Available online at www.sciencedirect.com
SCIENCE DIRECT®
ELSEVIER
Int. J. Human-Computer Studies 58 (2003) 415–445
International Journal of Human-Computer Studies
www.elsevier.com/locate/ijhcs

A unifying framework for intelligent DNS management

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

Department of Computer and Information Science, Computer and Network Center, National Chiao-Tung University 1001, Ta Hsueh Road, HsinChu, Taiwan 300

Received 20 October 2002; accepted 17 December 2002
Paper accepted for publication by Editor, B. Gaines

Abstract

The Domain Name System (DNS) is a special kind of distributed directory service for people to create and access the network information systems by (1) allowing local control of its segments and (2) making each segment's data available on the Internet using a client–server scheme. However, few administrators have the expertise to do the jobs well since this distributed mechanism is a double-edged sword; it allows DNS not only to scale to Internet size but also allows for incredible mis-configurations. The presented study is an analysis of what problems and difficulties most DNS administrators might encounter and provides the insights into how various DNS assistant sub-systems could be designed and deployed to solve the complex problems or alleviate these DNS administration job loadings. Ontologies become an important mechanism to build information systems. The role of ontologies is to capture domain knowledge and provide a commonly agreed upon understanding of a domain. The advantages include the sharing and re-use of knowledge, and the better engineering of knowledge-based systems with respect to acquisition, verification and maintenance. In this paper, we propose a unifying framework (e.g. including configuration, outstanding traffic monitoring and analysis, planning and management, tutoring, etc.) for supporting intelligent DNS management using web interface and expert system technology to help inexperienced administrators in insuring the smooth operation of their DNS systems. To help extract knowledge, we propose an efficient DNS Ontology Construction Algorithm to fast conceptualize DNS domain knowledge through a hybrid method of brainstorming and use cases modeling. While some sub-systems are still under development, others have been prototyped and deployed for everyday use. As our experience with a first simple prototype has shown, the paradigm of using DNS ontology to build a unifying framework for intelligent

*Corresponding author. Tel.: +886-3-573-1721; fax: +996-3-571-4031.
E-mail address: cschen@mail.nctu.edu.tw (C.-S. Chen).

1071-5819/03/$ - see front matter
doi:10.1016/S1071-5819(03)00012-0

DNS management works good and effective. It is supposed that all these might become an integral part of the DNS administration activities of an organization in the near future.

Keywords: DNS; Knowledge sharing and re-using; Ontology; Use case modeling

1. Introduction

The Domain Name System (DNS) is an essential part of the Internet infrastructure. Currently, with the exception of some specific applications (e.g. peer-to-peer applications, etc.), most internet services are based on the working model that there will be some DNS queries before the communication activities. Therefore, even if your web servers are in good shape, the firewalls are doing their jobs, and your backend application servers and databases are in perfect order, none of these matters. All might seem to fail if your DNS servers do not work properly as expected.

Technically speaking, the DNS is a special kind of distributed directory service that people use to create (for accessing) and access other network information systems. In essence, DNS is a distributed database which (1) allows local control of its segments and (2) makes data in each segment available across the entire network using a client–server scheme. Robustness and adequate performance are achieved by replication and caching (Albitz and Liu, 2001).

However, this distributed control and configuration is a double-edged sword; it allows the system to scale to Internet size, but it also allows for incredible misconfigurations. The phenomenon is especially obvious between novice administrators and inexperienced administrators that manage a small scale of network. Hence, this is the main reason that even though DNS is so important to network operation today, some latest DNS survey in November 2002 (Man-Mice Company, 2002) showed that over 70% of the DNS servers of commercial sites (e.g. ".COM" Zones) have some configuration errors.

The objectives of the work reported in this paper are to use the World Wide Web as a continuing base to support intelligent DNS management in which essential DNS issues (e.g. DNS configuration and maintenance, outstanding DNS traffic monitoring and analysis, DNS debugging, DNS planning and management, DNS tutoring, etc.) become an integral part of the DNS administration activities of an organization.

How could a DNS administrator tell if her/his systems work as expected? Table 1 summarizes common issues that could often be identified at typical DNS servers. On considering these problems, one notes immediately that the DNS problem domain is rather complex and varies greatly on different sites because too many things, like management strategies and resources, need to be considered. For example, we could often find that there are unfixed bugs on some systems. The system administrators might make mistakes in the deployment phase or in doing maintenance jobs from

Table 1
Typical DNS management issues

Issues	Descriptions
Configuration	Delegations of domain zones, illegal setting of DNS entries, etc.
Availability	Master/slave architecture, data synchronization among deleted servers, etc.
Performance	DNS caching, forwarding, etc.
Security	Access control, dynamic update, intrusion detection, etc.
Interoperability	BIND (version 4, 8, 9, etc.), Microsoft DNS, etc.

time to time. For years, BIND (Albitz and Liu, 2001) was the basis for most vendors' DNS implementations, but recent independent implementations by Microsoft and others have introduced interoperability issues that were largely invisible before. Or, there is always the possibility of an authorized user misusing his or her privileges. Of all these problems, some will be found quickly as the system(s) can no longer provide normal services. Other cases may be ignored at first for a long time (i.e. before the buggy information query is triggered.) since the DNS system(s) can still provide most of the normal services in a degraded mode.

On the other hand, 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 very complex one, with a few standard-type DNS servers and lots of caching-only servers for special considerations, etc. 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 stand by for those inexperienced administrators under emergency conditions.

As mentioned in Ou (2002), many companies and people develop assistance software to help DNS administrators manage their DNS systems. However, most of these software packages are built by using conventional methodology. Basically, they are mainly used to solve syntax problems and provide user friendly interface, help domain zone management, find domain zone configuration errors, etc. Few, if any, address the DNS semantics issues or the complex DNS management problems.

It is now widely recognized that constructing a domain model, or ontology, is an important step in the development of a knowledge-based system (KBS) (Chandrasekaran et al., 1999). The advantages include the sharing and re-use of knowledge, and the better engineering of KBSs with respect to acquisition, verification and maintenance (Musen, 1992). The role of ontologies is to capture domain knowledge and provide a commonly agreed upon understanding of a domain (Studer et al., 1998). 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. With the help of ontology, the knowledge is not only human readable but also machine readable.

There are at least 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.

In Chen et al. (2002c), we built a DNS ontology using the METHONTOLOGY (Fernandez et al., 1997) methodology and Protégé-2000 (Noy et al., 2000) system from scratch. The DNS ontology is used for helping build the subsystems proposed in this paper. To help extract knowledge from experts and construct the DNS ontology, we proposed a knowledge acquisition (KA) algorithm, DNS Ontology Construction Algorithm, to fast conceptualize DNS domain knowledge through the traditional brainstorming-and-trimming approach. The DNS ontology has been verified by the experts and shown to be useful. However, since the DNS is still evolving, there is a need for updating the DNS ontology by adding new entries (e.g. concepts, relations, etc.) from time to time. To incorporate this fact, in this paper, we try to redesign the same algorithm by using a hybrid method of brainstorming and use cases modeling. Our experience has shown the effectiveness of this revised method. In fact, with a little modification, the same algorithm could be applied to other domains for ontology construction.

In this paper, we propose a unifying framework (e.g. including configuration, outstanding traffic monitoring and analysis, planning and management, tutoring, etc.) for intelligent DNS management using web interface and expert system technology to help inexperienced administrators in insuring the smooth operation of their DNS systems. The DNS ontology can 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 and (4) most importantly to provide expertise to those administrators whenever they need it. As our experience with a first simple prototype has shown, the paradigm of using DNS ontology to build a unifying framework for intelligent DNS management works good and effective; in the near future, it is supposed that all these might become an integral part of the DNS administration activities of an organization.

The rest of the paper is organized as follows. Section 2 gives an overview of the DNS system. Section 3 describes the awareness issue of the DNS traffic. In Section 4, we describe the system architecture of the unifying framework. Section 5 briefs on the DNS ontology. Section 6 describes the DNS KA process. In Section 7, we give some discussions about the system implementation and evaluation. Finally, in Section 8, we provide some concluding remarks.

2. Preliminaries

DNS problems (e.g. configuration, planning and management, etc.) are rather complex and not easy to solve for most administrators. Furthermore, the shortage of DNS domain experts makes the situation worse.

2.1. Basics of the DNS

The DNS (Albitz and Liu, 2001) is responsible for translation 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 (e.g. ".com", ".tw", etc.). The working paradigm can be illustrated as shown in Fig. 1.

2.2. Overview of uses cases modeling

The Unified Modeling Language (UML) is the industry-standard language for specifying, visualizing, constructing and documenting the artefacts of software

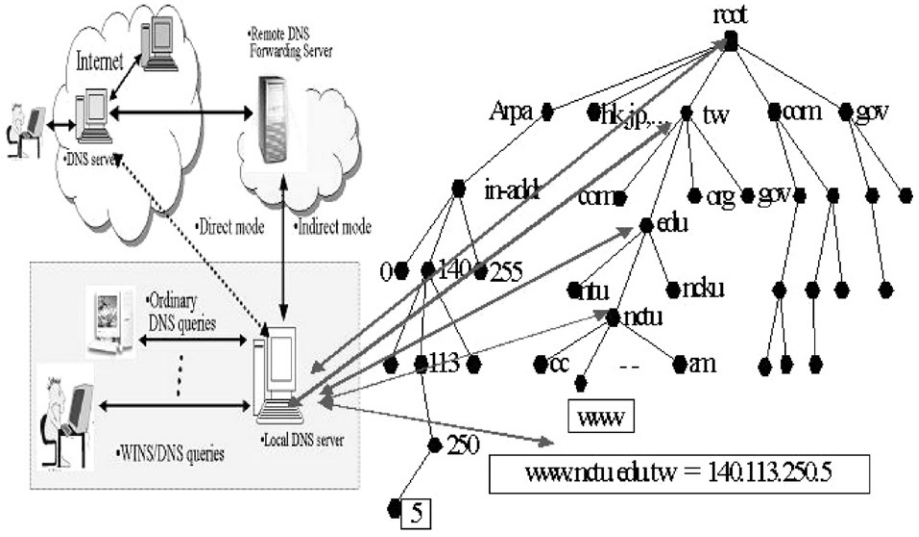


Fig. 1. DNS operation model.

systems (Kobryn, 1999). It simplifies the complex process of software design, making a "blueprint" for construction. An important part of the UML is the facilities for drawing use case diagrams. Use cases are powerful tools for analysts to use during the analysis phase to identify and partition system functionality (Cockburn, 1997). The methodology separates the system into actors and use cases. The basic model of uses cases is that actors interact to achieve their goals. Every path in use cases indicates a scenario and how a goal succeeds or fails.

An actor represents a role that an external entity such as a user, a hardware device or another system plays in interacting with the system. Actors must be external to the part of the system. They must supply stimuli to that part of the system, and receive outputs from it. Use cases describe the behavior of the system when one of these actors sends one particular stimulus. It describes the nature of the stimulus that triggers the use case; the inputs from and outputs to other actors, and the behaviors that convert the inputs to the outputs. Instead of showing "how to do", a use case shows "what to do" to reveal the functional requirements.

Since use case has been a standard notation, the using of use cases modeling might not only help us communicate with others, but also simplify the process of requirement gathering. We applied use cases modeling to help extract DNS knowledge while building the DNS ontology in the work of this paper.

2.3. *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 (Chandrasekaran et al., 1999). The role of ontologies is to capture domain knowledge and provide a commonly agreed upon understanding of a domain. 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.

As mentioned in Fernandez (1999), the ontology building process is still a craft rather than an engineering activity. Each development team usually follows its own set of principles, design criteria and phases on the ontology development process. In Fernandez et al. (1997), the authors of METHONTOLOGY explain that the life of an ontology moves on through the following states: specification, conceptualization, formalization, integration, implementation and maintenance. KA, documentation and evaluation are supporting activities that are carried out during the majority of these states. Since the DNS is still evolving, we have to update the DNS ontology whenever possible. The evolving prototype life cycle of METHONTOLOGY allows the ontologist to go back from one state to another if some definition is missed or wrong. So, this life cycle permits the inclusion, removal or modification of definitions at anytime of the ontology life cycle.

Tools 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 (Noy et al., 2000) is an easy-to-use KA tool that could construct the domain ontology and achieve the interoperability with other knowledge-representation systems. In Chen et al. (2002c), we built a DNS ontology using the METHONTOLOGY methodology and Protégé-2000 system from scratch. The DNS ontology has been verified by the experts and shown to be useful for helping build the sub-systems proposed in this paper.

3. DNS metrics: awareness of traffic distribution

How could a DNS administrator tell if her/his systems work as expected? Is there a set of golden metrics for checking? It seems that there is no easy answer. Any reasonable answer should depend heavily on the working environment of each site. With this mind, let us first check some DNS statistical data collected on Taiwan Academic Network in the past year to get the intrinsic ideas (MOECC, 2001).

3.1. Some DNS metrics from TANet

Taiwan Academic Network (TANet), the national teaching and research network co-developed by major universities and the Ministry of Education in July of 1990, is one of the major parts of the Taiwan Internet community (Tseng et al., 1996). TANet has backbone, regional network structure, and research-related information infrastructure to share information and offer co-operated opportunities that connect schools and research institutes at all levels.

Fig. 2 shows the distribution of the total traffic volume in bytes transferred between TANet and Internet in November 2001, in which the aggregation of DNS traffic occupies about 5.7% of the total. On the other hand, Fig. 3 gives a brief

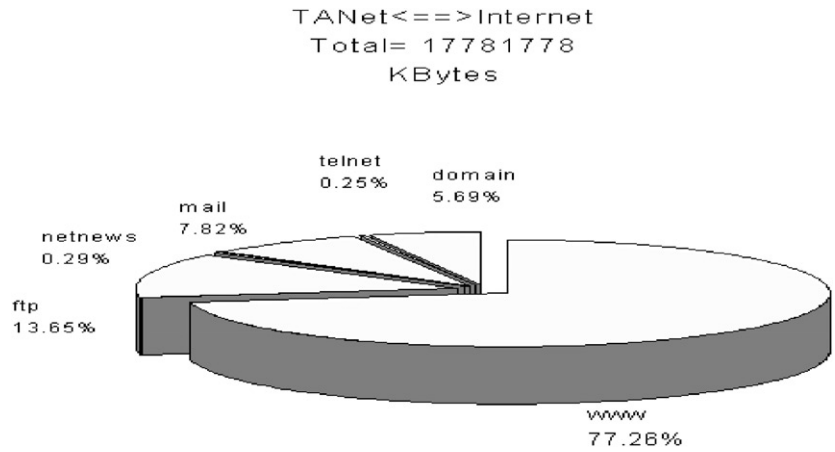
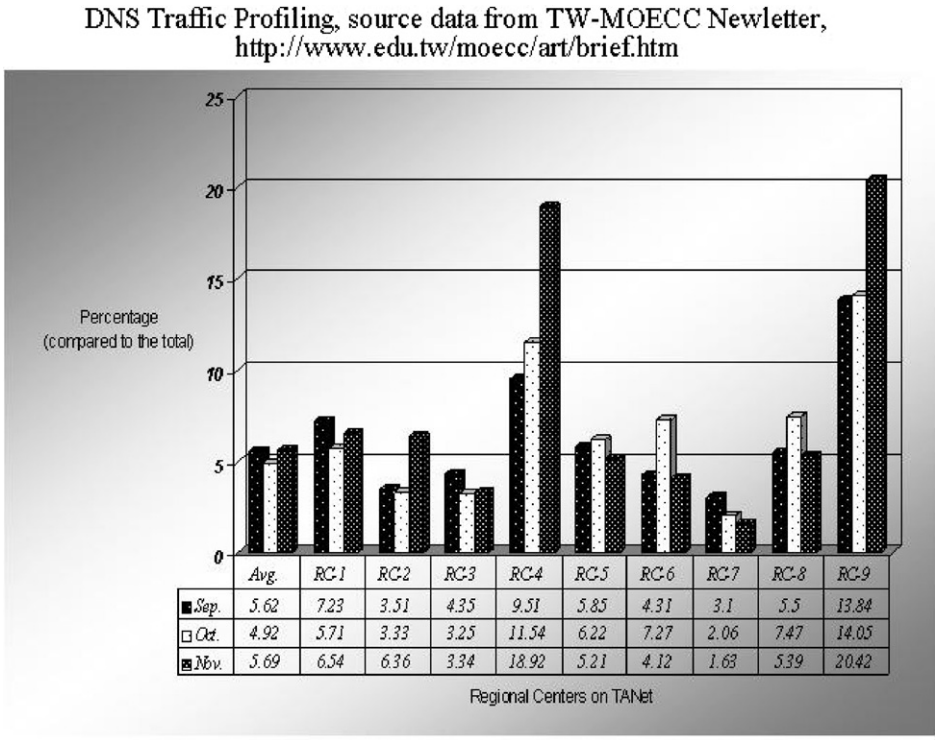


Fig. 2. Monthly traffic distribution on TANet, November 2001.



	Avg.	RC-1	RC-2	RC-3	RC-4	RC-5	RC-6	RC-7	RC-8	RC-9
Sep.	5.62	7.23	3.51	4.35	9.51	5.85	4.31	3.1	5.5	13.84
Oct.	4.92	5.71	3.33	3.25	11.54	6.22	7.27	2.06	7.47	14.05
Nov.	5.69	6.54	6.36	3.34	18.92	5.21	4.12	1.63	5.39	20.42

Fig. 3. DNS traffic distribution.

summary on the monthly DNS traffic statistical data for the nine regional centers on TANet, in the 3-month interval in September, October and November 2001. As shown in item 1 of Fig. 3, the average values of the DNS traffic distributions (as compared to the total traffic including WWW, SMTP, etc.) of the sites center around 5%. However, on some regional centers (e.g. RC-4 and RC-9), the monthly average values are more than 10% (or even bigger) and their traffic distributions obviously have larger variations than the others during the summary interval. In short, there might be some DNS configuration problems, network abusing, DNS anomalous activities or a mixture of the above, running on these sites during these time intervals.

3.2. *Unwelcome DNS traffic and other related problems*

According to domain expertise, the DNS traffic distributions vary from site to site and slightly from time to time on the same site. This makes sense for at least two reasons since (1) different time-to-live (TTL) values for domain entries and (2) different kinds of interesting Internet applications are deployed on each site. In general, DNS traffic consists of independent queries from different sources and of different types (though the number of different types is limited). However, in

Table 2
Potential top users/programs of a typical DNS server

Category	Examples
1. Normal server	Mail, proxy applications, etc.
2. DNS server	Downstream DNS forwarding servers
3. Malicious program	PC/network virus (mail, web, etc.), intrusion attempts, etc.
4. Underground servers	EDonkey (p2p), etc.

practice, most of the DNS queries are conducted on some major hosts in daily use. For example, Table 2 shows some potential top users (or programs) of a typical DNS server of our campus network. Under normal conditions, the servers listed in categories 1 and 2 are usually recognized and acceptable. According to our observations, the total traffic (of the above two) might occupy over 70–80%. However, on the other hand, the traffic introduced by hosts in categories 3 and 4 are usually not welcome. Either they are malicious programs or they are underground server processes. All of these might consume lots of network and system resources. If the administrators could not recognize the problem sources in time and handle them properly, under severe emergent situations, some may even crash the entire network or related systems.

According to the above reasoning, it seems that there might be something wrong on the sites with distribution values more than 10% as shown in Fig. 3. It is more than double (even triple) of the lowest value in the collected data sets and its traffic distribution has a far larger variation than the others. Assuming no configuration problems on the servers of the sites, there must be some anomalous activities going on during these time intervals unless there are some errors in the statistic data collected.

4. System architecture of the unifying framework

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 for debugging and tracing some network system issues.

4.1. Traditional approaches for fixing DNS problems

Table 3 shows a simple classification of DNS problems that most DNS administrators might encounter (Chen et al., 2002a). Many factors contribute to these and the important ones are listed below.

(1) Lots of novice DNS administrators 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 experts.

Table 3
A simple classification of typical DNS problems

Category	Examples
1. Configuration errors	Improper defaults, Lame Server, etc.
2. Planning and management	Invalid DNS dynamic update, WINS-to-DNS forwarding, etc.
3. Buggy software	Not immune to DNS spoofing, server vulnerability exploited, etc.
4. Attacks to the DNS systems	DDoS, forwarding attacks, etc.

(2) Many administrators who manage a small scale of network lack the experiences of dealing with global Internet traffic. Some serious problems (e.g. using buggy versions of DNS software, inappropriate configuration or planning problems, etc.) had not been identified or even been ignored on these sites. Initially, these small-scale anomalous activities may seen immaterial on the sites; however, these issues can become fatal problems when the overall traffic grows larger and larger.
(3) Moreover, given the importance of DNS servers, direct or indirect attacks on the DNS systems are common. The shutdown of Microsoft web sites on January 24, 2001 through the use of DoS attacks on their DNS servers (rather than their web servers) may be the beginning of a new wave of attacks against vulnerable DNS server infrastructures (Koh, 2001).

When inexperienced administrators find that there seems to be something wrong with their DNS servers, the traditional way for them to solve the problem is trying to find a domain expert who can offer expertise, or find some documents that describe similar situations, and then try to modify the solutions to satisfy their own situation. Usually, the domain expert will first 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 is verified to be correct, a plausible solution is proposed to fix the problem. 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. On the other hand, 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. 4 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. In practice, the expert will solve these kinds of DNS 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 thorough 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.

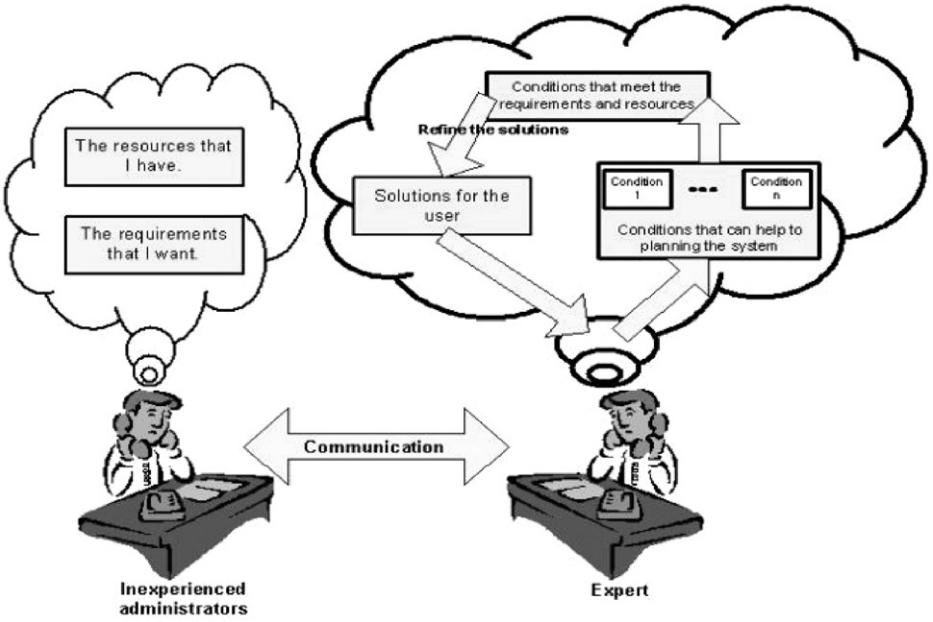


Fig. 4. Traditional ways for DNS planning and management.

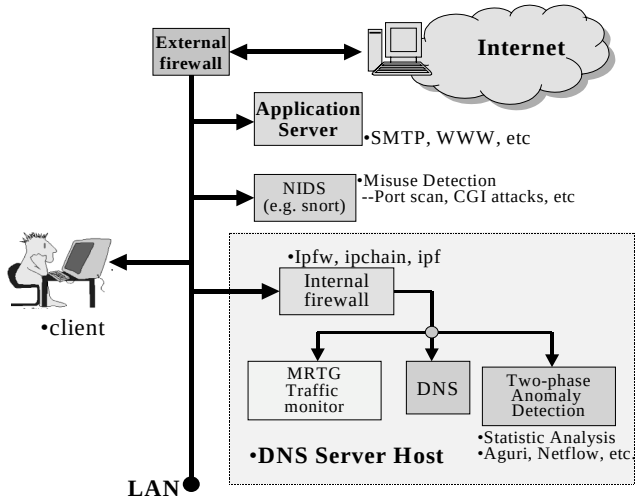


Fig. 5. A standard configuration of a DNS system.

4.2. *A standard architecture of a DNS system*

Fig. 5 shows a standard configuration of a single DNS system. In practice, to insure the proper operation, there are many issues that should be carefully designed and deployed. For example, according to domain expertise, the DNS-related security

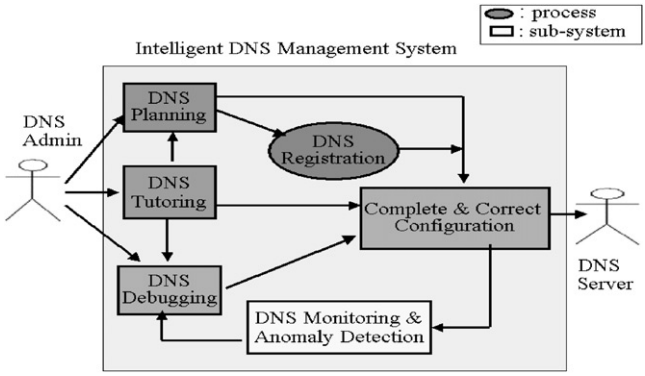


Fig. 6. Functional requirement analysis of the unifying system.

issues include: the protection of network segments (e.g. firewall), protection of DNS server hosts and DNS zone data (e.g. ACL). Moreover, these could be further turned into topics including DNS anomalies detection and identification, DNS dynamic update protection, etc. In theory, firewalls and ACL configuration on DNS servers could help block the intrusion attempts from outsiders. However, they might help little if the abusing activities are from insiders. In principle, network abusing or anomaly attempts from insiders could mainly be detected by the deployment of hybrid systems (e.g. Intrusion Detection System, analysis of the offline traffic statistics, mining of the trace log of the DNS servers, etc.). Finally, there are always cases that are undiscovered or missed by the local security system. The proper operation of a generic contact e-mail address like abuse@your.organization (or security@your.organizaion) could always help collect timely valuable messages from related remote sites for fighting these abusers. In short, these might not be easy jobs for most inexperienced DNS administrators.

4.3. Overall system architecture

In this paper, we propose a unifying framework as shown in Fig. 6 (e.g. including configuration, debugging, outstanding traffic monitoring and analysis, planning and management, tutoring, etc.) for supporting intelligent DNS management using web interface and expert system technology.

With the complexity of software development, the concept of software IC (i.e. Integrated Circuit) has been proposed for a long time. In principle, just like the design and deployment of the IC components in physical hardware systems, we can view some software components as the basic elements of the software systems under development. Therefore, people need not re-invent the wheel from scratch. For example, according to domain expertise (e.g. through use case analysis, etc.), the DNS management jobs could further be classified into the following main sub-domains: DNS configuration and maintenance, DNS debugging, DNS outstanding traffic monitoring and anomaly detection, DNS planning, DNS registration, DNS tutoring, etc. We can reuse the components and generate a new application by the combination of these components

whenever possible. To achieve the goal of re-using components, Object-Oriented Programming is an important roadmap. Before a software component comes out, we must test and debug the component first to make sure it works well. The OOP paradigm makes use of the concept of class, inheritance, polymorphism, etc. and follows the software engineering process to iteratively build and test software modules. These properties help the developers to abstract the problems and make high cohesion and low coupling software components possible. Therefore, we can ensure that the combination of these software components is bug free.

On the other hand, with the population of the World Wide Web, the web interface has been conceived and expected as a standard way for accessing typical information systems by most people. With these in mind, the objectives of the work reported in this paper are to use the World Wide Web on a continuing basis to support intelligent DNS management in which essential DNS issues (e.g. DNS configuration and maintenance, outstanding DNS traffic monitoring and analysis, DNS debugging, DNS planning and management, DNS tutoring, etc.) become an integral part of the DNS administration activities of an organization.

5. The DNS ontology

This DNS ontology is used as a basis to enhance the operation, planning and management of the DNS systems reported in this paper.

5.1. Building of a DNS ontology

As described in Gomez-Perez (1999), Fig. 7 shows problems in building a KBS from scratch. It is well known that KA is usually the bottleneck of building a KBS.

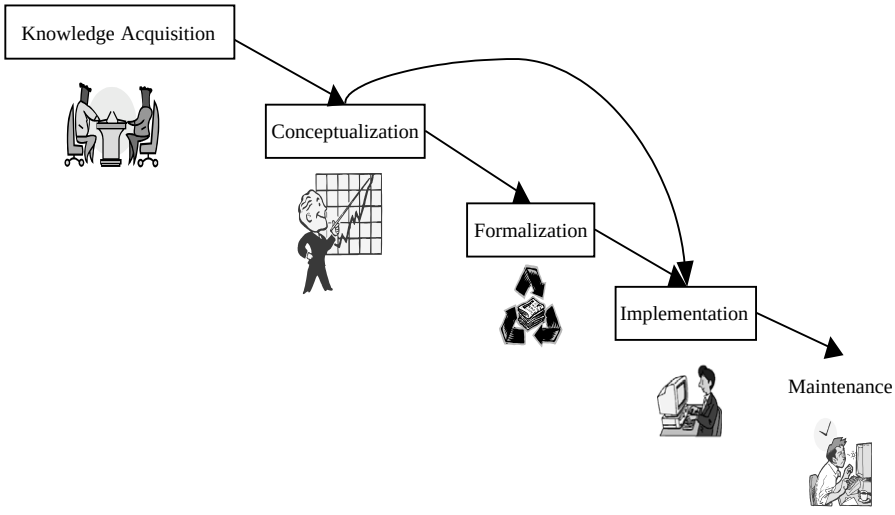


Fig. 7. Problems in building a knowledge-based system from scratch.

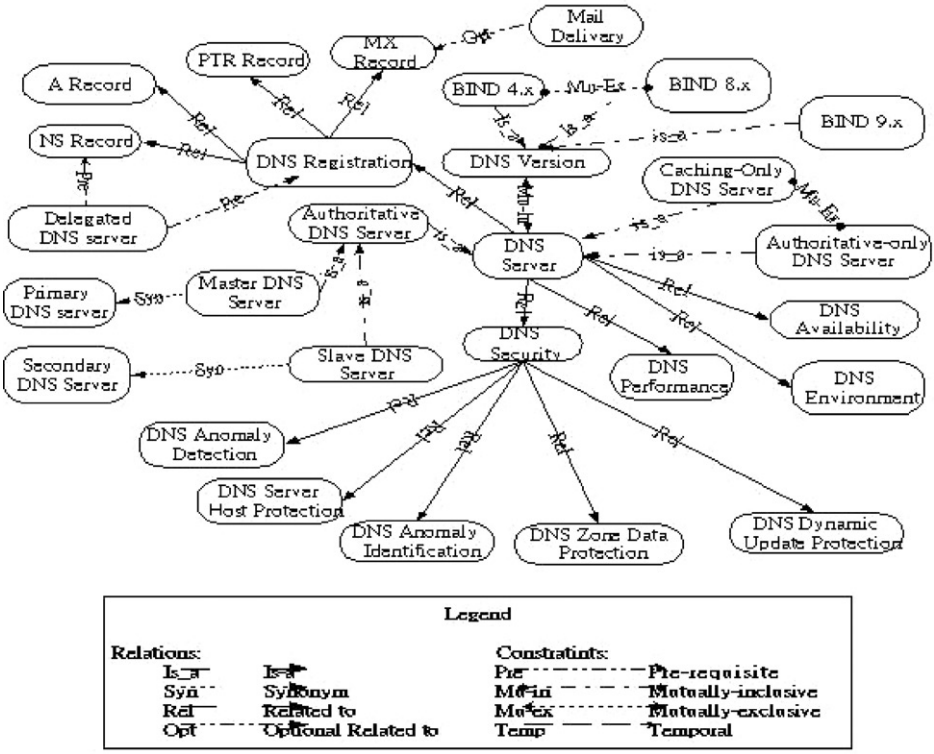


Fig. 8. Partial DNA ontology.

The process of the KA is to transfer domain knowledge into knowledge bases. The problems that we are faced with during the KA process are usually very hard. In general, KA involves: (1) elicitation (gathering) of data from the expert, (2) interpretation of the data to infer the underlying knowledge or reasoning procedure and (3) guided by this interpretation, creation of a model of the expert's domain knowledge and performance. In essence, each knowledge base is an extension of some application domain ontology, where the ontology provides a roadmap for the class of the concepts that will comprise the knowledge base. Just as a schema provides the organizing framework for a database, an ontology provides the framework for the domain knowledge base (Shadbolt et al., 2000). 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.

In the following, we will brief on some main points of the DNS ontology. From Fig. 8, we could find that there are mainly four types of relationships defined and several constraints identified:

- *Four types of relationships*: (1) "**is_a**" is generalization relationship, which can be used to describe the is_a relationship in the class hierarchy. For example, a master

DNS server is a DNS server; (2) "**Syn**" is synonym relationship, which could be used to denote two or more terms with the same meaning; (3) "**Rel**" is "related to" relationship, which denotes that there exists some relationship between these terms. For example, "Rel" relationship exists between the DNS server class and the DNS security class (of issues) and (4) "**Opt**" is optional relationship. For example, when we want to register a DNS server, the **DNS-MX** record is optional. According to domain expertise, while delivering a mail, a mail server would first try to look up some explicit MX record(s) corresponding to the destination of the message. But, if there is no explicit MX record found, the mail server would then try to find the implicit MX record by using its **DNS-A** record instead. In this way, if there is a corresponding A record, the mailing operation could still work; however, without explicit MX records defined, you cannot deploy the important mechanism of having relay sites for the smooth operation of delivering mails to your site.

- *Identification of Constraints*: (1) Pre-requisite constraint: one term/relationship depends upon another. For example, currently, if we would like to have a BIND DNS server with multi-threading capability support, we have to deploy BIND version 9. (2) Temporal constraint: one term/relationship must occur before another. For example, DNS registration is needed before a delegated DNS server exists. (3) Mutually inclusive constraint: one term/relationship requires another for its existence. For example, if we want to install a DNS server running BIND software packages, we need to choose from a list of available versions as well; (4) Mutually exclusive constraint: one term/relationship must not co-exist with another. For example, a caching-only DNS server is not an authoritative-only DNS server and vice versa.

5.2. Knowledge hierarchy of the DNS ontology

Fig. 9 shows the knowledge hierarchy of the DNS Ontology. 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 above, 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.

(2) *Resources*: In DNS ontology hierarchy, there are some components that do not belong to DNS group or management strategies. They can affect the

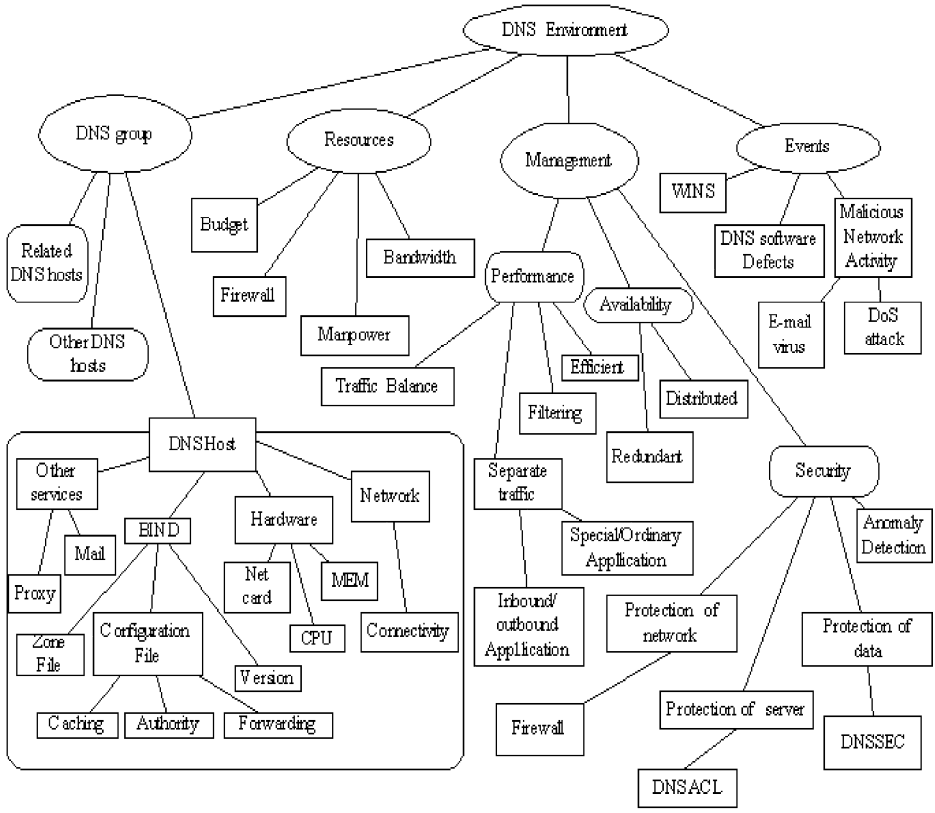


Fig. 9. Knowledge hierarchy of the DNS ontology.

management plans and some configurations; e.g. when budget is sufficient, hardware can be upgraded to high level to improve the efficiency.

(3) *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 are configurations in each DNS host different but also many tradeoffs need to be considered. Availability, performance, and security are major issues in this field.

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

6. DNS Knowledge acquisition

It is important to bear in mind that KA is an independent activity in the ontology development process (Fernandez et al., 1997). 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.

6.1. DNS KA 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 KA tools. Since ontologies specify the constraints that domain knowledge should satisfy, they can be used to direct the acquisition of domain knowledge (Heijst et al., 1997).

In this paper, to help extract knowledge from experts and construct the DNS ontology hierarchy, we propose an efficient KA algorithm (e.g. DNS Ontology Construction Algorithm) to fast conceptualize DNS domain knowledge by using a hybrid method consisting of the brainstorming and use case modeling. A simplified scheme can be illustrated as shown in Fig. 10 and the operation details will be described later. In fact, with a little modification, the same algorithm could be applied to other domains for ontology construction.

6.2. Use case modeling for DNS KA

As described in Sugumaran and Storey (2002), domain knowledge could typically be captured in use cases. In principle, the primary motivations for use case creation: (1) gaining an understanding of the problem, (2) capturing an understanding of the

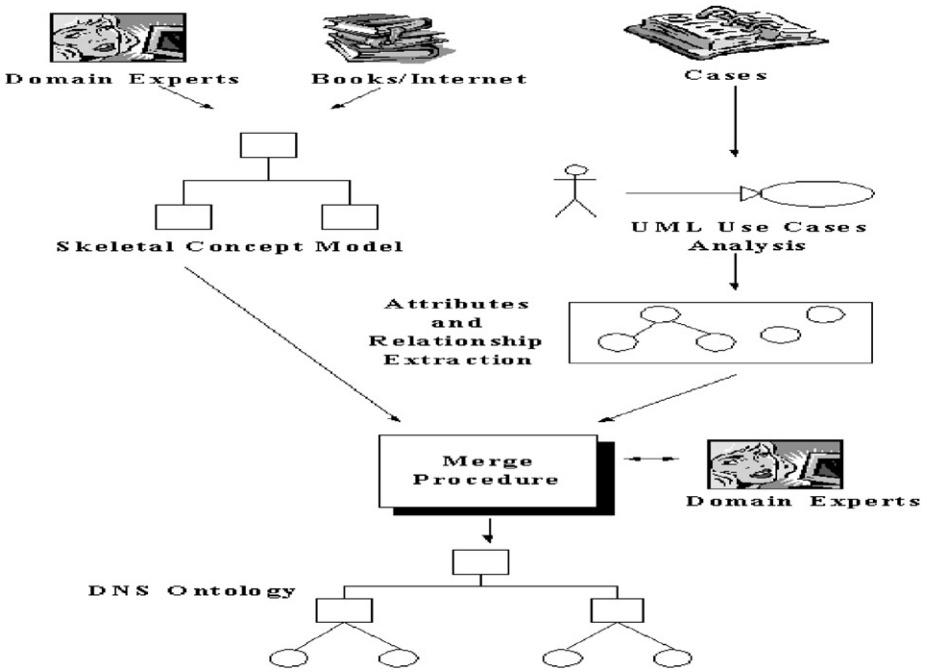


Fig. 10. DNS knowledge acquisition process.

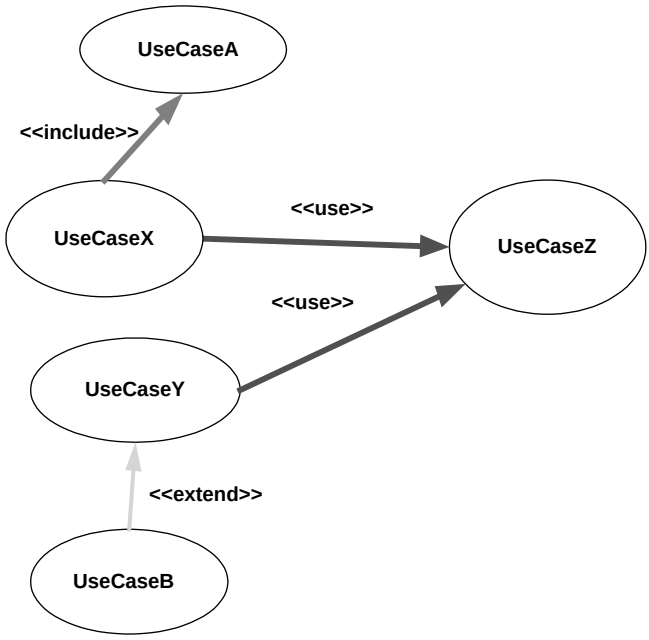


Fig. 11. Typical relationships of use cases.

proposed solution and (3) identifying candidate objects. Thus, we can derive important terms from use cases and view these terms as concepts. In other words, we can construct ontology classes. Moreover, use case relationships and the corresponding diagrams could help analysts to structure use cases; thus making them easier to maintain. Usually, as shown in Fig. 11, there are three relationships existing in use case diagrams.

(1) *Include*: One use case may include the functionality of another use case as parts of its normal processing. For example, we know that each time we call UseCaseX, we would call UseCaseA as well. In other words, we could build the "*composition*" relationship between these two classes. The composition relationship is like "Rel" relationship of an ontology diagram. Furthermore, the ordering of use cases can also help us set up the prerequisite, temporal, mutually inclusive and mutually exclusive constraint relationships as well.

(2) *Use:* A specific use case may be used by one or more use cases, so it helps to reduce duplication of functionality by factoring out common behavior into use cases that are re-used many times. After some abstraction, this introduces the "is_a" relationships of an ontology diagram. For example, as shown in Fig. 11, there exists "use" relationship between UseCaseX and UseCaseZ, we know that UseCaseZ might be used by many use cases, which include UseCaseX. Based on the observation, we know that UseCaseZ might be a "*critical case*" in our

system, and useCaseZ can be *reused* in many use cases (e.g. UseCaseX, UseCaseY, etc.). If we view each use case as a class, we can make use of the concept of OOP to construct our ontology hierarchy. Because of the reuse property, the abstraction of class Z is needed, such that we can generate many reused classes based on class Z. Hence, for utilizing the reuse property, we should first establish an abstraction class, and then the concrete classes could inherit the abstraction class. In this way, we could establish the "is_a" relationships of an ontology diagram.

(3) *Extend*: One use case may extend the behavior of another use case to accommodate for more complex problems. It is typically used when exceptional circumstances are encountered. For example, as shown in Fig. 11, there exists "extend" relationship between UseCaseB and UseCaseY, it means that UseCaseB is an exceptional case of UseCaseY. We can view UseCaseB as an extension of UseCaseY, and then concept B would "*inherit*" concept Y. The inheritance relationship is like the "is_a" relationship of an ontology diagram.

6.3. Use case modeling and DNS ontology building

As mentioned in Gaines and Shaw (1993), 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. 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 developed. However, as different from the repertory grid methodology, we propose another approach for extracting the concepts and attributes by using a hybrid method consisting of the brainstorming and use case modeling.

The power of a few critical cases described in terms of relevant attributes to build domain ontology is remarkable. This is because it is often easier and more accurate for the experts to provide critical cases rather than domain ontology. Hence, UML use cases analysis is adequate for our DNS KA algorithm because it is usually not hard for experts to enumerate critical cases and will not take too much time from them. In addition, we could also get lots of use cases from many well-known domain-related mailing lists that contain enough and not too much information, so the knowledge engineers can modify the ontology hierarchy easily. After the extractions, we would then decompose these concepts into a lot of sub-components by class abstraction and inheritance. A simplified scheme is outlined in Fig. 12.

In the following, we will give the details of the DNS ontology construction algorithm. The descriptions about the use relationships listed below follow the ones mentioned in Section 6.2.

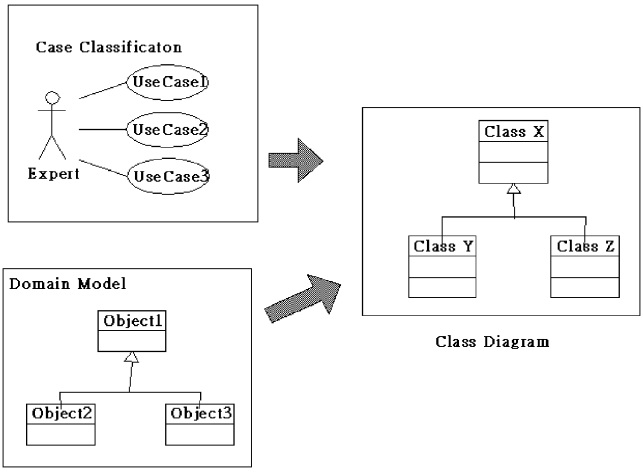


Fig. 12. DNS modeling and DNS ontology construction.

6.3.1. DNS ontology construction algorithm

Input: Every kind of DNS case.

Output: DNS Ontology.

Step 1. Build the skeleton DNS ontology (top-down).

Step 1.1: Interview with DNS domain experts (or read the DNS books, etc.) for extracting primary DNS concepts from scratch.

*Step*1.2: Build the skeletal concept model of the DNS ontology by following a top-down brainstorming method.

Step 2. Initiates (or conducts) use case modeling.

Step 2.1: Case Input from the email, newsgroup, or system administrators, etc.

Step 2.2: Transform these cases into UML Use Cases by following the principles below.

Step 2.2.1: Actors Identification.

Step 2.2.2: Define what the actors want to do with the system.

Step 2.2.3: Define (draw) the internal state transition diagram.

Step 2.2.4. Define what the actors would receive from the system.

Step 3: Conduct the attributes and relation extraction. After the transformation, attribute and relationship extraction is required.

Step 3.1: Analyse and decompose the cases into small components (e.g. availability, security, performance, etc.). Each use case, we could view as a concept (class).

Step 3.2: Define or identify the relationships between the use cases.

Step 3.2.1: If there exists some ''include'' relationship between UseCaseX and UseCaseY, build the ''*composition*'' relationship between these two classes.

- As described in Section 6.2, the composition relationship is like ''Rel'' relationship of the ontology diagrams. Furthermore, the ordering of use cases could also help set up the pre-requisite, temporal, and mutually inclusive constraints as well.

Step 3.2.2: If there exists some "use" relationship between UseCaseX and UseCaseZ, we could infer that UseCaseZ is used by many use cases, which include UseCaseX. After some abstraction, this introduces the "is_a" relationship of an ontology diagram as described in Section 6.2.

Step 3.2.3: If there exists some "extend" relationship between UseCaseX and UseCaseY, we can view UseCaseX as an extension of UseCaseY, and then concept X would "*inherit*" concept Y. The inheritance relationship is like the "is_a" relationship of an ontology diagram.

Step 4: Merge the ontological components collected in steps 1 and 3 above.

Step 4.1: Associate the results (e.g. concepts) of step 3 with the concepts of step 1 by consulting the synonym table first. If there are some matches, we could establish "Syn" relationships of the ontology diagram and go to step 5.

- We should have a synonym table that describes whether two different terms indicate the same meaning.

Step 4.2: If we can find the appropriate concept(s) from the concepts of Step 1, we have to identify the related parts in the ontology tree under construction and associate the concepts from step 3 to the concept from Step1 by putting the former in the lower level of the ontology hierarchy and link them to that component related to the concept from Step1. After that, go to step 5.

Step 4.3: If we could not find any appropriate known concept, we have to create a new concept for step 3 and link it to the ontology sub-trees.

- Since it is usually impossible to define a complete concept hierarchy from the beginning, we need to create new concepts whenever necessary.

Step 5: Experts verify the ontology under constructing.

Step 5.1: Verify the correctness of the skeletal ontology under construction, including those new created components and the hierarchical structure.

Step 5.2: If the experts find any ontological components not covered by the constructed skeletal ontology, then the construction process will go back to step 2.

Step 6: After experts' verification, the DNS ontology is constructed to cover DNS domain knowledge.

6.3.2. Examples for illustrating the DNS ontology construction algorithm

As shown in Fig. 12, the whole DNS KA process could be roughly divided into three parts, namely, Part 1 = {Step 1}, Part 2 = {Steps 2 and 3}, Part 3 = {Steps 4, 5, and 6}.

(1) *Part* 1: In essence, it is impossible for us to construct a complete ontology hierarchy from the beginning. In part 1, all we have to do is simply extract important concepts. After some brainstorming process, we could construct a rough skeletal concept model of the DNS ontology as shown in Fig. 13. We might lose some concepts, but that does not matter. We will have a more complete model after the whole process.

Step 1: Build the Skeleton DNS ontology (top-down).

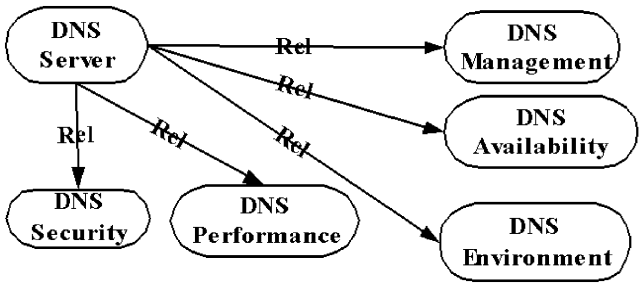


Fig. 13. An initial skeletal concept model of DNS.

In step 1, we interviewed with the domain experts for extracting DNS knowledge. The domain experts can help extract concepts. In additional, they can also help define or identify the relationships between these concepts as well (e.g. ConceptX should be performed before ConceptY, ConceptX would occur with the ConceptZ, etc.). Step 2 initiates (or conducts) use case modeling.

(2) *Part* 2: Since the skeletal concept model constructed in step 1 is not complete, it needs a refinement process to increase its completeness.
Step 2. Initiates (or conducts) use case modeling.
Step 3. Conduct the attributes and relationships extraction.

(3) Part 3: Part 1 is top-down and part 3 is bottom-up. Now we have to merge them in step 4. After that, the domain experts help verify the DNS ontology tree built up until now {e.g., steps 5 and 6}.
Step 5: Experts verify the ontology under construction.
Step 6. The ontology generated is confirmed OK.

In the following, we would give two examples for helping to demonstrate the operation details of the KA process during Steps 2–4.

- *Example* 6.1: How to register a DNS entry (e.g. NS server record, address record, etc.)?

Step 2: Fig. 14 indicates the DNS registration concept.
Step 3: As shown in Fig. 14, we can easily extract the DNS registration concept.
Step 4: We could not find any appropriate concept to relate the DNS Registration concept with Fig. 13 (e.g. the rough DNS skeletal concept model). Hence, as shown in Fig. 15, we have to create a new concept, which might be named "DNS Registration", for relating with DNS server concept.

- *Example* 6.2: We have a use case consisting of DNS Security class that contains many issues including: DNS zone data protection, DNS anomaly identification, DNS server host protection, and DNS Dynamic update protection, etc.

Step 2: Fig. 16 shows the concepts and relationships about the DNS security class.
Step 3: As shown in Fig. 16, we can first extract six concepts, namely, DNS security, DNS dynamic update, DNS zone data protection, DNS anomaly detection,

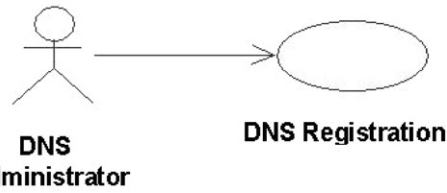


Fig. 14. A use case indicating the DNS registration concept.

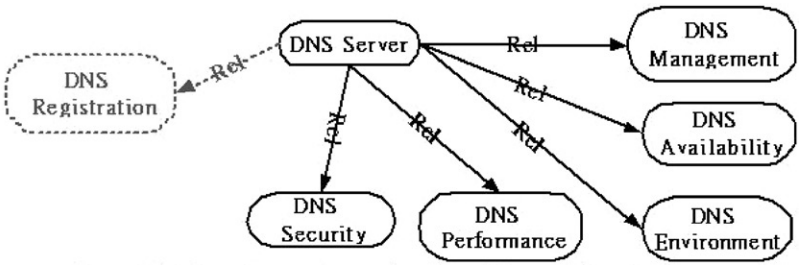


Fig. 15. The DNS ontology after the adding of the DNS registration concept.

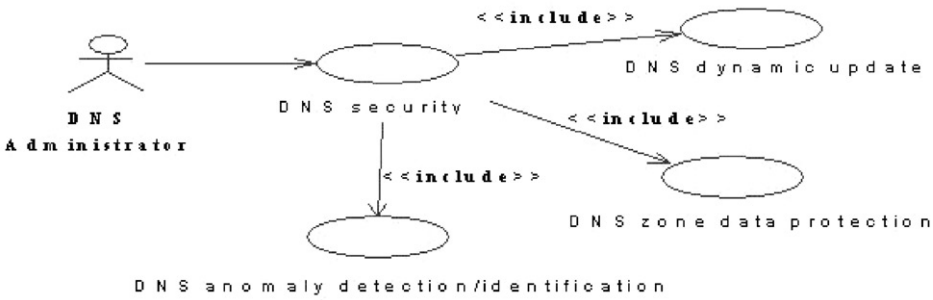


Fig. 16. Concepts and relationships about the DNS security class.

and DNS anomaly identification. Moreover, besides the concepts of extraction, we could find that the "include" relationship exists between the DNS security concept and other concepts. Therefore, by following our former heuristics, we can construct a part of the ontology hierarchy concerning the DNS security class as shown in Fig. 17.

Step 4: We find that there is a "DNS Security" concept related with "DNS Server" concept. Hence, we could connect these two concepts directly. After the operation, the DNS ontology could be illustrated as shown in Fig. 18.

7. Experiments and evaluations

In this paper, we propose a unifying framework for supporting intelligent DNS management using web interface and expert system technology. Fig. 19 shows the

Fig. 17. A concept model of the DNS security class.

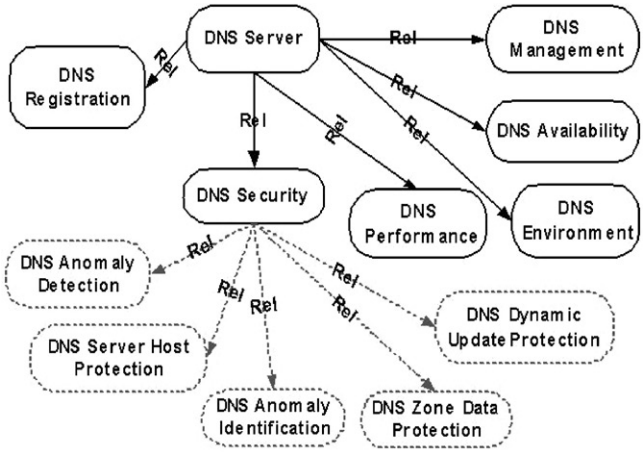


Fig. 18. The DNS ontology after the adding of six more DNS security-related concepts.

web page of our ongoing iDNS-MS (e.g. intelligent DNS Management System) project. Interested users could refer to the web site (e.g. http://idns-ms.nctu.edu.tw). For supporting local users' access, some of the diagrams listed below are shown in our native language (e.g. Traditional Chinese Big5 code, mainly used in Taiwan and Hong-Kong).

7.1. Overview of the iDNS-MS project

Fig. 20 shows the current status of the project. In practice, the daily DNS management jobs could be partitioned into the following main sub-domains: DNS configuration and maintenance, DNS debugging, DNS outstanding traffic monitoring and anomaly detection, DNS planning, DNS registration, DNS tutoring, etc. In essence, the DNS systems involved are distributed in nature and deployed across different network segments and different platforms. At the time of the writing: (1) the main operating systems deployed are FreeBSD 4.6.2, and Solaris 2.5 with necessary patches; (2) the primary DNS software package deployed are BIND 8.3.3

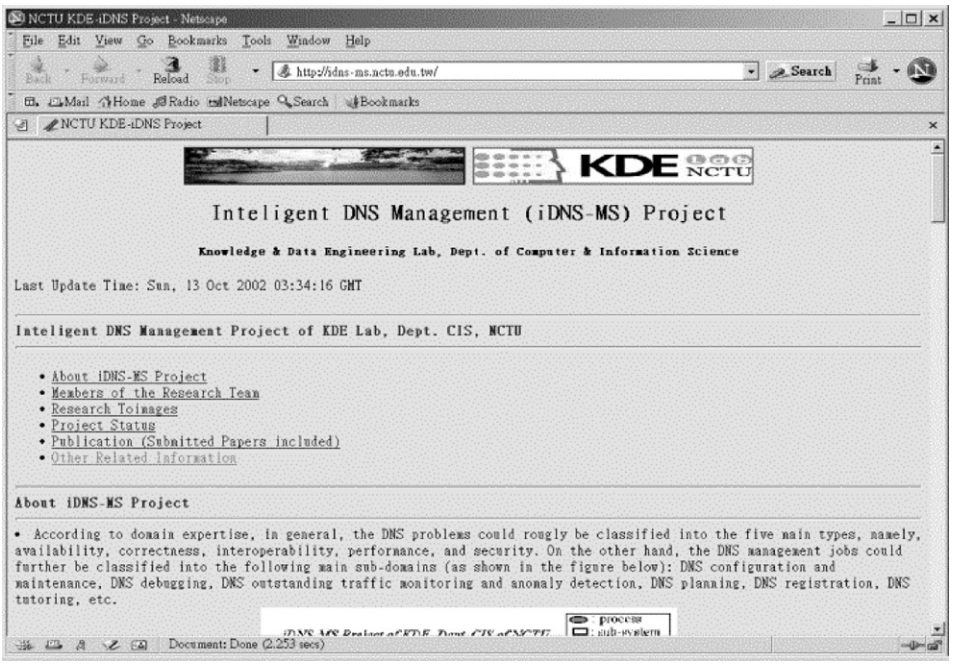


Fig. 19. Overview of the iDNS-MS project.

and 9.2.2rc1 and (3) the web server packages deployed is Apache 2.043. Some sub-systems have been prototyped and deployed for everyday use. Others are under development. For example: (1) Fig. 21 shows a scenario on the observed DNS server, it seems that there were anomalous DNS traffic. We implemented a set of simple PERL script for running with MRTG (Oetiker, 1998) to interpret the DNS traffic (i.e. port 53) statistics captured by the packet filtering program, or firewall; (2) Fig. 22 shows a snapshot of the inference result of the experimental DNS Planning and Management sub-system while it is used to help debug some configuration errors in the zone file upload and (3) Fig. 23 shows the home page of the Chinese DNS tutoring sub-system.

7.2. Evaluation and discussion

Table 4 shows a list of ideal DNS administration tools or sub-systems that could be useful for everyday use and hope to become an integral part of the DNS administration activities of an organization in the future. In the following, we will give some more descriptions about each of them.

(1) For some categories (e.g. categories 2, 3, etc.), there are many commercial products (e.g. DNS configuration tools like DNSexpert, or DNS debugging

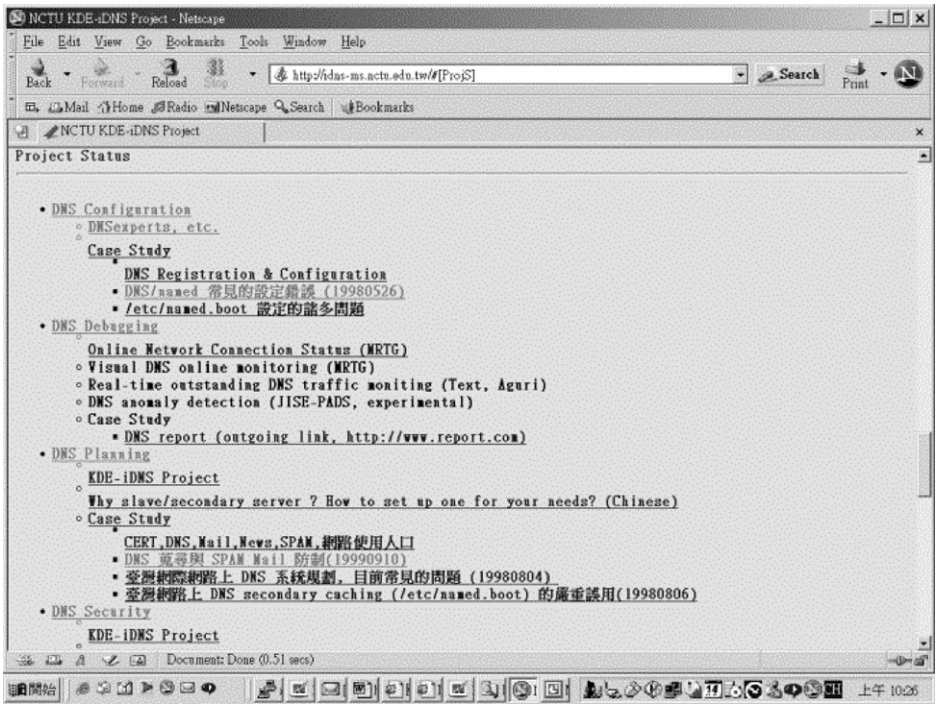


Fig. 20. Project status.

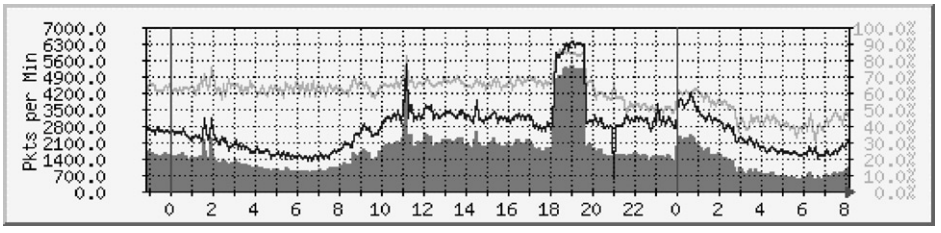


Fig. 21. Online passive DNS monitoring.

web-site, http://www.dnsreport.com, etc.). However, we can also find some free command-line text tools (e.g. dnswalk, doc, etc.) from public domain or on the system by default (e.g. ping, etc.) with similar functions. Usually only with some minor modifications, we can easily have such a set of similar tools with web interface.

(2) On the other hand, we could get some specific program sources from public domain or vendors and modify them to fit our own requirement. For example,
- As mentioned in (Chen et al., 2002b), we have designed and deployed the online DNS monitoring system (e.g. category 4) as shown in Fig. 21, and the online DNS outstanding traffic identification sub-system (e.g. category 1).

Fig. 22. An inference (e.g. debugging) result of our iDNS-PMS.

Fig. 23. Online DNS tutoring system with FAQ, case studies, etc.

Table 4
Overview of important DNS administration activities

Category	Modules/sub-systems in need
1. DNS anomaly detection	Online DNS summary report (modified aguri) as in Chen et al. (2002b). Our experimental anomaly detection engine as mentioned in Chen et al. (2002b).
2. DNS configuration tools	DNSexpert, Dlint, etc. as mentioned in Ou (2002).
3. DNS debugging tools	(1) www.dnsreport.com (e.g. commercial sites). web tool similar to doc, dnswalk (e.g. public domain tools, text command lines). (2) Ping (e.g. testing tool of network connectivity). A default tool on most systems.
4. DNS monitoring	(1) Online real-time DNS traffic monitoring (MRTG). (2) Online DNS summary report sub-system, with outstanding traffic identification capability as mentioned in Chen et al. (2002b).
5. DNS planning and management	Our experimental Intelligent DNS planning and management system (iDNS-PMS), still under development.
6. DNS traffic statistics (offline)	Network traffic statistical engine. For example, Cisco Netflow mentioned in Cisco NetFlow (2001).
7. DNS tutoring system	(1) Online basic DNS tutoring course. (2) FAQ (native-language support). (3) Case study.

- The network administration team at our campus had implemented an offline network traffic statistic engine (e.g. category 6). With proper input selections, we can get the functionality in category 6 above.

(3) Finally, for some sub-systems (e.g. especially the ones in category 1, 5, or 7), we have no better choice but to try developing them ourselves.

- Either there are no successful systems known to the public yet, or there are some obvious social barriers (e.g., languages supporting issues in DNS tutoring systems).

In the near future, it is supposed that all these might become an integral part of the DNS administration activities of an organization.

8. Concluding remarks

In this paper, we have shown two ways in which explicit DNS ontology can be used during knowledge engineering: for knowledge elicitation and for computational design. Our main contributions are: (1) to present the characteristics of the various DNS administration activities; (2) to propose a unifying framework of intelligent

DNS management environment that integrates DNS monitoring, debugging, planning, configuration, and tutoring sub-systems, etc. using web interface and expert system technology to help insure the healthy operation of the DNS systems of an organization.

The presented study is an analysis of what problems and difficulties most DNS administrators might encounter and provide insights into how various DNS assistant sub-systems could be designed and deployed to solve the complex problems or alleviate these DNS administration job loadings. In this paper, we propose a KA algorithm, DNS Ontology Construction Algorithm, to help knowledge engineers extract knowledge from experts and construct the DNS ontology. Our experience has shown the effectiveness of this method. On the other hand, the systematic analysis has provided useful support and has aided in improving the design. During computational design application, the ontology can be used to determine the suitability of problem solvers for the DNS systems including: (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, and (4) most importantly to provide expertise to those administrators whenever they need it. As our experience with a first simple prototype has shown, the paradigm of using DNS ontology to build a unifying framework for intelligent DNS management works good and effective; in the near future, it is supposed that all these might become an integral part of the DNS administration activities of an organization.

Future research will focus on several issues. First, we will complete the more complex sub-system, Intelligent DNS Planning and Management System (iDNS-PMS). 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 including: IPv6, multilingual DNS, intrusion detection mechanisms concerning DNS, etc. The very extensions should be reflected on each of the appropriate sub-systems in our proposed unifying framework. Finally, for letting more local DNS administrators gain the insight of DNS administration in a systematic and effective approach, more case studies will be integrated into the DNS tutoring sub-system with native language support (e.g. Chinese Big5, etc.) incorporated.

The authors wish to acknowledge that most of the presented research was conducted while they were affiliated in part with the MOE Program of Excellence Research of the Republic of China under Grant No. MOE89-E-FA-04-1-4.

References

Albitz, P., Liu, C., 2001. DNS and BIND, 4th Edition. O'Reilly & Associates Inc., Sebastopol, CA.

Chandrasekaran, B., Jorn, R., Josephson, V., Richard Benjamins, 1999. What are ontologies, and why do we need them? IEEE Intelligent Systems 14 (1), 20–26.

Chen, C.S., Tseng, S.S., Liu, C.L., Ou, C.H., 2002a. Design and implementation of an intelligent DNS configuration system. In: Proceedings of Fourth International Conference on Advanced Communication Technology, Korea, February, 2002, pp. 230–235.

Chen, C.S., Tseng, S.S., Liu, C.L., 2002b. A distributed intrusion detection model for the domain name system. Special issue on parallel and distributed systems. Journal of Information Science and Engineering 18, 999–1009.

Chen, C.S., Tseng, S.S., Liu, C.L., Ou, C.H., 2002c. Building a DNS ontology using METHONTOLOGY and Protege-2000. In: Proceedings of 2002 International Computer Symposium Workshop on Artificial Intelligence, December, 2002, Vol. 2, pp. 1853–1860.

Cisco NetFlow, 2001. NetFlow Services Solutions Guide. Cisco Systems, Inc. http://www.cisco.com/univercd/cc/td/doc/cisintwh/intsolns/netflsol/nfwhite.htm.

Cockburn, A., 1997. Structuring Use Cases with Goal. Journal of Object-Oriented Programming (Parts 1 and 2), the September–October issue, 35–40; and the November–December issue, 56–62.

Fernandez, M.L., 1999. Overview of methodologies for building ontologies. In: Proceedings of the IJCAI-99 Workshop on Ontologies and Problem-Solving Methods: Lessons Learned and Future Trends, Vol. 18 (4). CEUR Publications, pp. 1–13.

Fernandez, M.L., Gomez-Perez, A., Juristo, N., 1997. METHONTOLOGY: from ontological art towards ontological engineering. Workshop on Ontological Engineering, Spring Symposium Series. AAAI97 Stanford, USA.

Gaines, B.R., Shaw, M.L.G., 1993. Knowledge acquisition tools based on personal construct psychology. Special Issues on "Automated Knowledge Acquisition Tools" of the Knowledge Engineering Review 8 (1), 1993.

Gomez-Perez, A., 1999. Tutorial on Ontological Engineering. In: Proceedings of the IJCAI-99 Tutorial Program, No. 2. http://www.ontology.org/main/presentations/madrid/theoretical/pdf.

Heijst, G.V., Schreiber, A.T., Wielinga, B.J., 1997. Using explicit ontologies in KBS development. International Journal of Human-Computer Studies 46 (2/3), 183–292.

Kobryn, C., 1999. UML 2001: A standardization odyssey. Communications of the ACM 42 (10), 29–37.

Koh, J.L., 2001. Recent Developments and Emerging Defenses to D/DoS: The Microsoft Attacks and Distributed Network Security. SANS Institute, URL: http://www.sans.org/infosecFAQ/DNS/developments.htm.

Man-Mice, 2002. Domain Health Survey for. COM-November 2002, http://www.menandmice.com/6000/61_recent_survey.html.

Moecc, Taiwan, 2001. Network Traffic Statistics on TANet, MOECC Newsletter, URL:http://www.edu.tw/moecc/art/brief.htm.

Musen, M.A., 1992. Dimensions of knowledge sharing and reuse. Computers and Biomedical Research 25, 435–467.

Noy, N.F., Fergerson, R.W., Musen, M.A., 2000. The knowledge model of Protege-2000: Combining interoperability and flexibility. In: Proceeding of the Second International Conference on Knowledge Engineering and Knowledge Management (EKAW'2000), Juan-les-Pins, France.

Oetiker, T., 1998. MRTG: The Multi Router Traffic Grapher. In: USENIX LISA Conference, Boston, MA, December, 1998, pp. 141–147.

Ou, C.H., 2002. Design of an intelligent dns planning and management system. Master Thesis, Department of Computer and Information Science, National Chiao-Tung University, Taiwan.

Shadbolt, N., O'Hara, K., Cottam, H., 2000. The use of ontologies for knowledge acquisition. In: Cuena, J., et al. (Ed.), Knowledge Engineering and Agent Technology. IOS Press, Amsterdam, pp. 12–14.

Studer, R., Benjamins, R., Fensel, D., 1998. Knowledge engineering: principles and methods. IEEE Transactions on Data and Knowledge Engineering 25 (1–2), 161–197.

Sugumaran, V., Storey, V.C., 2002. Ontologies for conceptual modeling: their creation, use, and management. In Data & Knowledge Engineering 42, 251–271.

Tseng, S.S., Su, L.S., Chao E. H., 1996. TANet: Taiwan Academic Network. In: Proceedings of INET96, Montreal, Canada, June, 1996, pp. 24–28.