2002 International Computer Symposium (ICS2002)

Workshop on Artificial Intelligence

Building a DNS Ontology using METHONTOLOGY and Protégé-2000

Chang-Sheng Chen, *Shian-Shyong Tseng, Chien-Liang Liu, Chia-Hao Ou

Department of Computer and Information Science

National Chiao-Tung University

1001, Ta Hsueh Rd., HsinChu, Taiwan 300

E-mail: sstseng@cis.nctu.edu.tw

Tel: +886-3-5712121 Ext.56605

Fax: +886-3-5721490

Abstract

The Domain Name System (DNS) is an essential part of the Internet software infrastructure. Unfortunately, although DNS is so important to network operation today, rather few DNS administrators have the expertise to do the jobs well. Due to the complex and distributed nature of the DNS systems, we could often find lots of poorly performed DNS servers on lots of Internet sites. In this paper, we study the problem of building a DNS ontology from scratch using the METHONTOLOGY methodology and the Protégé-2000 system. Ontologies are becoming an important mechanism to build information systems. The advantages include the sharing of knowledge, the re-use of knowledge, and the better engineering of knowledge-based systems with respect to acquisition, verification and maintenance. To help extract knowledge from experts and construct the DNS ontology, we propose an efficient knowledge acquisition algorithm, DNS Ontology Construction Algorithm, to fast conceptualize DNS domain knowledge. This DNS ontology can then be used as a basis for some applications to enhance the operation, planning and management of the DNS systems.

Keywords: DNS, knowledge-based system, METHONTOLOGY, ontology, Protégé-2002

* The corresponding author

† This work was supported in part by the MOE Program of Excellence Research of the Republic of China under Grant No. MOE89-E-FA-04-1-4.

1. Introduction

The Domain Name System (DNS)[1,2] is an essential part of the Internet infrastructure. Although DNS is so important to network operation today, some research results showed that over 70% of the DNS servers of commercial sites (e.g., ".COM" Zones) have some configuration errors [3,4]. In principle, the hierarchical and distributed properties of the DNS system make the administration duties to be distributed among different organizations and departments, and these make the whole system more scalable and robust. However, they also make the whole system more difficult on debugging and tracing some network system issues.

There are many operational, planning and management issues that need expertise to improve the DNS system [4,5,6]. Unfortunately, new administrators or administrators that manage a small scale of network usually do not know the theoretical and practical knowledge of DNS system very well. It takes a long time for them to gain the related knowledge without the assistance of the experts. Currently, most DNS assistant software packages are built by using conventional methodology to solve problems [4]. These programs are developed to solve domain zone management, find domain zone configuration errors, provide friendly user interface, and provide network tools. No one uses the expert system methodology to solve DNS problems and deals with the complex DNS management problems.

Table 1. Simple Classification of Common DNS Problems [4,5,6,7]

Category	Examples
1. Configuration errors	Lame Server, etc.
2. Inappropriate planning and management (e.g., Improper defaults, etc)	Inappropriate dynamic update, WINS/DNS forwarding, etc
3. Inappropriate software implementation (e.g., not immune to cache poisoning, etc.)	DNS-spoofing, server root vulnerability exploited, etc.
4. Attacks to the DNS systems	DDoS, forwarding attacks, etc

Table 1 shows a simple classification of DNS problems that most DNS administrators might encounter. Due to the complex and distributed nature of the DNS system, we could often find lots of poorly performed DNS servers (i.e. by mis-configuration, inappropriate planning, etc.) on lots of Internet sites. Moreover, given the importance of DNS servers, direct or indirect attacks on the DNS systems are common [7,8].

Knowledge sharing and reuse have become one of the primary goals of Knowledge Representation Systems

[9,10]. To achieve the goal of knowledge sharing and reuse, we need to have a common language, ontology. Ontologies are becoming an important mechanism to build information system. Ontology defines the concepts, the attributes of the concepts, and the relationships among concepts. With the help of ontology, the knowledge is not only human-readable but also machine-readable [11,12].

In this paper, we study the problem of building a DNS ontology from scratch using METHONTOLOGY [13,14] and the Protégé-2000 system [15]. This DNS ontology can then be used as a basis for some applications to enhance the operation, planning and management of the DNS systems. To help extract knowledge from experts and construct the DNS ontology hierarchy, we propose an efficient knowledge acquisition algorithm, DNS Ontology Construction Algorithm, to fast conceptualize DNS domain knowledge. The DNS ontology has been verified by the experts and been shown to be useful. The DNS ontology could be used for facilitating inexperienced administrators in DNS planning and management issues: (1) to re-organize the resources of the DNS systems in the organization to make them become more efficient, (2) to upgrade with the scale of the DNS systems when the environment is changed, (3) to make sure that each DNS server is configured correctly and properly to avoid those common problems that come from configuration errors, (4) and most importantly to provide expertise to those administrators whenever they need it.

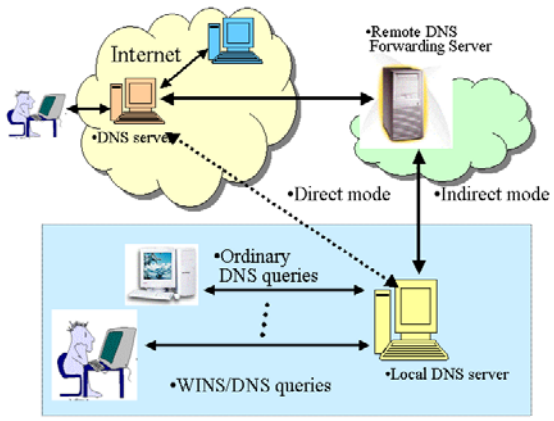
2. Preliminaries

DNS problems (e.g., planning and management, etc.) are very complex and not easy to solve. Furthermore, the shortage of DNS domain expert makes the situation worse.

2.1 Basic of the DNS System

The Domain Name System [1,2] is responsible for translating between hostnames and the corresponding IP addresses needed by software. The mapping of data is stored in a tree-structured distributed database where each name server is authoritative (responsible) for a portion of the naming hierarchy tree. The client side query process typically starts with an application program on the end user's workstation, which contacts a local name server via a resolver library. That client side name server queries the root servers for the name in question and gets back a referral to a name server who should know the answer. The client's name server will recursively follow referrals re-asking the query until it gets an answer or is told there is none. Caching of that answer should

happen at all name servers except those at the root or top-level domains (.com for example). The working paradigm could be illustrated as shown in Figs.1a. and 1b.



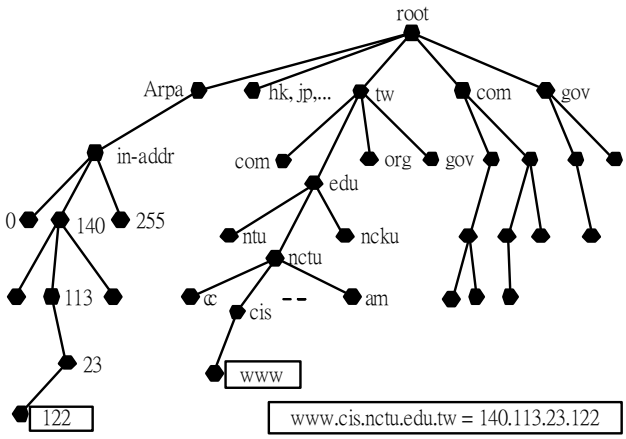


Fig. 1a. The operation of DNS

Fig. 1b. The hierarchical DNS structure

2.2 The needs of ontologies

Ontologies are useful in a range of applications, where they provide a source of precisely defined terms that can be communicated across people and applications. An information system cannot be written without a commitment to a model of the relevant world – commitments to entities, properties, and relations in that world [11]. The role of ontologies is to capture domain knowledge and provide a commonly agreed upon understanding of a domain. The common vocabulary of an ontology, defining the meaning of terms and their relations, is usually organized in a taxonomy and contains modeling primitives such as concepts, relations, and axioms [12]. With the help of ontology, the knowledge is not only human-readable but also machine-readable. Having developed a formal specification for a domain ontology, it is possible for database and software developers to agree on its use.

2.3 Overview of the METHONTOLOGY [13,14] methodology

Up till now, the ontology building process is still a craft rather than an engineering activity [12,14]. Each development team usually follows its own set of principles, design criteria and phases on the ontology development process. Fig. 2 shows the METHONLOGY methodology for ontology building. In [14], the authors

explain that the life of an ontology moves on through the following states: specification, conceptualization, formalization, integration, implementation, and maintenance. The evolving prototype life cycle allows the ontologist to go back from any state to other if some definition is missed or wrong. So, this life cycle permits the inclusion, removal or modification of definitions anytime of the ontology life cycle. Knowledge acquisition, documentation and evaluation are support activities that are carried out during the majority of these states.

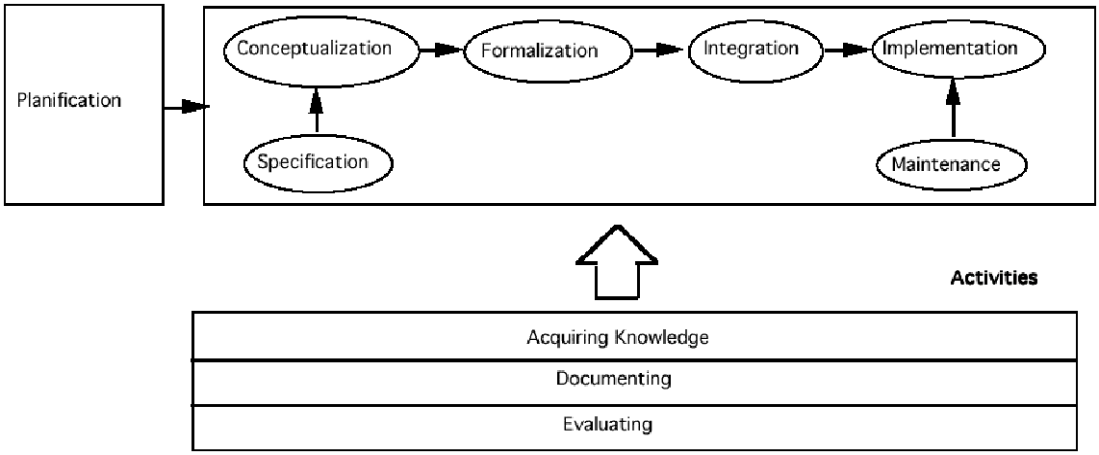


Fig. 2. States and activities in the ontology life cycle [14]

2.4 The tool for ontology construction

Tools are helpful to aid ontologists in constructing ontologies, and merging multiple ontologies since such conceptual models are often complex, multi-dimensional graphs that are difficult to manage. These tools also usually contain mechanisms for visualizing and checking the resulting models – over and above the logical means for checking the satisfiability of the specified models.

Protégé-2000 is an easy-to-use knowledge acquisition tool that could construct the domain ontology and achieve the interoperability with other knowledge-representation systems. In this paper, we build a DNS ontology using Protégé-2000. The operation details will be described later.

3 Motivating Scenarios for building the DNS ontology

DNS problem domain is very complex and varies greatly on different sites because too many things, like management strategies and resources, need to be considered [4,5,6]. For example, the DNS system architecture of a site can be a very simple one (e.g., with only two standard type DNS servers); or, it could be a

much complex one, with a few standard type DNS servers and lots of caching-only servers for special considerations. Due to many factors such as the cost-performance issue or security issue, sometimes it is necessary for a site to change its DNS system architecture from a simple type into a more complex one. However, this is not an easy job for most inexperienced administrators. On many occasions, it needs the guidance of the DNS domain experts. But, it is a pity that domain experts are so hard to find and cannot always standby for those inexperienced administrators under emergency conditions. This section describes sample scenarios about the requirements for applying DNS ontology to achieve one or more purposes.

3.1 Scenario: Diagnosing DNS-related Problems

When inexperienced administrators find there is something wrong with their DNS servers, the traditional way to solve the problem is trying to find a domain expert that can offer expertise, or find some documents that describe similar situations, and then try to modify the solutions to satisfy their own situation. Fig. 3a shows the process when an inexperienced administrator asks a domain expert for help. The domain expert will try to make a tentative diagnosis according to the facts that the inexperienced administrator provides, and try to verify the guess. Once the tentative diagnosis can be verified to be correct, solutions are proposed to fix the problem(s). If the tentative diagnosis is shown to be wrong, another tentative diagnosis will be proposed and it will be verified again. This cycle will be repeated until a final diagnosis can be made or confirmed.

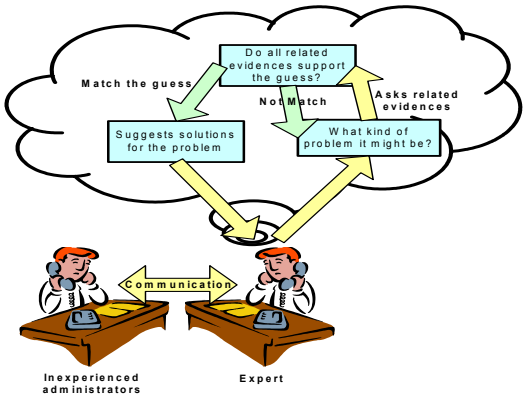


Fig. 3a Diagnosing DNS-related problems

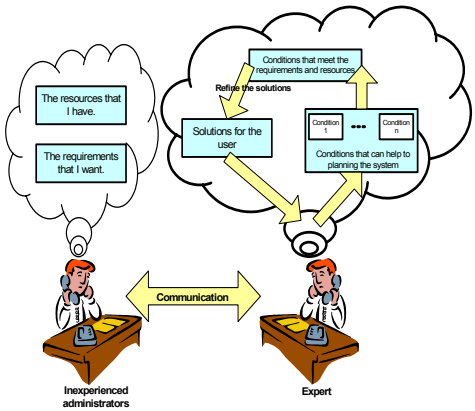


Fig.3b DNS Planning and management

3.2 Scenario: DNS Planning or Management Problems

When inexperienced administrators need to re-organize the resources of their DNS systems or design new DNS systems, they might want the experts to help design the DNS systems and teach them some management skills. Fig. 3b shows the traditional way when an administrator asks an expert for help. The expert needs to consider a lot of conditions; each of them depends upon the resources and requirements. The expert will solve the planning or management problems by reusing domain knowledge. Usually, the first model comes into the expert's mind is the basic DNS model. The expert will ask the inexperienced administrator what kind of resources she/he has and what requirements she/he wants in order to extend the basic model to a more complex one. After a thoroughly brainstorming, the expert makes decisions according to the resources and the importance of the requirements. Finally, a suitable DNS system plan is made for the administrator and the process is finished.

4. Building a DNS ontology using METHONTOLOGY

In this paper, we build a DNS ontology from scratch using METHONTOLOGY [14] and the Protégé-2000 system [15]. As specified in [14], the life of an ontology moves on through the following states: specification, conceptualization, formalization, integration, implementation, and maintenance. Next, we will give short descriptions about the phases required for building the DNS ontology in the following subsections.

4.1 Specification

As shown in Table 2, in the specification process, we describe the goal of the DNS ontology, the scope of the ontology, and the source of the knowledge. The goal of the specification phase is to produce an informal specification document written in natural language using competency questions [14]. It is difficult for us to design a complete ontology for a specific domain at the first time. Hence, we may design a subsection of the ontology first, and refine it later when necessary. One of the ways to determine the scope of the ontology is to sketch a list of competency questions that a knowledge base based on the ontology should be able to answer. The questions will serve as the litmus test later. In the future, that could help us to know which section we have completed and if someone wants to reuse our ontology, she/he could know whether the ontology is appropriate for her/him from the scope description. The source of knowledge is used to denote the source where we get our knowledge.

Table 2. Requirements specification document for the DNS ontology

- Domain: DNS
- Date: 2002/07/08
- Developed by C.S.Chen, C.L.Liu, C.H.Ou
- Purpose:
  - Ontology about DNS to be used when information about DNS is required in debugging/analysis, planning, configuration, etc.
- Level of formality: informal
- Scope:
  - List of 5 type of servers: master, slave, caching-only, …
  - List of concepts: DNS registration, DNS security, caching, …
  - Information about at least the following properties: caching, authority, forwarding, access control, ….
- Source of knowledge:
  - Interviews with the experts
  - DNS FAQ from the ISC, http://www.isc.org/
  - Books:
    - P. Albitz and C. Liu, DNS and BIND 4th, O'Reilly & Associates, Inc., Sebastopol, CA, 2001
    - The Domain Name Handbook, R&D Books an imprint of Miller Freeman, Inc., Lawrence, KS, 1998

■ Informal Competence questions

Table 3 shows a list of competency questions in the DNS domain. Judging from this list of questions, the ontology will include the information on various DNS characteristics and configuration types that could be used for DNS planning and management suggestions, fixing DNS operation problems, etc.

Table 3. Informal competency questions for building a DNS ontology

Issue	**Description**
■ Management of DNS space	What activities must a particular system administrator perform to accomplish the registration jobs for a specialized domain zone?
■ DNS server software	What is the appropriate server software to choose for implementing the DNS system of an organization? Which version and on what platform?
■ Available issue	While deploying a DNS system, what are the design principles to avoid single point of failure?
■ Performance issues	When deploying DNS systems, what are the design principles to improve the system performance on the following aspects: CPU, memory, Network, etc.
■ Security issues	When deploying DNS systems, what are the design principles to take care of the security considerations on the following aspects: Protection of servers, protection of data, DNS anomaly detection, etc.
■ Resource considerations	Manpower & Hosts maintenance

4.2 Knowledge Acquisition

It is important to bear in mind that knowledge acquisition is an independent activity in the ontology

development process [14]. However, it is coincident with other activities. Most of the acquisition is done simultaneously with the requirement specification phase, and decreases as the ontology development process moves forward. In this section, we will describe the ways in which the DNS knowledge acquisition process can support the ontological engineering process.

Expert, books, Internet experimental results, etc. are sources of knowledge from which knowledge can be elucidated using conjunction techniques such as: brainstorming, interviews, formal and informal analysis of texts, and knowledge acquisition tools. One important role for ontologies is that they can be used by knowledge acquisition tools to direct the acquisition of domain knowledge [12]. This is because ontologies specify which constraints domain knowledge should satisfy. In this paper, we study the problem of developing a knowledge acquisition method for constructing the DNS ontology systematically to help conceptualize the DNS problem domain. However, there is still one thing to be addressed first. In DNS domain, experts are used to describe their domain knowledge by enumerating cases (e.g., including both positive and negative cases). We must decide on how many cases they should give at each time.

Personal Construct Psychology (PCP), developed by George Kelly in the early 1950s, has wide application in modeling human knowledge processes. PCP gives an account of how people experience the world and makes sense of that experience [16]. The repertory grid was an instrument designed by Kelly to bypass cognitive defenses and give access to a person's underlying construction system by asking the person to compare and contrast relevant examples. The repertory grid methodology has evolved in the light of application experiences and a lot of related research applications have been development [16,17]. However, as different from the repertory grid methodology, we propose another approach by extracting the concepts and attributes from the input cases.

The power of a few critical cases described in terms of relevant attributes to build domain ontology is remarkable [16]. This is because it is often easier and more accurate for the experts to provide critical cases rather than domain ontology. Hence, three cases seem adequate for our DNS knowledge acquisition algorithm because it is usually not hard for experts to enumerate three cases and will not take too much time from them. Besides, three cases usually contain enough and not too much information, so the knowledge engineers can modify the ontology hierarchy easily. After the extraction, we would then decompose these concepts into a lot of

sub-components by class abstraction and inheritance. Some examples and the operation details will be described in Sections 4.3, 4.4 and 4.5 later. A simplified procedure is outlined as shown in Fig. 4. The detail of the algorithm is shown as follows:

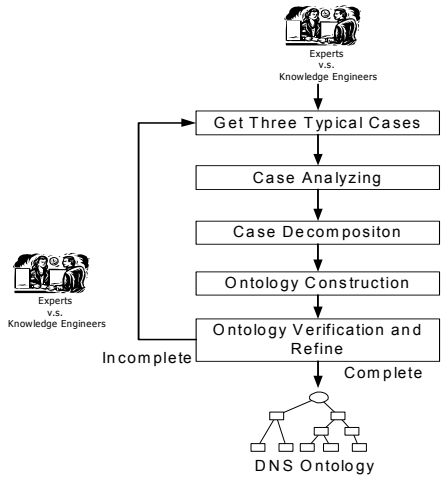


Fig. 4. Our proposed knowledge acquisition algorithm

- **Knowledge acquisition algorithm for facilitating DNS ontology construction**

Input: Every kind of DNS cases.

Output: DNS Ontology.

Step 1: Get three typical cases from experts whenever possible.

Step 2: Analyze and decompose the cases.

Step 2.1: Ask experts about how to describe these cases in the following issues: hardware, software, management, and configuration.

Step 2.2: Decompose these cases into small components (e.g. availability, security, performance, etc.)

Step 3: Build or Refine the DNS ontology.

Step 3.1: If two or more components can be conceptually covered by another component, then put them in the lower level of ontology hierarchy and link them to that component.

Step 3.2: If it cannot naturally form an ontology hierarchy, create new components to conceptually cover the original components and link the ontology sub-trees. Finally it will form an complete ontology hierarchy.

Step 4: Experts verify the hierarchical ontology.

Step 4.1: Verify the correctness of the skeletal ontology, including those new created components and the hierarchical structure.

Step 4.2: If the experts find any ontological components not covered by the constructed skeletal ontology, then they will repeat and try enumerating three more cases that might contain other ontology

components. The construction process will go back to Step 2.

Step 5: After experts' verification, DNS ontology is constructed to cover DNS domain knowledge and it is helpful when we build knowledge objects.

4.3 Conceptualization - taxonomy of concepts, relations, etc.

An ontology is an explicit, knowledge level specification of a conceptualization [12]. In principle, the organization of concepts could provide for at least two purposes: (1) identification (2) specialization and generalization. In the DNS domain, knowledge objects can be built up based on the representation of knowledge, but this cannot be done without the domain knowledge of DNS. By using the brainstorming and trimming approach, we could produce a list of potentially relevant concepts and delete irrelevant entries and synonyms as shown in Tables 4a, 4b, 4c and 4d.

Table 4a. Concept table for the DNS ontology

Concept	
Domain Name Space	(1). Domain Entries: A, PTR, NS, CNAME, MX, etc (2). Type-of-server: Master/Slave server, authoritative-only server, caching-only server, DNS forwarder (3). Type-of-zone: root hint files, forward vs. reverse zone, public vs. private zone, etc. (4). Misc: Domain Zone Delegation, Zone Authority, etc
DNS server programs	■ Software packages: BIND, Windows DNS, etc. ■ Version issues
DNS client program	Resolver library routines

Table 4b. Relation table for the DNS ontology

Relations
(1). DNS query request Clnt → Srv
(2). DNS query response Srv → Clnt
(3). DNS forwarding (a special case, indirect query) (Clnt →) Srv1 → Srv2;
(4). Zone transfer request: Slave_Srv → Master_Srv
(5). Zone transfer response: Master_srv → Slave_Srv
(6). Dynamic update request Clnt → Srv (authoritative server)
(7). …

We could view the concepts as the classes of the Object-Oriented Programming paradigms. From the concept table (Table 4a), we can get the kind of the classes for constructing the DNS ontology. For example, we know that an authoritative-only DNS server could be classified into being a master server, or a slave server. Since

a concept could be mapped into a class, so we could define a DNS server class. The master servers, slave servers, or the authoritative-only servers all belong to the DNS server class. Moreover, that means we could construct the master server class or the slave server class by following the inheritance relationship from the DNS server class. In this way, after we conceptualize the domain problem, we could get the domain model as illustrated in Fig. 5.

Table 4c. Function table for the DNS ontology

Function
(1). *DNS entry registration* - add, delete, or modify DNS entries (with the NS RR excluded) within a domain zone
(2). *Domain zone delegation* - add, delete, or modify NS RR's within a domain zone
(3). *Positive DNS caching* - caching of a DNS answer after a successful query and its response returned
(4). *Negative DNS caching* - caching of an invalid DNS result after a successful query and its response returned
(5). …

Table 4d. Axiom table for the DNS ontology

Axioms
1. Any domain zone should have at least two authoritative DNS servers for resolving DNS queries from concerning the zone from Internet.
2. Any DNS server should have a primed root hint file to operate properly (e.g., 13 root DNS server from ICANN).
3. Any DNS server of a site should not forward queries concerning private zones to other DNS server for resolving. Instead, queries about these zones should be resolved at local DNS servers (e.g., 127.in-addr.arpa, 10.in-addr.arpa, 168.192.in-addr.arpa, etc.)
4. …

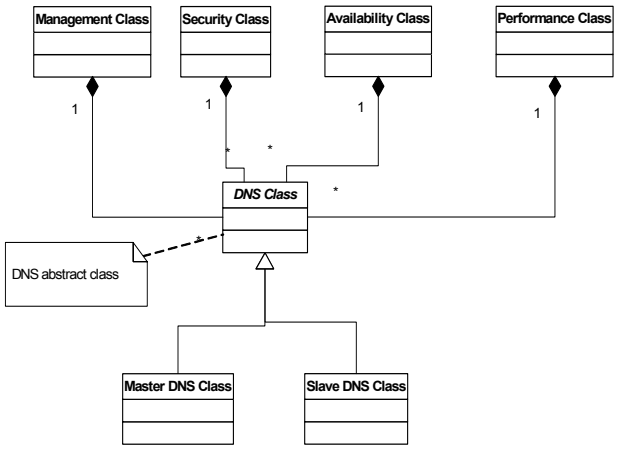


Fig. 5. The abstraction of DNS class

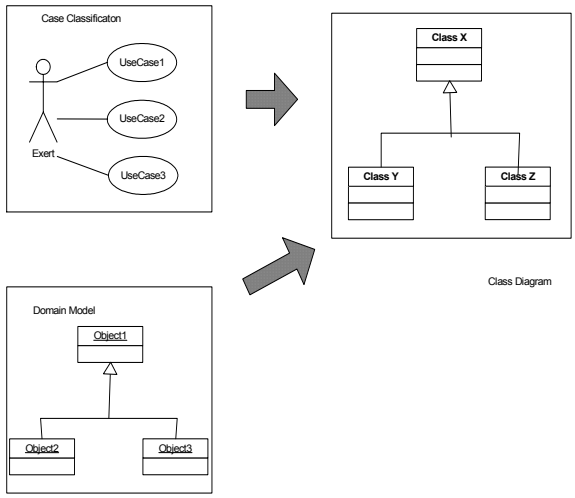


Fig. 6. DNS modeling and DNS ontology construction

Relation element is a type of interaction between concepts of the domain. For example, as shown in Table 4b, the zone transfer request relation is used to denote that the zone data is transferred from some DNS server A to another DNS server B. Functions element (as shown in Table 4c) is a special case of the relations, which denotes some special operation defined on certain DNS servers themselves. Axioms elements (as shown in Table 4d) are used to model sentences that are always true. These could be used to guide our ontology design. After we finish the ontology construction, the axioms could also be used to verify our design.

4.4 Formalization

For constructing the DNS ontology, we need to abstract our class design and build the class hierarchy. The abstraction process would make the classes become high cohesion and low coupling. That could greatly simplify the DNS ontology development process. However, in what ways should we do to perform the abstraction?

First of all, we might have to know the relationship from the domain expert's input cases and collect the common attributes among all the classes. If there is more than one class with the same (or almost the same) attributes, we could abstract and create the class that contains these basic attributes, and we could generate a lot of sub-classes, which have the common attributes, by inheriting from the class. For example, master servers and slave servers are both DNS servers. In principle, we know that there are some basic attributes for all the DNS servers. Hence, we could define the abstract DNS class (e.g., Domain Name Server class) to describe a DNS server. By definition, an abstract class (like the DNS class mentioned) would not have an instance. Instead of generating an instance of the DNS class, we would generate its sub-classes, namely, the master DNS server class, the slave DNS server class, etc. Since they are concrete classes, they would inherit all the attributes of its parent class (e.g., the abstract DNS class) as shown in Fig. 5.

In the formalization process, all we have to do next is try to use the case classification and domain modeling to generate the entire class diagram. With the help of domain modeling, we could know the attributes and behavior of the DNS problem domain, and with the help of case classification, we could build up the inheritance and the composition relationship. In this way, we could generate a more accurate hierarchy of the problems. The procedure could be illustrated as shown in Fig. 6.

4.5 Implementation

We adopt the METHONTOLOGY approach to build a DNS ontology using Protégé-2000 [15] from scratch.

The knowledge model of Protégé-2000 is frame-based and the ontology built consists of classes, slots, facets, instances. The class element is used to describe the concept, and we could build the class hierarchy of the taxonomy. For example, Fig.7a shows a class diagram with the DNS class mentioned above. In DNS ontology, since both master DNS server and slave DNS server are DNS servers, both of them belong to the subclasses of the DNS server class and they both inherit the DNS property. From the class hierarchy, we could know the relationship of these concepts.

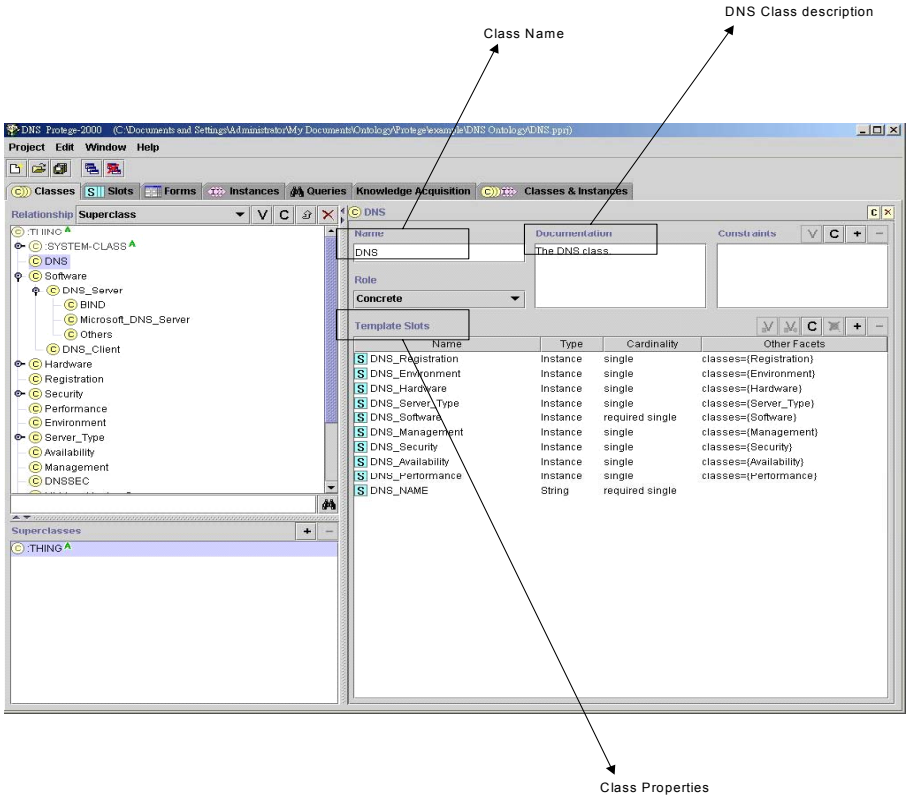
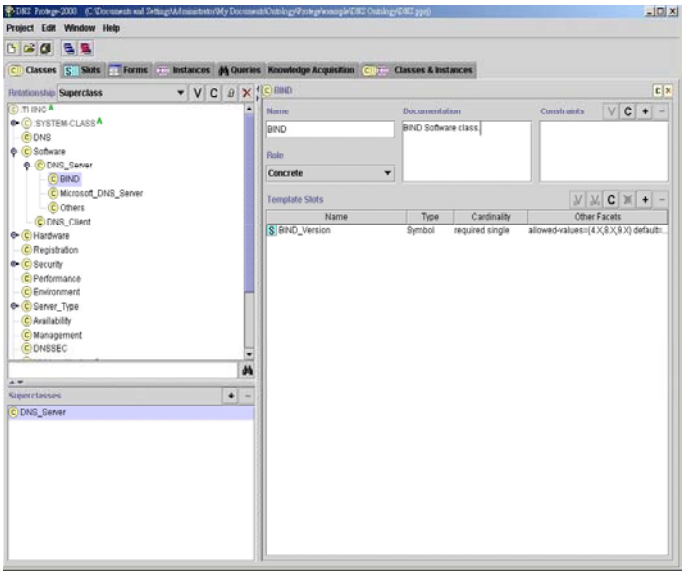


Fig. 7a. A class diagram showing part of the DNS ontology using Protégé-2000

Slots in Protégé-2000 describe the property of classes and instances, such as the configuration of the DNS server, or the software version of the DNS server. The slot could be created without being attached to a specific class. So a version slot could be used to denote the version of the BIND software or version of the Microsoft DNS Server. When we could bind the slot to a specific class, it could have value, so if we attached the version to the BIND software, it could have the value of 8.2.2 or 9.2.1.

Facets in Protégé-2000 are the constraint of the slot. We could set up the constraint of cardinality or the value type of the slot. For example, the cardinality of the version attribute in DNS is single and its type is symbol. We also could define the minimum and maximum value for the numeric slots.



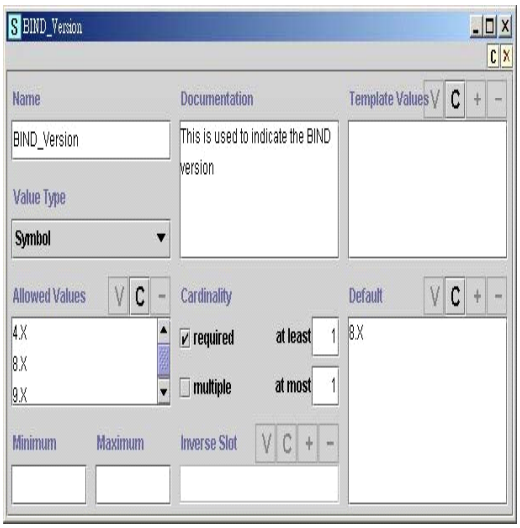


Fig. 7b. and Fig. 7c. showing the slots and facets of the DNS ontology respectively using Protégé-2000

5 Evaluation

In this paper, we build a DNS ontology using METHONTOLOGY and the Protégé-2000 system. The DNS ontology has been verified by the experts and been shown to be useful. We will show two ways in which explicit DNS ontology can be used during knowledge engineering: for knowledge elicitation and for computational design. Because the DNS ontology specifies which constraints domain knowledge should satisfy, it can be used to direct the knowledge elicitation process. During computational design application, the ontology can be used to determine the suitability of problem solvers for the DNS application.

5.1 Skeletal model of the DNS ontology

Fig. 8 shows the skeletal model of the DNS Ontology. The DNS ontology is intended to bridge the gap between the executable system and the real world it models. The root of DNS ontology is DNS Environment; it covers four major sub trees: DNS group, Management strategies, Events, and Resources. Each of them is described in the following:

(1) DNS group: The DNS_Server_Host class shows the related ontology structure concerning a DNS server. The servers within some Related_DNS_hosts class might have some close interactions with one another, e.g., relationship between a master and its slave DNS servers. The Other_DNS_hosts class is referred to the other

DNS systems all over the Internet. The Root_DNS_hosts class is a very special sub-class of the Other_DNS_server class. It plays an important role in the hierarchical and distributed DNS systems. As described in Sec. 2.1, all the DNS servers without DNS forwarding capability are supposed to consult some server(s) in the Root_DNS_hosts class first whenever they have no idea about some DNS queries.

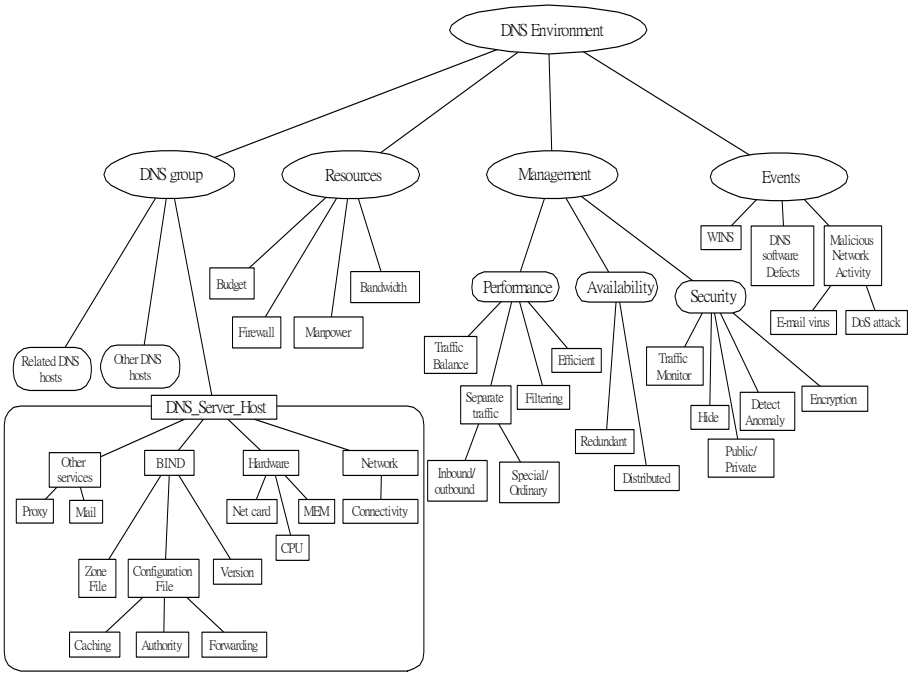


Fig. 8. The skeletal model of the DNS Ontology

(2) Management strategies: Management strategies are very important when we plan and manage the DNS systems. According to the different requirements of each DNS system's management strategies, not only configurations in each DNS host are different but also many tradeoffs need to be considered. Availability, performance, and security are major issues in this field.

(3) Events: Events are those activities that can affect the DNS system, e.g. DoS attacks can overload the DNS system.

(4) Resources: In DNS ontology hierarchy, there are some components that do not belong to DNS group or management strategies. They can affect the management plans and some configurations; e.g., when budget is sufficient, hardware can be upgraded to high level to improve the efficiency.

5.2DNS traffic distribution sample

Fig. 9 gives a brief summary on DNS traffic for the 9 regional centers on TANet (i.e. Taiwan Academic Network). It is generated from the monthly traffic statistical data, among some 3-month interval (e.g. September, October, November in 2001), collected on Newsletter of MOECC [18], Taiwan. We could find that the average values (e.g., 1st column of Fig.9) of the DNS traffic distributions (as compared to the total traffic including WWW, SMTP, etc.) of the sites center around 5%. However, on some sites, the monthly average values are more than 10% (or even bigger) and their traffic distributions obviously have larger variation than the others during the summary interval. In short, there might be some DNS configuration problems, some network abusing, or some DNS anomalous activities running on these sites. Hence, it is highly expected that we should have some intelligent systems to help configure and maintain the smooth operation of the DNS systems.

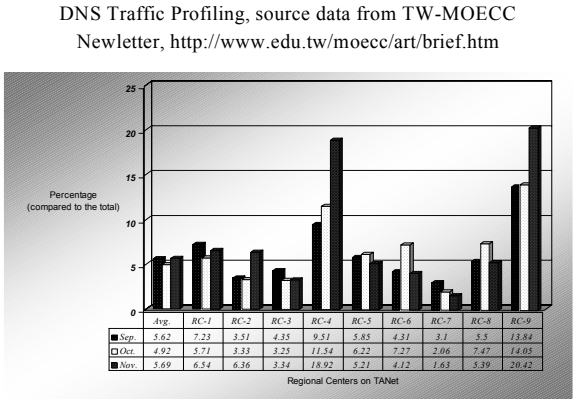


Fig. 9. DNS traffic distributions

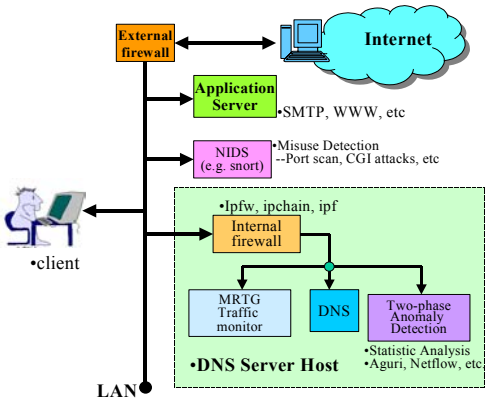


Fig. 10. A typical configuration of a single DNS system.

5.3 Facilitating computational design - DNS-related Security Issues

Fig. 10 shows a typical configuration of a single DNS system. For protecting the DNS system from being abused or attacked by malicious sites, some other security systems such as firewall and IDS are suggested for securing the related DNS system. After analyzing the case, we find that if we want to consider the DNS security issue, we should consider the DNS zone data protection, DNS anomalies detection and identification, DNS dynamic update protection and DNS server host protection. In the ontology, we construct a simple composition relationship such that security class would be composed of DNS zone data protection instance, DNS anomalies detection and identification instance, DNS dynamic update protection instance and DNS server host protection instance as shown in Fig.11a. Furthermore, we find that the Hinder_Illegal_Zone_Transfer class and

DNS_Dynamic_Update both perform different actions according to the DNS server software. Hinder_Illegal_Zone_Transfer class and DNS_Dynamic_Update class would have the same subclass, DNS_BIND and DNS_Microsoft_Server. Based on the class abstraction, we would build up an abstract DNS_BIND class and an abstract DNS_Microsoft_Server class as shown in Fig.11b since there would be no DNS_BIND and DNS_Microsoft_Server instance. Next, we define 3 subclasses for DNS_BIND based on their versions. Each version class would inherit the name, reason, and action attributes from DNS_BIND class. Name attributed is used to identify the class, the reason attribute is used to describe the reason why we want to prevent dynamic update or illegal zone transfer. The action attribute is used to describe the step we should follow to solve the problem.

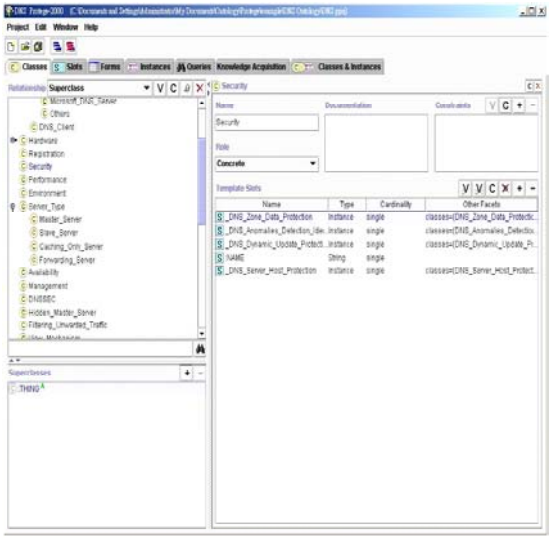


Fig. 11a. The composition relationship of security

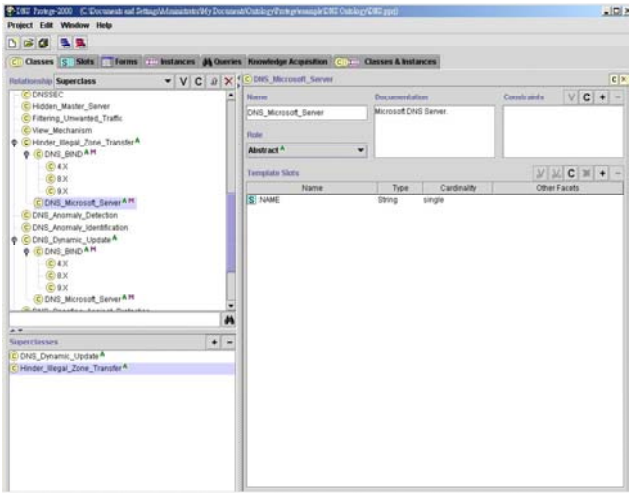


Fig.11b. DNS_BIND and DNS_Microsoft_Srv class

5.4 Extend the DNS ontology

In [14], the evolving prototype life cycle allows the ontologist to go back from any state to other if some definition is missed or wrong. So, this life cycle permits the inclusion, removal or modification of definitions anytime of the ontology life cycle. Since the DNS is still evolving, we will have to update the DNS ontology whenever possible. For example, (1) the IPv6 and wireless communications (hence DHCP and Dynamic update on DNS) start to become popular on many parts of the Internet. (2) Multilingual DNS environment is under development. In the future, there might be some DNS server and client implementation issues that need to be considered. Therefore, a lot of new concepts and relations have to be added or modified for the DNS ontology to

address these topics.

6. Concluding Remarks and Future Work

It is now widely recognized that constructing a domain model, or ontology, is an important step in the development of a knowledge-based system. The advantages include enabling the sharing of knowledge, the re-use of knowledge, and the better engineering of knowledge-based systems with respect to acquisition, verification and maintenance. In this paper, we apply the METHONTOLOGY approach to build a DNS ontology using Protégé-2000 from scratch. We present the characteristics of the DNS and an efficient knowledge acquisition algorithm, DNS Ontology Construction Algorithm, is proposed to help knowledge engineers systematically extract knowledge from experts and construct the DNS ontology. This ontology can then be used as a basis for some applications of DNS planning and management.

We have shown two ways in which explicit DNS ontology can be used during knowledge engineering: for knowledge elicitation and for computational design. Because the DNS ontology specifies which constraints domain knowledge should satisfy, it can be used to direct the knowledge elicitation process. During computational design application, the ontology can be used to determine the suitability of problem solvers for the DNS application.

For evaluating the potential of this ontology and for advanced research, an Intelligent DNS Planning and Management System (iDNS-PMS), based upon expert system methodology and web-interface, is under development to help inexperienced administrators find the DNS problems and solutions in their systems or provide planning and management solutions for them. Future research will focus on several issues. First, we will complete the iDNS-PMS prototype. Second, since the DNS system is still evolving, the corresponding DNS ontology needs refining. For example, we will extend the ontology to integrate more topics about IPv6, multilingual DNS, intrusion detection mechanisms concerning DNS, etc. That will be our future work.

REFERENCES

[1]. P. Mockapetris, "Domain Names - Concepts and Facilities," RFCs 1034, November 1987.

[2]. P. Mockapetris, ``Domain Names - Implementation and Specification" RFC 1035, Nov. 1987

[3]. Man-Mice Company, *"Domain Health Survey for .COM - May 2002"*, accessed on Jul 10, http://www.menandmice.com/6000/61_recent_survey.html

[4]. C.H.Ou, "Design of An Intelligent DNS Planning and Management System", Master Thesis, Dept. of Computer and Information Science, National Chiao-Tung University, Taiwan, 2002.

[5]. BIND (Berkeley Internet Domain), URL:http://www.isc.org, Accessed 10 July 2002

[6]. C.S.Chen, S.S.Tseng, C.J.Liu, C.H.Ou, "Design and Implementation of an Intelligent DNS Configuration System", 4th International Conference on Advanced Communication Technology, Feb 5-7, 2002, Korea.

[7]. C.S.Chen, S.S.Tseng, C.J.Liu, "A Distributed Intrusion Detection Model for the Domain Name System", to appear in Journal of Information and Science Engineering, Nov. 2002.

[8]. Jay L. Koh, "Recent Developments and Emerging Defenses to D/DoS: The Microsoft Attacks and Distributed Network Security" February 9, 2001, URL: http://www.sans.org/infosecFAQ/DNS/developments.htm, Accessed 16 July 2002.

[9]. Musen, M.A., Dimensions of knowledge sharing and reuse, Computers and Biomedical Research. 25: p.435-467,1992.

[10]. Studer, R., Benjamins, R., Fensel, D. *Knowledge Engineering: Principles and Methods*. IEEE Transactions on Data and Knowledge Engineering 25(1-2).pp:161-197. 1998

[11]. Chandrasekaran, B. and Jorn R. Josephson, V. Richard Benjamins. 1999. What Are Ontologies, and Why Do We Need Them? IEEE Intelligent Systems. 14 (1): pp. 20 - 26.

[12]. G Van Heijst, A.T. Schreiber and B.J. Wielinga, *Using Explicit Ontologies in KBS Development*, International Journal of Human-Computer Studies, Vol. 46, No. 2/3, February-March, 1997, pp. 183-292.

[13]. Fernandez Lopez. *Overview of methodologies for building ontologies*. In Proceedings of the IJCAI-99 Workshop on Ontologies and Problem-Solving Methods: Lessons Learned and Future Trends. CEUR Publications, 1999.

[14]. Fernandez, M.; Gomez-Perez, A.; Juristo, N., "*METHONTOLOGY: From Ontological Art Towards Ontological Engineering"***,** Workshop on Ontological Engineering. Spring Symposium Series. AAAI97 Stanford, USA.

[15]. N. F. Noy, R. W. Fergerson, & M. A. Musen. The knowledge model of Protege-2000: Combining interoperability and flexibility. 2th International Conference on Knowledge Engineering and Knowledge Management (EKAW'2000), Juan-les-Pins, France, 2000.

[16]. Brian R Gaines and Mildred L G Shaw, *Knowledge Acquisition Tools based on Personal Construct Psychology*. Special Issues on "Automated Knowledge Acquisition Tools" of the Knowledge Engineering Review, 1992.

[17]. Brian R Gaines and Mildred L G Shaw, *Eliciting Knowledge and Transferring it Effectively to a Knowledge-Based System*, IEEE Transactions on Data and Knowledge Engineering 5(1), pp.4-14. 1993.

[18]. MOECC Newsletter, http://www.edu.tw/moecc/art/brief.htm, Accessed 10 July 2002.