A Context-Aware Approach Enhancing XML Semantics Integration

Dershing Luo[1,2] Ching-Cha Hsieh[1]
dsluo@cc.chit.edu.tw *cchsieh@im.ntu.edu.tw*
[1] *Department of Information Management, National Taiwan University, R.O.C.*
[2] *Department of Information Management, China Institute of Technology, R.O.C.*

Abstract

XML is designed to facilitate data exchange between applications. However, XML and its schema languages do not express semantics but rather the document structure, such as the nesting information. Therefore, in the integration of XML, we should emphasize the semantics more than the document structures. In this paper, we provide a context aware framework, named BICAA, which aims to enhance the XML semantic integration. In BICAA, we generate both DAML+OIL context ontology and context instances. Through this bidirectional approach, we hope to achieve better context-aware computing in the XML semantic integration.

Keywords: *ontology, XML, semantic, context, DAML+OIL*

1. Introduction

As Tim Berners-Lee mentioned, the semantic web is a meta-web, building on the WWW, which enables web content for machines which are both accessible and interpretable (W3C, 2001). How can machine**s** *understand* the semantics? These machines may link to the relevant web or resources through an RDF (resource description framework) and a URI (universal resource index). Then it may locate the keywords through the hyperlink. Finally, it may define the keywords through ontology and make some inferences (Huang, 2002). Thus, in the integration of XML, we should emphasize the semantics more than the document structures. However, the real semantics are based on not only an ontology, but also depend on the context (Madnick, 1996).

In this paper, we provide a context aware framework that aims to enhance the XML semantic integration. The remainder of this paper is organized as follows. Section 2 introduces context and context-aware computing. Section 3 describes techniques supporting context-aware computing. In section 4, we propose a context aware framework that aims to enhance the XML semantic integration. Finally, conclusions and future work are given in Section 5.

2. Context and Context-Aware Computing

The term *"Context"* is overloaded with a wide variety of meanings, depending on the purposes of the particular application and/or on the research community standpoint (Pokraev et al., 2003). The domain is so versatile**,** as in information bases, artificial intelligence, or ubiquitous computing, etc. However, a more generic definition (more operational approach) is as follows: "*Context is 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."*

In a traditional application, we can hardly handle different contexts. While adding on context-aware computing, as Figure 1 shows, we may process such context as state of users, state of physical environment, state of computational environment, and history of user-computer interaction, etc.

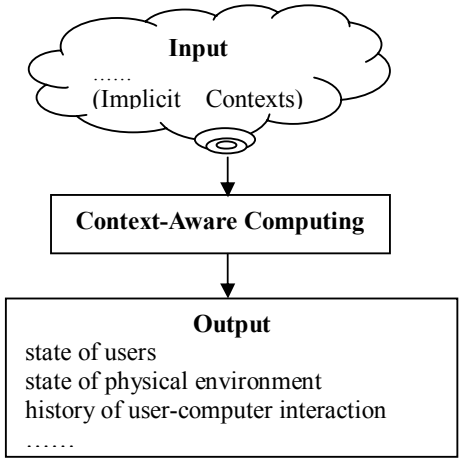

Figure 1. Context-aware applications (Re-drawn from Costa, 2003)

Pokraev et al (2003) gave a more general definition of context-aware computing: *A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task*.

In general, context-aware computing falls into two categories: one is using context as one of its parameters, the other is adapting the system to context changing. To sum up, a context-aware system should detect, interpret and respond to the context.

The strategy for context-aware computing may include the following stages: gathering/sensing, modeling, storing, distributing, and monitoring context. These stages should justify the need for proper architectural support (Costa, 2003; Gandon & Sadeh, 2004). The Ontology based approach is commonly employed.

The sensing stage may be divided into three phases; the first phase, the sensing phase is to sense the low-level context, such as the location (formats, determination), proximity, time, user status, device, network, etc. The second phase, the building of a higher-level context information phase is to model current activity, social context, and physical context. The last phase is sensing context changes, such as continuous sensing, event-based sensing.

The modeling stage describes the nature of contextual information. The primitive contexts are: locations, users, objects/devices, time, and a combination of the techniques for device modeling and service modeling. A context representation should be structured, interchangeable, composable/ decomposable, uniform, extensible, standardized.

The context monitoring stage could be achieved through the following steps: generation, processing, dissemination, and presentation. The requirements for a context monitoring service are: filtering and correlation, scalability, dynamic changes, and intrusiveness.

3. Techniques Supporting Context-Aware Computing

The current architectures and elements supporting context-aware computing are described in this section.

3.1 Current Architectures Supporting Context-Aware Computing

We surveyed several architectures which support context-aware computing. The conceptual frameworks define a conceptual basis to support the development of context-aware and adaptive systems, as well as the applications. Such examples are the Cooltown project developed by Hewlett-Packard and the context toolkit designed by Georgia Tech.

The service platforms aim at the rapid creation and deployment of context-aware services. They also offer dynamic service discovery, dynamic deployment of adaptive applications addressing issues of scalability, security and privacy. Similar architectures are platforms for adaptive applications completed from the Lancaster University, U.K., and M3 architecture proposed by the University of Queensland, Australia.

The appliance environments are supporting interoperability among collections of appliances. A Universal Information Appliance (UIA) by IBM and Ektara by MIT are two of the samples.

The Computing Environments address the ubiquitous computing goal. There are two example frameworks, these being the Portolano framework proposed by the University of Washington and PIMA framework designed by IBM.

From the point of view of context awareness issues, Costa (2003) has made a comparison between the above architectures, as shown in Table 1. The issues compared in Table 1 are all context awareness related. The fourth one is especially for the supporting issues applied to distributed systems. Through the comparison, we can conclude that some of the issues are present in the above research projects, though it is still challenging to develop an infrastructure that integrates solutions to all the presented issues.

Table 1. Comparisons between the initiatives on the context awareness (Source: Costa, 2003)

Issue	Conc. Frameworks	Service Platforms	Appliance Environments	Computing Environments
Support for device heterogeneity			•	
Support for device mobility			•	
Management of application mobility and distribution				
Support for context-aware issues	•	•		
Support for adaptation		•		•
Support for rapid development / deployment of applications		•		•
Management of user context	•			
Support for user mobility				

Another context-aware architecture is WASP, which stands for Web Architectures for Services

Platforms project. WASP is proposed by the University of Twente, Enschede, Netherlands and funded by the Freeband Knowledge Impulse joint initiative of the Dutch government, knowledge institutes and industry (Costa, 2003; Costa et al., 2004; Pokraev et al., 2003).

The WASP platform can gather contextual information from different sources and adapt them according to the user needs and system capabilities.

There are several possible models of information provisioning, e.g. request-response, time-driven, event-driven. The *context interpreter* gathers contextual information from context providers (sensors or third party context providers) which may use different communication protocols and semantically different contextual representation, making contextual information uniformly available to the platform. Therefore, it is responsible for tackling the support for different kinds of context provider requirements. The three main parts are for *context gathering, for the aggregator and for the inference machine.*

Above them, the *context inference* is accomplished by a *context interpreter* with a variety of sources such as *context provider, user profile, and entity repository*. The *context interpreter* will provide information to the *subscription manager.* The *user profile* may be built with elements, which are explicitly and/or implicitly provided by the user, these being history, ratings, interests, preferences, characteristics, and identities (Pokraev et al., 2003). The *entity repositories* are the architectural components that contain and maintain the information represented in the data entity model. The entity types, entity instances, action types and function types are essential information to allow dynamic deployment of applications through the subscription language.

The core of the WASP is the *monitor module*. It is responsible for interpreting and managing the application subscriptions. Therefore, it tackles the requirements reactive behavior and coordination among different applications. In order to perform its operations, the *monitor module* makes use of the data available in the repositories and the contextual information is provided by the c*ontext interpreter*.

Figure 2 shows the example of configuration with entities for a mall and an employer together with their relationship with context location. The UML diagram depicted in Figure 2 describes a possible configuration of the context model proposed for the WASP platform, called the *entity model.*

To sum up, the above comparisons and critical elements of SWAP show the following three addressing issues. One is related to building a ubiquitous infrastructure, i. e. a dynamic deployment of context-aware applications on top of a services platform. The second issue is that most of the existing systems are based on ad-hoc context models, which lack much of the desired flexibility, expressiveness and interoperability**,** making it hard for the development of abstractions and tools. The final issue is that none of the existing context-aware applications, except the WASP, deal with privacy of the users, which is a critical issue.

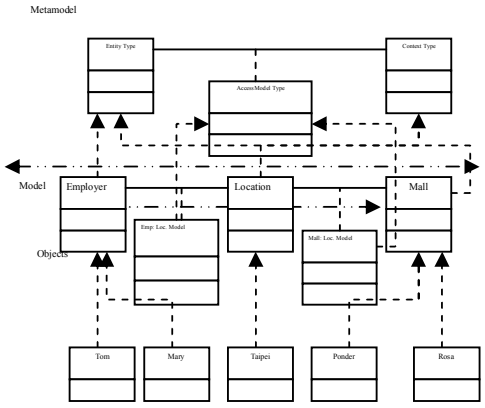

Figure 2. Object-model- metamodel instantiation levels

3.2 Ontology Languages

There are seven famous languages which support ontologies as shown in Figure 3. They are: Ontology Exchange Language (XOL), Simple HTML Ontology Extension (SHOE), Ontology Markup Language (OML), Resource Description Framework (RDF) and also RDF Schema, Ontology Interchange Language (OIL), DARPA Agent Markup Language (DAML) + OIL, and Web Ontology Language (OWL). Gómez-Pérez and Corcho (2002) compared the ontology languages for the semantic web from the following dimensions: concepts, relations, functions, axioms and instances. From their comparisons, the DAML+OIL offers intensive support as well as its supporting software, such as OILEd, WebODE, etc.

The DARPA Agent Markup Language (DAML) Program officially began in August 2000. Its goal is to develop a language and tools to facilitate the concept of the Semantic Web. McGuinness et al. (2002) have a concrete analysis and introduction on DAML+OIL. Their point is the DAML+OIL is a web transformation from a message representation forum to an interoperatable, understanding and inferencing resource.

The implementation of a context-aware application should allow the acquisition of

contextual information, consider it, and then adapt itself to changing the context. This implementation may use ontologies described in DAML+OIL to define different types of contextual information. Pokraev et al. (2003) stated that these ontologies describe different kinds of applications, services, devices, users, data sources and other contextual agents. Ontologies are also used to define the structure of contextual information. These are useful for checking the validly of context information, assuring that the system has the same semantic understanding of different pieces of information.

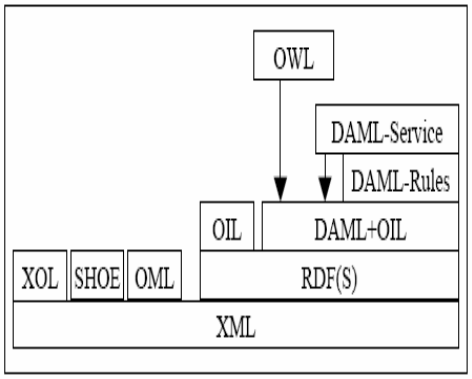

Figure 3. The ontology languages for the semantic web (Source: Yu, 2004)

4. The Proposed Context-Aware Framework

Since the semantic web is rapidly becoming a reality through the development of semantic web markup languages, such as DAML+OIL, it follows that a fundamental component of the semantic web is the markup of web services to make them computer-interpretable, use-apparent, and agent-ready. The ontology based approach for context integration is widely used (Pokraev et al., 2003).

From the above discussion, we may conclude that there are at least two methods of context integration. Our proposed framework is based on the following two methods. One is the top-down approach, in which the Context-Ontology building and User Profiles are employed. The other is the bottom-up approach, which employs a Repository of Context-Instances. This framework is named BICAA, short for bidirectional context-aware approach.

Concerning the technology applied, current web service standards, such as UDDI or WSDL are not sufficient (Gandon & Sadeh, 2004). Separating the process of acquiring contextual information from actual context-aware applications are keys to facilitating application development and maintenance (Gandon & Sadeh, 2004; Yu, 2004).

Semantic markup languages such as DAML-S are as well as early efforts to define Web Service ontologies and markup in the context of languages. The DAML + OIL is popularly used (Gandon & Sadeh, 2004; Hsu, 2003) More recently, OWL has been used to represent contextual information (e.g. location, calendar activities, social and organizational relationships, etc.), privacy preferences, on Semantic Web service concepts to support the automated discovery, and access of personal and public resources.

At the same time, the occurrence of (a combination of) events may allow users to set reminders to be triggered (Costa, 2003). Keeping history of contextual information is particularly interesting to allow context inference based on past occurrences, e.g., the inference of the speed of a user from the latitude and longitude changes over time. Such applications are those such as medical, security, bus, etc.

In our proposed framework, we generate context ontology and user profile with DAML+OIL, and context instances with an XML repository.

5. Conclusions and Future Work

In this paper, we have presented a context-aware approach based on DAML+OIL to support semantics integration. This approach, BICAA, employs a combination of two approaches. One is **a** top-down approach, in which the *Context-Ontology* building and *User Profiles* are employed. The other is a bottom-up approach, which employ**s** a *Repository of Context-Instances*. The applied technologies are semantic markup languages as well as early efforts to define Web Service ontologies and markup in the context of languages, DAML + OIL. The proposed framework could improve the context-aware computing in the XML semantic integration.

The future work will be an implementation of a prototype and its evaluation under some applied domains. Such work could have a better understanding in context-aware computing to enhance the XML semantic integration.

6. References

Costa, P. D. (2003). *Towards a Services Platform for Context-Aware Applications*. Thesis for a Master of Science degree in Telematics from the University of Twente, Enschede, The Netherlands.

Costa, P. D., Ferreira Pires, L., van Sinderen, M., and Pereira Filho, J. G. (2004). Towards a Services Platform for Mobile Context- Aware Applications. In Proceedings of the First

International Workshop on Ubiquitous Computing - IWUC 2004, (Eds: Mostefaoui, Maarmar and Rana).

Gandon, F. L. & Sadeh, N. M. (2004). Semantic web technologies to reconcile privacy and context awareness. *Web Semantics: Science, Services and Agents on the World Wide Web. 1*. pp. 241–260.

Gu, T., Pung, H. K. & Zhang, D. Q. (2005). A service-oriented middleware for building context-aware services. *Journal of Network and Computer Applications. 28*, pp. 1–18.

Gómez-Pérez, A. and Corcho, O. (2002). Ontology Languages for the Semantic Web, *IEEE Intelligent Systems, January/February 2002*, pp.54-60.

Hsu, S. H. (2003). The Annotation and Access Based on the Ontology. The Master Thesis of Department of Information Engineering, National Don-Hwa University.

Huang, J. Z. (2002). The Future Direction on Programming Linguistics and Chinese Language Processing: The Semantic, Phrase Net, and Ontology. Presentation for The COLING2002 in Taipei.

McGuinness, D. L., Fikes, R., Hendler, J. & Stein, L. A. (2002). DAML+OIL: An Ontology Language for the Semantic Web. *IEEE Intelligent Systems, September/ October 2002*, pp.72-80.

Madnick, S. (1999). Metadata Jones and the Tower of Babel: The Challenge of Large-Scale Semantic Heterogeneity. *Proceedings of the 1999 IEEE Meta-Data Conference, April 6-7, 1999*, pp. 1-13.

Pokraev, P., Costa, P. D., Pereira Filho, J. G., Zuidweg, M., Koolwaaij, J. W. & van Setten, M. (2003). *Context -aware services state-of-the-art*. Telematica Instituut, Ericsson, CTIT.

W3C (2001). *Semantic Web*. Retrieved Apr 12, 2005, from the World Wide Web: http://www.w3.org/2001/sw/.

Yu, I. J. (2004). The Ontology in the Semantic Web and Blog. *Technical Report, Dec. 2004*.

7. Referenced Websites

DAML+OIL. http://www.daml.org/2001/03/daml+oil-index.html

Ontology Inference Layer (OIL). http://www.ontoknowledge.org/oil/