Semantic Context Reasoning Using Ontology Based Models

Rodrigo Mantovaneli Pessoa[1], Camilo Zardo Calvi[2], José Gonçalves Pereira Filho[2], Cléver Ricardo Guareis de Farias [3], Ricardo Neisse[1*]

[1] University of Twente, Enschede – The Netherlands
[2] Federal University of Espírito Santo, Vitória – Brazil
[3] University of São Paulo, Ribeirão Preto – Brazil

{mantovanelir, r.neisse}@ewi.utwente.nl
{camilozc, zegonc}@inf.ufes.br
{farias}@ffclrp.usp.br

Abstract. New mobile computing technologies and the increasing use of portable devices have pushed the development of the so-called *context-aware applications*. This new class of applications aims at improving human-computer interactions by supporting dynamic adaptations according to context changes. This paper discusses the suitability of using ontologies for modeling context information and presents the design, implementation and applicability of an ontology based context interpreter. The proposed interpreter is responsible for inferring new context information in a context-aware services platform.

Keywords: Context-aware, context modeling, context reasoning, ontologies.

1 Introduction

The new mobile standards and technologies, and the increasing use of portable devices have stimulated the development of a new computing paradigm called Pervasive Computing. In contrast to the more traditional desktop-based computing paradigm, Pervasive Computing is characterized by constant changes in the environment caused by the mobility of its users. In this scenario, a new class of applications called context-aware has raised on increasing interest in the research community. Context-aware applications take into account in their processing not only explicit user supplied information, but also implicit information related to user's physical and computational environment. These applications are programmed to react to and explore the constant changes in user's context within a dynamic domain.

The development of context-aware applications deals with a number of technological challenges and requires the existence of a suitable support infrastructure to facilitate the construction and execution of these applications. Particularly, support is needed to deal with different context information sources and types. In this sense, a

* Supported by CNPq Scholarship – Brazil

number of initiatives related to the development of support platforms have been proposed in the literature, e.g., [6], [5], [4], [9]. This work is part of the Infraware Service Platform [12][7] which aims at providing a number of services and suitable architectural support for the development of context-aware applications.

One particular problem that has to be addressed by a context-aware service platform is the definition of a model to describe the contextual domain in which a given application or service is defined. Several context models are available in the literature, such as Key-value pairs models [13], Markup Scheme models [6], Object-oriented models [1], and ontology based models [4][9]. The objective of these models is to provide a high level abstraction of context information in order to store, manage, and process the context. However, most of these models lack a formal representation of its syntax and semantics in order to guarantee the consistency between different representations of context used by applications, context providers and service platforms. Furthermore, the level of abstraction provided by the proposed models and the expressivity of their representation language has a significantly impact on the reasoning process.

This paper discusses the suitability of using ontologies for modeling context information and presents the design, implementation and applicability of an ontology based context interpreter. The proposed interpreter is responsible for reasoning context information in a context-aware services platform. The remaining of this work is structured as follows: section 2 presents ontology-based models; section 3 introduces architectural aspects of our context interpreter; section 4 presents implementation details; section 5 describes a usage scenario; section 6 discusses some related work; finally, section 7 presents our conclusions and future work.

2 Ontology-Based Context Models

As already mentioned, one particular problem that should be addressed by context-aware service platforms is the definition of a model describing the context information and the application domain in which one application is inserted.

Ontology based models provide logic characterization for the interpretation of objects, classes and relationships. The formal notation used by this type of model permits the specification of the domain in an ambiguous way, allowing semantically consistent inferences, and also assuring one shared and reusable representation of the contextual information among context providers, service platforms and applications. This enables the development of systems that are able to derive implicit information through the analysis of information represented explicitly.

The OWL (Ontology Web Language) [2] was chosen to specify our context model as it allows the modification and reuse of concepts definition and relations through a well defined and sufficient expressive semantic. OWL is also the language recommended by W3C for ontology and content representation in the semantic web.

3 Conceptual Architecture of Context Interpreter

Our Context Interpreter is divided into five modules as presented in Figure 1: Query Engine, Event Notifier, Working Memory, Inference Engine, and Knowledge Database. These modules together are able to carry on the context reasoning process.

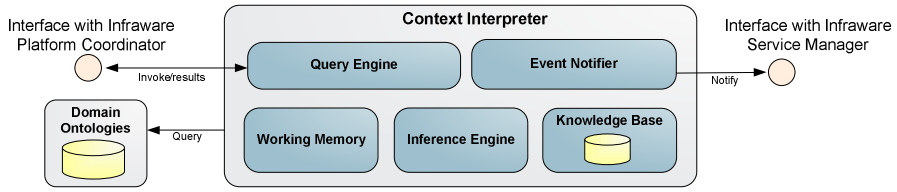

Fig. 1. Conceptual Architecture of Context Interpreter.

The five modules share the Domain Ontologies that describe general concepts related to the application domain and their relationships. The storage of the domain ontologies in this repository promotes a shared knowledge organization and allows reuse. Furthermore, the isolation of these specifications permits the extension and redefinition of these ontologies for different domains.

The Knowledge Database stores instantiated knowledge from the Domain Ontologies or, in other words, it represents a specific knowledge representation. This knowledge is specified through a set of if-then rules and known facts. One rule establishes one relationship between clauses (assertions and facts) and, depending on the situation, can be used to generate new information or to fire an action.

The working memory stores known facts and assertions made by the rules. The Inference Engine combines facts from the Working Memory with the Rules Database in order to assert new facts or to identify specific contexts. The Query Engine allows access to the actual state of interpreted facts by the other modules in the Infraware platform. The Event Notifier is responsible for the dispatch of an action from the occurrence of a new event or monitored context.

4 Implementation

In the current implementation, our Context Interpreter uses the Jena Semantic Web Framework API [10] to manipulate and infer contextual information. Jena is a framework based on Java, developed by HP Labs Semantic Web Programme to build applications to support Semantic Web scenarios. Beyond the support of RDF, DAML+OIL and OWL languages, Jena API also offers components to build rule inference engines.

Subscription requests are compiled using inference rules written in GRL (Generic Rule Language), specified by Jena API. These rules express possible contexts and are evaluated by the Context Interpreter, which uses the information sent by context providers to dispatch actions at the occurrence of several events. Using the RDQL (Resource Description Query Language) declarative language [14], the applications can query relevant context information with support to inference and semantic validation.

Figure 2 shows a UML component diagram of the implemented Context Interpreter. The Context Interpreter uses two data files. One file contains OWL domain ontologies (*DomainOntology*) that defines terms to describe and represent an application domain. In this mode, domain ontologies also define a structured vocabulary with possible metadata constraints describing contextual information and related entities. The second file (*DataInstances*) contains instances of the Domain Ontology representing real-world entities and individuals. The process of context interpretation occurs in response to instance data changes, reflecting the real world's state captured by context providers.

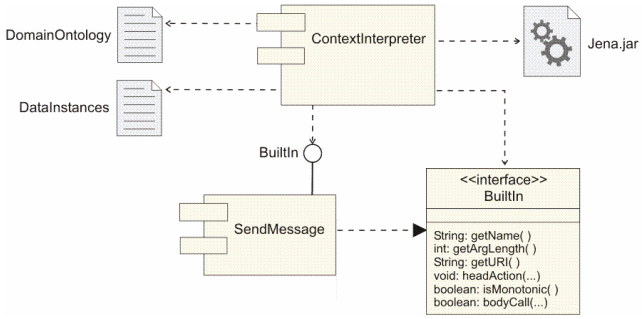

Fig. 2. Context Interpreter Component Diagram.

The component *SendMessage* executes a notification service and implements a *BuiltIn* interface. The interface defines a set of methods which implements operators that realize primitive procedure calls. Beyond the operators supplied by Jena API, inference rules can contain personalized operators, like implemented by the *SendMessage* component.

4.1 Inference Engine

The Jena inference system is designed to allow the support of a wide range of inference engines or reasoners. Such engines are used to derive additional assertions which are entailed from some knowledge base together with any ontology information and rules associated with the reasoner. In particular, Jena includes a general purpose rule-based reasoner (Generic Rule Reasoner) which is used to implement both the RDFS and OWL reasoners but is also available for general use.

In our prototype system, we use Jena Generic Rule Reasoner since it supports rule-based inference over RDF graphs and provides forward chaining, backward chaining and a hybrid execution model.

4.2 Generic Rule Language

The Generic Rule Reasoner defines the Generic Rule Language (GRL) to describe inference rules. These rules can derive new facts or dispatch action at the occurrence of specified conditions (events). A rule is defined by a Java *Rule* object with a list of body terms (premises), a list of head terms (conclusions) and an optional name and

optional direction. Each term or *ClauseEntry* is a triple pattern, an extended triple pattern or a call to a builtin primitive.

4.3 RDF Data Query Language

Besides the inference of new contextual information and the capability of triggering the dispatch of actions based on event occurrence, the Context Interpreter also allows applications to query for the current state of input and derived context information. These queries are expressed in RDQL [14], a declarative query language for RDF supported by Jena's inference models. RDQL uses a declarative SQL-like syntax for querying information contained in an inference model, often expressed as a set of triples. An RDQL query consists of a graph pattern, expressed as a list of triple patterns. Each triple pattern is comprised of named variables and RDF values. Additionally, a RDQL query can have a set of constraints on the values of those variables, and a list of the variables required in the answer set.

5 Usage Scenario

As a proof of concept in this section we present the use of our Context Interpreter in the development of a tourism application. The implementation of this application has allowed us to evaluate the project decisions related to the modeling and inference of context information as well as to evaluate the technical aspects of the technology used in the implementation of our Context Interpreter component.

Taking as a reference scenario one tourist (user) strolling in an unknown city, applications such as interactive maps, personalized tourist advices about tourist interesting points (following the tourist profile and preferences), and access to infrastructure services like gastronomy, and entertainment (hotels, restaurants, cinemas, theaters, etc.) could be able to help. Users are identified as actors in the system and interact with the applications running in portable mobile devices. Figure 3 illustrates a simplified version of the domain ontology we defined for tourism applications describing the main elements of our application domain.

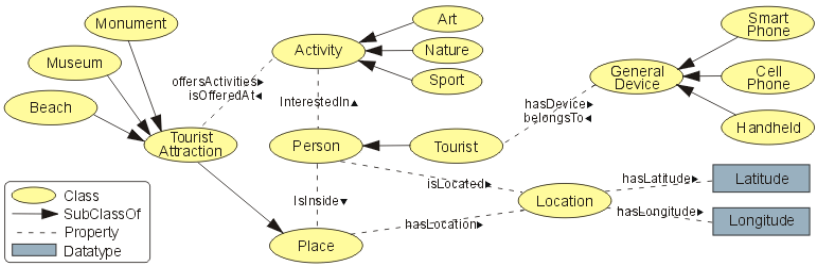

Fig. 3. A simplified tourism domain ontology.

One application could be interested in sending users tourist information about a tourist point in the moment the user is in the location. Even though mobile devices allow the access to the information any time and place, it may be interesting to

personalize the content of the information sent to users considering the type of the user's device. For example, users of mobile phones with small displays capabilities may fell uncomfortable in reading long messages. Another factor that justifies the personalization of content sent to users is related with the great variation in the resources for presentation of multimedia content in heterogeneous devices. Figure 4 describes two subscription requests (A and B) for tourist information for different classes of devices.

```
[subsc01:(?t rdf:type Tourist)
       (?t hasDevice ?d)
       (?d rdf:type MobilePhone)
       (?t isInside ?p)
       (?p rdf:type TouristAttraction)
 -> SendSMSTouristInformation(?d,?p)]
```

(A)

```
[subsc02:(?t rdf:type Tourist)
       (?t hasDevice ?d)
       (?d rdf:type SmartPhone)
       (?t isInside ?p)
       (?p rdf:type TouristAttraction)
 -> SendMMSTouristInformation(?d,?p)]
```

(B)

Fig. 4. Examples of rules to send tourist information by SMS or MMS messages.

The first subscription request (Figure 4–A) calls the information tourist service using SMS messages every time one tourist handling a mobile device of class *MobilePhone* visits one tourist point of interest in the city. The second subscription request (Figure 4–B) send the information to tourists handling devices of the class *SmartPhone* using MMS messages. In both service requests the user device and the point of interest are parameters through the variables "?d" and "?p". Figure 5–A shows how the received messages look like in different types of devices.

(A) (B)

Fig. 5. Tourist information being received by different types of devices.

Another application might be interested in sending users suggestions of places to be visited, according to the users' personal interests. Figure 6 shows an application subscription request using RDQL query in which the tourist points of interest and activities are selected according to the preferences of one specific user. In the example we assume that the user's activities of interest are *Nature* and *Sport*. Figure 5–B shows the result of this query presented to the application user.

```
SELECT ?touristAtraction, ?activity
WHERE (?person, <rdf:type>, <v:Peron>),
      (?person, <v:name>, ?name),
      (?person, <v:InterestedIn>, ?activity),
      (?activity, <v:isOfferedAt>, ?touristAtraction)
 AND ?name eq "Rodrigo Mantovaneli Pessoa"
```

Fig. 8. An example RDQL query retrieving tourist attractions based on user personal interests.

6 Related Work

Different approaches for context modeling and reasoning have been proposed in literature. These approaches reflect the diversity of contextual information and its use in different application domains [11].

Schilit et al. [13] propose the use of key-value pairs for the representation of contextual information, which are then used as environment variables. This approach supports only simple value matching comparison, thus limiting its applicability to simple scenarios. Brézillon [3] propose the use of contextual graph for modeling contextual information as nodes in acyclic graphs. New facts about contextual information can be derived by following the graph. Gray and Salber propose a context model that use first order logic to formally represent the transformations and relationships involving contextual information [8].

Ontologies have also been used for context modeling. The Context Broker Architecture (CoBrA) [4] provides an agent-based architecture for the development of context-aware application. Central to this architecture is an intelligent agent called Context Broker that maintains an ontology-based shared model of context on the behalf of a community of agents, services, and devices.

7 Conclusions

This paper proposes the use of a context interpreter to support the development, deployment and execution of context-aware applications. The proposed context interpreter relies on ontology-based semantic descriptions of contextual information to reason and interpret context.

The use of an ontology-based context interpreter allows the definition of formal and extensible models to describe a particular application domain through the specification of a number of concepts, relations and axioms. The extensibility property provided by the use of ontologies allows the introduction of changes in the underlying context model with minimal impact on the applications that depends on this model. Additionally, the specification of the contextual information semantics in a particular application domain allows the interpretation and inference of new contextual information, based on the described contextual model.

Currently, we are working towards the definition of a more generic context ontology using OWL that can be used in different application domains. The existence of such ontology would facilitate the mapping of context information between

different context models, thus increasing the degree of portability and interoperability between context-aware applications. Additionally, we will consider the definition of concepts related to the quality and consistency of contextual information.

Acknowledgments

This work has been supported by CNPq under project number 50.6284/2004-2.

This work is part of the Freeband AWARENESS and A-MUSE projects. Freeband is sponsored by the Dutch government under contract BSIK 03025.

References

1. Bardram, J. E.: The Java Context Awareness Framework (JCAF). In: Third International Conference on Pervasive Computing, volume 3468 of LNCS, pages 98–115. Springer.
2. Bechhofer, S., van Harmelen, F., et al.: OWL Web Ontology Language Reference: www.w3.org/TR/2004/REC-owl-ref-20040210/, W3C Recommendation (2004).
3. Brézillon, P.: Context-based modeling of operators' practices by contextual graphs. In Proceedings of the 14th Conference on Human Centered Processes. pp. 129-137, (2003).
4. Chen, H.: An Intelligent Broker Architecture for Pervasive Context-Aware Systems. Ph.D. Thesis, University of Maryland, USA (2004).
5. Costa, P. D.: Towards a Services Platform for Context-Aware Applications. Master Thesis, University of Twente, Enschede, The Netherlands, 2003.
6. Dey, A.K.: Providing Architectural Support for Building Context-Aware Applications. Ph.D. Thesis, Georgia Institute of Technology, USA (2000).
7. Farias, C. R. G.; Leite, M. M.; Calvi, C. Z.; Pessoa, R. M.; Pereira Filho, J. G.: A MOF Metamodel for the Development of Context-Aware Mobile Applications. Proceedings of the 2007 ACM Symposium on Applied Computing, p. 947-952, Seul (2007).
8. Gray, P. D. and Salber, D.: Modelling and Using Sensed Context Information in the Design of Interactive Applications. In: Little, M. R. and Nigay, L. (eds). Engineering for Human-Computer Interaction. LNCS (2254), Springer-Verlag, pp. 317-336, 2001.
9. Gu, T., Pung, H.K., Zhang, D.Q.: A Service-Oriented Middleware for Building Context-Aware Services. Elsevier Journal of Network and Computer Applications, Vol. 28, Issue 1, pp. 1-18, January (2005).
10. McBride, B.: Jena: Implementing the RDF model and syntax specification. In: Proceedings of the 2nd International Workshop on the Semantic Web, 2001.
11. Mostéfaoui, G. K., Pasquier-Rocha, J., Brezillon, P.: Context-Aware Computing: A Guide for the Pervasive Computing Community. In Proceedings of the IEEE/ACS International Conference on Pervasive Services (ICPS´04). IEEE Computer Society, pp. 39-48, 2004.
12. Pereira Filho, J.G.; Pessoa, R.M.; Calvi, C.Z. et al.: Infraware: Um Middleware de Suporte a Aplicações Móveis Sensíveis ao Contexto. (In Portuguese) (Infraware: Middleware Support for Context-Aware Mobile Applications). Proceedings of the: 24º 24th Brazilian Symposium on Computer Networks, Curitiba-PR, Brazil, 2006.
13. Schilit, W. N., Adams, N. I. and Want, R.: Context-Aware Computing Applications In Proceedings of the Workshop on Mobile Computing Systems and Applications. IEEE Computer Society, p 85-90, 1994.
14. Seaborne, A.: RDQL - A Query Language for RDF: www.w3.org/Submission/2004/SUBM-RDQL-20040109/