DOI:10.2298/CSIS091011009J

A Grammar-based model for the Semantic web

Hyosook Jung[1] and Seongbin Park[1]∗

[1] Department of Computer Science Education,
Korea University, Seoul, Korea
{est0718, hyperspace}@korea.ac.kr

Abstract. The Semantic Web is an extension of the Web where information is represented in a machine processable way. In this paper, we present a two-level model for the Semantic Web from the perspective of formal language theory. The model consists of two grammars where the first level grammar is for creating ontologies and the second level grammar is for creating ontological instances. Based on the model, we implemented a system by which one can easily construct a small-scale Semantic Web environment.

Keywords: Semantic web, ontology, grammar.

1. Introduction

The Semantic Web is a vision for the future of the Web in which information is given explicit meaning, making it easier for machines to automatically process and integrate information available on the Web [5].

In this paper, we present a grammar-based model for the Semantic Web. As in [3], we view the Semantic Web as the set of ontologies and ontological instances, where an ontology is a document or file that defines the relations among concepts. The proposed model consists of two grammars. The first level intends to represent an ontology about a domain of interests. Ontologies are strings generated by the first level grammar. The second level intends to represent ontological instances which are resources described using concepts and relationships based on the ontology defined at the first level.

While there are approaches to model the Semantic Web [3,9,10], the advantage of the proposed model is that users can easily create a small-scale Semantic Web environment where various experimentations such as whether a current Web browser needs a new functionality or not can be done. To construct the environment, one can define a grammar for an ontology and generate ontological instances.

Our system can serve as an education tool for teaching the Semantic web. For example, non-experts learn about the conceptualization and formalization of ontologies during lectures. Although there are different ontology language

∗ To whom correspondence should be addressed.

standards, in teaching level the general understanding about the ontology language might be more important than the specific understanding about a certain ontology language standards. Using our system, they can practice how to create ontologies by defining their simple ontology languages. While the languages that they define are not full-fledged ontology languages, they can understand the roles of ontologies and how ontologies are used.

Our system is different from ontology development tools such as Protégé-OWL editor [13], OntoEdit [15], OntoKick [16], WebODE [17] etc. in that it allows non-expert users to generate ontologies by using simple languages defined by themselves without the knowledge of OWL and RDF which are not simple concepts to understand [14]. They can understand the essential elements of ontologies and how they are used.

Figure 1 shows the idea behind our approach.

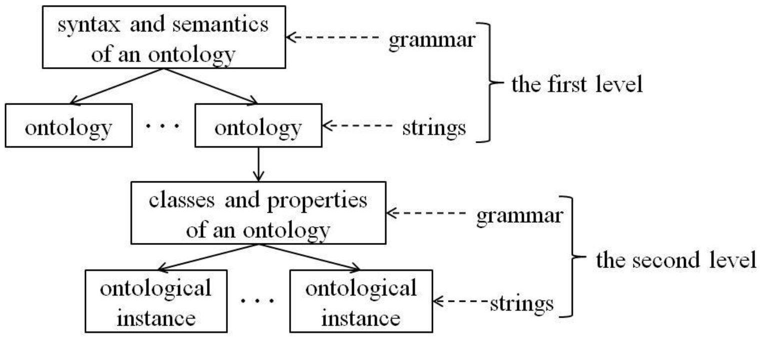


Fig. 1. A grammar-based model for the Semantic Web

This paper is structured as follows. Section 2 describes related works. In section 3, a two-level model is explained. Illustrative examples are given in section 4 and section 5 describes how a small-scale Semantic Web can be constructed using the proposed model. Finally, 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 [4]. There are several languages to represent machine interpretable content on the Web. XML offers a surface syntax for structured documents and XML Schema is a language for restricting the structure of XML documents. RDF is a data model for objects and their relations and supports a simple semantics for the data model. RDF Schema is a vocabulary for representing properties and classes of RDF resources. OWL adds more vocabulary for describing properties and classes:

among others, relations between classes, cardinality, equality, richer typing of properties, characteristics of properties, and enumerated classes [1].

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 a domain and also knowledge that extends domains [6]. An OWL instance is a description about a resource created by using properties and classes defined in the OWL ontology [2,8].

In Ontobroker [7], ontologies are defined in a representation language based on Frame-Logic which supports queries by using instances of an ontology. The representation language used to define ontologies enables elementary expressions such as classes, attributes, relationships, and axioms. It also allows complex expressions such as facts, rules, double rules and queries. The defined ontology is composed of concept hierarchy which defines the subclass relationship between different classes, attribute definitions given for classes and a set of rules which defines relationships between different concepts and attributes.

The Semantic Web has been modeled in various ways. [9] describes a Semantic Web space as two-tuple <O, R>, where O is a set of ontologies and R is a set of resources such as web pages, databases, and sensors.

[10] describes the semantic web as a Notebook + Memex where, the Memex emphasizes on engaging with information, developing it, and working with it, the notebook focuses on both the more writerly and the more personal side of engaging with information. It can perform the automatic and logical processing of repetitive thought tasks and the creation of associative links across different resources by connecting into either the similar tasks or creative thought processes.

[3] describes a semantic network as a directed labeled graph. For the Semantic Web, a semantic network substrate is represented by the constraints of the RDF which describes a semantic network as a set of triples where a subject resource points to an object resource according a predicate resource. Subject and predicate resources are identified by URI (Uniform Resource Identifier) and object resources are a literal or URI. The Semantic Web can be defined as $G \subseteq (U \times U \times (U \times L))$, where U is the set of all URIs and L is the set of all literals.

Linked Data is about using the Web to create typed links between data from different sources. It basically uses the RDF data model to publish structured data on the Web and RDF links to interlink data from different data sources [18]. It is associated with the semantic web because the semantic Web isn't just about putting data on the web, but about making links, so that a person or machine can explore the web of data [19]. Our tool lets people represent data based on ontologies, which is a basic process to make semantic links between data.

Cloud computing is a term to describe both a platform and a type of application. A cloud is a pool of virtualized computer resources. A cloud computing platform dynamically provisions, configures, reconfigures, and deprovisions servers as needed. Cloud applications are extended applications to be accessible through the Internet. These cloud applications use large data centers and powerful servers that host Web applications and Web services [20]. Although the cloud computing providers are publishing various clouds over the Internet, there are no standard, open protocols and discover mechanisms for different kinds of clouds [21]. So, the Cloud Computing Interoperability Forum (CCIF) focus on being placed on the creation of a common agreed upon framework or ontology that enables the ability of two or more cloud platforms to exchange information [22]. A common cloud ontology can support the expression of cloud computing and its related parts by using a common data model. Our tool allows people to define ontologies to represent data semantically. They can experience the way of creating a data model for cloud computing.

Social semantic web is related to the creation of explicit and semantically rich knowledge representations. It can be seen as a Web of collective knowledge systems that which can provide useful information based on human contributions and get better as more people participate. Instead of relying entirely on automated semantics with formal ontology processing and inferencing, humans are collaboratively building semantics aided by socio-semantic information systems [23]. Our tool enables users to create ontologies and represent data based on the ontology by using their own description languages instead of RDF/OWL. The users can also create a small-scale social semantic web that supports semantic browsing by using user-defined ontologies.

A reasoner is a service that takes the statements encoded in an ontology as input and infers new statements from them. In particular, OWL reasoners such as FaCT++ and Pallet can be used to reveal subclass or superclass relationships among classes, determine the most specific types of individuals, and detect inconsistent class definitions [24]. Our tool checks whether the ontology is defined without syntactic and semantic errors and whether the instances are defined by using the classes and properties of the ontology.

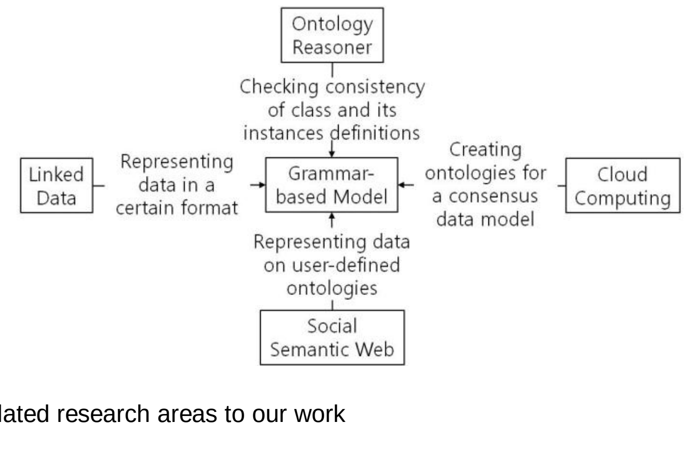

Ontology Reasoner
Checking consistency of class and its instances definitions
Linked Data
Representing data in a certain format
Grammar-based Model
Creating ontologies for a consensus data model
Cloud Computing
Representing data on user-defined ontologies
Social Semantic Web

Fig. 2. Related research areas to our work

Figure 2 shows how our research is related to Linked Data, Cloud computing, Social Semantic Web, and Ontology reasoners.

3. A Grammar-based Model

In this section, we describe a grammar-based model for the Semantic Web. The proposed model consists of two grammars. The first grammar is for generating ontologies and the second grammar is for generating ontological instances. More specifically, the first level grammar in our model is used to generate ontologies that describe information about classes or properties. A class has certain restrictions, where a restriction is a data type of a class or a condition about data value. Datatype properties define relations between instances of classes and RDF literals and XML Schema datatypes. Object property defines relations between instances of two classes by connecting instances in a domain class into instances in a range class. A data range is used as the range of a data-valued property such as string, integer, Boolean, and float.

The syntax of the first level grammar[1] is as follows.

```
ontology ::= 'Ontology' ontologyID directive*;
directive ::= import | class | property;
import ::= ‘NS:’ namespaceID=referrenceID
class ::= 'Class' classID description*;
description ::= 'SubClassOf' classID | restriction*;
restriction ::= 'Restriction On
Property'(datatypePropertyID datatype |
objectPropertyID objecttype);
datatype ::= dataRange | cardinality;
objecttype ::= classID | cardinality;
cardinality ::= 'min' digit+ | 'max' digit+| 'equals'
digit+;
property ::= datatypeProperty | objectProperty;
datatypeProperty ::= 'DatatypeProperty'
datatypePropertyID
 ('domain' classID)* ('range' dataRange)*;
objectProperty ::= 'ObjectProperty' objectPropertyID
 ('domain' classID)* ('range' classID)*;
dataRange ::= 'string' | 'integer' | 'boolean' |
'float';
ontologyID ::= identifier;
```

1 Terminals are quoted (i.e., ‘Ontology’) and non-terminals are not quoted (i.e., ontologyID). Alternatives are either separated by vertical bars(|) or are given in different productions. Components that can occur at most once are followed by ‘+’ and components that can occur any number of times including zero are followed by ‘*’.

Using this grammar, an ontology about a movie can be defined. For example, a movie ontology can have Class Film and Genre, and property genreOf which creates a relation between Film's and Genre's instances. Class Film has at least one instance of Class Genre as the value of ObjectProperty genreOf. Using the grammar, a Movie ontology can be derived as follows.

```
'Ontology' ontologyID directive*
→'Ontology' Movie directive*
→'Ontology' Movie class directive*
→'Ontology' Movie 'Class' Film description* directive*
→'Ontology' Movie 'Class' Film restriction directive*
→'Ontology' Movie 'Class' Film 'Restriction On
Property' objectPropertyID objecttype directive*
→'Ontology' Movie 'Class' Film 'Restriction On
Property' genreOf cardinality directive*
→'Ontology' Movie Class Film 'Restriction On Property'
genreOf 'min' 1 directive*
→'Ontology' Movie 'Class' Film 'Restriction On
Property' genreOf 'min' 1 class directive*
→'Ontology' Movie 'Class' Film 'Restriction On
Property' genreOf 'min' 1 'Class' Genre directive*
→'Ontology' Movie 'Class' Film 'Restriction On
Property' genreOf 'min' 1 'Class' Genre property
→'Ontology' Movie 'Class' Film 'Restriction On
Property' genreOf 'min' 1 'Class' Genre objectProperty
→'Ontology' Movie 'Class' Film 'Restriction On
Property' genreOf 'min' 1 'Class' Genre
'ObjectProperty' objectPropertyID ('domain' classID)*
('range' classID)*
→'Ontology' Movie 'Class' Film 'Restriction On
Property' genreOf 'min' 1 'Class' Genre
'ObjectProperty' genreOf 'domain' Movie 'range' Genre
```

Now, this ontology serves as the second level grammar. According to the Movie Ontology, Class Genre has Instance Fantasy and Adventure and Class Film has Harry_Potter_and_the_Sorcerers_Stone whose genre is Fantasy and Adventure. The classes and properties of the ontology become terminals and strings are variables. An instance of the movie ontology can be derived as follows.

```
('Genre' genreInstance)+ ('Film' filmInstance 'genreOf'
(genreInstance)+)+
→'Genre' Fantasy 'Genre' Adventure (Film filmInstance
('genreOf' genreInstance)+)+
→'Genre' Fantasy 'Genre' Adventure 'Film'
Harry_Potter_and_the_Sorcerers_Stone ('genreOf'
genreInstance)+
→'Genre' Fantasy 'Genre' Adventure 'Film'
Harry_Potter_and_the_Sorcerers_Stone 'genreOf' Fantasy
'genreOf' Adventure
```

Figure 3 shows Movie ontology and its instances.

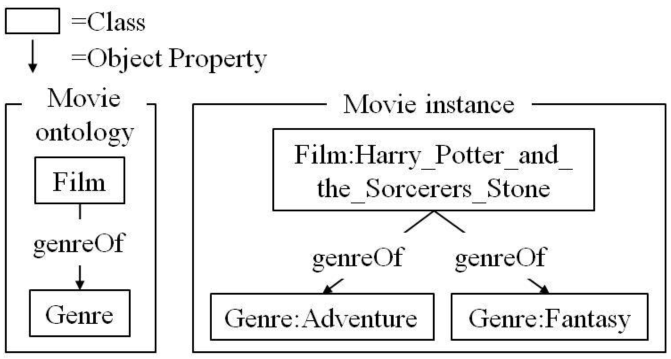


Fig. 3. Movie ontology and its instances

4. Illustrative Examples

In this section, we show how the proposed model can be used to describe various aspects of the Semantic Web environment.

4.1. Scenario 1

In the Semantic Web, an instance of an ontology can be semantically related to an instance of another ontology. For example, figure 4 shows how two domains (Movie and Travel) are connected.

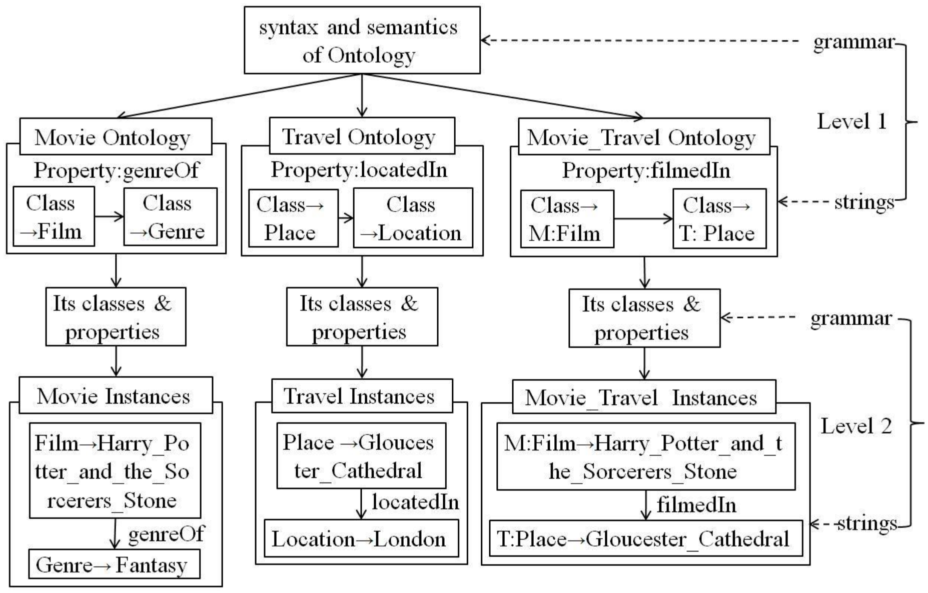


Fig. 4. Movie, Travel, Movie_Travel ontology and their instances

This situation can be easily described using the proposed model. Movie ontology has Film and Genre class, and genreOf property which creates a relation between Film's and Genre's instances.

```
Ontology Movie
Class Genre
Class Film
  Restriction on Property genreOf equals 1
ObjectProperty genreOf domain Film range Genre
```

Based on Movie Ontology, Film class has Harry_Potter_and_the_ Sorcerers_Stone in-stance and Genre class has Fantasy.

```
Genre
  Fantasy
Film
 Harry_Potter_and_the_Sorcerers_Stone genreOf Fantasy
```

Travel Ontology has Location and Spot class, and locatedIn property which creates a relation between Location's and Spot's instances.

```
Ontology Travel
Class Location
Class Place
  Restriction on Property locatedIn equals 1
ObjectProperty locatedIn domain Place range Location
```

Based on Travel Ontology, Location class has London instance and Place class has Gloucester_Cathedral.

```
Location
  London
Place
  Gloucester_Cathedral locatedIn London
```

We create Movie_Travel ontology by importing Movie and Travel ontology. It uses Film class of Movie ontology and Spot class of Travel ontology. It has filmedIn property which creates a relation between Film's and Spot's instances.

```
Ontology Movie_Travel
NS:M=Movie
NS:T=Travel
Class T:Place
Class M:Film
Restriction on Property filmedIn equals 1
ObjectProperty filmedIn domain M:Film range T:Place
```

Based on Movie_Travel Ontology, Film class has Harry_Potter_and_the_Sorcerers_Stone instance and Place class has Gloucester_Cathedral.

```
T: Place
 Gloucester_Cathedral
M: Film
 Harry_Potter_and_the_Sorcerers_Stone filmedIn
loucester_Cathedral
```

People can get information for traveling a place where a famous movie is filmed by combining Film and Travel ontology.

4.2. Scenario 2

Historical study often focuses on events and developments that occur in particular blocks of time. Therefore, the events and developments might be organized based on historical periods such as Ancient history, Middle Ages, Early modern period, Modern era and Post-Modern. Assume that there are three types of ontologies which define different classes and properties as follows.

```
Ontology Ancient_History
Class Nation
Class Machine
  Restriction on Property inventedBy equals 1
ObjectProperty inventedBy domain Machine range Nation

Ontology Middle_Ages
Class Area
Class Invention
  Restriction on Property introducedFrom equals 1
```

```
ObjectProperty introducedFrom domain Invention  range
Area

Ontology Modern_Era
Class Country
Class Technology
  Restriction on Property developedIn equals 1
ObjectProperty developedIn domain Technology range
Country
```

Assume that a person wants to organize information on technology in history by using an ontology, but historians already organized the information on technology as well as war, religion, or science based on different ontologies by the historical periods. The person tries to extract the classes or properties related to science and technology from different ontologies and to integrate them in an ontology. That can be done by importing the three ontologies. So, the person can define the namespace for each ontology.

```
Ontology History_Of_Technology
NS:AH = Ancient_History
NS:MA = Middle_Ages
NS:ME = Modern_Era
AH:Nation
  Egypt
AH:Machine
  Ramp AH:inventedBy Egypt
  Lever AH:inventedBy Egypt
MA:Area
  East
MA:Invention
  Compass MA:introducedFrom East
  Gunpower MA:introducedFrom East
  Silk MA:introducedFrom East
  Astrolabe MA:introducedFrom East
ME:Country
  Britain
ME:Techology
  StreamEngine ME:developedIn Britain
Program Code
```

4.3. Scenario 3

People organize resources based on their interests or needs. One resource can be classified differently because their interests or needs are different. The proposed model allows users to create a specification file which contains the lexical definitions and the grammar of their own ontology language and define ontology which represents the meaning of terms and the relationships between those terms by using the ontology language. If the web resources are reorganized based on their own ontologies, the users can conveniently navigate the web resources according to their interests or needs without wasting a lot of time.

As an example, assume that a user wants to search web resources in an Internet art museum. An Internet art museum has lots of art works and users navigate them based on their interests. Some search the art works of the artists who they like such as van Gogh, Picasso, Millet, etc. Others search the art works according to the trend of art such as realism, impressionism, cubism, etc. The others search the art works of art forms which they are interested in such as drawing, painting, sculpture, etc. If the artworks re-organized by their interests are displayed, the users navigate them conveniently and find out desired resources easily.

If user A wants to browse the art works based on painting styles, the user can use the following ontology.

```
Ontology Painting_Style
Class Work
   Restriction on Property belongTo equals 1
Class Art_Movement
ObjectProperty belongTo domain Work range Art_Movement

Art_Movement
   Realism
   Impressionism
   Cubism
Work
   Work_1 belongTo Realism
```

If user B wants to browse the art works based on artists, the user can use the following ontology.

```
Ontology Painting_Artist
Class Work
   Restriction on Property belongTo equals 1
Class Artist
ObjectProperty paintedBy domain Work range Artist

Artist
   Picasso
   Van_Gogh
   Millet
Work
   Work_1 paintedBy Millet
```

If user C wants to browse the art works based on art forms, the user can use the following ontology.

```
Ontology Painting_Medium
Class Work
   Restriction on Property madeOf equals 1
```

```
Class Medium
ObjectProperty madeOf domain Work range Medium

Medium
   Fresco
   Oil
   Watercolor
Work
   Work_1 madeOf Oil
```

Figure 5 shows how the resources can be organized according to users' interests. .

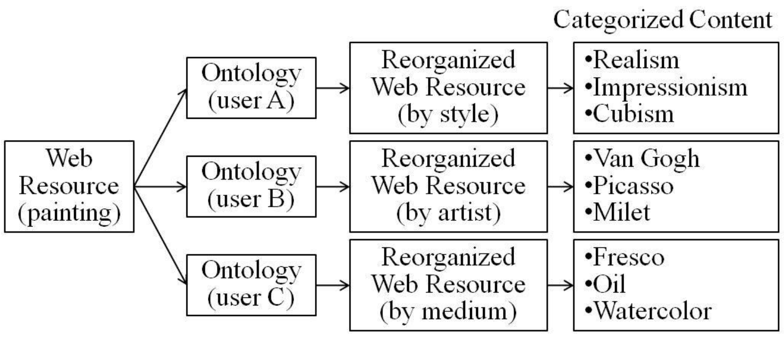


Fig. 5. Organization of resources based on users' interests

4.4. Scenario 4

The proposed model can be used to create Linked Data. Each user can build an ontology about a certain domain by using our model and create instances based on the ontology. Each instance can be regarded as the description about a raw data that each user has and so they are similar to the descriptions of data in Linked Data by using the standards like RDF.

Linked Data is an approach to expose, share, and connect pieces of data, information, and knowledge on the Semantic Web using URIs and RDF. If some users can define a common ontology together, create instances to describe their raw data, and share the instances by using our model, it is possible to construct Linked data. For example, there are people who are interested in art. They open a community in a social network for sharing their data. They first build a general ontology about art domain as follows;

```
Ontology Art
Class Painting
Class Artist
ObjectProperty artist domain Painting range Artist
```

```
DatatypeProperty title domain Painting range String
DatatypeProperty year domain Painting range String
```

They also create the instances which describe their data based on the Art ontology as follows;

Table 1. Instances created by each user

User A	User B	User C	User D
Ontology ART Artist Van_Gogh Painting work_1 artist Van_Gogh Painting work_1 title Sunflower Painting work_1 year 1889	Ontology ART Artist Van_Gogh Painting work_2 artist Van_Gogh Painting work_2 title Self_Portrait Painting work_2 year 1886	Ontology ART Artist Picasso Painting work_3 artist Picasso Painting work_3 title Guernica Painting work_3 year 1937	Ontology ART Artist Picasso Painting work_4 artist Picasso Painting work_4 title Massacre_in_Korea Painting work_4 year 1951

Then, they share the instances in their community. It is possible kind of linked data services. For example, they can find all data linked to a certain artist such as Van_Gogh or Picasso. Even though each user has small data, they can get an amount of linked data and also create new information from the linked data.

5. Constructing a small scale Semantic Web environment

In this section, we show how a user can construct a small-scale Semantic Web environment using the system we implemented. In order to create a small-scale Semantic Web environment, a user defines a grammar for the ontology language and creates a parser by using SableCC [11] that is a parser generator which creates object-oriented frameworks for building compilers, interpreters, and other text parsers. For describing the ontology, the user needs to create a SableCC specification file which contains the lexical definitions and the grammar productions of an ontology language.

Figure 6 shows how a user can construct a small-scale Semantic Web environment using the system. First, a user defines an ontology language grammar. The grammar file is written in a SableCC specification file format. Then, the user creates an ontology compiler by launching SableCC on the grammar file. The user writes and compiles Java sources for an ontology compiler that checks grammatical errors of the ontology defined by the ontology language and is aware of its classes and their relationships. If there are no errors, an ontology in XML format is written. The user writes and saves the instances in a XML document if it is defined based on the classes and properties of the defined ontology.

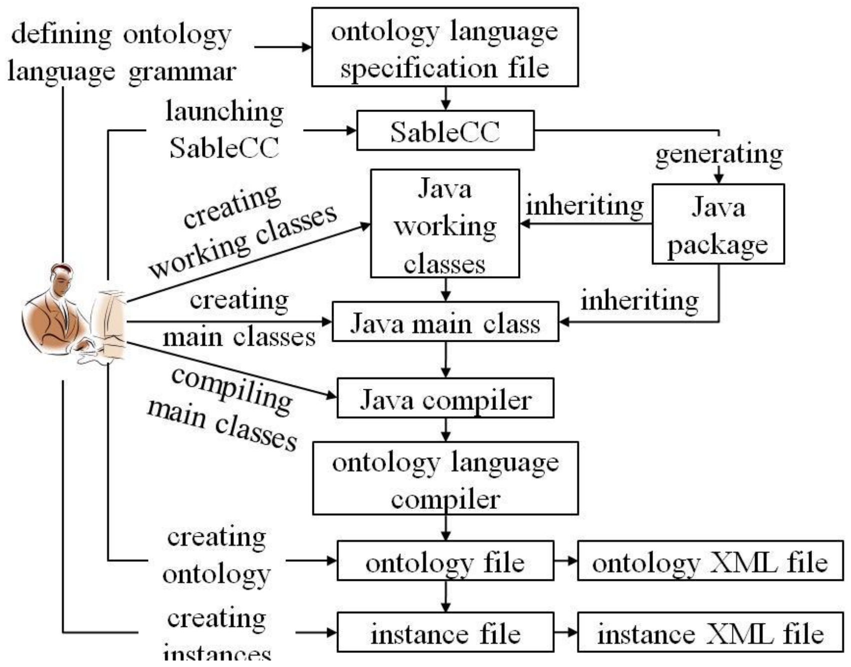
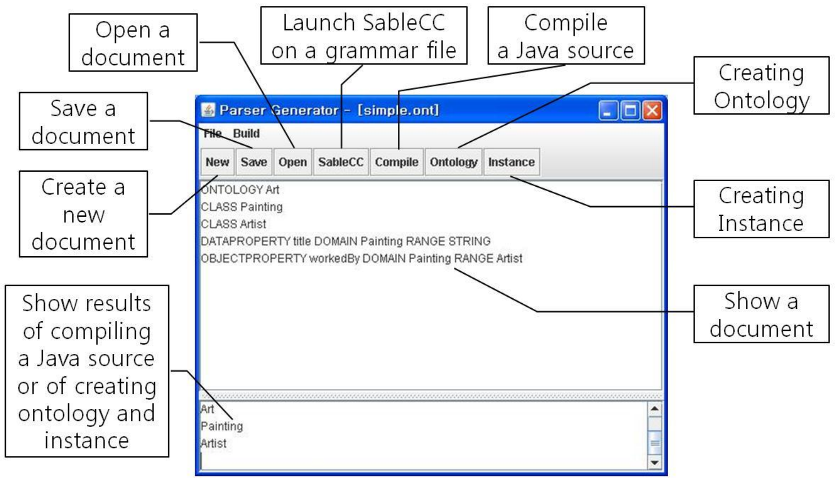


Fig. 6. Steps for constructing a small-scale Semantic Web environment.

Figure 7 shows the screenshot of the user interface captured when a user creates an ontology.



```
ONTOLOGY Art
CLASS Painting
CLASS Artist
DATAPROPERTY title DOMAIN Painting RANGE STRING
OBJECTPROPERTY workedBy DOMAIN Painting RANGE Artist
```

```
Art
Painting
Artist
```

Fig. 7. Art ontology and its instances

Figure 8 shows the situation used in the example that follows.

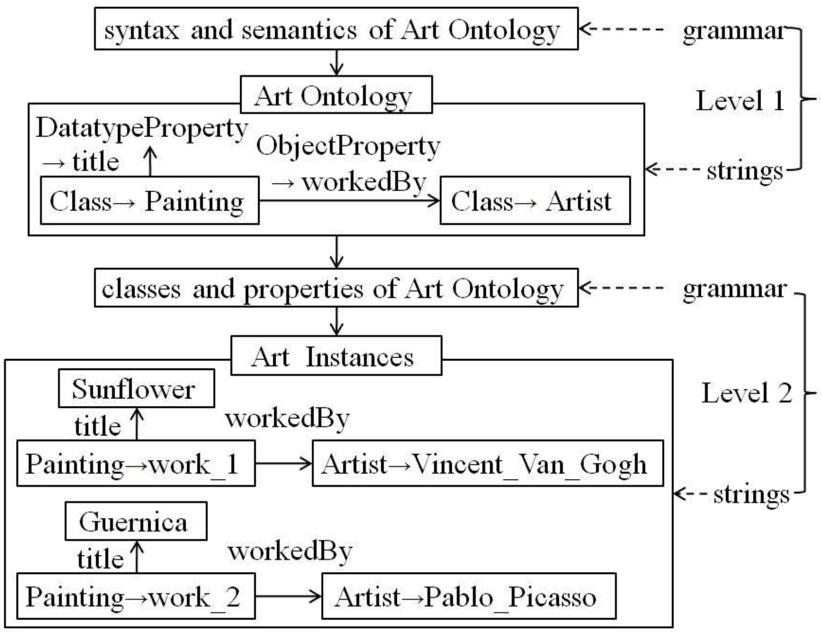
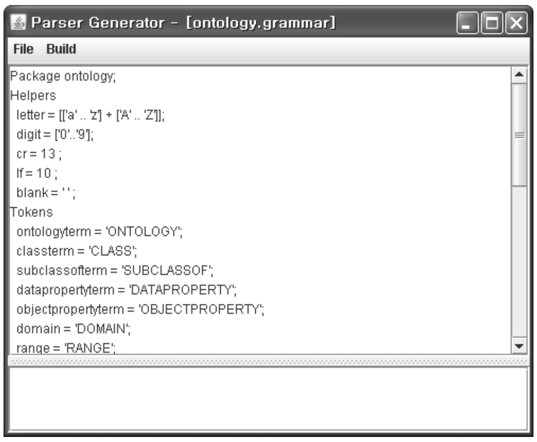


Fig. 8. Art ontology and its instances

We assume that the environment consists of an Art ontology and its instances. More specifically, in the Art ontology, there are a Painting class and an Artist class. The title datatype property defines the Painting class as its domain and a string type as its range. An individual of the Painting class has a title value. The workedBy object property defines the Painting class as its domain and the Artist class as its range. An individual of the Painting class has an individual of the Artist class as its workedBy value. Instances of of Art ontology are created by creating two individuals and assigning their properties. We define a Painting with an ID of work_1 and specify that it is worked by (workedBy) Vincent_Van_Gogh and its title is Sunflower. We also define a Painting is an ID of work_2 and specify that it is worked by (workedBy) Pablo_Picasso and its title is Guernica.

The steps for constructing a small-scale Semantic Web environment are as follows.

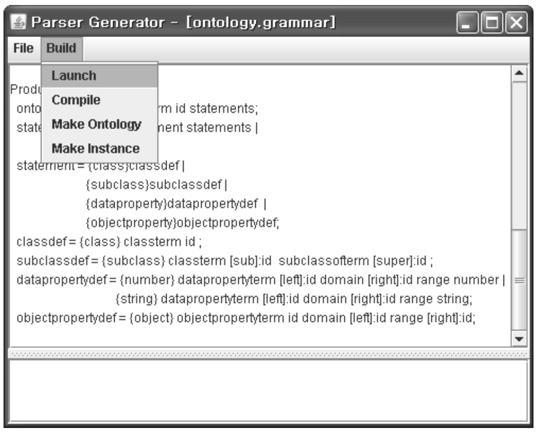
1. A user defines the grammar of an ontology language to be compiled and saves it as a specification file. Figure 9 shows the screenshot of the the specification file.



```
Package ontology;
Helpers
 letter = [['a' .. 'z'] + ['A' .. 'Z']];
 digit = ['0'..'9'];
 cr = 13 ;
 lf = 10 ;
 blank = ' ' ;
Tokens
 ontologyterm = 'ONTOLOGY';
 classterm = 'CLASS';
 subclassofterm = 'SUBCLASSOF';
 datapropertyterm = 'DATAPROPERTY';
 objectpropertyterm = 'OBJECTPROPERTY';
 domain = 'DOMAIN';
 range = 'RANGE';
```

Fig. 9. Screenshot of the specification file

2. The user launches SableCC on the specification file by clicking [Build]-[Launch] (Figure 10). It generates a framework which consists of four packages such as lexer, parser, node and analysis.



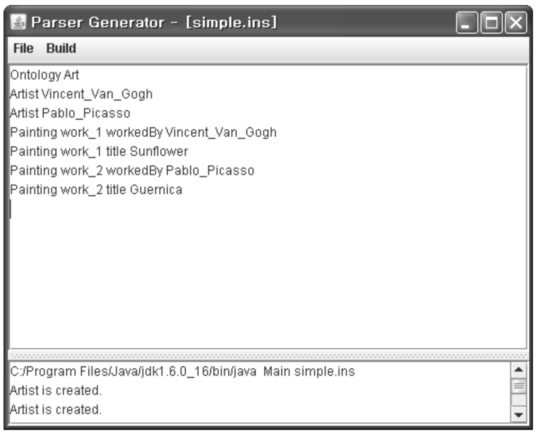
```
Produ
 onto                  rm id statements;
 state                 nent statements |

 statement = {class}classdef |
             {subclass}subclassdef |
             {dataproperty}datapropertydef  |
             {objectproperty}objectpropertydef;
 classdef = {class} classterm id ;
 subclassdef = {subclass} classterm [sub]:id  subclassofterm [super]:id ;
 datapropertydef = {number} datapropertyterm [left]:id domain [right]:id range number |
                   {string} datapropertyterm [left]:id domain [right]:id range string;
 objectpropertydef = {object} objectpropertyterm id domain [left]:id range [right]:id;
```

Fig. 10. Launching SableCC on the specification file

3. The user creates working classes which inherit fields and methods from the classes of the Java packages. The working classes contain the core compiler functionalities. If an input is an ontology file, it finds classes and properties of the ontology and saves them and their relationships as an XML file. If an input is an instance file, it finds individuals, their properties and values, and saves them as an XML file.
4. The user also creates a main compiler class which activates lexer, parser, and working classes. The main class reads an ontology file which is

defined by the user. If an input is an ontology file, it checks whether the ontology is defined according to the grammar of the ontology language. If an input is an instance file, it checks whether the instances are defined based on the vocabulary of the ontology.

5. Then, the user compiles the main compiler with a Java compiler and the application generates an ontology language compiler. If the main compiler has any error, the user can debug it. In this example, the user saves a compiler program as "Main.java" and compiles it by clicking [Build]-[Compile].
6. The user creates an ontology which contains classes and properties about a domain according to the ontology language grammar and compiles it with the ontology language compiler. The application generates an ontology XML file. For example, the user defines an Art ontology and saves it as "simple.ont". The ontology can be created if the user clicks [Build]-[Make Ontology]. If there is no syntax and semantic error, the system produces an ontology.
7. The user also creates its instances which are defined by the classes and properties of the ontology and compiles it with the ontology language compiler. The application generates an instance XML file. In this example, the user defines instances based on Art ontology and saves it as "simple.ins". The user creates instances based on the ontology by clicking [Build]-[Make Instance]. If the instances are defined by using the classes and properties of the ontology, the system produces an ontology(Figure 11).



```
Ontology Art
Artist Vincent_Van_Gogh
Artist Pablo_Picasso
Painting work_1 workedBy Vincent_Van_Gogh
Painting work_1 title Sunflower
Painting work_2 workedBy Pablo_Picasso
Painting work_2 title Guernica
```

```
C:/Program Files/Java/jdk1.6.0_16/bin/java  Main simple.ins
Artist is created.
Artist is created.
```

Fig. 11. Creation of instances

6. Comparison with other ontology development tools

In this section, we compare the proposed system with two well-known ontology development tools, Protégé [13] and Apollo [27]. Specifically, we show how the scenario given in section 5 can be realized using Protégé and Apollo in section 6.1 and 6.2, respectively. In addition, we show parts of an ontology and an instance from three systems to illustrate the differences in section 6.3. The following table summarizes briefly the differences.

Table 2. Comparison between Protégé-OWL, Apollo, and our system

System	Language	Prerequisite	Usage
Protégé-OWL	OWL/RDF	Understanding OWL/RDF(S) vocabularies	Building general ontologies & their instances
Apollo	OKBC model	Understanding OKBC Knowledge Model	Building general ontologies & their instances
Our system	User-defined language	Understanding formal languages & parsing	Constructing a semantic web environment easily

Protégé-OWL and Apollo are developed for implementing metadata of ontology using the languages used to encode the ontology. They generally require users to be trained for the languages, knowledge representation, and predicate logic. For example, Protégé-OWL supports the Web Ontology Language (OWL) and exports ontologies to OWL/RDF (Resource Description Framework). It requires users understand the vocabularies of RDF(S) and OWL and their functions. Apollo is a knowledge modeling application based on the internal model of the OKBC (Open Knowledge Base Connectivity) protocol and export ontologies to CLOS (Common LISP Object System) and OCML (Options Configuration Modeling Language). It also requires users understand the meaning of each concept, the operations, and the naming and argument conventions provided in OKBC specifications. Our system allows users to use languages that they define. It requires the users to have a basic understanding of formal languages and parsing which undergraduate students generally learn from a compiler course.

Protégé-OWL and Apollo are developed for all stages of the ontology lifecycle such as creation, population, validation, deployment, maintenance and evolution. However, our system is developed for undergraduate students to construct an environment that is structurally similar to the Semantic web which consists of ontologies and their instances so that they can understand the structural properties of the Semantic Web while studying the Semantic Web. The combination of theory and practice can help them understand the Semantic Web clearly. On top of this, they can also conduct experimentations on Semantic Web applications that run in the environment so constructed.

The ontologies in Protégé-OWL or Apollo are represented by using general languages or knowledge representation such as RDF(S), OWL, OKBC model, etc. They can be reused and shared with other applications using the same language. However, users should understand the technical terminologies of the language or its specification and it can be difficult [28,29]. In our system, users can simply build ontologies and their instances which do not use technical terminologies and logic. Even though the ontologies are not represented by the general ontology language, the system can be helpful for the undergraduate students who have basic knowledge about computer science to understand the Semantic web. In addition, although our system now represents the ontology by XML, it can be easily extensible to represent the ontology by OWL/RDF(S).

6.1. Protégé-OWL

The steps to create an Art ontology given in section 5 are as follows.

1. We start Protégé-OWL and create a new OWL project by clicking “Create New Project”. When “Create New Project” wizard appears, we select a project type, “OWL/RDF Files” and specify a unique URI that will become the identifier for the ontology. Then, we select an OWL/RDF dialect such as OWL DL.
2. We create classes for concepts in the ontology. We select the “OWL Classes” tab. It shows the hierarchy of classes. All the classes will be created subordinate to owl:Thing. We click the Create subclass button. A class is created with a generic name such as “Class_1”. We rename the class using the “class name widget” to “Artist”.

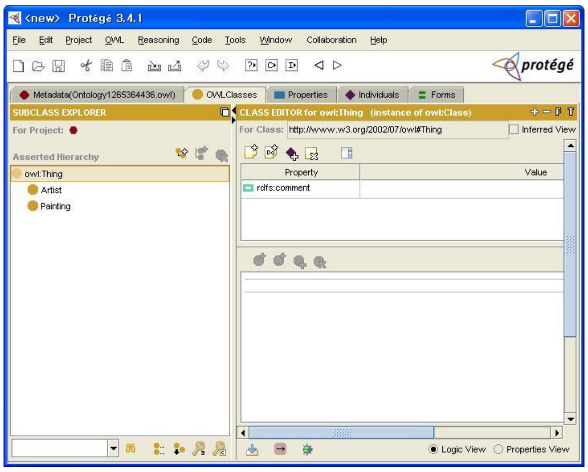
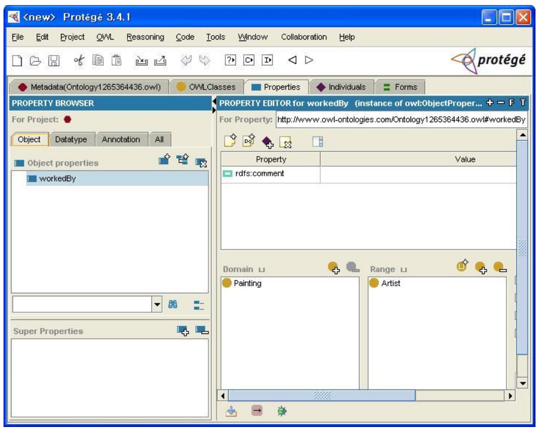


Fig. 12. Creation of classes

We repeat the previous step to add the class "Painting" (Figure 12).



<new> Protégé 3.4.1
File Edit Project OWL Reasoning Code Tools Window Collaboration Help
protégé
Metadata(Ontology1265364436.owl) OWLClasses Properties Individuals Forms
PROPERTY BROWSER
For Project:
Object Datatype Annotation All
Object properties
workedBy
Super Properties
PROPERTY EDITOR for workedBy (instance of owl:ObjectProper...
Property Value
rdfs:comment
Domain
Range

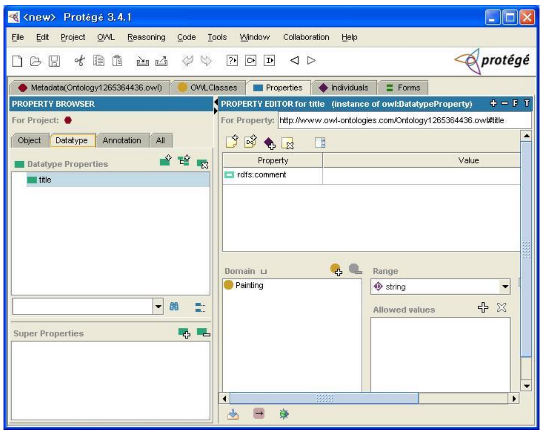
For Property: http://www.owl-ontologies.com/Ontology1265364436.owl#workedBy

Domain: Painting

Range: Artist

Fig. 13. Creation of an Object property

3. We create properties of these classes, for example, the title of the painting and the artist that painted it. We switch to the "Properties" tab. We click the "Create Object Property" button to create a new Object property. An Object property is created with a generic name. We rename the property to "workedBy". Then, we specify a domain and a range of the Object property. We press the "Add named class" button on the "Domain Widget" and select the class "Painting". We also press the "Add named class" button on the "Range Widget" and select the class "Artist" (Figure 13).



<new> Protégé 3.4.1
File Edit Project OWL Reasoning Code Tools Window Collaboration Help
protégé
Metadata(Ontology1265364436.owl) OWLClasses Properties Individuals Forms
PROPERTY BROWSER
For Project:
Object Datatype Annotation All
Datatype Properties
title
Super Properties
PROPERTY EDITOR for title (instance of owl:DatatypeProperty)
Property Value
rdfs:comment
Domain
Range
Allowed values

For Property: http://www.owl-ontologies.com/Ontology1265364436.owl#title

Domain: Painting

Range: string

Fig. 14. Creation of a Datatype property

4. We click the "Create Datatype Property" button to create a new Datatype property. A Dataproperty is created with a generic name. We rename the

property to “title”. Then, we specify a domain and a range of the Datatype property. We press the “Add named class” button on the “Domain Widget” and select the class “Painting”. We select the item “string” on the “Range Widget” (Figure 14).

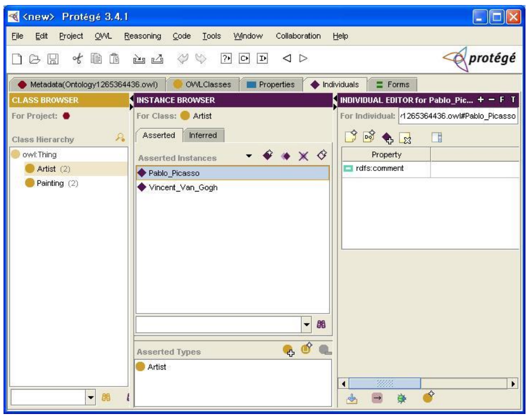


Fig. 15. Creation of instances of the class Artist

5. We create some instances of the classes. We switch to the “Individuals” tab. We select the class “Artist”. We press the “Create Instance” button. An instance is created with a generic name. We rename the instance to Vincent_Van_Gogh. We also create another instance called “Pablo_Picasso” (Figure 15).

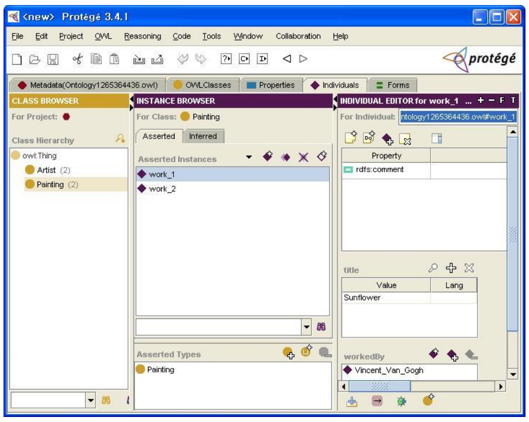


Fig. 16. Creation of instances of the class Painting

6. We select the class “Painting” and press the “Create Instance” button. An instance is created with a generic name. We rename the instance to “work_1”. We press the “Add new value” button in the Datatype property

"title" and type "Sunflower" as its value. We also press the "Add new value" button in the Object property "workedBy" and select the instance "Vincent_Van_Gogh" as its value.

7. We repeat step 6 to create another instance called "work_2". We press the "Add new value" button in the Datatype property "title" and type "Guernica" as its value. We also press the "Add new value" button in the Object property "workedBy" and select the instance "Pablo_Picasso" as its value (Figure 16).

6.2. Apollo

The steps to create an Art ontology given in section 5 are as follows.

1. We start Apollo and a new project. We click "Create New Project" and open the "Create new ontology" dialog and enter its name, "Art".
2. We create classes for concepts in the ontology. We open the "New class" dialog in the focused ontology and type the class name "Artist". We repeat the previous step to add the class "Painting" (Figure 17).

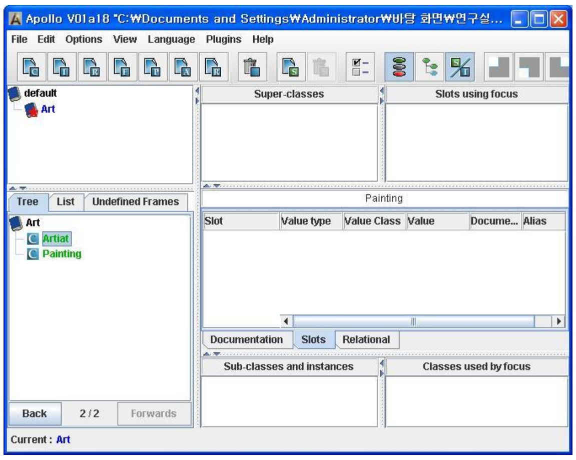
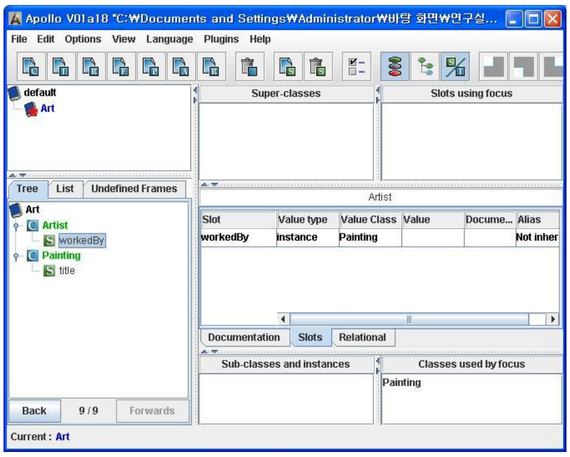


Fig. 17. Creation of classes

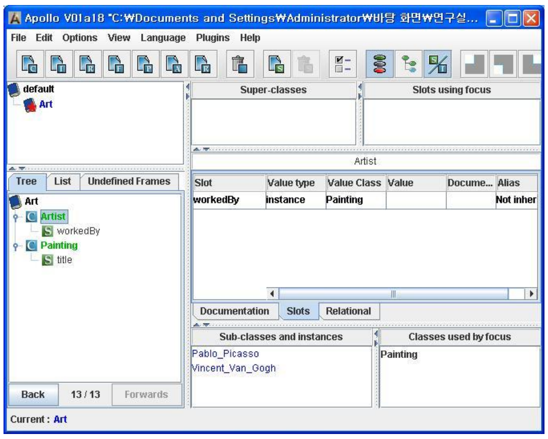
3. We create property slots of the classes. A slot contains a number of facets such as value, value type, and value class, etc. We open the "New slot" dialog in the class "Artist". We type its name "workedBy" and set its value type "instance" and its value class "Painting". We also open the "New slot" dialog in the class "Painting". We type its name "title" and set its type "string" (Figure 18).


Apollo V01a18 "C:₩Documents and Settings₩Administrator₩바탕 화면₩연구실...
File Edit Options View Language Plugins Help
default
Art
Tree List Undefined Frames
Art
Artist
workedBy
Painting
title
Back 9 / 9 Forwards
Current : Art
Super-classes
Slots using focus
Artist
Documentation Slots Relational
Sub-classes and instances
Classes used by focus
Painting

Slot	Value type	Value Class	Value	Docume...	Alias
workedBy	instance	Painting			Not inher

Fig. 18. Creation of property slots

4. We create some instances of the classes. We select the class “Artist” and open the “New instance” dialog. We type the instance name “Vincent_Van_Gogh” and its type “Artist”. We also open the “New instance” dialog again. We type the instance name “Pablo_Picasso” and set its type “Artist” (Figure 19).

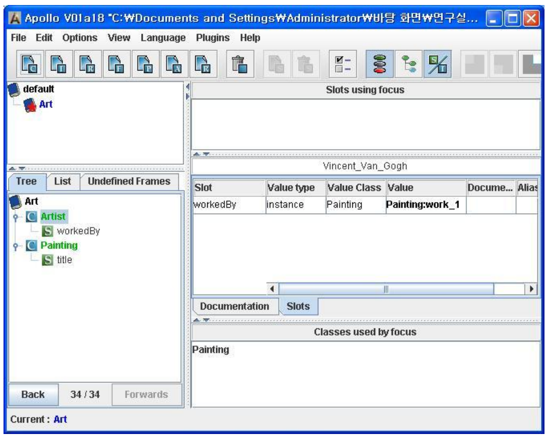

Apollo V01a18 "C:₩Documents and Settings₩Administrator₩바탕 화면₩연구실...
File Edit Options View Language Plugins Help
default
Art
Tree List Undefined Frames
Art
Artist
workedBy
Painting
title
Back 13 / 13 Forwards
Current : Art
Super-classes
Slots using focus
Artist
Documentation Slots Relational
Sub-classes and instances
Pablo_Picasso
Vincent_Van_Gogh
Classes used by focus
Painting

Slot	Value type	Value Class	Value	Docume...	Alias
workedBy	instance	Painting			Not inher

Fig. 19. Creation of instances of the class Artists

5. Similarly, we select the class “Painting” and open the “New instance” dialog. We type the instance name “work_1” and set its type “Painting”. We also open the “New instance” dialog again. We type the instance name “work_2” and set its type “Painting”.
6. We specify the facets of the instances of the class “Artist”. We select the class “Artist” and the instance “Vincent_Van_Gogh” in sub-classes panel.

We double-click the value of the slot “workedBy” and select the instance “work_1”. We select the instance “Pablo_Picasso” in sub-classes panel. We double-click the value of the slot “workedBy” and select the instance “work_2” (Figure 20).



Slot	Value type	Value Class	Value	Docume...	Alias
workedBy	instance	Painting	Painting:work_1		

Fig. 20. Specifying the facets of the instances of the class Artist

7. Similarly, we specify the facets of the instances of the class “Painting”. We select the class “Painting” and the instance “work_1” in sub-classes panel. We edit the string value of the slot “title” to “Sunflower”. We select the instance “work_2” in sub-classes panel. We edit the string value of the slot “title” to “Guernica”.

6.3. Comparison of the results

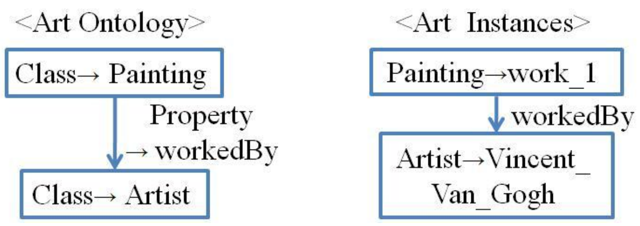


Fig. 21. Two classes and instances

To illustrate the differences and similarities among our system, Apollo, and Protégé-OWL, we extract parts of the documents where they save the ontology and its instances. Protégé-OWL saves them in OWL format, and Apollo and our system do in XML format. There are two classes “Artist” and “Painting” and the property “workedBy”. There are also instances “work_1” and “Vicent_Van_Gogh” which are connected with the property “workedBy” (Figure 21).

The following table shows the ontology generated from the three systems.

Table 3. Ontologies generated by Protégé-OWL, Apollo, and our system

Task	Apollo (XML)	Protégé-OWL (OWL)	Our system (XML)
Creating Ontology	<classes> <class name=”Artist”/> <class name=”Painting”/> <slots> <slot name=”workedBy”> <type value=”instance”/> <is_own value=”false”/> <value_class value=”Artist”/> <value_type value=”instance”/> </slot> </slots> <classes>	<owl:Class rdf:ID=”Painting”/> <owl:Class rdf:ID=”Artist”/> <owl:ObjectProperty rdf:ID=”workedBy”> <rdfs:range rdf:resource=”#Artist”/> <rdfs:domain rdf:resource=”#Painting”/> </owl:ObjectProperty>	<CLASS name= ‘Painting’/> <CLASS name=‘Artist’/> <OBJECTPROPERTY Name=’workedBy’> <DOMAIN name=’Painting’/> <RANGE name= ‘Artist’/> </OBJECTPROPERTY>

The ontological instances generated from the systems are as follows.

Table 4. Ontologial instances generated by Protégé-OWL, Apollo, and our system

Task	Apollo (XML)	Protégé-OWL (OWL)	Our system (XML)
Creating Instance	<instance name = “Vincent_Van_Gogh”Class =”Artist”/> <instance name =“work_1” class=“Painting”> <slots> <slot name=”title”> <value value= “Sunflower”/> </slot> <slot name= “workedBy”> <value value=”Artist: Vincent_Van_Gogh”/> </slot> </slots> </instance>	<Painting rdf:ID= “work_1”> <workedBy> <Artist rdf:ID=”Vincent_Van_Gogh”/> </workedBy> </Painting>	<Artist name= ’Vincent_Van_Gogh> <Painting name= ‘work_1’> <workedBy value = ’Vincent_Van_Gogh’> </workedBy> </Painting>

7. Conclusions and Future Works

In this paper, we presented a two-level model for the Semantic Web. The model consists of two grammars, where one grammar is used to model ontologies and the other grammar is used to model ontological instances. We implemented a system by which a user can easily construct a small-scale Semantic Web environement.

Our model can be utilized as follows.

First, a personalized semantic web can be easily constructed. The Semantic Web is a linked information space where data is being enriched and added based on the standards to formalize the syntactic and semantics of web contents. It encourages users to create, share, and reuse resources related to their needs and interests. Especially, the rapid increase of communities promotes the interaction with each other and development of a shared repository of resources. However, it is not easy for average users to handle the languages to construct the semantic web such as RDF or OWL. Our model enables the users to design an ontology language for a domain of their interests and represent web contents by using the language. The users can represent their own contents and connect to others in a shared domain of interests easily because they use languages that are easier than RDF or OWL. They can construct their personalized semantic webs.

Second, constructing knowledge is easily done. People construct their knowledge by connecting existing knowledge into new knowledge, but their knowledge construction is different each other. Although they are given the same resources, they organize them in different ways because they have various views about the resources. The proposed model enables users to define and utilize resources according to their views easily. They define ontologies and describe resources based on the ontologies to organize the resources. They can reuse resources made by others in new and exciting contexts as well. It can help them build knowledge.

Third, a semantic social network can be easily created. In the Semantic Web, users create online communities where they can create, collect and share resources. Especially, a social network is a community where members with a shared interests interact and develop shared contents. If they can construct a small semantic web suited to their own community, they can represent resources semantically and share meaningful information. The proposed model enables them to construct a semantic social network according to their interests and needs.

We are currently investigating ways by which a logical inference mechanism can be supported in the proposed model. We are also working on a tool that that can exploit the structural properties of the Semantic Web such as Magpie [25], Piggy Bank [12], Potluck [26], etc. using a structurally similar environment to the Semantic Web created by our system.

References

1. McGuinness, D. L. and van Harmelen, F.: OWL Web Ontology Language Overview, W3C Recommendation, http://www.w3.org/TR/owl-features. (2004)
2. Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D. L., Patel-Schneider, P. F. and Stein, L. A.: OWL Web Ontology Language Reference, W3C Recommendation, http://www.w3.org/TR/owl-ref. (2004)
3. Rodriguez, M. A. and Bollen, J.: Modeling Computations in a Semantic Network, Computing Research Repository (CoRR), ACM, abs/0706.0022. (2007)
4. Antoniou, G., and van Harmelen, F.: A Semantic Web Primer, The MIT Press (2004)
5. Berners-Lee, T., Hendler, J., and Lassila, O.: The Semantic Web, Scientific American Special online Issue. (2001)
6. Heflin, J., Volz, R. and Dale, J.: Web Ontology Requirements, Proposed W3C Working Draft, http://km.aifb.uni-karlsruhe.de/projects/owl/index.html. (2002)
7. Decker, S., Erdmann, M., Fensel, D. and Studer, R.: Ontobroker: Ontology Based Access to Distributed and Semi-Structured Information, Kluwer Academic Publishers. (1998)
8. Patel-Schneider, P. F., Hayes, P. and Horrocks, I.: OWL Web Ontology Language Semantics and Abstract Syntax, W3C Recommendation, http://www.w3.org/TR/owl-semantics. (2004)
9. Pan, Z., Qasem, A. and Heflin, J.: An Investigation into the Feasibility of the Semantic Web, In Proceedings of the Twenty First National Conference on Artificial Intelligence (AAAI 2006). (2006)
10. Schraefel, M. C.: What is an analogue for the semantic web and why is having one important?, In Proceedings of the eighteenth conference on Hypertext and hypermedia, 123-132. (2007)
11. Gagnon, E. M. and Hendren, L. J.: SableCC, an Object-Oriented Compiler Framework, In Proceedings of the Technology of Object-Oriented Languages and Systems, IEEE Computer Society. (1998)
12. Huynh, D., Mazzocchi, S., and Karger, D.: Piggy Bank: Experience the Semantic Web Inside Your Web Browser, International Semantic Web Conference. (2005)
13. http://protege.stanford.edu
14. Martin, B., Mitrovic, A., Suraweera, P.: ITS Domain Modelling with Ontology. Journal of Universal Computer Science, Vol. 14, No. 17, 2758-2776. (2008)
15. Sure, Y., Erdmann, M., Angele, J., Staab, S., Studer, R., Wenke, D.: OntoEdit: Collaborative Ontology Development for the Semantic Web. Proceedings of ISWC 2002, Springer, LNCS 2342, 221-235. (2002)
16. Mizoguchi, R.: Tutorial on ontological engineering - Part 2: Ontology development, tools and languages, New Generation Computing, OhmSha&Springer, Vol.22, No.1, 61-96. (2004)
17. Corcho, O., Fernandez-Lopez, M., Gomez-Perez, A., Vicente, O.: WebODE: An Integrated Workbench for Ontology Representation, Reasoning and Exchange, Proceedings of EKAW2002, Springer, LNAI 2473, 138-153. (2002)
18. Bizer, C., Cyganiak, R., Heath, T.: How to Publish Linked Data on the Web. http://www4.wiwiss.fu-berlin.de/bizer/pub/LinkedDataTutorial. (2007)
19. Berners-Lee, T.: Linked Data. http://www.w3.org/DesignIssues/LinkedData.html. (2009)
20. Boss, G., Malladi, P., Quan, D., Legregni, L., Hall, H.:Cloud Computing. IBM Corporation (2007)

21. SHEU, P. C-Y, Wang, S., Wang, Q., Hao, K., Paul, R.: Semantic Computing, Cloud Computing, and Semantic Search Engine. IEEE International Conference on Semantic Computing, 654-657. (2009)
22. http://www.cloudforum.org
23. http://en.wikipedia.org/wiki/Social_Semantic_Web
24. http://syntheticbiology.org/Semantic_web_ontology/Software.html
25. http://projects.kmi.open.ac.uk/magpie/main.html
26. http://simile.mit.edu/potluck
27. http://apollo.open.ac.uk
28. Wang, H., Horridge, M., Rector, A.: Debugging OWL-DL Ontologies: A Heuristic Approach, Proceedings of ISWC 2005, LNCS 3729 (2005)
29. Chung, M., Oh, S., Kim, K., Cho, H., Cho, H.: Visualizing and Authoring OWL in ezOWL, International Conference on Advanced Communication Technology, 528-531. (2005)

Hyosook Jung received bachelor's degree in the department of elementary education from Seoul National University of Education in Seoul, Korea, master's degree in the department of elementary computer education from Seoul National University Graduate School of Education in Seoul, Korea and doctoral degree in the department of computer science education from Korea University in Seoul, Korea. Her research interests include Semantic Web, adaptive hypermedia and computer science education.

Seongbin Park received bachelor's degree in the Department of Computer Science from Korea university in Seoul, Korea, and both master's degree and doctoral degree in the department of computer science from the University of Southern California. He is currently an associate professor at the department of computer science education of Korea University in Seoul, Korea. His research interests include Semantic Web, adaptive hypermedia and computer science education.

Received: October 11, 2009; Accepted: February 23, 2010.