and dynamic context information are necessarily managed by the context-aware intelligent museum system. However, most of exiting systems lack of this scalability, which brings on the problem of difficult, fallible, and time-consuming maintenance.

To address this problem, we propose a scalable context-aware intelligent museum system, namely **iMuseum.** It provides visitors with customized relic information via handheld devices in visiting. It decouples context acquisition from context usage through an underlying context server. By defining pluggable and sharable context vocabularies, third parties can independently develop their own context-aware applications or context providers, and then add or end them at run-time of the system.

The rest of this paper is structured as follows. In Section 2, we discuss related work about intelligent museum systems. In Section 3, details of the iMuseum, including context model, system architecture, and key features are described. Section 4 presents the implementation of the prototype system and experimental results. Finally, Section 5 concludes the paper and points out several directions for future research.

2. Related work

In the past few years, intelligent museum systems have become a hot topic that attracts many researchers' interests.

* Corresponding author. Address: School of Computer Science, Northwestern Polytechnical University, Xi'an, Shaanxi 710072, P.R. China. Tel.: +81 075 753 7480; fax: +81 075 753 7480.
E-mail addresses: yuzhiyong@mail.nwpu.edu.cn (Z. Yu), zhouxs@nwpu.edu.cn (X. Zhou), yu@ccm.media.kyoto-u.ac.jp (Z. Yu), jhpark1@kyungnam.ac.kr (J.H. Park), jianhua@hosei.ac.jp (J. Ma).

0140-3664/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2008.05.004

Several systems provide web services for browsing museum collections through the Web enhanced with personalization technologies. Pechenizkiy and Calders [3] proposed a framework that learns visitor's interest model online and offline. It supports user navigation, information filtering, and other important processes of a user-centered interactive information exchange between museum websites and their visitors. Rutledge et al. [4] presented an approach for determining user interests in a semantically annotated museum collection with the help of interaction dialog. The Rijksmuseum project [5] explores different users' characteristics and personalizes users' museum experiences within the virtual and physical collections. These systems cannot supply visitors with the live experiences in a museum since the digital information is parted from the physical world.

In some other projects, handheld devices are used in museums along with the visiting. The Multimedia Tour [6] provides visitors with rich and captivating audio–visual information about the collection. The MoMo project [7] includes a mechanism for browsing large collections of explanatory items on PDAs and provided social interaction within museums. These systems do not have the feature of context-awareness. They collect limited information of surroundings, and are unable to synthesize multiple contexts into high-level context. As a result, they cannot make services intelligent enough.

A few researchers developed context-aware systems for intelligent museums. Koch and Sonenberg [8] presented a demonstration application that sends tailored content to the visitor's mobile phone in MMS (Multimedia Messaging Service) according to his/her location and profile preloaded. The Remember tool [9] records real-time photographs and typed notes about the visited exhibits, which provides a starting point for later exploration, discussion and reflection on the observed phenomena. However, these systems are all tight coupled architectures. Entities such as sensors and the context manager are not independent in a tight system. In other words, the systems lack of scalability. If a new application is needed to create and deploy, since it is unable to reuse partial modules of former applications, overall redeployment is needed. This is unacceptable in terms of cost and time.

Our work mainly aims to address the scalability issue in a context-aware intelligent museum system. It differs from previous systems in several aspects. First, unlike the systems presented in [3–5], our system is an assistant system for real visiting. Visitors are in a smart physical space instead of virtual space such as on the Web. Second, it is context-aware, i.e., providing adaptive visiting services according to the visitor's changing context. A variety of environmental states can be sensed and synthesized to recognize visitors' high-level context through ontology technologies. Third and the most remarkable, our system is a loose coupled hierarchical system with run-time scalability. It outstands from all tight coupled systems (e.g., [8,9]). It separates context-aware applications from the context server and context providers efficiently by shared context knowledge and simple interfaces.

3. System design

The general principle of our system is that it not only provides client access to retrieve context data, but permits the simple registration of new distributed heterogeneous data sources [10]. It is based on a new context model that benefits from ontology as well as database.

3.1. Context model

The ontology-based model [11] can share vocabularies across different tools and systems, and perform reasoning based on domain knowledge. But it has shortages for its inefficient storing and querying as it is always represented in semi-structural data. The hierarchical model [12] comes from the traditional database that does well in data maintenance and retrieve. However, it is not good at expressing knowledge in a human intelligible way and reasoning from the knowledge. Therefore, we propose a **2** Sets and **3** Layers **C**ontext **M**odel (**2*3CM**), which is illustrated in Fig. 1. It integrates the advantages of ontology-based model and hierarchical model.

We first divide all context knowledge into two sets: concepts and instances. Concepts are used to describe vocabularies, relations or states about a class of individuals, while instances to describe the state of a specific individual. They are from the TBox and ABox of description logic [13] respectively. TBox statements describe a system in terms of controlled vocabularies. They are associated with object-oriented classes and properties. ABox are TBox-compliant statements about that vocabulary, and are associated with instances of those classes.

Then all context knowledge is partitioned into three layers. The basic layer contexts come from data sources directly such as sensors. The high-level layer contexts can not be collected from sensors directly but need some fusion or reasoning, i.e., synthesizing. The application-oriented layer contexts are an extraction from all contexts, which only include the parts needed by a certain application.

Therefore, all context knowledge is classified into six categories: basic concepts (B-C), basic instances (B-I), high-level concepts (H-C), high-level instances (H-I), application-oriented concepts (A-C), and application-oriented instances (A-I). We combine OWL-DL [14] and SWRL [15] to express all context knowledge. OWL-DL is the ontology language with the most powerful expression ability to guarantee the reasoning completeness. SWRL is the semantic web rule language extended from OWL-DL. It can be integrated with OWL-DL seamlessly. It is able to define both rules and queries. These features allow us to express and process all types of context knowledge in a uniform way.

For example, the following segment represents B-Cs defined from a temperature sensor. It denotes a *Thermometer* that can provide temperature integer values.

```
<owl:Class rdf:ID = "Thermometer"/>
<owl:DatatypeProperty rdf:ID = "temperature_is">
    <rdfs:domain rdf:resource = "Thermometer"/>
    <rdfs:range rdf:resource = "&xsd;int"/>
</owl:DatatypeProperty>
```

On the other hand, B-Is come from this temperature sensor will be:

```
<Thermometer rdf:about = "Thermometer_1">
  <temperature_is rdf:datatype = "&xsd;int">30</temperature_is></Thermometer>
```

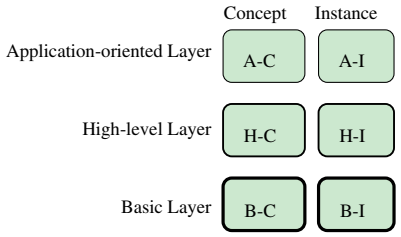

Fig. 1. 2*3CM context model.

It means *Thermometer_1*, which is a particular of *Thermometer*, senses that the temperature is 30°.

An H-C example defined from an application is shown as follows:

Thermometer(?x)∧Room(?y)∧is_in(?x,?y)∧temperature_is(?x,?z)∧swrlb: greaterThan(?z,28) -> is_hot(?y,YES)

This rule is a human readable SWRL sentence, whose syntax also follows OWL. It infers a *Room* is hot if the temperature of a *Thermometer* in the *Room* is greater than 28°.

H-Is can be expressed in the same way of B-Is while A-Cs can be expressed similarly with H-Cs. A-Is need not to be expressed in a formal way as they are sent to applications directly.

3.2. System architecture

There are three types of computing entities in our system (see Fig. 2): context providers, context-aware applications, and a context server.

Context Providers are entities that provide basic instances to the context server in predefined expression and transmission form. They register to the context server by defining new basic concepts, which declare their abilities of context production.

Context-Aware Applications use application-oriented instances to implement particular services adapting to these contexts. They register to the context server by defining new application-oriented concepts, which describe their requirements of context consumption. In the expression of requirements, some rules can be added to define new high-level concepts. The server will employ these user-defined rules to infer high-level instances.

The Context Server is the centric node that manages all entities and context knowledge. All context providers and context-aware applications register to it. The Mapper creates the relationship between provider abilities and application requirements, and generates new application-oriented instances when basic instance update happens. The Synthesizer is responsible for synthesizing high-level instances that applications need. The Ontology Knowledge Base is used for storing context that is expressed and inferred based on ontology.

The key features of our system are two mechanisms: distributed acquisition of context knowledge and its centralized sharing. With these two mechanisms, iMuseum is able to support defining new concepts, synthesizing high-level context, and querying application-oriented instances at run-time. We will describe the details of them in the following sections.

3.3. Distributed acquisition of context knowledge on demand

The mechanism of distributed acquisition of context knowledge on demand ensures that each category of context knowledge comes from the entity with the most competence to define it. It makes the context knowledge sufficient but not redundant. At run-time, six acquisition ways can be used to add new context knowledge into the system:

(1) The server defines some basic concepts at the beginning of the system development, forming as Original Ontology (see Fig. 3). All the following new context knowledge is based on this semantic foundation.
(2) Applications add new application-oriented concepts or high-level concepts, i.e., requirements or rules. Since applications are consumers of application-oriented instances or high-level instances, they have the responsibility of defining what is needed and how to obtain.
(3) Providers add new basic concepts (i.e., abilities) as they encapsulate data sources.
(4) Providers add new basic instances. The providers that correlate with some applications are in active state, otherwise in waiting state. Only the active providers will send new instances when the surrounding changes.
(5) The server infers high-level instances since only the server can collect all basic instances and high-level concepts.
(6) The server generates application-oriented instances. This is because only the server can gather all context instances and application-oriented concepts.

The Global Ontology is defined as the ontology that imports all ontologies in the system. It is updated continuously by adding new knowledge in these six ways on the base of the Original Ontology (see Fig. 3).

Pub/Sub (Publish/Subscribe) communication is adopted to asynchronously transmit all types of context knowledge among various entities. Therefore the system is decoupled in time and space. The server maintains the correlations between requirements and abilities dynamically. It aims to merely subscribe the basic instances related with application layer, only synthesize the high-level instances related with application layer and updated with basic layer, and only query the application-oriented instances updated with basic layer. Hence acquisition of context knowledge on demand is achieved.

3.4. Centralized sharing of context knowledge with double-repository

Sharing of context knowledge provides an approach for various entities distributed in the system to access or update this context knowledge. Knowledge sharing mainly has two functions:

(1) Partial new knowledge is *defined* based on the former knowledge, i.e., importing the old global ontology. This part of knowledge includes some basic concepts, basic instances, high-level concepts, and application-oriented concepts.

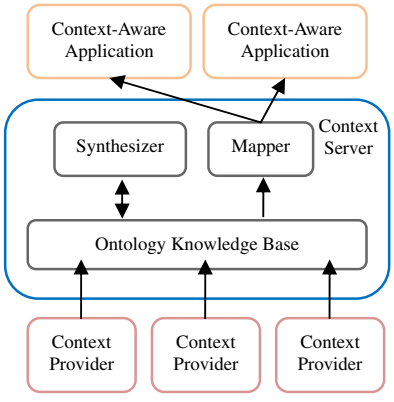

Fig. 2. System architecture.

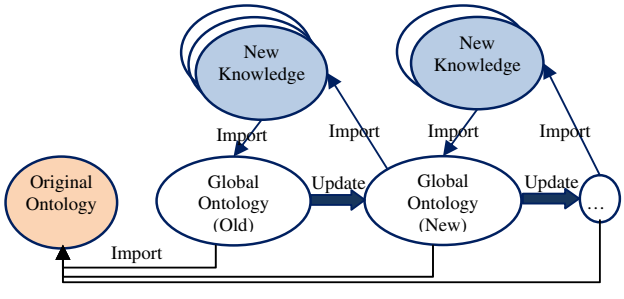

Fig. 3. Run-time updating procedure of Global Ontology.

(2) Partial new knowledge is *discovered* based on the former knowledge, i.e., reasoning (generally means consistency checking, taxonomy classification, synthesizing, and querying) from the old global ontology. This part of knowledge includes high-level instances and application-oriented instances.

In systems that use high-level context, centralized sharing of context knowledge can improve the synthesizing efficiency [10]. The iMuseum system realizes centralized context sharing via a double-repository approach. Fig. 4 shows the schematic architecture. The server maintains a File Repository and a Memory Repository. The file repository offers a reference address pointing to the global ontology. All entities on networks can import existing context knowledge through this address, and then produce new context knowledge. The file repository also provides an uploading address that allows all knowledge (formatted as files) to be uploaded into this repository. The memory repository inputs context from the file repository, checks consistency and classifies taxonomy about the global ontology, synthesizes high-level context instances, and queries application-oriented instances.

Consistency checking consists of two functions. First, it detects classes that cannot have any instances. DL Reasoners can be used to accomplish this based on OWL ontologies. (The SWRL sentences should be excluded before checking, because SWRL is beyond DL.) Second, it determines semantic inconsistencies. Semantic inconsistencies cannot be found by any general reasoners. For example, the system infers that Tom is in Room1 based on the location of his mobile phone, whereas at the same time, his voice is detected in Room2, hence an inconsistency occurs. This kind of cases can be solved by developers of particular applications, e.g. [16].

4. Implementation and evaluation

The iMuseum prototype system is deployed in a museum exhibition room (see Fig. 5). The context server, context-aware applications and other basic services are implemented on the background computer. Every visitor is given a PDA using as an assistant tour tool. It connects with the background computer through a WLAN AP (Access Point). The PDA is equipped with a RFID (Radio Frequency Identification) reader that can detect the tags attached on the relics.

4.1. Software platform

To support multiple applications loading or unloading independently and dynamically, we adopt OSGi [17] to build the software platform on the background computer. Fig. 6 shows the platform architecture. From bottom to top it consists of three layers: physical interface layer, system layer, and OSGi layer. The OSGi layer is composed of service framework and service bundles. The service framework provides a service hosting environment. All services are built and running as bundles on top of the framework.

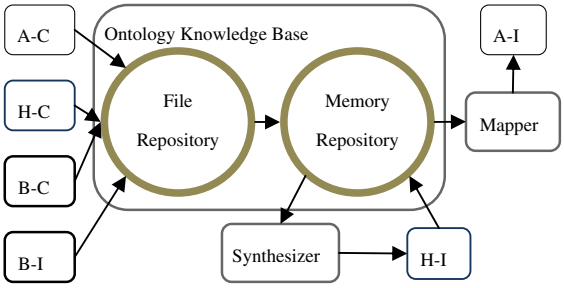

Fig. 4. Centralized sharing with double-repository.

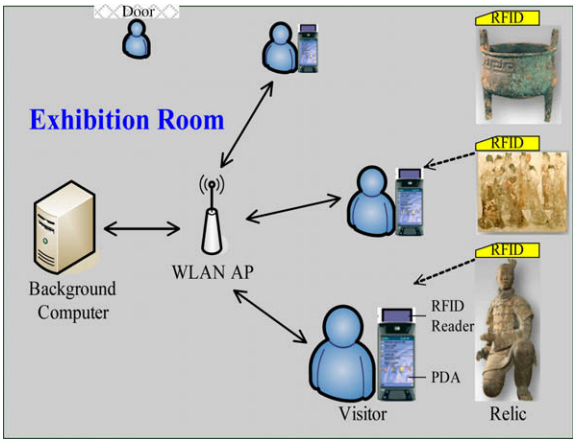

Fig. 5. Prototype system of iMuseum.

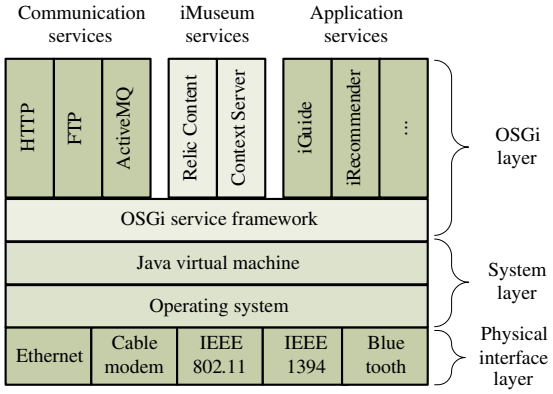

Fig. 6. Background platform based on OSGi.

Three types of services are constructed according to the requirements of iMuseum: Communication services, iMuseum services, and Application services. Communication services are general and basic services including HTTP (offers a reference address to the global ontology), FTP (offers an uploading address for new ontology files), and ActiveMQ [18] (a JMS [19] realization providing Pub/Sub communication). iMuseum services rely on communication services. They are special supporting services including Relic Content and Context Server. The Relic Content refers to the database for storing relic information such as category, dynasty, location, ID number, etc., and media files such as texts, pictures, audios, videos, etc. Application services are context-aware applications relying on both communication services and iMuseum services. Application services include iGuide, iRecommender, and other applications developed by the third parties, which will be described in detail in Section 4.3.

4.2. Context server

The context server is a general functional entity that manages context without focusing on any particular applications. We developed the context server with J2SE 5.0 and Protégé [20]. The whole program is a Java class containing several properties and methods (see Fig. 7).

There are a total of six properties:

AbilityList – It is used to store objects of Ability. When a provider registers to the server, an Ability object is created and added into this ability list.

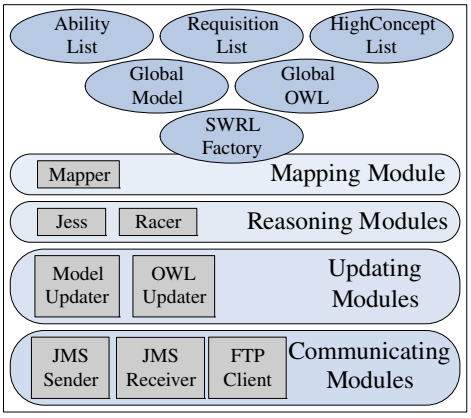

Fig. 7. Components on the context server.

RequisitionList – It is responsible for storing objects of Requisition. When an application registers to the server, a Requisition object is created and added into this requisition list.
HighConceptList – It is utilized to store objects of HighConcept. If a new high-level concept is defined during the application registration, a HighConcept object is created and added into this high-level concept list.
GlobalModel – It contains all fresh context knowledge used for context reasoning.
GlobalOWL – When a new OWL file is added in the system, the global OWL files must be updated to import this new file.
SWRLFactory – It can extract SWRL sentences from the *GlobalModel* to create Ability objects, Requisition objects, and HighConcept objects.

There are four types of methods correspondingly:

The mapping module (*Mapper*) utilizes the *SWRLFactory* to maintain *AbilityList*, *RequisitionList*, and *HighConceptList*, and also construct relationships between them.

The reasoning modules (*Jess* [21] and *Racer* [22]) reason from the *GlobalModel*, i.e., checking consistency, classifying taxonomy, and processing SWRL rules and queries.

The updating modules (*ModelUpdater* and *OWLUpdater*) update the *GlobalModel* and the *GlobalOWL* when an OWL file is added or updated.

The communicating modules (*JMSSender*, *JMSReceiver* and *FTPClient*) are responsible for information exchanging with outside, e.g., applications, providers, and the FTP server.

4.3. Applications

We have developed two context-aware applications: iGuide and iRecommender. When a visitor is interested in a cultural relic nearby, the iGuide will automatically play corresponding multimedia commendatory of this relic on the visitor's PDA. The iRecommender can recommend related relics that a visitor might be interested in after he/she views some relics. The relics recommended are sorted by relevancy and indicated with their locations. These two applications were proposed mainly for illustrating the scalability of our framework. The approach for capturing contexts is not the focus of this paper. In iGuide, the user's location (near which relic) and the staying time were acquired through a RFID system. In iRecommender, the user viewing history was recorded in order for inferring which type of relics the user might like most. User preference also can be incorporated in the recommendation, which can be learned using different techniques, e.g. [23,24].

Taking the iGuide as an example, we here describe the running procedure of an application to demonstrate the scalability of the iMuseum system.

Step 1: Communication services and iMuseum services start up first. At this moment the context knowledge only contains basic concept, "Visitor" (see Fig. 8a), which comes from the original ontology.
Step 2: Then the PDA (installed with a RFID reader, an ID provider, and the iGuide client) starts up, and declares its abilities of context production at the same time. Here the context knowledge expands with basic concepts "Relic", "localize_near' and "localize_for" (see Fig. 8b).
Step 3: iGuide launches and indicates its requirements of context consumption. At this time the context knowledge grows to that shown in Fig. 8c. The high-level concept "interest_in" (e.g., if the visitor has localized near a relic for more than 5 s) and corresponding application-oriented concept are defined.
Step 4: The server determines that context abilities can satisfy context requirements, and then subscribes context updates from the ID provider.
Step 5: A visitor holding this PDA approaches a relic attached with a RFID tag, which can be detected by the RFID reader. In this case, the ID provider sends the tag's ID and continuance time to the server. The context knowledge may contain the basic instance like "visitor Visitor_1 localizes near relic Relic_1 for 7 s" (Fig. 8d).
Step 6: The server infers whether the visitor is interested in the relic. If interested, the context knowledge changes to the one shown in Fig. 8e. The high-level instance, "Visitor_1 is interested in Relic_1" is synthesized. Then a query is triggered to obtain application-oriented context instances and send them to iGuide.
Step 7: iGuide retrieves the corresponding multimedia commentaries of this relic, and delivers them to the visitor's PDA.

There are different modal commentaries available for each relic, which could be used according to different conditions. For example, when a visitor wants to know more detailed information and the network speed is very slow, he/she can view the text description shown as Fig. 9a. A child would be interested in the image one (see Fig. 9b). And the video commentary (see Fig. 9c) is more attractive but needs broader network bandwidth.

If a visitor who holds a new PDA enters the exhibition room, he can use the service of iGuide just by registering as described in Step 2. If a new application is developed at run-time, it can be added into the system by registering as Step 3. Visitors or applications also can leave the system with little influence on others. Along with these dynamic changes, context knowledge increases or decreases at run-time.

4.4. Evaluation

We evaluated iMuseum in terms of time performance and user acceptance. The background computer is a PC with Intel Pentium CPU 3.0GHz, 1.0GB RAM, running MS Windows 2000. The visitor's handheld device is a PDA (HP iPAQ hx2490b) running MS Windows CE 5.0. The context server involves three phases that are time-consuming: initialization, entity registration, and context update. The average running time of 20 tries is shown in Table 1.

Time increases slightly when a new piece of knowledge (3.5 KB) is added into. The reason is that our KB is very small (41 KB), and time of processing different sizes of KB is not substantially different. We can see that the initialization time is relatively long (about 5 s). In this phase, all components on the context server are created and pre-

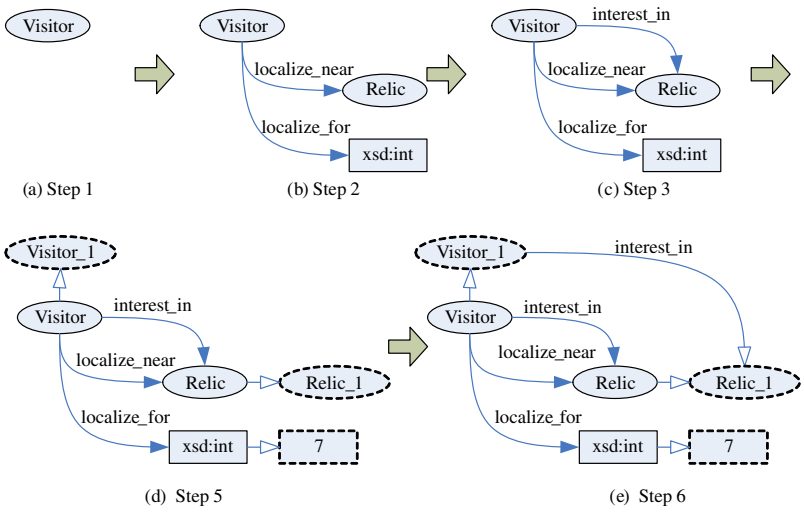

Fig. 8. Context knowledge update of iGuide.

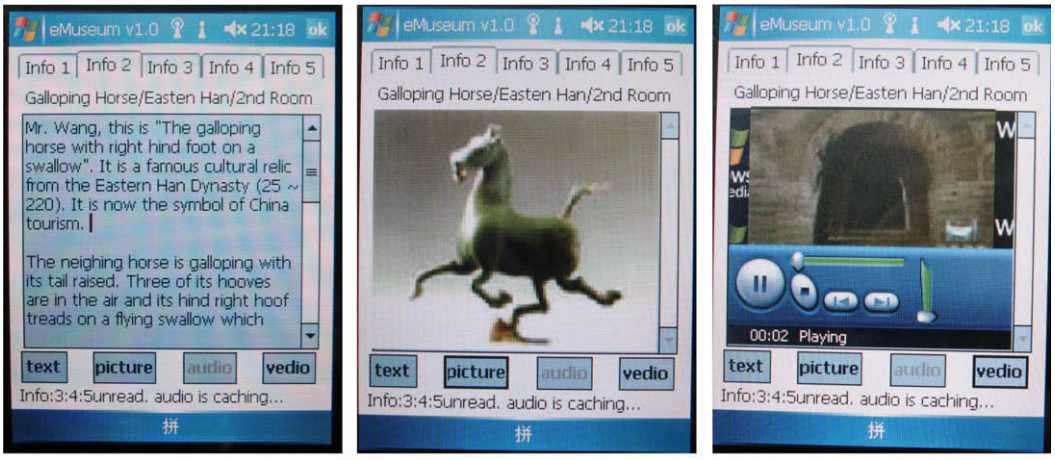

Fig. 9. Client side snapshots of iGuide.

Table 1
Running time of context server in different phases

Phase	Initialization	Entity registration	Context update
Time (ms)	4969	422	813

pared. Fortunately, this overhead is merely generated during the system startup and it is transparent to visitors. It does not affect the performance of other services at running time. Both response time of entity registration and context update is shorter than 1 s. It adequately satisfies the demand of dynamical registration and immediate context-awareness. Hence the time overhead is acceptable.

From the user's perspective, we conducted a user study to evaluate the user acceptance of our system. Ten participants from the Northwestern Polytechnical University, China were invited to visit the intelligent museum. Each person was given a PDA as client side to run two applications (iGuide and iRecommender). After visiting, they completed a questionnaire. The questions and results are shown in Table 2.

All the subjects were satisfied with the multimedia contents provided by iGuide, and the response time was also acceptable.

Table 2
User study results

Application	Question	Average rating*
iGuide	The multimedia contents of relics were attractive for me.	5.0
	The distance to sense relics was suitable.	3.5
	I was satisfied with the response time.	4.0
iRecommender	The recommended relics are exactly the ones I want to see.	4.5
	The indication of relic's location is very useful to find it quickly.	4.8
Both	The interface and operations of the system were easy to master.	3.7
	I would use this tool again for its helpfulness in visiting.	5.0

* 5 = strongly agree; 4 = agree; 3 = neutral; 2 = disagree; 1 = strongly disagree.

But they had different opinions about the distance to sense relics. Several participants complained that they had to be very near to the relic and hold the PDA in certain direction to detect the attached tag. The reason is that there is a fixed identification distance

(about 1 m in our prototype) between RFID tags and the reader. The signal is easily influenced by metal objects. It can be improved by choosing another type of RFID system with better performance. Both the recommended relics and location indications provided by iRecommender were interesting and useful. Some visitors suggested that this tool should be operated without a pen, i.e., can be controlled by speech or totally automatically. We can add a speech recognition engine into the PDA to meet this requirement. Despite these issues, all of the participants indicated that they would use this tour assistant tool again.

5. Conclusion and future work

In this paper, we proposed a scalable context-aware intelligent museum system called iMuseum. It captures context about visitors and surroundings and then provides users with appropriate relic information. It addresses the application scalability and visitor ambulation issues in museum. The main contributions include: (1) introducing a 2 Sets and 3 Layers Context Model (2*3CM) that supports uniform expression and various types of context knowledge integration by combining ontology-based model and hierarchical model; (2) proposing mechanisms of distributed acquisition and centralized sharing of context knowledge, which enables run-time new concepts definition, high-level context synthesizing, and application-oriented query; (3) presenting a context-aware application iGuide to illustrate the running process of the system and its scalability.

There are still several issues not solved in the current system. First, since some basic concepts are defined by new distributed entities, their semantics would not be well understood. Second, the Mapper in the context server requires efficient algorithm to construct the relationship between abilities and requirements. In the future, we will conduct research on these issues. We also plan to integrate some other recommendation methods [25] to enrich personalization services and consider the common preferences among group members so as to recommend museum collections to a group of visitors [26].

Acknowledgements

This work was partially supported by the High-Tech Program of China (863) (No. 2006AA01Z198) and the Innovation Fund of Northwestern Polytechnical University (NPU) of China (No. 2006CR13). It was also partially supported by Kyungnam University of Korea. The authors thank all the members from Ubiquitous Computing Lab of NPU for their discussion and implementation of the system.

References

[1] P. Busetta, M. Merzi, S. Rossi, et al., Group communication for real-time role coordination and ambient intelligence, 2003. Available from: <http://citeseer.ist.psu.edu/article/busetta03group.html>.
[2] M. Satyanarayanan, Pervasive computing: vision and challenges, IEEE Personal Communications 8 (4) (2001) 10–17.
[3] M. Pechenizkiy, T. Calders, A framework for guiding the museum tours personalization, in: Proceedings of Workshop on Personalization Enhanced Access to Cultural Heritage Joint with the 11th International Conference on User Modeling, 2007, pp. 11–28.
[4] L. Rutledge, L. Aroyo, N. Stash, Determining user interests about museum collections, in: Proceedings of the 15th International Conference on World Wide Web (WWW'06), 2006, pp. 855–856.
[5] L. Aroyo, P. Gorgels, Y. Wang, et al., Personalized museum experience: the Rijksmuseum use case, in: Proceedings of Museums and the Web 2007 (MW'07), 2007. Available from: <http://www.archimuse.com/mw2007/papers/aroyo/aroyo.html>.
[6] G. Wilson, Multimedia tour programmer at Tate Modern, in: Proceedings of Museums and the Web 2004 (MW'04), 2004. Available from: <http://www.archimuse.com/mw2004/papers/wlson/wilson.html>.
[7] J. Jaen, J.M. Esteve, J.A. Mocholi, et al., MoMo: enabling hybrid museums, IEE Proceedings Software 152 (5) (2005) 245–251.
[8] F. Koch, L. Sonenberg, Using multimedia content in intelligent mobile services, in: Proceedings of the WebMedia & LA-Web 2004 Joint Conference 10th Brazilian Symposium on Multimedia and the Web 2nd Latin American Web Congress (LA-Webmedia'04), 2004, pp. 41–43.
[9] M. Fleck, M. Frid, T. Kindberg, et al., Rememberer: a tool for capturing museum visits, in: Proceedings of the 4th International Conference on Ubiquitous Computing (Ubicomp'02), 2002, pp. 48–55.
[10] M. Baldauf, S. Dustdar, F. Rosenberg, A survey on context-aware systems, International Journal of Ad Hoc and Ubiquitous Computing 2 (4) (2007) 263–277.
[11] X. Wang, D. Zhang, T. Gu, et al., Ontology based context modeling and reasoning using OWL, in: Proceedings of the 2nd IEEE Conference on Pervasive Computing and Communications (PerCom'04), 2004, pp. 18–22.
[12] A. Schmidt, A layered model for user context management with controlled aging and imperfection handling, in: Proceedings of the 2nd International Workshop on Modeling and Retrieval of Context (MRC'05), 2005, pp. 86–100.
[13] F. Baader, D. Calvanese, D. McGuinness, et al., The Description Logic Handbook: Theory, Implementation and Applications, Cambridge University Press, 2003.
[14] M. Dean, G. Schreiber, S. Bechhofer, et al., OWL Web Ontology Language Reference, 2004. Available from: <http://www.w3.org/TR/owl-ref>.
[15] I. Horrocks, P.F. Patel-Schneider, H. Boley, et al., SWRL: a semantic web rule language combining OWL and RuleML, 2004. Available from: <http://www.daml.org/rules/proposal>.
[16] C. Xu, S.C. Cheung, Inconsistency detection and resolution for context-aware middleware support, in: Proceedings of the 10th European Software Engineering Conference Held Jointly with 13th ACM SIGSOFT international Symposium on Foundations of Software Engineering (ESEC/FSE-13), 2005, pp. 336–345.
[17] D. Marples, P. Kriens, The open services gateway initiative: an introductory overview, IEEE Communication Magazine 39 (12) (2001) 110–114.
[18] ActiveMQ. Available from: <http://activemq.apache.org>.
[19] JMS. Available from: <http://java.sun.com/products/jms>.
[20] Protégé. Available from: <http://protege.stanford.edu>.
[21] Jess. Available from: <http://www.jessrules.com/jess>.
[22] RacerPro. Available from: <http://www.racer-systems.com>.
[23] Z. Yu, D. Zhang, X. Zhou, C. Li, User preference learning for multimedia personalization in pervasive computing environment, in: The 9th International Conference on Knowledge-Based Intelligent Information & Engineering Systems (KES 2005), pp. 236–242.
[24] V. Schickel-Zuber, B. Faltings, Inferring user's preference using ontologies, in: Proceedings of American Association for Artificial Intelligence 2006 (AAAI'06), 2006, pp. 1413–1418.
[25] Z. Yu, X. Zhou, D. Zhang, et al., Supporting context-aware media recommendations for smart phones, IEEE Pervasive Computing 5 (3) (2006) 68–75.
[26] Z. Yu, X. Zhou, Y. Hao, et al., TV program recommendation for multiple viewers based on user profile merging, User Modeling and User-Adapted Interaction 16 (1) (2006) 63–82.