韓國 컴퓨터情報學會 論文誌
第13卷 第4號, 2008. 7.

2008-13-4-4-1

지능형 기상 서비스를 위한 기상 온톨로지의 설계

정 의 현*

A Design of Weather Ontology for Intelligent Weather Service

Eui-Hyun Jung*

요 약

IT 기반의 기상학과 기상 서비스의 급속한 발전에도 불구하고, 아직까지 사람들이 직접 기상 정보를 받아와 판단하는 전통적인 방식으로 기상 정보가 이용되고 있다. 특히 지능화된 기상 정보 처리가 유비쿼터스 컴퓨팅과 개개인의 생활에 매우 유용할 것으로 기대됨에도 불구하고, 기계 주도의 자동화된 기상정보 처리에 대한 연구는 오랫동안 주목을 받지 못했다. 본 논문에서는 지능형 기상 정보처리를 가능하게 하는 GRIB 기반의 온톨로지의 설계에 대해서 논한다. GRIB은 세계적으로 널리 사용되는 범용 목적의 기상 데이터 포맷으로 세계 기상기구에 의해 승인된 형식이다. 설계된 온톨로지와 Jess 엔진으로 구성된 추론 시스템으로 지능형 기상 애플리케이션을 구현하고 실험하여, 기계 주도의 기상 정보 처리에 대한 효과를 검증하였다.

Abstract

In spite of rapid development of IT-related meteorology and services, human users still ought to check the weather information manually as they did before because traditional weather information retrieval is based on pull-type and human interpretation. Furthermore, the automatic machine-driven weather information processing has been neglected for a long time although the intelligent weather information processing is expected to be very useful for personal daily life and ubiquitous computing. In this paper, we discussed a design of GRIB based ontology to enable smart weather information processing. GRIB is the general purposed and world-wildly used weather data format approved by the World Meteorological Organization. With the designed ontology and the inference system containing Jess engine, several intelligent weather applications have been implemented and tested to verify the virtue of machine-driven weather information processing.

▸ Keyword : Ontology, Semantic Web, Meteorology

• 제1저자 : 정의현
• 접수일 : 2008. 3. 29, 심사일 : 2008. 5. 24, 심사완료일 : 2008. 7. 25.
* 안양대학교 컴퓨터학과 교수

Ⅰ. Introduction

Weather is one of the fundamental processes that shape the Earth and it has played a large and sometimes direct part in our daily life. It affects not only clothes that people should wear tomorrow, but also the national sowing plan of the crops. In the US, weather and climate sensitive industries, both directly and indirectly, account for about one-third of the nation's GDP, or $4 trillion dollars, ranging from finance, insurance, and real estate to services, retail and wholesale trade and manufacturing [1]. Weather forecasting is a basic effort to predict the state of the atmosphere at a future time for economic benefit, safety, and convenience of the public.

The traditional approach to the forecasting process is product-based. That means the target of the process is making the end product such as city forecast or coastal forecast. Each forecast product is responsible for providing weather information to specific group of clients who reside in a fixed place. Since a traditional forecast is valid only for a fixed region during specific time, users have to decide target location and time before selecting forecast product. This pull-type and non-personalized information retrieval force human users to check weather data constantly and it would be annoying thing to busy people.

Recently, RSS(Really Simple Syndication) and mobile technology are used to deliver up-to-date weather information as push service. For example, Yahoo weather RSS service[2] provides dynamically-generated RSS feed based on zip code of users and My-cast[3] sends the weather events of user's current location to his/her cellular phone. Although these push services are useful and satisfy basic criteria of personalized weather service, they can not achieve high level of personalization considering user's preference based on his/her job or health condition yet. Furthermore, machine-driven automatic processing of weather data such as giving water a plant or controlling humidity of room is hard to be realized because current data formats can not be understood by machines.

There have been a lot of research on the meteorology but they mainly concentrated on the meteorology itself and IT-related research projects are relatively few. Among them, noticeable IT-related research studies are data exchanging scheme[4], upgrading prediction resolution[5], and providing Web services[6]. The WMO(World Meteorological Organ ization) has been approving a general purpose, bit-oriented data exchange format named GRIB(General Regularly-distributed Information in Binary) that is an efficient vehicle for transmitting large volumes of gridded data to the automated centers over high-speed telecommunication lines. To satisfy the requests of particular weather element at a specific point on the NDFD(National Digital Forecast Database) grid for a user defined time period, DWML[4] is developed by NWS(National Weather Service). DWML can reduce the volume of data being transmitted and ensure that the retrieved data are available to be used for further processing by Web applications. Currently, NWS runs SOAP(Simple Object Access Protocol) server that provides the public, government agencies, and commercial enterprise with DWML-encoded data from the NWS digital forecast database over the Internet[6].

However, ontology related research for personalization and machine understanding are scarce in the field. Although LEAD(Linked Environment for Atmospheric Discovery)[7] project and SWEET (Semantic Web for Earth and Environment Terminology)[8] ontology are announced, their target is not weather itself but mainly Earth and Atmospheric science. There are some research in which ontology is used such as weather forecasting process[9], data warehousing[10], or weather agent[11]. However, these research used the ontology technology as internal purpose only[9][10] or designed proprietary ontology not based on widely used standard data format like GRIB[11].

Generally, if an ontology is not designed based on standard data format, it is nearly impossible to map and gather the ontology data from the existing data sources[12][13].

There can be several issues that current weather IT applications should resolve. Among them, we pay attention to two subjects of personalization and machine understanding. First, highly structured personalization of weather service can be undoubtedly useful. The same weather condition can be differently applied to people according to their job, health condition, or vacation plan. For example, when raining is forecasted on the same region, a farmer may prepare for his/her plants while a commuter just only decides either on the rain coat or the umbrella. Second, machine understanding of the weather condition is a basic requirement for the smart weather service. If it is possible, machines can comprehend the weather information and take the proper actions such as controlling humidity of the hospital room. Furthermore, machines can directly take the weather information from Web services without human intervention and exchange it between other machines to cooperate. However, to achieve these two issues, developing weather ontology based on the approved weather data format should be preceded.

In this paper, a GRIB-based ontology, WeatherOnto, is designed to resolve these issues. The WeatherOnto provides a semantic base for the smart personalized weather service and the weather information exchanging between machines. A translator that converts DWML data to RDF/OWL statements is implemented with Java. We have also implemented an inference system containing Jess engine to show the feasibility of machine-driven weather information processing with semantic weather information and user's rules.

The remainder of this paper is divided into four sections. Section 2 looks over the DWML for machine understandable data. Section 3 describes the detailed structure of the designed ontology. Section 4 explains the process of the translation from DWML to RDF/OWL statements and the implemented smart weather application. Finally, section 5 concludes the paper.

II. DWML

DWML(Digital Weather Markup Language) is selected as the base data format for ontology design in this paper, because DWML is XML encoded GRIB data format and NWS's SOAP server provides DWML encoded weather data via the Internet. Currently, GRIB can be converted to DWML with degrib[14] software and DWML is only sole XML format for GRIB[14]. The piece of the database distributed in DWML will correspond to some subset of the available weather elements, times, and grid points. NDFD data can be modeled as 3-dimensional array as shown in Fig. 1. Users can select some needed weather element by indicating the specific space and time.

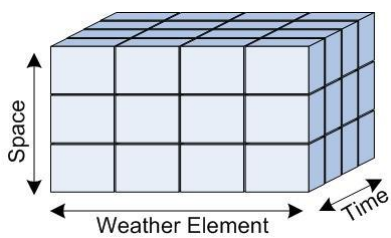

그림 1. 3차원 배열로 표현되는 NDFD
Fig 1. NDFD represented as 3-dimensional array

Since the structure of DWML is based on the 3-dimensional data model, location and time are the key factors to indicate the valid period of weather elements in DWML document The snapshot of DWML XML schema is shown in Fig. 2.

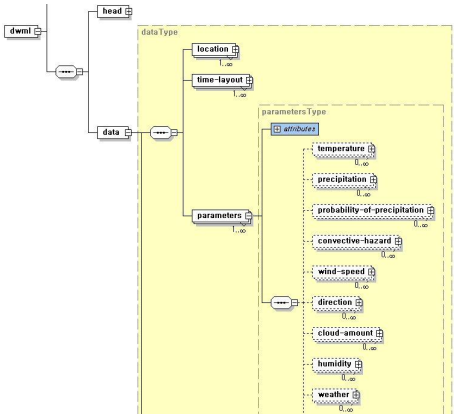

그림 2. DWML의 XML 스키마
Fig 2. XML schema of DWML

The head element contains metadata that provides information about DWML message and the data it contains. The data element consists of location, categorical-definitions, time-layout, parameters, and conversion-definitions. Among them, location, time-layout and parameters elements are mandatory. The location element indicates location data with geographical point or city name. The time-layout element can be repeated in several times and it contains start-valid-time, end-valid-time and layout-key element. The reason why one location can have multiple time-layouts is that weather data may be changed as time goes by. Last element is parameter that contains multiple sub elements indicating the detailed weather data such as temperature, wind-speed, weather-condition, precipitation, probability-of-precipitation, etc. Each sub element has the attribute of time-layout, which means each data is only valid for the duration indicated by time-layout. For example, the snippet of DWML sample data in Fig. 3. indicates that maximum temperature is expected to be 76 °F during 08:00 ~ 20:00.

```
<time-layout
  time-coordinate="local" summarization="none">
<layout-key>k-p24h-n1-1</layout-key>
<start-valid-time>2007-06-11T08:00:00-04:00
</start-valid-time>
<end-valid-time>2007-06-11T20:00:00-04:00
</end-valid-time>
</time-layout>
<parameters applicable-location='point1'>
<temperature type='maximum' units='Fahrenheit'
time-layout='k-p24h-n1-1'>
<name>Daily Maximum Temperature</name>
<value>76</value>
</temperature>
```

그림 3. DWML의 코드 샘플
Fig 3. A snippet of DWML

III. Structure of the WeatherOnto

3.1 Top Level Classes

To represent user's movement of several locations, each user's personalized weather is modeled as PersonalWeather class, which is composed of several SpotWeather classes. Each Spotweather takes charge of maintaining weather data of specific location within user's staying time. If a user moves to 3 places, PersonalWeather will be composed of 3 SpotWeather instances. A SpotWeather class has Location and Term class to indicate user's location and the staying time at the location. The SpotWeather has also several Parameters that contain detailed weather data such as temperature, humidity, or wind speed. In addition, the SpotWeather is linked to WeatherCondition class that indicates weather type, areal coverage and intensity. When a user stays over the time that a single weather condition data can't express, two or more WeatherCondition instances will be linked to SpotWeather. The RDF graph of the top level classes is shown in Fig. 4.

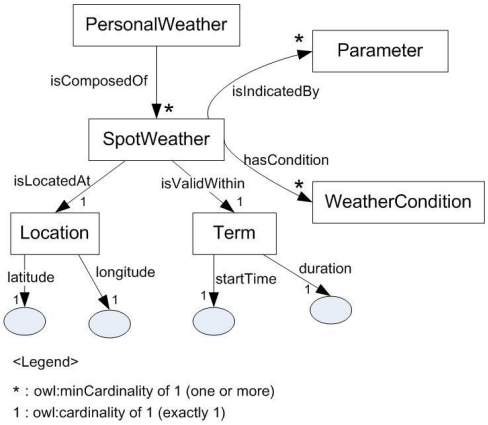

그림 4. 최상위 클래스들의 RDF 그래프
Fig 4. RDF graph of top level classes

3.2 Parameter Classes

Parameter class takes charge of indicating detailed weather elements. Parameter class has 3 properties isValidWithin, hasValue, and hasUnit. The isValidWithin property has a instance of the Term, which indicates the validation duration of the parameter. The hasValue property indicates the weather value that the parameter has and the hasUnit indicates the unit of value such as Fahrenheit or inches. Parameter class itself is not used in the RDF/OWL statements but subclasses play an actual role of indicating the weather data. The hasSkyCoverCode and the hasPopCode properties are used to link the weather constants such as PartlyCloudy or Occasional. The isTypeOf property is used to indicate the type of each Parameter. For example, a Temperature instance may be one of the HourlyTemperature, ApparentTemperature, DewPointTemperature, MaximumTemperature or MinimumTemperature according to the value of TemperatureType. The RDF graph of the parameter classes is shown in Fig. 5.

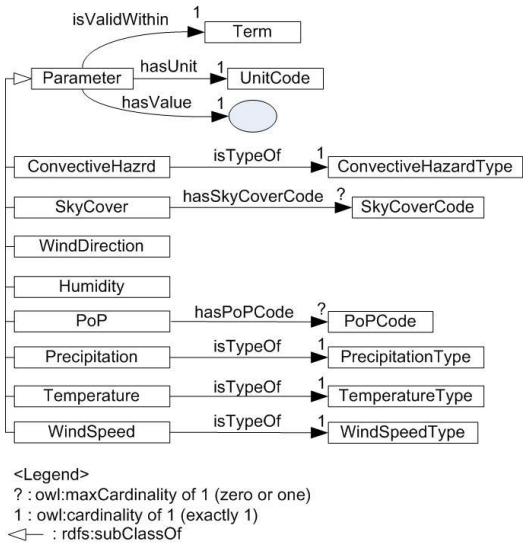

그림 5. 파라미터 클래스들의 RDF 그래프
Fig 5. RDF graph of parameter classes

Meanings of each Parameter are described in Table 1.

표 1. 파라미터의 의미
Table 1.Meaning of parameters

Parameter	Meaning
ConvectiveHazard	Probability of convective hazard such as tornadoes
Humidity	Percent value of humidity
PoP	Probability of precipitation
Precipitation	Precipitation of snow or liquid
SkyCover	Cloud amount of the sky
Temperature	Hourly, dew point, apparent, maximum and minimum temperature
WindSpeed	- Probability of tropical cyclone wind speed over some values (incremental and cumulative) - Speed of gust, sustained, and transport wind
WindDirection	Direction of wind

3.3 Weather Condition Classes

WeatherCondition class is used to describe the familiar weather information to human users such as rain, fog or snow. This class has 4 mandatory properties that indicate areal coverage, weather intensity, weather type, and valid duration. Optional properties are related to the visibility of weather condition. The RDF graph of the WeatherCondition class is shown in Fig. 6.

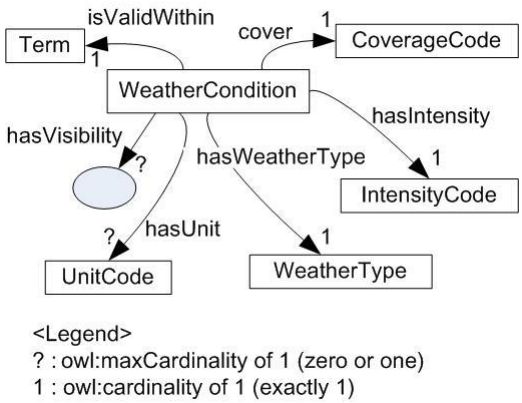

그림 6. 기상 조건 클래스들의 RDF 그래프
Fig 6. RDF graph of WeatherCondition classes

IV. Intelligent Weather Application

4.1 Process of Translation

Latest weather information encoded in DWML can be easily retrieved via Internet if proper parameters of longitude, latitude, and time duration are given. However, to translate retrieved DWML to RDF/OWL statements using the WeatherOnto, proper policy is required. Two factors of the staying time of users and the mapping of the weather constants are significantly considered in the process of translation. In Fig.7, the sample process of translation is shown.

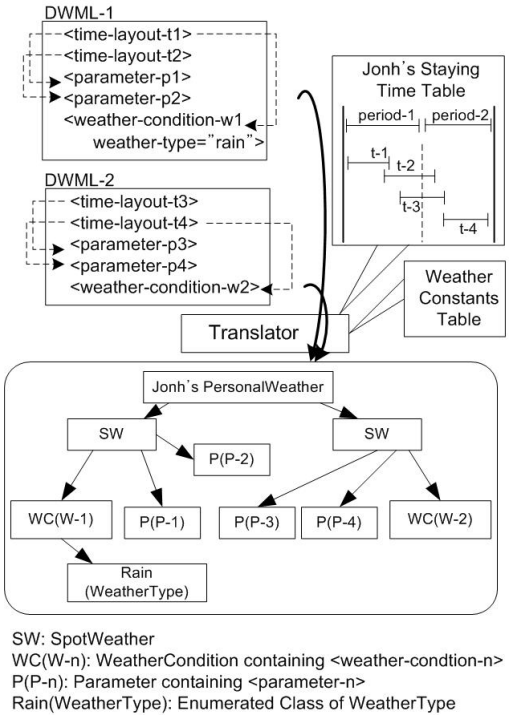

그림 7. 번역 과정
Fig 7. Process of Translation

Two sample DWML documents are assumed to have a single weather condition and 2 weather parameters that are valid within the indicated each time-layout. These XML weather elements are mapped to PersonalWeather instance with the target user's staying time table. Since there are 2 periods in John's staying time table and it means John moved two locations, 2 SpotWeather instances are created to hold the weather information related to the periods. Each SpotWeather instance includes parameters and weather condition even if their time-layout and user's staying time partly coincide. In the Fig. 7, weather-condition-1 element is assumed to have a weather-type attribute indicating "rain". During the translation, this attribute is mapped to Rain instance, which is linked to WeatherCondition using the mapping table in the translator. With this mechanism, correct semantic vocabulary can be constructed in the result

RDF/OWL statements. For the translation, we have implemented a DWML-to-WeatherOnto translator that can translate online weather data from NWS's SOAP server with SOAP protocol.

4.2. Inference Mechanism

To make a smart weather application, an inference mechanism that understands user's rulesand weather information based on the designed ontology is needed. There are several inference engines for the Semantic Web applications such as Jess[15], Jena[16] or Racer[17]. From them, Jess is selected in this paper because of its strong inference function and independent rule writing. Jess engine loads a RDF/OWL document as facts and combines these facts with the designed ontology and user's rules. To insert facts into Jess engine, the DWML-to-WeatherOnto downloads data from NWS's SOAP server and translates it. This process is shown in Fig.8.

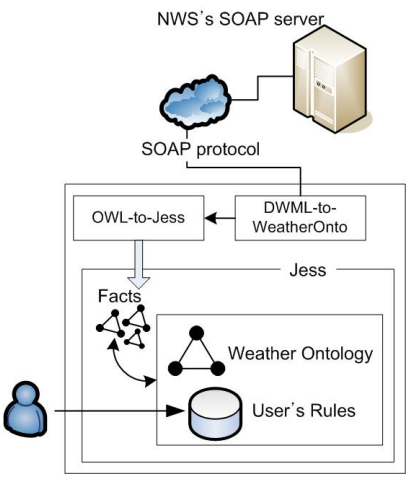

그림 8. 추론 시스템의 내부 구조
Fig 8. Internal Architecture of the Inference System

In the process, sample Jess rules in Fig. 9 can be used to warn a cold and notify preparing umbrella. Several Jess rules can be dynamically inserted or removed in the inference system. For example, to tell users whether preparing an umbrella or not, the first Jess rule can be used. This rule checks both the probability of precipitation is over 50% and weather condition is the Rain weather type. When applications require a customized inference function for their own purpose, they can freely add their new classes without spoiling the consistency of the designed ontology. The second Jess rule shows different reaction with same weather information by adopting a new class, "User". Although weather condition is satisfied, this Jess rule will not fire if user's condition is not "weak". Since Semantic Web applications adopting the WeatherOnto can understand the semantic meaning of the weather information, the inference rules can be shared to add new intelligent functions of the applications without code modification.

```
(defrule check_rain
(object (is-a WeatherType) (OBJECT ?wt) (:NAME "Rain"))
(object (is-a WeatherCondition) (OBJECT ?wc)
(hasWeatherType ?wt))
(object (is-a PoP)  (hasValue ?value&: (> ?value 50)))
=>
(printout t "Prepare umbrella" crlf))

(defrule warning_cold
"Warn: max and min temperature is over 20"
(object  (is-a TemperatureType) (OBJECT ?max_type)
(:NAME "MaximumTemperature"))
(object  (is-a TemperatureType) (OBJECT ?min_type)
(:NAME "MinimumTemperature"))
(object (is-a Temperature) (isTypeOf ?max_type)
(hasValue ?max_value))
(object (is-a Temperature) (isTypeOf ?min_type)
(hasValue ?min_value))
(object (is-a User) (condition "weak") (name ?name))
(test (> (- ?max_value ?min_value) 20))
=>
(printout t ?name ", be careful of cold: high daily range "
crlf))
```

그림 9. 온톨로지 테스트를 위한 Jess 코드
Fig 9. Jess Codes for testing the WeatherOnto

4.3 Smart Weather Page

We have implemented this prototype application

with the inference system and Google Map API[18]. To make the prototype system, we have tested the WeatherOnto ontology with many Jess rules. The facts based on the ontology are imported in the Jess engine and they are used with given rules. Fig. 9 depicts sample Jess test code to verify the WeatherOnto ontology.

In the implemented application, a user can select Jess rules and set his/her own schedule of staying location. After the Jess rules are setup, weather condition and advice are automatically shown in the Google Map depending on the weather information of the specified day and selected user's rules. The screenshot of this application is shown in Fig. 10. This screenshot is different according to each user's rule and condition.

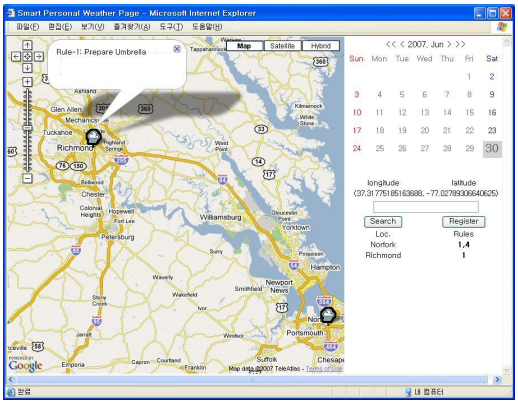

그림 10. 지능형 날씨 페이지의 실행화면
Fig 10. A screenshot of the Smart Weather Page

V. Conclusion

Weather information is getting more and more important in our daily life and the national economy. However, the traditional pull-type information retrieval forces human users to check the weather information constantly and interpret this information on their own. Recently, RSS and mobile phone technology alleviate this problem and provide users with the push-type information delivery service. However, machine-driven processing of weather information has been out of focus yet in spite of its great potentialities. To do this, a general purposed weather semantic ontology should be designed, but current research have concentrated on making a small set of ontology for their internal purpose only or a proprietary format not based on the approved weather data format.

In this paper, we select the world-widely used and approved data format, GRIB, as a base format to design a weather ontology. The designed ontology is used to create RDF/OWL statements dynamically from the latest weather information of NWS's SOAP server according to user's movement. For this purpose, the DWML-to-WeatherOnto translator has been implemented. To show the feasibility of smart weather information processing based on the ontology, the inference system containing Jess engine is implemented and tested with several weather situations. Also, the inference system is used to make the personalized weather Web page with Google Map API. In the future, the DWML-to-WeatherOnto translator will be reprogrammed as Web Services. This translator is going to provide online semantic weather information to other Semantic Web applications that need to add smart weather functions.

참고문헌

[1] NOAA, Economic Statistics for NOAA, http://www.publicaffairs.noaa.gov/pdf/economic-statistics2005.pdf, May, 2005.

[2] Yahoo, "Yahoo! Weather RSS Feed," http://developer.yahoo.com/weather/, 2007.

[3] MyCast home page, http://www.my-cast.com/, 2007.

[4] NOAA/NWS, "Digital Weather Markup Language XML Specification (Version 1.0)," Mar., 2007.

[5] IBM, "Deep Thunder, Precision forecasting for weather-sensitive business operations," http://services.alphaworks.ibm.com/deepthunder/

〔6〕 NOAA/NWS, "National Digital Forecast Database XML Web Service," http://www.weather.gov/xml/
〔7〕 M. Ramamurthy and K. Droegemeier, "Linked Environment for Atmospheric Discovery (LEAD): Transforming the Sensing and Numerical Prediction of High Impact Local Weather Through Dynamic Adaptation," American Geophysical Union, Fall Meeting. 2006
〔8〕 NASA, "Semantic Web for Earth and Environmental Terminology (SWEET) Ontologies," http://sweet.jpl.nasa.gov/ontology/
〔9〕 J. Bally, T. Boneh, A. E. Nicholson, and K. B. Korb, "Developing An Ontology for the Meteorological Forecasting Process," The 2004 IFIP International Conference on Decision Support System (DSS 2004), pp. 70-81, 2004.
〔10〕 Y. Zhu, "A Framework for Warehousing the Web Content," LNCS 1749, pp.83-92, 2004.
〔11〕 WeatherAgent@Aberdeen, http://www.csd.abdn.ac.uk/research/AgentCities/WeatherAgent/
〔12〕 김병곤, 오성균, "시멘틱웹 구축을 위한 스키마 관리 기법 연구," 컴퓨터정보학회 논문지, 12권 1호, 2007.
〔13〕 윤보현, 서창호, "시맨틱웹을 위한 효율적인 온톨로지 객체 모델," 컴퓨터정보학회 논문지, 11권 2호, 2006.
〔14〕 NOAA/NWS, "Degrib: Tutorial," http://www.nws.noaa.gov/mdl/degrib/tutorial.php
〔15〕 E. Friedman-Hill, "Jess in Action," Manning Press, 2003.
〔16〕 B. McBride, "Jena: A Semantic Web Toolkit," IEEE Internet Computing, vol.6, no.6, Nov. 2002.
〔17〕 V.and R.M öller, "Racer: A Core Inference Engine for the Semantic Web," Proc. of the 2nd International Workshop on Evaluation of Ontology-based Tools (EON2003), pp. 27-36, 2003.
〔18〕 Google, Google Maps API documents, http://www.google.com/apis/maps/documentation

저 자 소 개

정 의 현

1992년 2월 : 한양대학교 전자공학사
1994년 2월 : 한양대학교 전자공학과 석사
1999년 2월 : 한양대학교 전자공학박사
2004년 ~ 현재 : 안양대학교 컴퓨터학과 전임강사
관심분야 : 시맨틱 웹, 센서 네트워크