Context Modelling Using Semantic Web Technologies

Hyosook Jung, Sujin Yoo, Seongbin Park

Department of Computer Sciente Education, Korea University,
Seoul, Korea

Abstract: In this paper, we propose an ontology-based context model that generically describes different contexts surrounding each user. By using ontology, it can categorize all contexts and find contextual information from relations between contexts so that it can offer adaptive services according to user's context. A user can write a context rule using a context description language that we propose and our system compiles the context rules and find relevant information. We demonstrate how our system can be used using examples from popular social network services.

Keywords: context modelling, semantic web

1. Introduction

With the development of context-aware computing, different approaches have been proposed for providing services suited to a particular person. Context refers to any information that can be used to characterize the user's current situation such as time, place, device, task, etc. One of the most important things for context-awareness applications is context modeling because a well-designed context model can support expressing and analyzing the context. Early applications addressed a specific context model for just one application, but recently they are interested in generic context models that facilitate context sharing and reusing, or interoperability between different applications.

In this paper, we propose a generic context model based on ontologies [1] that specify concepts and relations between the concepts about the context. The ontologies can represent the context as structured data that computers can understand, and makes it easy to combine, share, or reuse the contextual information. We first define a context ontology written in Web Ontology Language (OWL) [2] for building the vocabulary to describe context. Then, we represent the contextual information written in Resource Description Framework (RDF) [3] using the context ontology. Our system can not only express the context meaningfully, but reason new contextual information from existing contextual information. We also develop a context rule that allows users to represent the contextual information and define what they want a system to do. The context rule consists of conditions and actions and is written using a context description language. A context compiler compiles the context rule, analyzes contextual information and provides a suitable content or services. Each context is represented by a 4-tuple structure (<ContextType>, <Subject>, <Verb>, <Object>) and can be combined using operations. The action defines what kind of content or service is offered. The context rule can be dynamically modified according to user's need and provide the services specialized to each user.

This paper is structured as follows. Section 2 describes related works to our research. Section 3 explains our approach to model contexts. The system architecture that we propose is described in section 4. Section 5 describes scenarios of how our system can be used. Section 6 concludes the paper.

2. Related Works

The Semantic Web is an environment where Web contents are represented in a form that is machine processable [1]. There are several languages to represent machine interpretable contents on the Web such as RDF and OWL. RDF is a data model for objects and their relations and supports a simple semantics for the data model. OWL adds more vocabularies for describing properties and classes such as relations between classes, cardinality, equality, richer typing of properties, characteristics of properties, and enumerated classes [4].

An OWL ontology represents a domain by defining classes and properties of those classes and defines individuals and asserts properties about them. Ontologies contain computer-usable definitions of basic concepts in the domain and the relationships among them. They encode knowledge in one domain or in extended domains [5].

The context aware computing needs to develop a formal context model for facilitating context representation and context sharing meaningfully among heterogeneous systems.

The use of an ontology-based context model enables to share context information among different systems and use the existing logic reasoning on the ontology [6].

ConChat is a context-aware application that allows users to communicate by offering contextual information. An atomic context is represented by (<ContextType>, <Subject>, <Relater>, <Object>). More complex contexts can be created using the atomic contexts. ConChat obtains contextual information through a smart space infrastructure called Gaia [7]. Gaia provides services to manage and program a space which is a ubiquitous computing environment [8]. In ConChat, users can determine which context shows to other users such as their status and mood. They also specify which context of others they want to know about. While ConChat lets users set their contexts or request others' contexts, our system allows users to add new context rules or change existing context rules to provide adaptive services according to their contexts.

[9] is a JXTA-based system for a smart home environment. Each appliance has a context interpreter and discovers other appliances joined in a JXTA network and shares their contextual information. It lets users define a rule that makes the system find proper resources. Each user can obtain a different searching result according to their rules. While the JXTA-based system allows users to specify the conditions of resources that they want to obtain, our system enables them to specify conditions and which functions they want to execute.

An adaptive hypermedia system provides personalized presentation and link structure based on a user model. For adaptation, it lets authors define adaptation rules that generate different content and link structure based on a user model [10]. However, it is difficult to share the resources between different domains or applications because each application offers adaptive services based on the domain model designed by the authors in advance. In our system, we use an ontology in order to define a common context model that describes contextual information of different systems and shares it.

[11] proposes an ontology-based context engine architecture that supports logic-based context reasoning and context-based knowledge creating. It models various contexts about a user as entities using a formal context model based on RDF and extracts data from RDF documents by using RDF Data Query Language. The context engine uses a two-tier reasoning mechanism; pre-programmed rules definition and semi-automated rules generation based on available context. It allows the system administrator to add rules manually, but our system enables end-users as well as developers to generate the additional rules.

AMAYA is a context-aware system that provides recommendations for situations related to user's interaction with the system by mapping personalized data to specific situations. It also supports services suited to user's need by introducing a content item that describes an atomic unit of information based on an ontology and offering a service-independent interface to a number of learning algorithms and predication method [12]. AMAYA recommends relevant information using well-known learning algorithms and predictions methods. While it might be convenient because developers do not care about any algorithm-specific and implementation-specific details, it requires that all situations are defined explicitly in advance and it is hard to modify the system according to user's need. On the other hand, our system can accept different user's needs because it offers a service or content according to a context rule defined by developers or end-users.

Table 1 shows the features of the systems.

3. Context Modelling

A context model that we propose enables expressing different types of contexts and sharing them between different applications or services. It is based on first-order logic and operations such as Boolean operations and quantifications. The context model uses the vocabulary of an ontology that defines classes and properties about the context. The ontology is used to check if the context model is available or not in a certain domain. The context model is not fixed. It means that it is easy to add different types of contexts to the original. We represent the ontology in OWL and the contextual information in RDF.

A single context consists of four elements such as (<ContextType>, <Subject>, <Predicate>, <Object>). <ContextType> is a type of context related to <Predicate>. It means that

Table 1. Features of related works

feature / system	**Context extensibility**	**User-defined rule generation**
ConChat	It is possible because of using ontology-based context model	End-users just describe their contexts or others', not rules.
Smart home	It is possible because of using ontology-based context model	End-users describe the condition of their target resource.
Adaptive hypermedia	It is impossible because of using fixed domain model	Authors can define the adaptation rules.
Context engine	It is possible because of using ontology-based context model	System administrators can add context rules.
AMAYA	It is impossible because of using ontology only for categorizing content items	It does not allow the modification of the reasoning rules because of using learning algorithms and predications methods.
Our system	It is possible because of using ontology-based context model	End-users can define the context rules

<ContextType>is one class in the ontology like 'Network' and <Predicate> is one of the properties related to the class like 'registered'. <Subject> is an instance of the domain class of the property like 'Peter'. <Object> is an instance of the range class of the property like 'Facebook'. For example, (Network, Peter, registered, Facebook) means that Peter already registered for Facebook [13].

As an example, consider the following context rule.

```
+Person "X" (Network, X, fbfriend, Peter)
```

'+' expresses an existential quantifier that indicates that there exists at least one value of a variable in instances of 'Person'. 'Person' and 'Network' are classes and 'fbfriend' is a property in the ontology. 'X' is a variable and 'Peter' is a string value. Both of them express persons' names. So, the above context rule specifies that X is one of Peter's friends in Facebook.

Using the context model, users can create various context rules. A context rule including a set of contexts has the purpose of finding contextual information or executing personalized functions. The context rule consists of conditions and actions. If the condition is true, the action is executed. It is written by a context description language that we develop so that non-expert users can define context rules easily.

Table 2 shows the grammar of the language.

Table 2. The grammar of context description languages

```
digit = ['0'..'9'];
uppercase = ['A' .. 'Z'];
letter = [['a' .. 'z']+['A' .. 'Z']];
string = letter(letter|digit)*;
identifier = '"' + uppercase + '"';
conjunction = '&&';
disjunction = '||';
negation = '!';
universal = '*';
existential = '+';
arrow = '=>';
context = condition arrow action;
condition = form |
    form expression condition;
action = form | form expression action;
form = basic_sentence |
    quantified_sentence;
quantified_sentence =
    quantification contexttype identifier
    basic_sentence |
    quantification contexttype identifier
    basic_sentence;
basic_sentence =
    (contexttype, subject, predicate, object) ;
expression = conjunction | disjunction |
    negation;
quantification = universal |existential;
contexttype = string;
subject = identifier | string;
predicate =  string;
object = identifier | string;
```

A user can define a context rule that consists of multiple conditions. As an example, consider the following context rule.

```
+Person "X" (Network, Peter, isfollowerof, X) &&
(Network, Peter,  login, Twitter)  => (Network,
Peter, seehomeof, X)
```

The context rule is divided by '=>'; one is a condition part and the other is an action part. More complex context is written by using Boolean operations. '&&' means the conjunction between two conditions. The condition specifies that Peter is one of X's followers and logs in Twitter [14]. The action refers that our system offers a map that shows where X lives for Peter if the condition is true.

4. Architecture

Figure 1 displays the architecture of our system.

In the viewer, a user defines a context rule and gives a command for parsing it to the system. The Context-rule parsing engine parses the context rule and checks whether the rule is grammatically correct or not. It also sends the rule to the Context analysis engine and shows the parsing result to the user. The Context rule manager stores the rules in the Context rule repository or loads them from the repository.

Result viewer' is associated with Comparing and Executing module that is composed of two parts. Context analysis engine receives the context rule from Context rule parsing engine

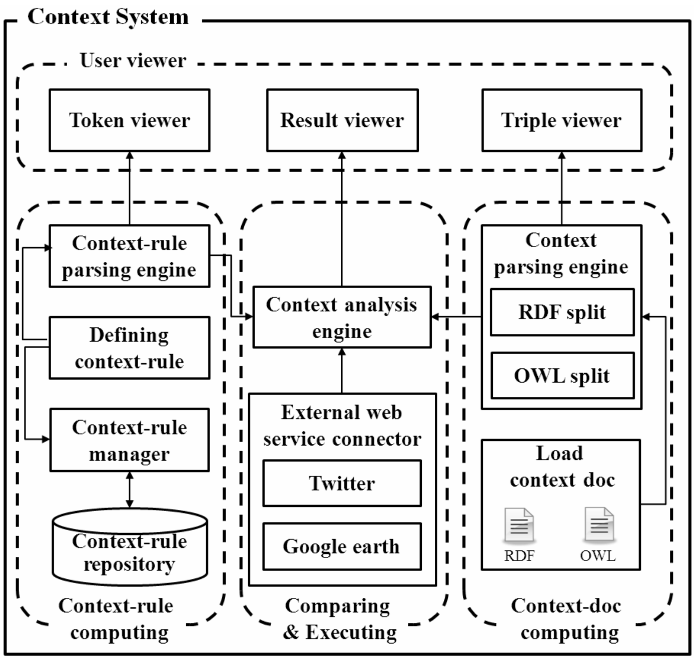


Figure 1. The architecture of our system

'User viewer' is a user interface where users can deal with the contextual information and context rules such as comparing context, finding context matching with certain conditions, and showing the result after executing context rule. It consists of three viewers.

'Triple viewer' is connected to the Context-doc computing module that loads and parses RDF or OWL document. When an RDF document is loaded, the context parsing engine parses it and the viewer shows all entities which are a set of subject, predicate and object. When an OWL document is loaded, the engine parses it and the viewer shows the classes and properties in OWL. It makes users understand the structure of an ontology and the contextual information, and how to define the context rule.

'Token viewer' is related to Rule-context computing module that consists of three parts.

and looks for contextual information to meet the condition part of the rule by using Context parsing engine. If there is the contextual information that satisfies the conditions, the Context analysis engine executes the action part of the rule with External web service connector. The connector enables the use of other web services by using open API such as Twitter or Google earth. It also updates the contextual information.

The viewer shows the result of the rule if there is contextual information to satisfy the conditions or the execution of the requested action.

A user can obtain content or service suited to the context information by using our system as follows (See Figure 2.);

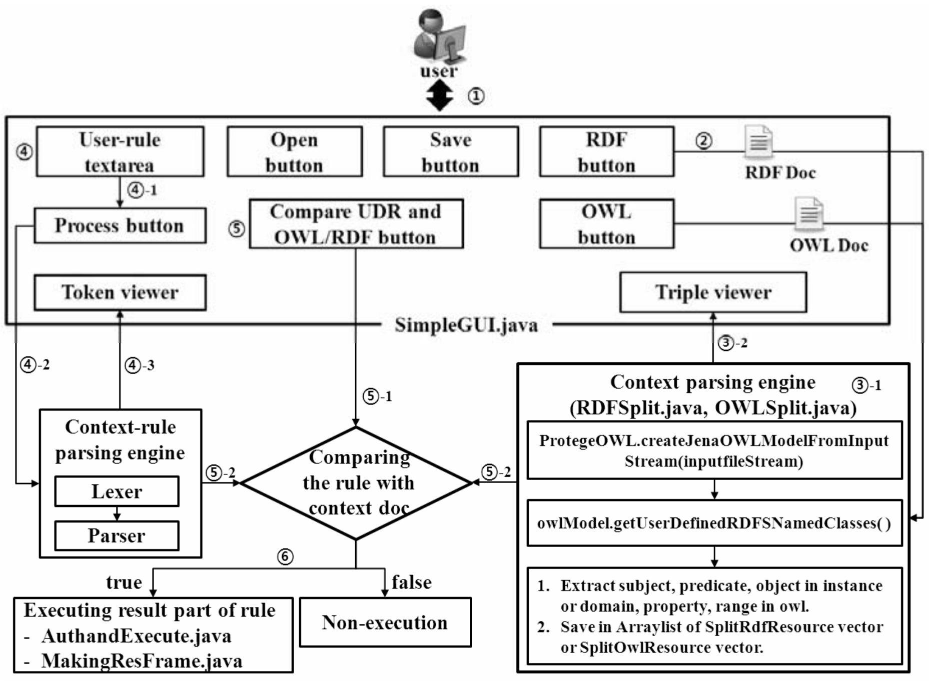

user
①
④
User-rule textarea
Open button
Save button
RDF button
②
RDF Doc
④-1
Process button
⑤
Compare UDR and OWL/RDF button
OWL button
OWL Doc
Token viewer
Triple viewer
SimpleGUI.java
③-2
④-2
④-3
⑤-1
Context parsing engine (RDFSplit.java, OWLSplit.java)
③-1
Context-rule parsing engine
Lexer
Parser
⑤-2
Comparing the rule with context doc
⑤-2
ProtegeOWL.createJenaOWLModelFromInputStream(inputfileStream)
owlModel.getUserDefinedRDFSNamedClasses( )
⑥
true
false
Executing result part of rule
- AuthandExecute.java
- MakingResFrame.java
Non-execution
1. Extract subject, predicate, object in instance or domain, property, range in owl.
2. Save in Arraylist of SplitRdfResource vector or SplitOwlResource vector.

Figure 2. The executing processes of our system

① A user executes our system in a local computer and sees the user interface that there are some viewers and command buttons.

② The user opens an RDF document that defines contextual information.

③ Context parsing engine parses the RDF document and extracts contextual information that consists of a set of RDF triplets such as subject, predicate and object. The user can see the RDF triplets in Triple viewer.

④The user defines a context rule in User-rule textarea by using context description language and requests parsing the rule. Context rule parsing engine verifies the rules and shows the result in Token viewer. In other words, it checks whether the context rule is grammatically correct and its vocabulary and values are valid. For example, a context type should be one of classes and a subject should be an instance of a domain class of a certain property in the ontology.

⑤ The user requests the comparison of the condition and the execution of the action in the context rule. The system searches the contextual information from the RDF document that satisfies the conditions in the context rule by using Context parsing engine.

⑥ If there is contextual information that satisfies the conditions, the system executes actions requested by the user such as providing certain information or functions and updating contextual information.

5. Application

In this section, we show how a user can obtain different content or service according to the context of the user. With the development of social networking services, most users are sharing picture, video, bookmark, message, etc. They make relationships such as contact in Flickr [15], friend in Facebook, follower or following in Twitter, etc. Each site enables users to search other users by using name, email, ID, etc. However, they do not know whether their potential friends have already registered for the sites or not.

When a user accesses to Twitter, if knowing that those who have relationships with the user in Facebook do not have relationships in Twitter yet, the user can easily make new relationships with them in Twitter without searching friends. It makes users actively participate in social networks.

Scenario 1

When we have a relation with our friends in a social network, it is quite possible that we also do in other social networks. For example, if two users have a friend relation in Facebook, they might do in Twitter or Flickr. However, we cannot know if our friends in Facebook are the members of a certain social network such as Twitter or Flickr or not. If we can know whether our friends in one social network have already joined other social network, it is easy to make new relationships with them.

In this scenario, a user looks for friends who have friend relations in Facebook, but do not in Twitter yet. If there are the right friends, the user tries to follow them in Twitter after accessing to their Twitter pages. For the purpose of executing the new action, the user executes our system. The context rule defined by the user is as follows;

```
+Person"X"(Network, testtwo, fbfriend, X) &&
(Network, X, registered, twitter)&&(Network,
X, login, twitter) => (Person, X,
isrecommendedto, testtwo)
```

The user's ID is testtwo and X is the ID of the potential friend in Twitter. 'fbfriend' means the friend relation in Facebook. If there is X, the system will show it to the user. We assume that the users use the same IDs in different social networks.

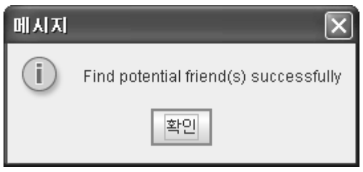
Most social networks let the users know who the potential friends are or directly search them with name, ID, or e-mail based on the relationships within themselves. However, this context rule recommends the potential friends by using the relationships in other networks.

First, the user opens a context file in RDF by clicking 'Open RDF file'. The system shows context information as a set of RDF triplets.

Second, the user types a context rule or opens it by clicking 'Open UDR(User Defined Rule)' (See Figure 3.)

Third, the user clicks 'Parsing UDR'. The system parses the rule and shows the parsing results in token area.

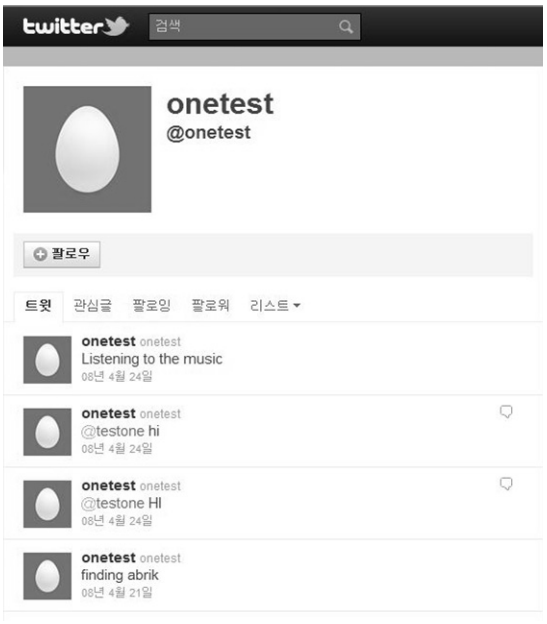
Forth, the user clicks 'Compare UDR and OW/RDF'. The system looks for the persons who have contextual information matching with the conditions defined in the rule. Figure 4 shows the result of the context rule.



Find potential friend(s) successfully

Figure 4. The result of the context rule

Fifth, if there are the right persons, the system shows their Twitter pages. Figure 5 is one of the Twitter pages.



onetest
@onetest

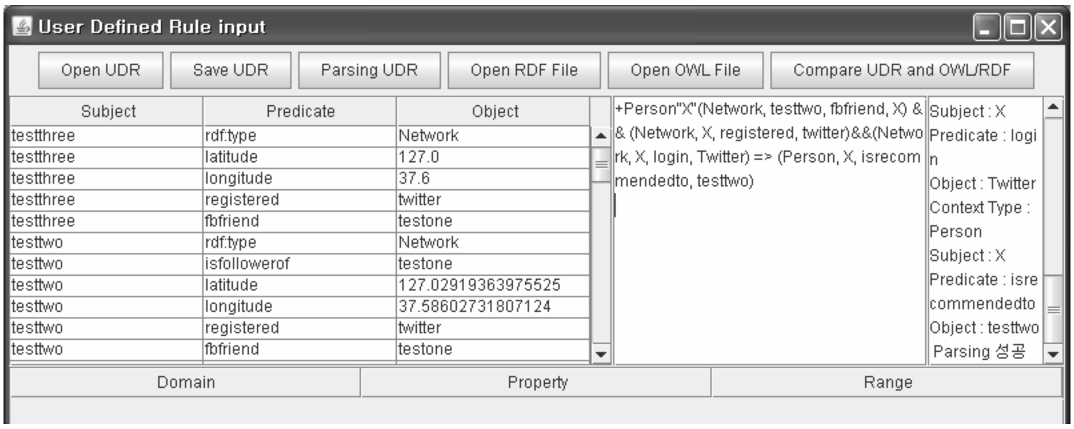
onetest onetest
Listening to the music
08년 4월 24일

onetest onetest
@testone hi
08년 4월 24일

onetest onetest
@testone HI
08년 4월 24일

onetest onetest
finding abrik
08년 4월 21일

Figure 5. The result of the context rule



Subject	Predicate	Object
testthree	rdf:type	Network
testthree	latitude	127.0
testthree	longitude	37.6
testthree	registered	twitter
testthree	fbfriend	testone
testtwo	rdf:type	Network
testtwo	isfollowerof	testone
testtwo	latitude	127.02919363975525
testtwo	longitude	37.58602731807124
testtwo	registered	twitter
testtwo	fbfriend	testone

```
+Person"X"(Network, testtwo, fbfriend, X) &
& (Network, X, registered, twitter)&&(Netwo
rk, X, login, Twitter) => (Person, X, isrecom
mendedto, testtwo)
```

```
Subject : X
Predicate : logi
n
Object : Twitter
Context Type :
Person
Subject : X
Predicate : isre
commendedto
Object : testtwo
 Parsing 성공
```

Figure 3. Defining and parsing the context rule in the our system

Scenario 2

In this scenario, a user looks for friends who have friend relations in Facebook but do not in Twitter yet. If there are the right friends, the user immediately becomes a new follower of the friends in Twitter. The context rule defined by the user is as follows;

```
+Person"X"(Network, testtwo, fbfriend, X) &&
(Network, X, registered, twitter) =>
(Person,testtwo, isfollowerof, X)
```

The user's ID is testtwo and X is the ID of the potential friend in Twitter. If there is X, the system will make the user a follower of X. After executing the rule, we can find that a new predicate 'isfollowerof' is inserted.

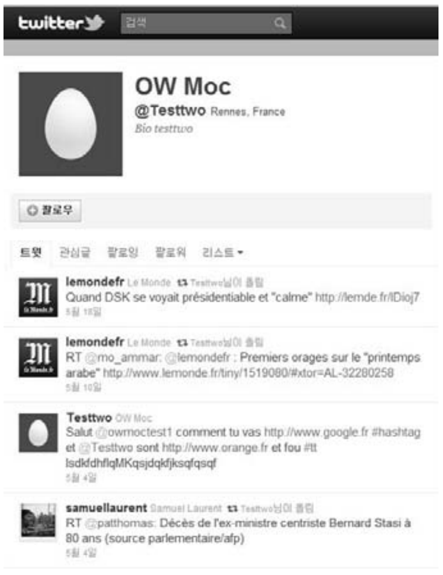
Scenario 3

In this scenario, a user wants to find the potential friends in Twitter and see where they live. The context rule defined by the user is as follows;

```
+Person"X"(Network, testtwo, fbfriend, X) &&
(Network, X, registered, twitter) => (Network,
testtwo, seehomeof, X)
```

The user's ID is testone and X is the ID of the potential friend in Twitter. If there is X, our system recommends a user of potential friends in Twitter and shows maps of their residential areas by using Google Earth [16]. The user first defines the context rule in the text area and compiles it.

If there are the right persons, the system shows their Twitter pages and the residential maps (See Figure 6 and 7).



OW Moc
@Testtwo Rennes, France
Bio testtwo

lemondefr Le Monde Testtwo님이 출팀
Quand DSK se voyait présidentiable et "calme" http://lemde.fr/lDioj7
5월 18일

lemondefr Le Monde Testtwo님이 출팀
RT @mo_ammar: @lemondefr : Premiers orages sur le "printemps arabe" http://www.lemonde.fr/tiny/1519080/#xtor=AL-32280258
5월 10일

Testtwo OW Moc
Salut @owmoctest1 comment tu vas http://www.google.fr #hashtag et @Testtwo sont http://www.orange.fr et fou #tt lsdkfdhflqMKqsjdqkfjksqfqsqf
5월 4일

samuellaurent Samuel Laurent Testtwo님이 출팀
RT @patthomas: Décès de l'ex-ministre centriste Bernard Stasi à 80 ans (source parlementaire/afp)
5월 4일

Figure 6. The page of the potential friend

Figure 7. The Google map of the potential friend

6. Conclusions

In this paper, we presented a context model that describes contextual information based on ontologies. We also proposed a system that provides service or content based on the context model. By using semantic web technologies, our system can reuse and share contextual information among heterogeneous systems. Our contributions are as follows;

First, the context model based on ontologies provides a generic form to describe context. It is possible to share and reuse contextual information and combines the information from different domains.

Second, our system allows users to define context rule while general systems make their developers define the context rule. When new needs happen such as the existing context rule should be modified or new ones should be added, our system can support the needs. In addition, each user can get personalized services because they can ask a system for certain services through their own context rules.

Third, users can define context rule by using context description language although they are not experts about semantic web technologies. The context description language is simplified for non-expert users. The system also parses the contextual information defined in OWL and RDF and shows it in a table. So the users can easily understand that what kind of contextual information there exists, and how they define the context rule by using the contextual information.

In this paper, even though we shows some applications connecting to Twitter and Google Earth, we are going to improve our system so that it can support different functions by using other Open APIs.

REFERENCES

1. ANTONIOU, G., F. van HARMELEN, **A Semantic Web Primer**, The MIT Press, 2004.
2. **OWL**, http://www.w3.org/TR/owl-features/
3. **RDF**, http:// www.w3.org/RDF/
4. McGUINNESS, D. L., F. van HARMELEN, **OWL Web Ontology Language Overview**, W3C Recommendation, http://www.w3.org/TR/owl-features, 2004.
5. HEFLIN, J., R. VOLZ, J. DALE, **Web Ontology Requirements, Proposed W3C Working Draft**, http://km.aifb.uni-karlsruhe.de/projects/owl/index.html, 2002.
6. WANG, X. H., D. Q. ZHANG, T. GU, H. K. PUNG, **Ontology based Context Modelling and Reasoning using OWL**, Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops, 2004, pp. 18-22.
7. RANGANATHAN, A., R. H. CAMPBELL, A. RAVI, A. MAHAJAN, **ConChat: A Context-aware Chat Program**, Pervasive Computing, 2002, vol. 1, no. 3, pp. 51-57.
8. CERQUEIRA, R., C. K., HESS, M. ROMÁN, R. H. CAMPBELL, **Gaia: A Development Infrastructure for Active Spaces**, Workshop on Application Models and Programming Tools for Ubiquitous Computing, 2001.
9. JUNG, H., J. AHN, H. KIM, S. PARK, **Contextual Information in Smart Home**, International Workshop on Ubiquitous Web Applications, 2008.
10. WU, H., G. HOUBEN, P. De BRA, **AHAM: A Reference Model to Support Adaptive Hypermedia Authoring**, Proceedings of the Conference on Information Science, 1998, pp. 51-76.
11. GOH, E., D. CHIENG, A. K. MUSTAPHA, Y. C. NGEOW, H.-K. LOW, **A Context-Aware Architecture for Smart space Environment**, Proceedings of the 7th International Conference on Multimedia and Ubiquitous Engineering, 2007, pp. 908-913.
12. RACK, C., S. ARBANOWSKI, S. STEGLICH, **Context-aware, Ontology-based Recommendations**, International Symposium on Applications and the Internet Workshops, 2006, pp. 23-27.
13. **Facebook**, http://www.facebook.com
14. **Twitter**, http://twitter.com
15. **Flickr**, http://www.flickr.com
16. **Google Earth**, http://earth.google.com