CRL: A Context-aware Request Language for Mobile Computing

Alvin T.S. Chan, Peter Y.H. Wong, Siu-Nam Chuang

Department of Computing, The Hong Kong Polytechnic University
Hung Hom, Kowloon, Hong Kong
{cstschan, csyhwong, cssiunam}@comp.polyu.edu.hk

Abstract. This paper introduces an XML-based generic Context Request Language (CRL), whose construction is part of a web services framework in the domain of mobile context sensing. The paper describes an implementation of the technique that is in accordance with the formal mathematical representational model, using first-order temporal language [6]. The language is an attempt to introduce intelligence into context-aware computing by defining context-sensing elements into logical entities. The use of first-order calculus in this language definition serving on web service technology allows users to utilize context aggregation and to embed user control in contextual information. By carrying out on-the-fly context inferences at the middleware level, we can achieve a complete separation of concerns between user application and context sensing. Moreover, the declaration of contextual knowledge based on situations and events within the predicate domain allows users to express changes in contextual information and to quantify these elements among times and durations.

1 Introduction

A review of current context-aware applications [1], [2], [3], [4] suggests some important areas in designing context-aware applications still need to be addressed. In particular, we believe there needs to have a methodology that allows applications to utilize and intelligently reason about contextual semantics and achieves a complete separation of concerns between applications and contextual information. Such methodology is invaluable in making intelligent context-aware applications. In this paper we describe our attempt to implement a generic rule-based language called **C**ontext **R**equest **L**anguage (**CRL**) that provides a solution for an intelligent technique to context sensing. Our effort focuses on incorporating temporal predicate calculus, which is used in agent planning [5], with an XML-based syntax for extensibility [6]. This language allows applications to specify inferences for monitoring contextual information. The use of predicate calculus in this language definition enables users to utilize context aggregation and precise control over contextual information. CRL forms an integral part of our new Web Services architecture, CRL Framework, which supports dynamic reflective reconfiguration, asynchronous communication, and predefinition of context composition through CRL-rule definitions. In this architecture we have adopted a layered approach to relax applications participation in context sensing while still allowing applications to gain control over their context requirements. This layered approach enables the separation of mobile applications and their surrounding environment. Its key layers and components are briefly described as follows:

- **Application Layer.** This describes mobile and remote clients such as PDAs, Pocket PCs, etc. This layer is where local applications conceptually sit and they can either be dependent or independent of the context environment.
- **Web service requester.** This is a client agent that uses a stub that discovers context-aware web services through service registry [7]. It also hosts the Context Request Builder for constructing CRL-instances. A CRL-instance is an XML document that can either be a real-time construction of inference rules about one or more contexts or a specific definition for a particular common environment that is sometimes referred as smart space [4]. These CRL-instances are then enveloped as SOAP messages and transported to the particular web service provider that provides context inference via HTTP.
- **Web service provider.** This is a mediator between the context requester and the contextual environment. As it receives an SOAP call from a remote application, it uses the Context Request Parser (CRParser) that validates both the semantic and the syntax of the CRL-instances against the CRL schema, and transforms it into a set of CRL-based inference rules before passing it for context sensing.
- **Context Layer.** This layer contains a set of components that offers context-sensing ability at the hardware level and context inferences using the CRL-inference engine. There is also a CRL-rule repository that contains common CRL-rules so that applications are able to refer to pre-defined instances in the repository using key referencing.
- **CRL-rule Repository.** It contains CRL-rules used in smart spaces [4], such as an intelligent meeting room or a smart vehicle. Since CRL-instances are written based on the CRL XML schema, these rules are intended to be organized in a tree structure. The process of updating a CRL-instance on the context layer is thus modeled as a specific algorithm in relation to the addition and deletion of nodes in a tree.
- **CRL-Management Module.** This module is part of the Context Layer and it contains a collection of components that provides administrative functionalities such as registering context sensors, providing meta-controlling, processing notification policy.

This architecture approach addresses the important issues of application participation. Using SOAP allows clients' applications to submit CRL-instances asynchronously, which means that applications are not engaged in listening to contextual feedback from the Context Layer. This is an important feature which provides the separation of concerns. The second feature is allowing user to embed predicate logic into each CRL-instance. This enhances the inference process at the context layer. The notion of reflective reconfiguration, inspired by the unique approach employed in MobiPADS [1], allows CRL-rules, either parsing through the CRL-inference engine or residing at the CRL-rule repository, to be updated at real time. Thus a highly transparent framework that does not compromise the preferred feature of separation of concerns is thus achieved.

2 An Overview of CRL

CRL plays an important role in CRL Framework, enabling it to reason about context. The complete CRL is a collection of language grammars expressed in XML for context request definition, context definition and meta-control definition. During the construction of the CRL framework we have developed a total of one core inference language (CRL-instance) and six other supplementary languages.

CRL-instance is the core language component for constructing context request rules. Its grammar is in accordance with temporal predicate calculus [5]. We have chosen to use temporal predicate calculus as the underlying logic for the following reasons:

- It offers basic propositional logics that are common to all inference systems
- Unlike the logical systems used by other context-aware application, predicate calculus extends the premature propositional logics in that it provides a mechanism to express, and reason generically which is the key advantage of CRL.
- Furthermore, temporal predicate calculus extends predicate calculus to provide a mechanism for reasoning about contextual information and their changes at different time intervals.
- It offers a set of fundamental control structure to increase the flexibility and the complexity of CRL-instance.
- It provides a formal mathematical base in context-awareness which is novel in the field of context-aware mobile computing.

By encapsulating a proven logical system into a standard language such as XML, CRL-instance becomes a truly generic, platform independent and flexible rule-based language. The aims of CRL-instance are to provide the following:

- A language for users/applications to construct context request rules to reason about past and present contextual information.
- A set of control structures for users/applications to condition the enquiry of present and future contextual information.
- A mechanism for users or applications to specify sequential and concurrent sensing of context.
- A facility for users or applications to actively interact with the context environment using user-specific actions.
- A well-known syntax for users and applications to construct context inference rules.

Up to six supplementary languages are defined under the **CRL$_{\text{sup}}$** language group. They are system languages designed to enable meta-control, flexible user context feedback and error handling. They are briefly described as follows:

- **CRL-rule.** This language is designed to cache common CRL rules in the CRL-rule repository used in smart spaces [4].
- **CRL-feedback.** This is used to assist the transitioning of context information back to the Application Layer.
- **CRL-control.** This is a meta-language that helps the monitoring and the manipulation of CRL rules that have already been defined. This enables meta-

adaptation and therefore ensures the CRL framework is working in a conflict–free environment.

- **CRL-CTree.** This defines the context entities that a CRL-inference engine monitors.
- **CRL-user.** This defines users' hierarchy, including user-specific information.
- **CRL-error.** This language is used to assist the CRL framework in error handling during context inference process.

3 Context Layer

While constructing CRL, an implementation of the Context Layer was developed. We have chosen Java™ 2 Standard Edition (J2SE™) [8] to be the implementation language, as it is a platform-independent, object-oriented programming language that supports web service architecture and XML processing. In this implementation, Java™ API for XML Processing 1.2 specification is required to validate any CRL document against the CRL Schema. For developmental purposes, our current implementation of the Context Layer contains the CRL-Inference Engine, CRL Management Module and it also contains a fully functional CRParser which, although not part of the Context Layer, is a vital part of the framework to carry out logical inferences. Moreover, the Context Layer also contains a collection of Java objects that represent contextual sensors and a graphical user interface for purposes of emulation.

The current implementation of the Context Layer allows individual rules to be proxy-assigned. Proxy-assigned CRL-rules means that while the rules reside in the inference engine, each of them will have constructed a notification policy. These policies are a collection of triples, each containing the condition, the name of the rule that has set up the policy and the engine thread which contains the rules. Part of the implementation of the CRL Management Module is the CRL Proxy; once policies are constructed they are sent to the CRL (sensor) proxy which monitors them against any changes in the relevant contexts. There are two types of proxy-assignments – temporal notification and event notification. Two separate Java packages have been implemented to emulate our web services framework.

4 Conclusion

By introducing temporal predicate calculus into context sensing, an XML-based Context Request Language (CRL) has been defined to support the passing of a user's request to a Web Service framework for context retrieval. Importantly, CRL supports the novel concept of out-of-band separation of context co-ordination and rules from the application. The representation of CRL is intended to bridge the gap of linking the formalism in theoretical logics to the implementation of intelligent context-aware applications. A CRL definition forms an important step in bringing intelligent inferences into context sensing. We have employed the well-known temporal calculus which is an extension of predicate logics and applied this formal logic to a context-aware environment. In making CRL to be a truly flexible logical rule-based language, we have defined a collection of supplementary languages to assist in controlling context inferences and rules manipulation. We have also adopted a layered-approach to formulate the CRL Framework which leverages on web service's standard messaging protocol to achieve a complete separation of application concerns toward context environments. During the stage of designing the framework, we implemented the Context Layer, which includes a CRL-inference engine to test and demonstrate the usability of CRL.

Furthermore, the CRL, its supplementary languages and CRL-inference engine have been tested with RedPoint which is a locally developed local positioning system evolved from [9]. Their performance results have been measured and are subject to future publications.

5 Acknowledgement

This project is supported by the Hong Kong Polytechnic Central Research Grant GT-877.

References

1. Alvin T.S. Chan, Siu Nam Chuang, "MobiPADS: A Reflective Middleware for Context-Aware Mobile Computing", IEEE Transactions on Software Engineering, vol. 29, no. 12, Dec 2003, pp. 1072-1085.
2. A. K. Dey, "Providing Architectural Support for Building Context-Aware Applications", PhD thesis, College of Computing, Georgia Institute of Technology, December 2000.
3. H. Chen, S. Tolia, C. Sayers, T. Finin, A. Joshi, "Creating Context-Aware Software Agents", Article, First GSFC/JPL Workshop on Radical Agent Concepts, September 2001
4. H. Chen, T. Finin, A. Joshi, "An Intelligent Broker for Context-Aware Systems", InCollection, Adjunct Proceedings of Ubicomp 2003, October 2003.
5. E. Davis, "Representations of Commonsense Knowledge", Morgan Kaufmann Publishers, 1990
6. H.S. Thompson, D. Beech, M. Maloney, N. Mendelsohn, "XML Schema Part 1: Structures", May 2001, Available at http://www.w3.org/TR/xmlschema-1
7. F. Curbera, W. Nagy, S. Weerawarana, "Web Services: Why and How", OOPSLA 2001 Workshop on Object-Oriented Web Services, Florida, USA, 2001
8. Java™ 2 Platform, Standard Edition, Available at http://java.sun.com/j2se/
9. Alvin T.S. Chan, Hong Va Leong, Joseph Chan, Alan Hon, Larry Lau, Leo Li, "BluePoint: A Bluetooth-based Architecture for Location-Positioning Services", Proceedings of the ACM Symposium on Applied Computing (SAC2003), 9-12 March 2003, Florida, USA, pp. 990-995.