2005년 8월
석사학위논문

도메인 온톨로지 구축을 위한 효율적인 방법론 제안

조선대학교 대학원

전 자 계 산 학 과

유 해 도

도메인 온톨로지 구축을 위한 효율적인 방법론 제안

Proposing an Effective Methodology for Developing Domain Ontology

2005년 8월 일

조선대학교 대학원

전 자 계 산 학 과

유 해 도

도메인 온톨로지 구축을 위한 효율적인 방법론 제안

지도교수 김 판 구

이 논문을 이학석사학위신청 논문으로 제출함.

2005년 4월 일

조선대학교 대학원

전자계산학과

유 해 도

유해도의 석사학위논문을 인준함

위원장 조선대학교 교수 _____

위　원 조선대학교 교수 _____

위　원 조선대학교 교수 _____

2005년　5월　일

조선대학교 대학원

Contents

Figure Contents

Table Contents

ABSTRACT

Proposing an Effective Methodology for Developing Domain Ontology

Haitao Liu
Advisor : Prof. Kim, Pan-Koo Ph.D.
Department of Computer Science,
Graduate School of Chosun University

Ontology developing process has aroused a lot of controversy among knowledge engineers and knowledge users. The recent surges on ontology building methodologies and practical ontology applications have explored a broad spectrum of knowledge management challenges. On the one hand, the abundant methodology theories provide us with a set of useful heuristic rules, from which we get the overview of ontology building process. But on the other hand, every research groups would like to justify their theories by listing their specific characteristics and unique method when trying to get on the right track. However, there is still no one "correct" way or methodology for developing ontologies. In this case, the methods used to evaluate only a subset of specific domain do not make any sense to the commonsense users.

Through the investigation of existing ontology methodologies, we get the conclusion that most of them have in common that they start from the identification of the purpose of ontology and the need for domain knowledge acquisition, differ in their following steps to be taken. Each has its own characteristics but evaluates only a subset of the specific

domain. the hope of getting direct guideline from the just beginning, focus the ontology structure design in details, which cover from the ontology vocabulary selection to ontology evaluation, a requirement of integrating the whole process in knowledge management systems. As a result, an effective methodology of domain ontology is imperative and still in great demand.

1. Introduction

Ontologies are a core element in the knowledge management architecture, its explicit specification of the terms in the domain and relations among them has been widely used for a computer understandable architecture—enable the people and the computer working together to achieve a more meaningful World-Wide Web.

Sharing common understanding of the structure of information among people or software agents is one of the driving forces for the development of ontologies. Often ontology is not a goal in itself, it defined a common vocabulary for researchers who need to share information in a domain, allows a common understanding which will contribute a lot to the reuse of certain knowledge.

The recent surge on ontology research, ranges from large taxonomy of categorized Web sites (such as on Yahoo!) to the classifications of individuals and their features in a certain domain (such as Wine ontology, food ontology).at the same time, ontology web languages and web applications for ontology standardization have been promoted and complemented in the last few years.

Many methodologies which guide the building process of ontologies have been proposed independently by different research groups. However, Most of them have in common that they start from the identification of the purpose of ontology and the need for domain knowledge acquisition, differ in their following steps to be taken. Each has its own characteristics but evaluates only a subset of the specific domain.

Ontology design is a creative process, not only for the rules: “There is no single correct ontology for any domain”[1], but also for no two ontologies by different people would be the same. The potential methodology for guiding the building process and the designer’s understanding and view of the domain will lead the design road and change the result. To this extent, the best solution for the variable

alternatives when modeling a domain always depends on the application you use and the outcomes that the end users are expecting to get from the knowledge base.

Concerning different methodology-based ontology development, there is a great need for combining ontology development with capabilities for collaboration and getting a comprehensive sight of the entire work. In this paper, we put forward the investigation of mainstream ontology methodologies, and use our methodology to evaluate ontology building process. A case study on Local Museum Collections Ontology is provided to specify each building stage. We hope to help knowledge workers form a comprehensive sight of the entire process for specific domain.

2. Related Works

Due to the fact that ontology engineering is still a relatively immature field, each research group suggests its own methodology to guide the building process of ontologies, each has its own characteristic and supported by certain applications.

“What type of methodology is uses to develop ontology?”
”How can we avoid the chaos of choosing a methodology among so much candidates?”
“Why we need a methodology at all?”

We would like to emphasize the necessary of using a methodology by first look at some puzzles almost everyone might face up to. To get an answer, one might suggest you to look up the paper: Ontology Development 101— the first standardized document with listed rules and descriptions. You might look it as a methodology and immediately doing your work according to the procedures as described. (In fact, it would better be classified as a set of useful heuristics rather than as a methodology. But in any case, it’s a valuable resource especially for beginners). The theories in this paper enlighten on the importance of methodology in ontology field, make is easier for the reader to get on the right way to the iterative process.

In the following, we will give a brief overview of the mainstream methodologies.

2.1 On-To-Knowledge (OTK) Methodology [2]

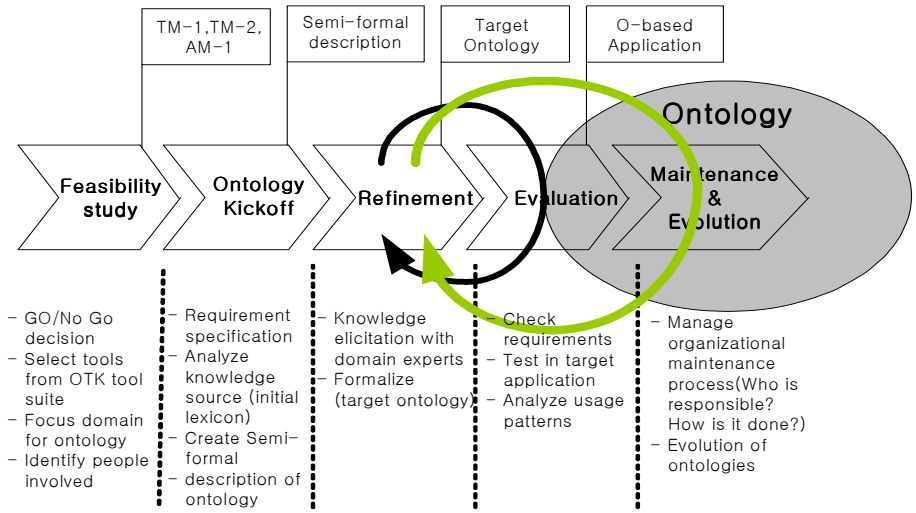


Figure 1. Steps of the on-to-knowledge methodology

It describes a methodology for application driven ontology development, covering the whole project lifecycle from the kick off phase to the maintenance phase. They put ontology development into a wider organizational context by performing a priori feasibility study, which is based on CommonKADS (cf.Schreiber et al., 1999). They modified certain aspect of CommonKADS for a tight integration of the feasibility study into their methodology.

The feasibility study they adopted offers an analysis of the users and use cases. Carry out a scoping and problem analysis study, consisting of two parts:

a. identifying problem/opportunity areas and potential solutions, and putting them into a wider organizational perspective;

b. deciding about economic, technical and project feasibility, in order to select the most promising focus area and target solution.

After these modified CommonKADS steps, the domain scope which goes through the feasibility study is supposed to be falling into 3 aspects of functions dealing with three levels of tasks, lead to the modified result as indicated in the dark shading in Figure 2.

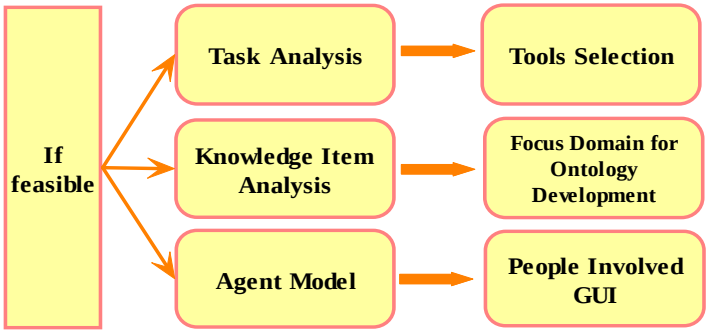


Figure 2. Feasibility Study Results

The results as described above serve as input for the kick off phase, and begin the first stage of ontology development.

The methodology is initially designed for knowledge management system, and the feasibility study mainly serves as decision support for economical, technical and project feasibility factors. As a result, it is too complicate for single knowledge workers to take account every aspect of their methodology into consideration when the problem field is onlydeal with a subset of specific domain.

2.2 Skeletal Methodology [3]

This methodology is based on the experience of building the enterprise ontology (Ushold and King, 1995). The following guidelines for developing ontologies are proposed:

1. Identify purpose. Clarify goal and intended usage of the ontology.
2. Building the ontology, which is broken down into three steps:

a. Ontology captures. Identify key concepts and relationships.
b. Coding. Represent the knowledge acquired in 2(a) in the formal language.
c. Integrate existing ontologies.

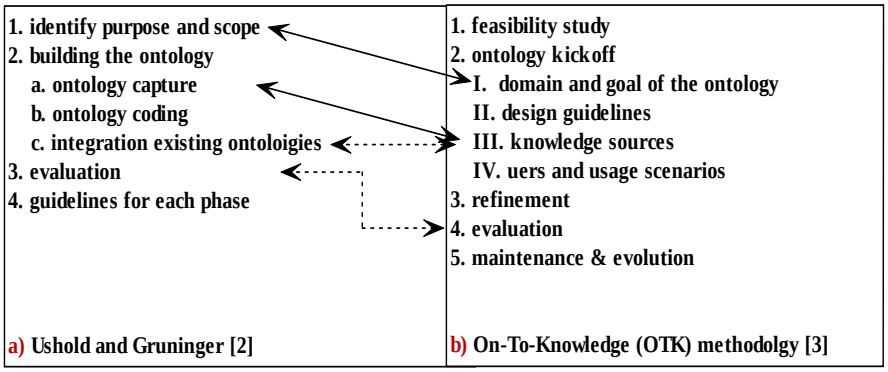
3. Evaluation. Make a judgment of the ontologies with respect to a frame of reference, which may be require specifications or competency questions.

4. Documentation. Document ontologies according to the type and purpose.

They catch up with the idea of competency questions and expand their usage not only for evaluation, but also for formulate the lexical entries for concepts and relations, etc. A disadvantage of this methodology is that it does not precisely describe the techniques for performing different activities. For example, it remains unclear how the key concepts and relationships should be acquired; only a very vague guideline, involving the use of brainstorming techniques, is given. A lifecycle is not recommended. There are no guidelines about the maintenance of evolving ontologies.

2.3 Methodology Comparability

Table 1. Relationships between phases of two methodologies



We listed phases of two current prevailing methodologies, and compared some of the phases, which actually describe the same functionalities. Meanwhile, the variable alternatives in each process of these two methodologies verified the chaotic situation to be the most terrifying obstacle to ontology engineers.

3. Towards a New Methodology for Building Domain Ontology

3.1 Current Methodology Drawbacks

1. There is not a "perfect" way for developing ontologies.
Compare with existing methodologies, each has its own characteristic while remains some problem fields and a wide-open research area uncovered. Some groups emphasize on the feasibility study before entering the ontology life cycle, involving giving out the target solution and narrowing the promising focus area, on the other hand, lacking the support for the rest design of an integrate ontology, the maintenance and evolutionary aspects of ontologies; some argues that under what circumstance the detailed competency questions should be used, is it too specific to guide the early development of ontology? Or Can it serves as a useful technique in the later stage of evaluating ontology coding?

2. The key factors for developing a methodology.
Concerning with different methodologies. Focuses on the main steps for ontology development falls into variable alternatives, which depends on the application that you prefer in mind and the extensions you anticipate. Although ontology development is an iterative process, however, we must face the challenge to integrate the whole life-cycle of ontology process from focus the domain to the maintenance and evolution stage.

3. Collaborative work is necessary.
Ontology design is a collaborative work. Typically, ontology engineers and domain experts are joined in a team, working together on description of the domain and the goal of the ontology. These collaborative aspects will also occur during each of the ontology design step.

Since it is necessary to come forwards a shared understanding within a group of people, and the achieved consensus will be reflected during the whole life cycle of ontology development, the support for these collaborative works within a methodology, consisting of both potential users and use case's contribution and supported applications, is crucial with respect to the construction of complex ontology structures.

3.2 Preliminary Study

The definition of our proposed methodology is analogous to methods of Knowledge Based System. It does not begin from scratch, but a refinement cycle, adding the new aspects and perspectives of the systems and integrating the successful ingredients of previous methodologies.

We focus the preliminary study on two aspects:

Step1 : Conceptualization.[9] Elicit task to obtain original description of problem field and determine user cases.

The aim of Step1 is to have an overview of the supposed domain; put together all the useful information; take all factors into consideration, and then, a semi formal description of the domain is created, which is including The outcome of this step should be evaluated by the following step.

Step2 : Analysis. Three aspects of analysis would be used to justify the probability of the semi formal description: Task Analysis, Knowledge Domain Analysis, Agent Analysis, the result will be the requirements specification of the intended domain field.

Supposedly, these analyses work should be handled by knowledge provider, ontology engineer and knowledge worker. Ontology engineers provide the structure for the ontology, Initially they are responsible for the ontology development, but also for handling the maintenance afterwards. Knowledge providers are responsible for the content provision. Typically the main sources of that knowledge stem from intranet pages, documents in electronic formats and databases. Before they are filled into knowledge base, knowledge providers should define the semantic interlinkage between these

sources and knowledge base. This is called annotation. Web pages and documents have to be annotated to explicitly represent the semantics of their content. The annotations might be stored within the knowledge base or externally. The management to annotations plays an important role in knowledge management systems.[5]

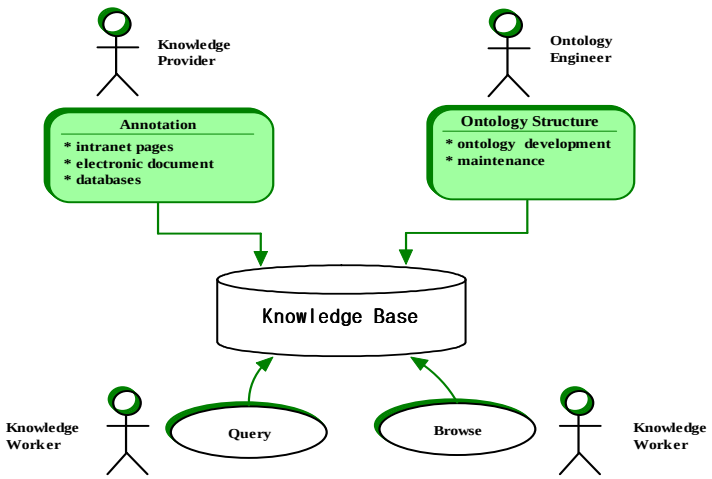


Figure 3. User and use cases management diagram

The user and use cases study has been introduced in most of the application driven methodologies, especially in the earliest stages of ontology development. Here we put forward using the diagram above in order to specify their roles during the analyses. As described in the diagram, knowledge workers play as agents and users of the knowledge base, they push knowledge sharing, navigate and browse a knowledge base, query a knowledge base and seek knowledge.

The method of user and use cases modeling determines the requirements of ontology, leave the results to be filled into three aspects of systematical analysis: task analysis, knowledge domain analysis and agent analysis.

The CommonKADS methodology [4] which serves for the development of Knowledge based system offers three models have the same function as described above, here only the extensions to CommonKADS are given. Since our work is only to provide a comprehensive view of domain ontology,

integrate every aspect into the organization is optional. Here we continue Step2 by specifying each analysis stage.

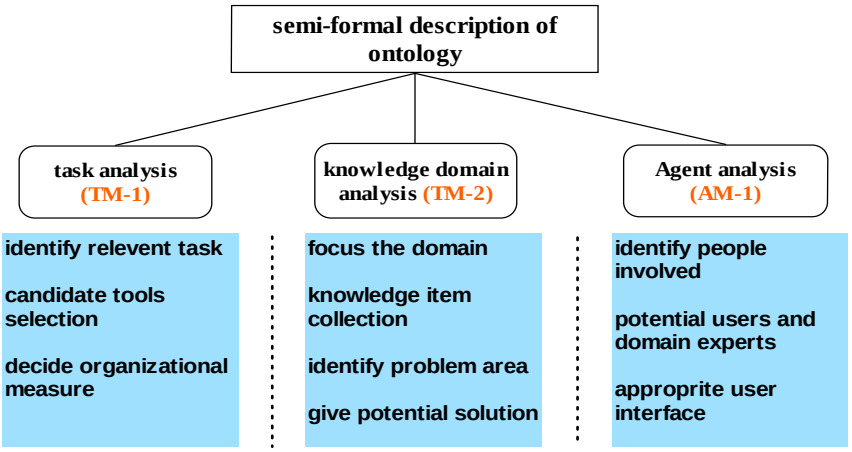


Figure 4. Modified CommonKADS steps

According to CommonKADS, task assignment should be performed in a fixed way, however, we take good use of the result from conceptualization stage to carry out a flexible task analysis, in the meanwhile, remains the restrict forms of the task model; we also enjoyed accepting the agent model as an executor of a task as well as integrating other two CommonKADS models, that are Communication model and Expertise model. Because of the particular role of human agent—users and potential domain experts, ontology engineers are facilitated to get the exchange of information between different agents, and then have them as input resource for sake of the decision making of expertise evaluation. In this way, knowledge workers begin to evaluate the final knowledge base, implement the requirement, and report to ontology engineers for ontology maintenance.

After specifying our delicate work, we would like to discuss our case study on Local Museum Collections Ontology. Suppose a local museum wants to classify and rebuild the catalogs in a more convenient way, they adopt ontology to facilitate local users when they are querying and browsing

the museum collections information on its web site. So the task has been distributed, but who will be responsible for this job? In what aspect will the chosen domain suppose to facilitate specific end users? Here we give out a primitive description, which referred to as a model for this specific domain.

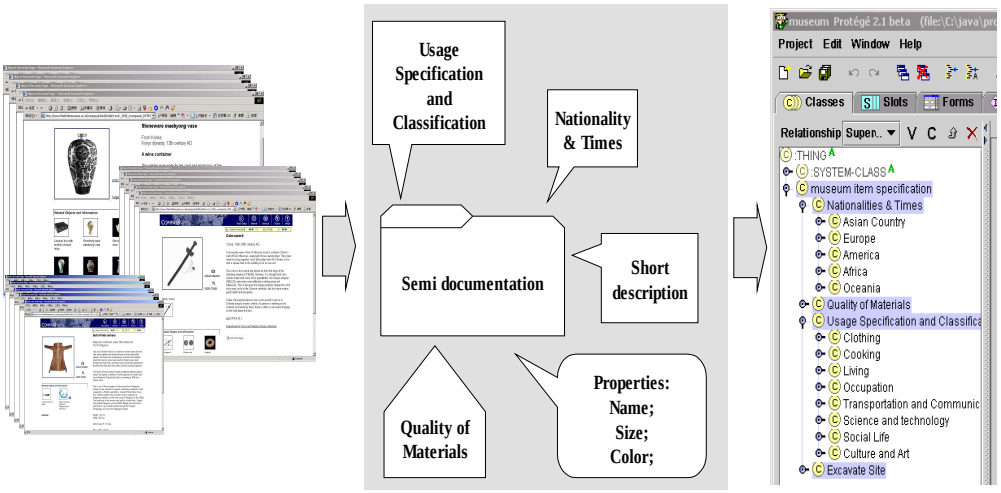


Figure 5. proposed methodology model in the case study

The Figure 5 shows an initial information process model, through this case study model, we would like to explain the use of Step1 *Conceptualization,* and put forward the analyses stage smoothly and sufficiently. The semi documentation in Figure 5 is something like extracting the necessary information from media, and collecting a first description of the ontology domain, the annotations in the electronic format and original information in the web site database contains most of structure one ontology may need. Other works to fulfill the specific usage of the intended ontology is left to Step 2 Analysis to take responsibility for an overall analysis before ontology engineers begin to think about the hierarchy of ontologies like described in the right part of Figure 5. [10,11]

Following the analysis study, a formalized description of domain knowledge will be send into the next step, and begin with ontology development.

3.3 Architecture Design

With the semi-formal description of ontology, ontology engineers are about to provider a structure, which is suitable for both the developing process and the maintenance afterwards. It is impossible to cover every aspect of issues that an ontology engineer may need to grapple with and we are not trying to address all of them. Here, the general architecture design is subdivided into two levels.

1) Knowledge level: Several design decisions should be taken with respect to the management of ontologies:

a) Domain of the ontology. The ontology engineer may use the outcomes of the task analysis to describe the domain of the ontology, specify the particular domain in use.

b) Guidelines for Design. We must make clear how the key concepts and relationships should be acquired, it should not only involves brainstorming techniques, but also contains capture vocabularies from competency questions, reuse of existing ontology to collect a first hand script of class hierarchies. Also it must follow certain rules in certain situations. Such as deciding whether a particular concept is a class or an individual instance.

For example, consider the museum collection birthplace. We may define main birthplace regions, such as China, South Korea, Japan, Europe, South America, and so on. Suppose *Jilin Province* is an instance of Chinese collection birthplace region, However, *ChangBai Mountain* is a *Jilin Province* region, therefore, *Jilin Province* must be a class in order to have subclass or instances. It is very hard to distinguish which region are classes, and which are instances. Therefore, we define all collection birthplace regions as classes.

c) Define the classes and the class hierarchy.

1. *Top-down or Bottom-up Approaches.*

Here we argue that combining both top-down and bottom-up approaches.

for constructing the Local Museum Collections Ontology hierarchy, we

might start with a few top-level concepts such as *Collections*, *Regions, Quality of Materials.* and a few specific concepts, such as *gold* as subclass of Metal under the super class of *Quality of Materials.* Through the reuse of original classification in the old generation knowledge base of the museum database, we can get a systematic top-down view of the domain, then it may be easier to use the top-down approach. However, if the knowledge worker prefer to get informed by the specific examples, we 'd rather start by getting grounded with the bottom-up approach.

Neither top-down or bottom-up approach is inherently better than any of the others. the approach to take depends on the personal view of the domain. whichever approach we choose, we select the terms that describe objects having independent existence rather than terms that describe these objects. These terms will be classes in the ontology and will become anchors in the class hierarchy.

2. ***Object-Oriented Design for Class hierarchies.***

Although some ontology-design ideas originated from the literature on object-oriented design, and ontology classes are more likely to be associated with ***Classes*** in a object-oriented programming languages at first sight, ontology development is different from designing classes and relations in object-oriented programming. A class structure and relations among classes in an ontology are different from the structure for a similar domain in an object-oriented program, a programmer makes design decisions based on the operational properties of a class, whereas ontology designer makes decisions based on the structural properties of a class. However, we noticed that the understanding of classes and their properties in the object-oriented programming does make sense to elementary level ontology learners. In this case, we have listed out some typical class concepts and relations.

1) **Transitivity of the hierarchical relations**

A subclass relationship is transitive: If B is a subclass of A and C is a subclass of B, then C is a subclass of A. we call class B is the direct subclass of class A, and class c is an indirect subclass of class A because there is a class B between class A and class C.

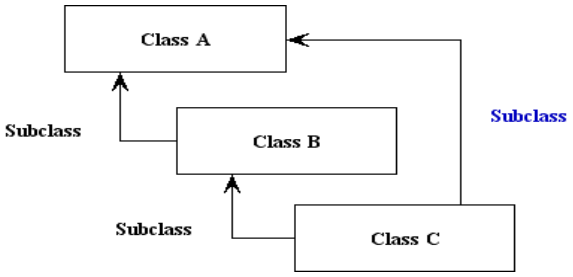


Figure 6. Transitivity of class hierarchy

For example, we define a class *Items*, and then define a class *clothing* as a subclass of *Items*, Then we define a class *Hat* as a subclass of *clothing.* Transitivity of the subclass relationship means that the class *Hat* is also a subclass of *Items*. In this case, *Hat* is a direct subclass of *clothing* and is a indirect subclass of *Items*.

2) **Inheritance and Multiple inheritance.**

In the object-oriented programming, the subclasses can use variables in their parents' class directly, and inherit methods from parent classes, the programming centers primarily around the methods on classes, and programmers make use of properties of classes and relations between subclasses to make decisions. In ontology design, inheritance is also plays a very importance role. subclasses-either direct or indirect-can inherit slots (properties) from their parent classes. after you have defined classes and subclasses, we must describe the internal structure of concepts. the terms which are likely to be properties of these classes are then attached at the most general class that can have that property, so the slot to classes have been created, and all subclasses of a class can **inherit** the slots of that class.

For example, all the slots of the class *Items*, will be inherited to all subclasses of *Items,* including *clothing, living, cooking and so on. we* add an additional slot, *belongings* (male, Female), to the *clothing* class. the *belongings* slot will be inherited by all the classes representing clothing items. Figure 8 shows the snapshot of inheritance in class items and its subclasses, such as clothing, cooking, occupation and transportation. the slot attached to class *Items* have been inherited by its subclasses, so in *Living*

subclasses of *Items* class, the slot Name, Description, Size, Shape, MadeOf can be seen listed in the Template Slots frame.

Slot can have different facets describing the value type, allowed values, the number of the values (cardinality), and other features of the values the slot can take. we could easily understand these slot types by associating with different types of variables that predefined in parent classes or subclasses in object-oriented programming language.

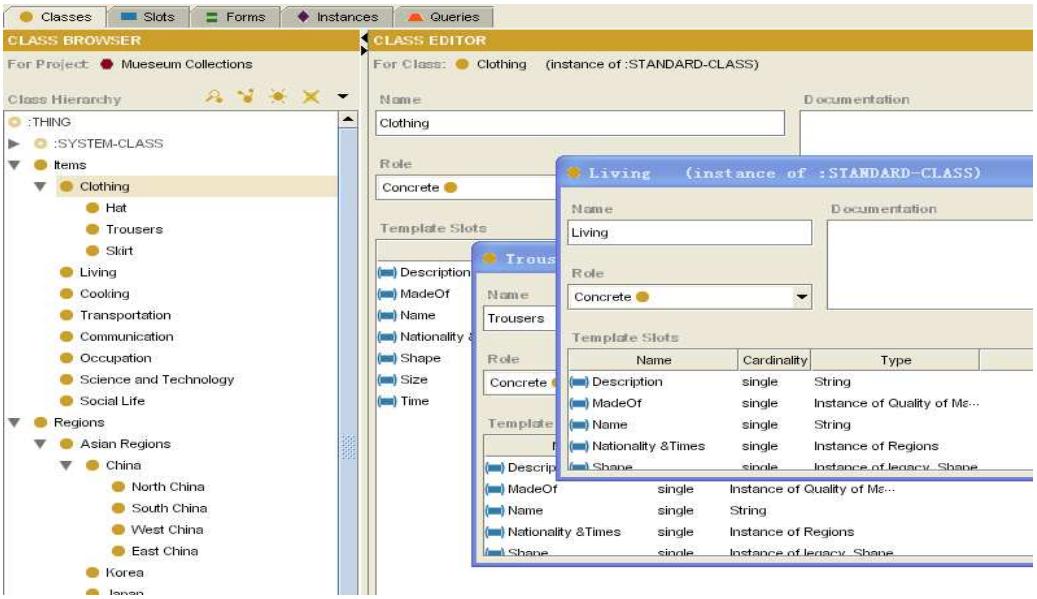


Figure 7. Protégé Snapshot of Slot Inheritance

Another facet of inheritance is allow **multiple inheritance** in the class hierarchy: a class can be a subclass of several classes. Suppose we would like to create a separate class of *Artificial Collections* class, under the direct subclass of *non-existing Items* , we define a class *Items* to represent entities that have been destroyed and rebuilded according to the writing records. Therefore,

3) **Siblings in a class hierarchy and Class cycles**

Classes that are direct subclasses of the same class in called siblings in the hierarchy, they must represent the same level of generality. However,

the concepts at the root of the hierarchy, which are often represented as direct subclasses of some very general class, such as Thing, represent major divisions of the domain and do not have to be similar concepts. in our case study, the root concept is divided into 6 classes: *Items, Nationalities, Times & Periods, Qualities of Materials, Regions, Properties.* in this level, none of these concepts represents the same similarity, but the subclasses of these root classes must be classified carefully, within each class hierarchy, only the same level concepts are included. For example, the direct subclasses of Regions is divided into *Asia Region, Europe Region, American Region, Other Regions,* and *Asia Region* is subdivided into China, Korea, Japan, India, Other Asian Regions.

Another important rule for designing class hierarchy is **avoid class cycles**, that is when a class A has a subclass B, and at the same time B is a superclass of A. Creating such a cycle in a hierarchy amounts to declaring that the classes A and B are equivalent: all instances of A are instances of B and all instances of B are also instances of A. the situation here is the same in object-oriented programming, if one class is declared to be a superclass of another class, it can not be declared to be the subclass of that one later, for the variables of subclass maybe re-declared and the types of variable and parameters can be changed after inherited from its parent class, in this way, it will be very dangerous to redirect the subclass as the superclass with respect to system compile errors.

d) Supported Application. The domain expert may take advantage of outcomes from the candidate tools study, which is part of the TM-1, to get a clear picture about which applications fit the proposed domain and give the proper interface to the potential users. These applications not only take charge of the ontology structure building, but also be able to deal with other pieces of processing in the ontology life cycle, such as evaluation, reasoning, provide an inferencing support, and so on. Therefore, the toolkits you choose does affect your ontology building process and consequently essential to the quality of your ontology.

Environments like **Protégé** [12] (Fridman et al., 2000) or Chimaera (McGuinnesset al., 2000) offer sophisticated support for ontology

engineering and merging of ontologies. **Protégé** also has a modular Dplug-in design rational like OntoEdit, but lacks sophisticated support for collaborative engineering. They provide limited methodological and collaborative support for ontology engineering.

Some ontology design toolkits, like OntoEdit, serves as an ontology-engineering environment by combining methodology-based ontology development with capabilities for collaboration and a tight integration into a tool framework. especially on the integration and evolution of ontologies.

The web-based Ontosaurus (Swartout et al., 1996) combines support for collaboration with reasoning and allows individuals to expand an ontology only when consistency is retained within the ontology as a whole. This approach takes advantage of the reasoning and consistency checking abilities of the underlying representation language LOOM. Ontosaurus was inspired by the Ontolingua system (Farquhar et al., 1996), which does not have an inferencing support as an integral part of the ontology development environment. Due to the simple 'state-less' HL interaction, both systems have several limitations. For example, a server does not maintain any state information about users, that is, clients. Nor is it possible for a server to initiate an interaction on its own, for example, alerting users to simultaneous changes by others. In general, no other approach is known to us that implements fine-granular locking of ontologies like we do.

2) Coordination level [10]: On this level, we will focus on the precious agent analysis (AM-1) in order to clarify the user and user case study.

Lists of potential users and description of each usage scenario should be reported from the real experience: In what situation did they wish such an ontology? How did they proceed without it? Each individual user gets a point of view of his scenario, which will finally be reflected on the building process of ontology.

As for human agents, who are likely to be potential domain experts and

might be a valuable resource at the stage of refinement phase of ontology development. Human agents might also be users of the knowledge base, and therefore might indicate a need for appropriate user interface.

The result that covered our case study in this architecture design integrating the semi document of **Step 1** with all the analysis result and form a formalized domain ontology specification.

Table 2. Domain Ontology Requirements specification

Domain Ontology requirements specification
Domain: Local Museum Collections
Made-by: Haitao Liu
Purpose: Ontology about local museum collections to be used as a deposited metadata to facilitate certain searches on the museum web site, provide users with meaningful result, share the information, e.g.
Ontology Scope: a list of 209 elements, containing 6 direct subclasses: *Items, Properties, Materials, Times & Periods, Artificial Collections, Regions. slot to class elements, such as name, MadeOf, Possession-by, has-color, excavate-year, special-symbols, Excavate Site.*
Source of Knowledge reused: *the current hierarchical catalogues used by South Korea Central Museum.*
Edit Application: *Protege Ontology Editor*

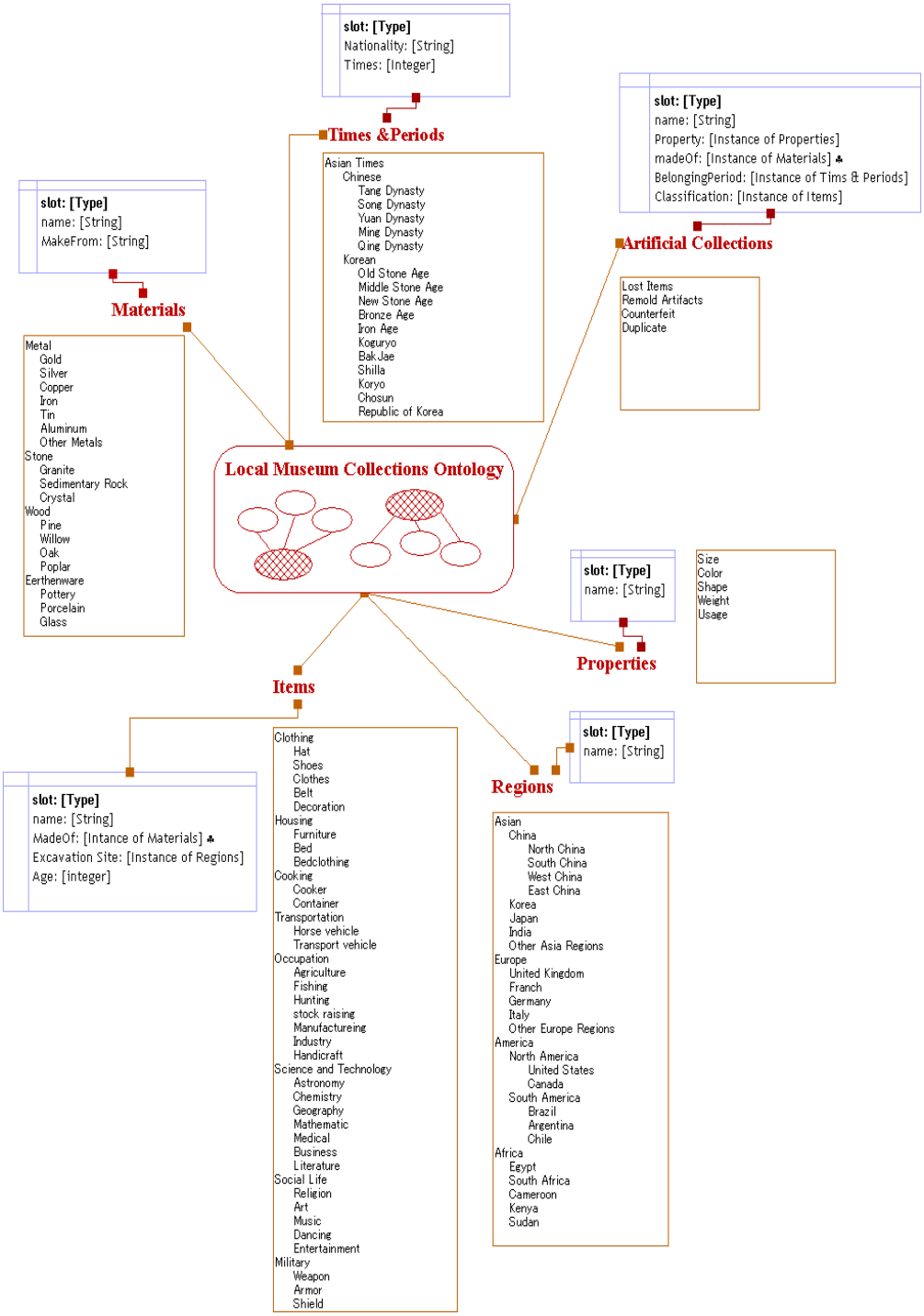


Figure 9. Local Museum Collections Ontology Structure

4. Evaluation and Refinements

Evaluation means to carry out a technical judgment of the ontologies: Check out the gaps among existing class elements, detect misconceptions and gather user's satisfaction, the outcome of these efforts might be necessary to help the knowledge engineer to correct his previous work, and begin the update/insert/delete circle.
An ontology, is composed of concepts and relations, some of which are explicitly defined, others which follow from a set of axioms.

4.1. Ontology Mapping

According to the case study, some of the instances can be presented to justify our methodology. Suppose there are some new items should be added to the knowledge base, a matching between the fundamental information and ontology is described as the following:

Table 3. Matching with Ontology metadata

Ontology Slot / *Item Description*	*Slot Name*	*Type*
Name	**Name**	**String**
size	**Size**	**Symbol**={small, medium, large}
shape	**Shape**	**Symbol**={round, ellipse,…,}
Earthenware	**MadeOf**	Instance of **Quality of Materials**
Belongs to Tang Dynasty	**Times**	Instance of **Time & Period**
Excavation Site: China, ShangDong Province	**Nationality & Times**	Instance of **Country**
Short Description	**Description**	**String**

Through this table, we classified original item into the ontology scope

by identifying typical description, and then matching with ontology slot. This process seems easy to achieve due to varieties of ontology edit tools, however, the satisfactory of the proposed ontology structure to fulfill the user requirement is depending on the original structure set by ontology engineer. So the instance here plays a role in evaluating the feasibility of ontology structure.

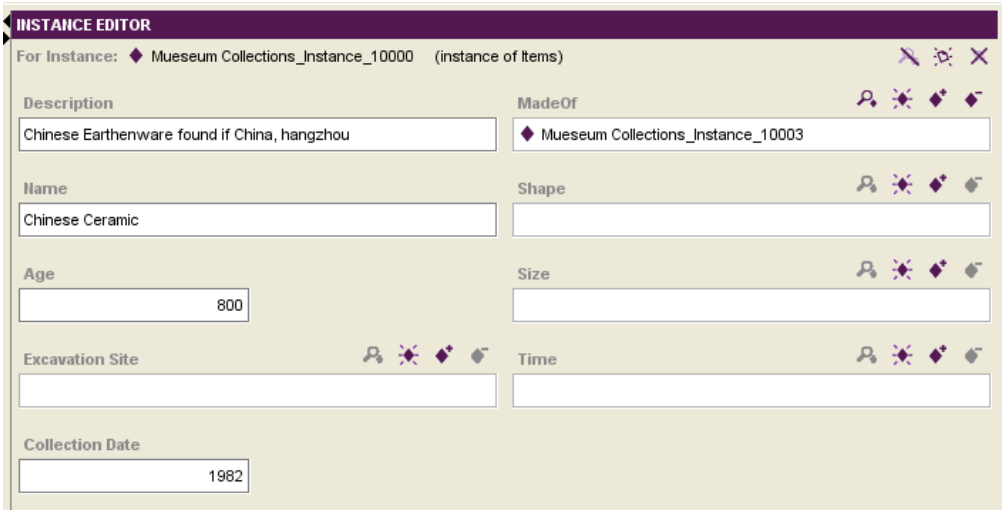


Figure 9 Edit Sample Instance with Protege Instance Editor

4.2 Evaluation of Granularity and Competence

Granularity – evaluation of granularity refers to whether or not the upper-level of the ontology reflects the identification of major concepts from the domain. The ontology uses hierarchies to describe the domain, using abstraction and avoiding multiple inheritances. Thus it further supports the claim for conceptual coverage.

Competence – the ontology responds to the question of how to develop an OM, what should be in it, how it should operate and by whom. By doing this, the ontology reaches its goal. Preliminary findings from the case study support this claim. The analysis of the case study is done with relation to the ontology as previously described.

It is impossible to automatically evaluate directly the fit between a knowledge artefact such as an ontology and a person's knowledge of a domain, let alone the knowledge of a group. One standard approach would be to compare a new ontology with an existing 'gold standard' one. Such has been the approach espoused by authors such as (Grefenstette, 1994). The problem here is that if the results differ from the gold standard, it is hard to determine whether that is because the corpus is inappropriate, the methodology is flawed or there is a real difference in the knowledge present in the corpus and the gold standard. In any case, this approach is more applicable when one is trying to evaluate ontology *learning methodologies.* In the Semantic Web scenario, it is likely that one has to choose from a range of existing ontologies the most appropriate for a particular domain, or the most appropriate to adapt to the specific needs of the domain/application.

5. Maintenance and Evolution

It is most important to clarify who is responsible for the maintenance and how it is performed. Is it a single person or a consortium responsible for the maintenance process? There has to be strict ways to the update/insert/delete process of ontologies.

We should separate the maturity of the ontology from that of its associated toolset. For the ontology, maturity is partly the degree in which it meets its stated development goals and the quality with which it does so We could also consider the completeness of the ontology with respect to the target domain. Needless to say, drawing that boundary is non-trivial, and for an upper ontology the lower bound is somewhat arbitrary. It is probably obvious to this group but I'll say it anyway. Size is a not terribly relevant indicator of "goodness"; better to think in terms of completeness for a purpose and quality. Longevity is a heuristic indicator only an ontology that continuously develops over a long period can improve or can become as much of an accessary mess as a long-lived program can. A final dimension of maturity might be the degree to which the ontology recognizes and handles issues within its scope. An ontology of process had better have a way to deal with the flow of time and probably needs to be able to represent situations that have changed without running into contradiction.

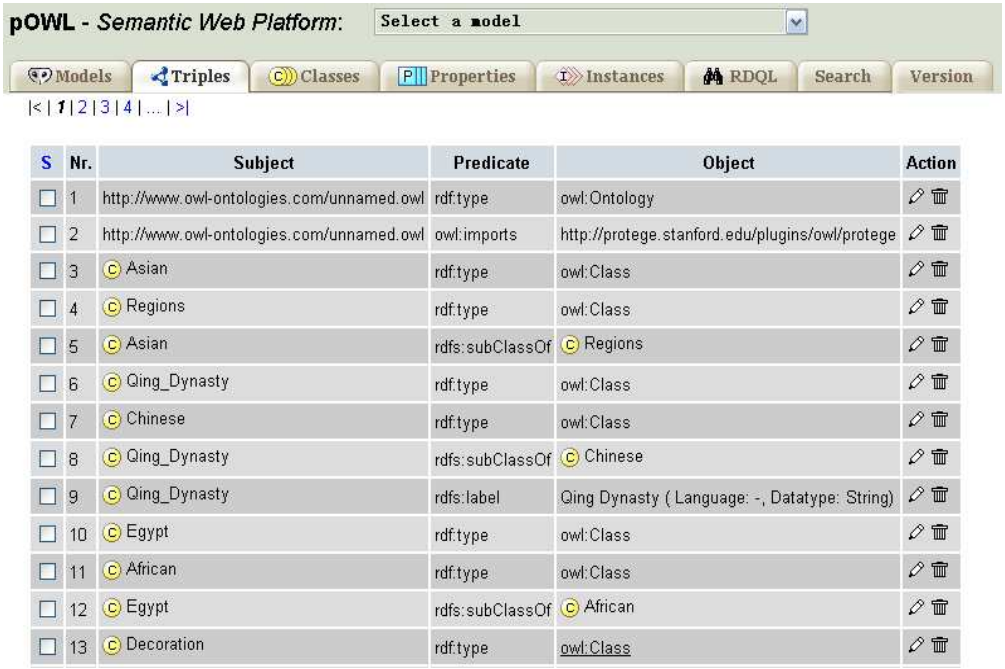
6. Application-Driven Methodology Evaluation

The next step is putting forward an application-driven methodology evaluation by adopting an Open Source community software - pOWL Semantic Web Development Platform. Although there is a bunch of OWL ontology editing and management solutions available, some of them are complicated to deploy or handle, some do not support strategies for collaborative, distributed development of ontologies, some are not Open Source or not available for the most distributed web technologies.

pOWL supports viewing, editing of RDFS/OWL ontologies of arbitrary size, the semantic web paradigm will probably only be successful in a broad perspective if there are applications and tools available tightly interacting with this language. The aim of the pOWL project is thus to deliver a PHP and web-based ontology editing and management solution to the Open Soure community.

The methodology evaluation is organized as a three-steps method for managing the proposed domain ontology. Firstly, export the ontology structure as an owl file from the protege, save as owl file, and then import to pOWL as a new model, which defined as a localhost URI. Secondly, in the platform interface of pOWL, the imported model has been identified and represented separately , we take advantage of the identified result to focus our evaluation on four aspects, that is , **triples, classes, properties** and **instances**. each of which has a overview of the existing hierarchies themselves, by viewing, editing and managing the hierarchies, we can do the evaluation work according to each part of ontology usages.

Triples are the RDF-Triples [13] composed of Subject, Predicate and Object. the syntax of the ontology class hierarchies is listed in details, ready to edit and evaluate. through the figure below, we can see the rdf:type predicate, which indicate the dependent classes that we call - **direct subclasses** of the museum ontology; the rdfs:subClassOf predicate is simply specified the subclasses which belongs to the object list.



S	Nr.	Subject	Predicate	Object	Action
☐	1	http://www.owl-ontologies.com/unnamed.owl	rdf:type	owl:Ontology	
☐	2	http://www.owl-ontologies.com/unnamed.owl	owl:imports	http://protege.stanford.edu/plugins/owl/protege	
☐	3	Asian	rdf:type	owl:Class	
☐	4	Regions	rdf:type	owl:Class	
☐	5	Asian	rdfs:subClassOf	Regions	
☐	6	Qing_Dynasty	rdf:type	owl:Class	
☐	7	Chinese	rdf:type	owl:Class	
☐	8	Qing_Dynasty	rdfs:subClassOf	Chinese	
☐	9	Qing_Dynasty	rdfs:label	Qing Dynasty (Language: -, Datatype: String)	
☐	10	Egypt	rdf:type	owl:Class	
☐	11	African	rdf:type	owl:Class	
☐	12	Egypt	rdfs:subClassOf	African	
☐	13	Decoration	rdf:type	owl:Class	

Figure 11 Classes hierarchies represented by RDF－Triples

The Properties and Instances management is another important factor to test the robustness of proposed ontology model. In this regard, the instance are added to test the properties compatability and comparability. Through this process, as has shown in figure 10, the property Possessed_By is added under the items domain, its values are set as a enumeration of two strings: **private** and **common**. this modified result reflected in Properties list is one new added value, but in the instance list, there will be a checkbox specifying the added instance is either belongs to private or common community. this added property also cause a rising interest of the knowledge engineer if is necessary to consider adding one direct subclass of **Possession Party** for specialized usage.

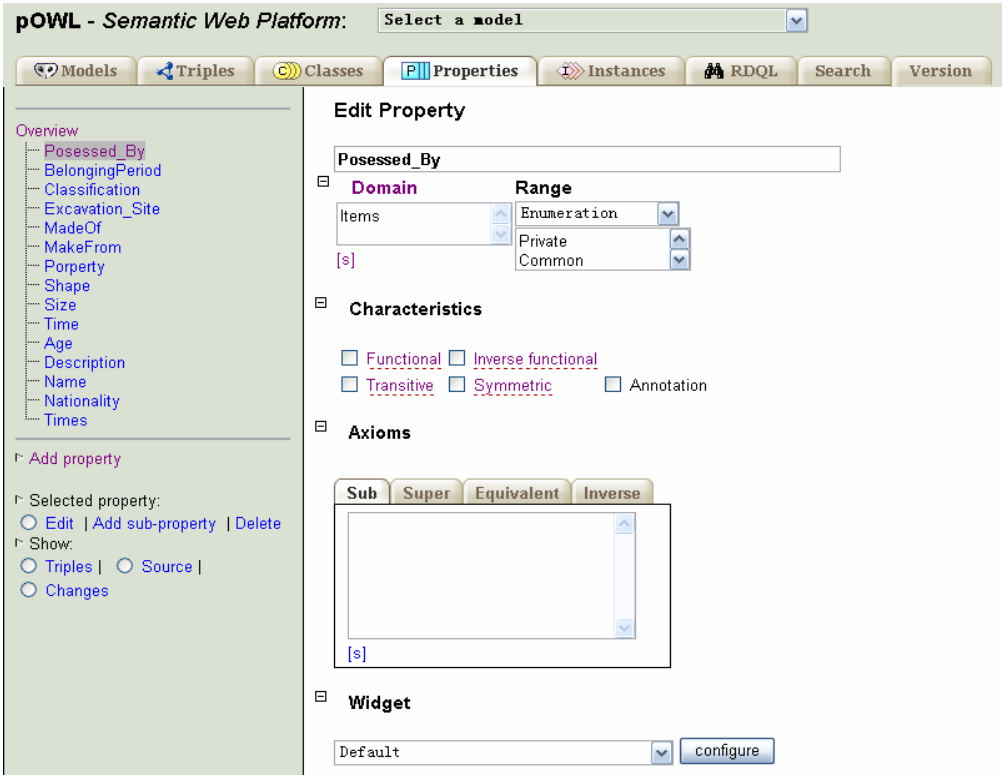


Figure 10. pOWL interface for editing properties

In fact, we can get a skeleton of the use scenario by looking the layout of properties in the new instance. the properties structures specified here is more likely to appear in the user interface to the people who browse and search in the ontology-implemented knowledge base. although it is impossible to make more than the widest guess at how many precent of ontology structure is still remains as it is shown now after the life-cycle of continuous ontology evolving, this prototype of information representation does give us a better impression of how to formulate and improve the evolution process.

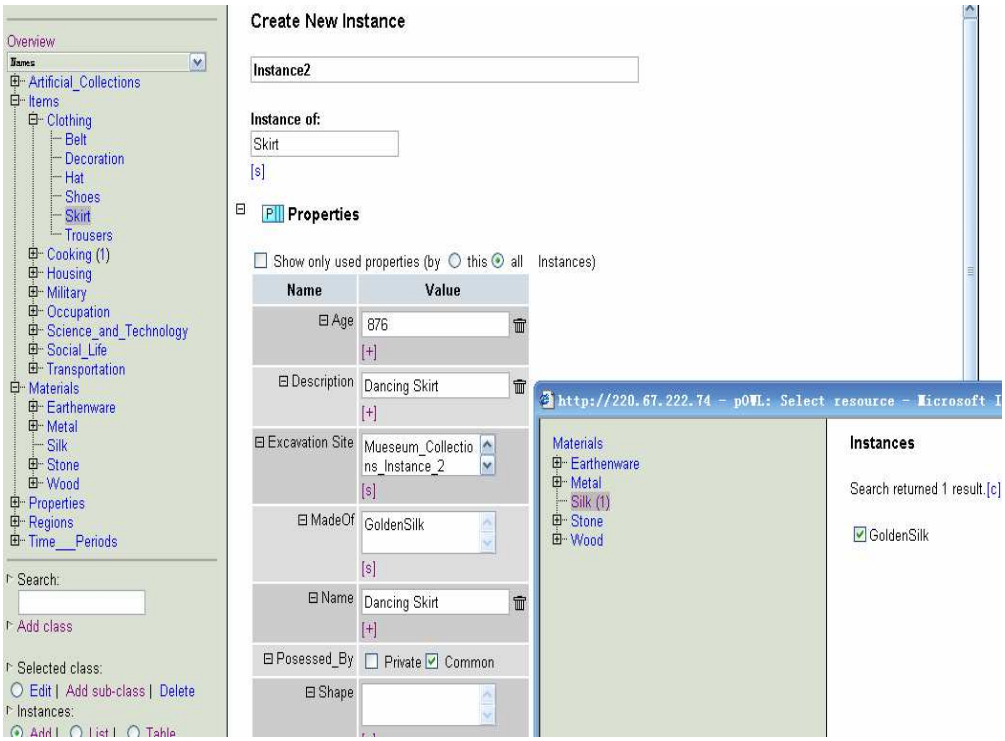


Figure 12. Instances editing and properties specification

After manipulating ontology model in the platform, we can validate and publish our model on the Internet, and left the reuse and evaluation in wider scale to be done by collaboration and standardization. during the validation, owl vocabulary is tested again to fulfill the requirement of syntax and integrity, the untyped object and data is implemented before publishing.

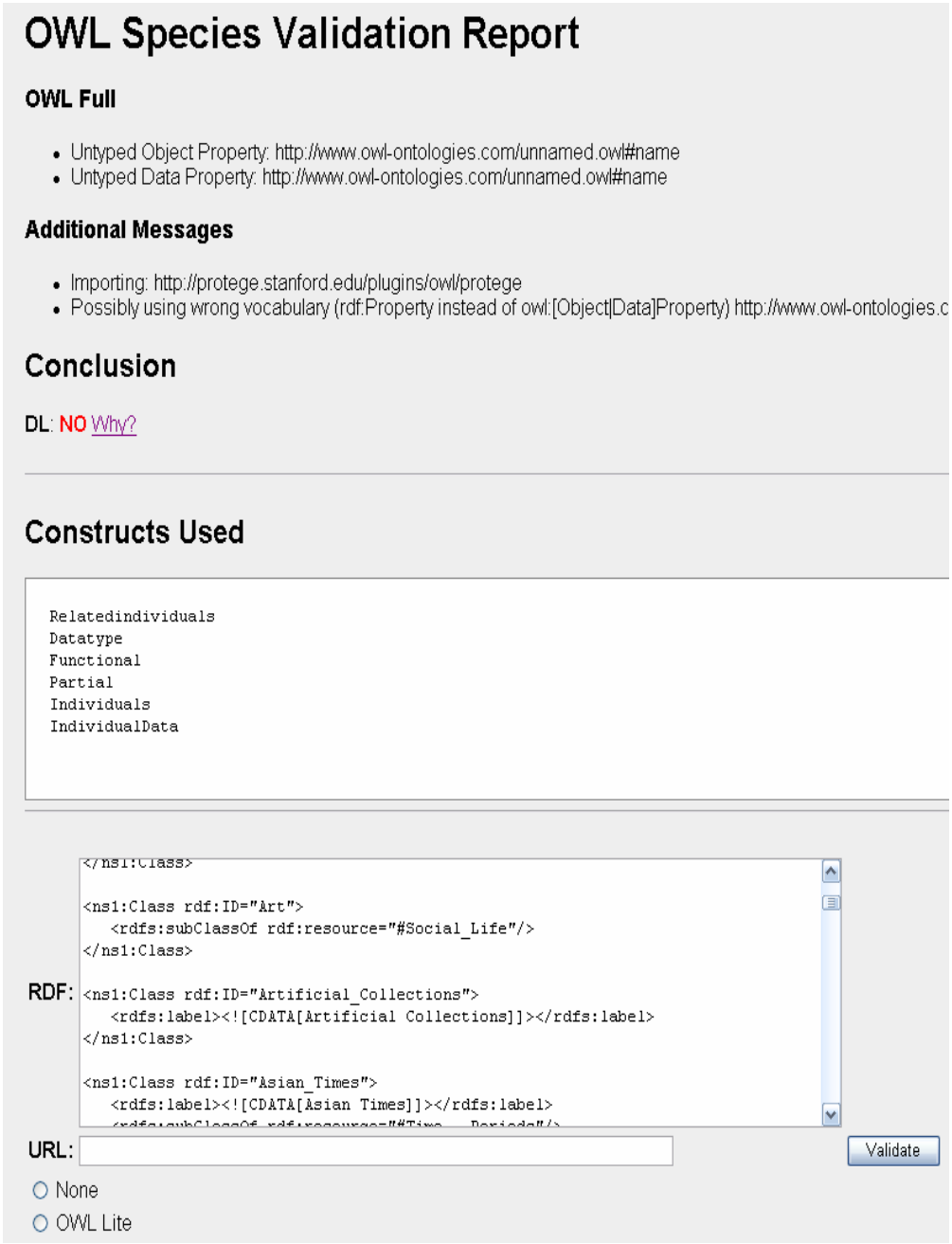


Figure 13. Online Validation Report of the modified ontology model

7. Conclusion

A comprehensive methodology has been outlined, and we have suggested some improvement derived from our delicate work on existing methodologies studies.

Through the investigation on existing ontology methodologies, we have been able to provide a comprehensive methodology for building specific domain ontology. In this paper, our methodology supported by a case study on Local Museum Collections Ontology. However, our methodology is not supposed to cover every aspect of domain ontology development, and it is impossible to provide every detail that should be addressed for ontology engineer. With respect to collaborative capabilities, we hope to share our understanding of ontology building process with companions, and in our future work, we would like to focus on more detailed evaluation process and maintenance steps.

As we have mentioned, some ontology design toolkits, like OntoEdit [10], serves as an ontology-engineering environment by combining methodology-based ontology development with capabilities for collaboration and a tight integration into a tool framework. Through our proposed methodology, we hope to find out a right way for our future research, especially on the integration and evolution of ontologies.

We make evaluations of plans for a number of different reasons. In one case we may draw together a set of related features of a plan in order to gain a picture of the plan's essential structure, the things that make it qualitatively different from another possible plan. For example we may look at a plan's use of resources and its assumption of risk. These features may either be deduced or inherent in the plan's definition. In our ontology, the evaluation structure groups together related analyses, each of which mentions some object property, such as fuel usage, and one or more estimates of a value. The analyses can have more than one estimate because the evaluation system may have more than one algorithm at its disposal for providing an estimate. For example, a system may begin with a gross estimate of fuel usage based

on heuristic, and then add a finer estimate based on a detailed simulation of the plan.

We hope to help knowledge workers form a comprehensive sight of the entire process for specific domain, in our future work, we would like to focus our study on ontology evaluation and more detailed maintenance work.

References

[1]. R. Dieng, O. Corby, A. Giboin, and M. Ribiere. "Methods and tools for corporate knowledge management". Int. Journal of Human-Computer Studies, 51(3): pages 567-598, 1999.

[2]. Natalya F. Noy and Deborah L. McGuinness. "Ontology Development 101: A Guide to Creating Your First Ontology". Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880, March 2001.

[3]. J. Davies, D. Fensel, and F. van Harmelen, editors. "Towards the Semantic Web Ontology-Driven Knowledge Management". Wiley, 2002.

[4]. G. Schreiber, H. Akkermans, A. Anjewierden, R. de Hoog, N. Shadbolt, W. Van de Velde, and B. Wielinga. "Knowledge Engineering and Management The CommonKADS Methodology". The MIT Press, Cambridge, Massachusetts. London, England, 1999.

[5]. D. O'Leary. "Using AI in knowledge management: Knowledge bases and ontologies". IEEE Intelligent Systems, 13(3): pages 34-39, May-June 1998.

[6]. Ushold and King, "Towards a Methodology for Building Ontologies". Workshop on Basic Ontological Issues in Knowledge Sharing, held in conjunction with IJCAI-95, Montreal, Canada, 1995.

[7] Y. Sure and R. Studer. On-To-Knowledge Methodology final version. On-To-Knowledge deliverable D-18, Institute AIFB, University of Karlsruhe, 2002.

[8] Stojanovic, L., Maedche, A., Motik, B., & Stojanovic, N. "User-driven ontology evolution management". In Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW), LNAI 2473, pages 285-300, Siguenza, Spain, 2002.

[9]. T. Gruber. "Towards principles for the design of ontologies used for knowledge sharing". International Journal of Human-Computer Studies, (43): pages 907-928, 1995.

[10]. Y. Sure, M. Erdmann, J. Angele, S. Staab, R. Studer, and D. Wenke. "OntoEdit: Collaborative ontology development for the semantic web". In

Proc. of the International Semantic Web Conference 2002 (ISWC 2002), June 9−12 2002, Sardinia, Italia, 2002.
[11]. Y. Sure, S. Staab, and J. Angele. OntoEdit: "Guiding ontology development by methodology and inferencing". In Proceedings of the International Conference Ontologies, Databases and Applications of Semantics ODBASE 2002, University of California, Irvine, USA, Springer, LNCS, 2002.
[12]. N. Fridman Noy, R. Fergerson, and M. Musen. "The knowledge model of Prot´eg´e−2000: Combining interoperability and flexibility". In Proceedings of EKAW 2000, LNCS 1937, pages 17−32, Springer, 2000.
[13]. Klyne, G., Carroll, J: "Resource Description Framework (RDF):Concepts and Abstract Syntax", W3C, 10 February 2004.

저작물 이용 허락서

학 과	전자계산학과	학 번	20037086	과 정	석사

성 명	한글: 유해도 한문 : 刘海涛 영문 : Haitao Liu
주 소	
연락처	E-MAIL : htliu@stmail.chosun.ac.kr
논문제목	한글 : **도메인 온톨로지 구축을 위한 효율적인 방법론 제안** 영문 : **Proposing an Effective Methodology for Developing Domain Ontology**

본인이 저작한 위의 저작물에 대하여 다음과 같은 조건아래 -조선대학교가 저작물을 이용할 수 있도록 허락하고 동의합니다.

- 다 음 -

1. 저작물의 DB구축 및 인터넷을 포함한 정보통신망에의 공개를 위한 저작물의 복제, 기억장치에의 저장, 전송 등을 허락함
2. 위의 목적을 위하여 필요한 범위 내에서의 편집형식상의 변경을 허락함. 다만, 저작물의 내용변경은 금지함.
3. 배포전송된 저작물의 영리적 목적을 위한 복제, 저장, 전송 등은 금지함.
4. 저작물에 대한 이용기간은 5년으로 하고, 기간종료 3개월 이내에 별도의 의사표시가 없을 경우에는 저작물의 이용기간을 계속 연장함.
5. 해당 저작물의 저작권을 타인에게 양도하거나 또는 출판을 허락을 하였을 경우에는 1개월 이내에 대학에 이를 통보함.
6. 조선대학교는 저작물의 이용허락 이후 해당 저작물로 인하여 발생하는 타인에 의한 권리 침해에 대하여 일체의 법적 책임을 지지 않음
7. 소속대학의 협정기관에 저작물의 제공 및 인터넷 등 정보통신망을 이용한 저작물의 전송출력을 허락함.

2005 년 5 월 일

저작자: (서명 또는 인)

조선대학교 총장 귀하